DAInamic V.Z.W.
Heide 4
3171 WESTMEERBEEK
016/698623

TOOLKIT 5

TOOLKIT 5

TOOLKIT 5

DCR 5 A

```
01-2-ARR- Toolkit 5  (c) BAInaaic 1983 rel. 1.0
02-0-BAS- BASICODE II TOELICHTING / EIPLANATION
03-1-OBJ-SOURCE BASICODE II
04-1-OBJ- OBJ  BASICODE 2  02EC-0E00 : CALLH0300
05-0-BAS- BAInaaic Debugging Tool (BDT)
06-2-ARR-ROUTINE0
07-2-ARR-MOVE0
08-2-ARR-LENGTH0
09-2-ARR-MHEN0
10-2-ARR-REG20
11-1-OBJ-SOURCE BAICON V3.3 G.BRUITERS TOOLKIT 5
        (c) BAInaaic
12-1-OBJ-400 AAA OBJECT BAICON V3.3 TK5 (c) BAIna
        aic
```

update 830710

TOOLKIT 5

DCR 5 B

```
01-0-BAS- BOOTSTRAP FGT Gothisch (300-FD0)
02-1-OBJ-FGT GOTHISCH 300-FD0
03-0-BAS-DEMO FGT GOTHISCH see line 2010 for hard
        copy
04-0-BAS- BOOTSTRAP PAINT
05-1-OBJ-PAINT 300 3AF
06-0-BAS-DEMO PAINT
07-0-BAS- BOOTSTRAP PAINT II
08-1-OBJ-300 4AF PAINT II
09-0-BAS- PAINT DEMO 1
10-0-BAS- PAINT DEMO 2
11-0-BAS- DCR LIST IX.I
12-0-BAS- DCR LIST + SPOOLER 20K  (DDL)
13-1-OBJ- SPOOLER 20K V2.3E
14-0-BAS- INIT SPOOLER
15-0-BAS- DCR LIST IX.ID FOR SPOOLER
16-0-BMA-SPOOLER 20K V2.3E (bufferdia. 12k)
17-2-ARR-.
```

update 830710

TOOLKIT 5

How to use the programs

1/ BASICODE

Load & run " BASICODE II TOELICHTING/EXPLANATION"

The Source-code of BASICODE II is available on tape for documentation and minor modifications. (type DNA "1"-file)

2/ DAInamic Debugging Tool

See documentation P.1-5

3/ DAICOM communication program

see Newsletter 15 p. 112 - 125

4/ FGT "GOTHISCH"

see DEMO program FGT GOTHISCH

5/ PAINT I & PAINT II

see DEMO-programs

6/ DCR LIST IX.I

first version is without spooler, second version is with 20K buffer-spooler.

DCR-cover is for printers with subscript/underscript facilities. control characters are for EPSON type III, EPSON FX and RX printers. (and compatible printers e.g. STAR)

BASICODE II

```
500  PRINT CHR$(12):PRINT " BASICODE"
510  PRINT
520  PRINT
530  PRINT " NEDERLANDSE UITLEG / ENGLISH EXPLANATION ? (N/E)"
540  A=GETC:IF A<>78.0 AND A<>69.0 THEN 540
550  IF A=78.0 THEN 2000
560  GOTO 1000
1000 PRINT CHR$(12)
1010 PRINT "THE BASICODE MACHINE LANGUAGE PROGRAM.":PRINT
1020 PRINT "WITH THIS PROGRAM IT IS POSSIBLE TO EXCHANGE PROGRAMS WITH"
1030 PRINT "OTHER COMPUTERS."
1035 PRINT "THIS PROGRAM CAN BE LOADED BY KEYING (FROM BASIC):"
1040 PRINT "'UT','R' AND RETURN . WHEN THIS IS DONE TYPE 'B'"
1050 PRINT "THE COMPUTER IS THEN BACK IN BASIC MODE."
1060 PRINT "THIS TRANSLATION PROGRAM IS STARTED BY CALLM #300."
1070 PRINT "THERE IS THEN A CHOICE OF 7 POSSIBILITIES:"
1080 PRINT
1090 PRINT "1 LOADING A BASICODE PROGRAM FROM TAPE:HERE,THE MID$-"
1100 PRINT "  STATEMENT WILL BE AUTOMATICALLY CORRECTED (THIS"
1110 PRINT "  STATEMENT IS DIFFERENT FROM OTHER COMPUTERS)."
1120 PRINT "2 SAVING FROM LINE 1000 OF THE PRESENT BASICODE PROGRAM:"
1130 PRINT "  ALSO HERE THE MID$-STATEMENT WILL BE CORRECTED;BESIDES,THE"
1140 PRINT "  SPACES BETWEEN LINE NUMBER AND LINE WILL BE SUPPRESSED."
1150 PRINT "3 LOADING OF THE BASICODE PROGRAM (WITHOUT ANY CORRECTION)."
1160 PRINT "4 SAVING OF THE TOTAL PROGRAM (WITHOUT ANY CORRECTION)."
1170 PRINT "5 CANCELLING THE PRESENT PROGRAM AND (RE)STORING THE"
1180 PRINT "  BASICODE SUBROUTINES."
1190 PRINT "6 MERGING THE BASICODE SUBROUTINES WITH THE PRESENT"
1200 PRINT "  PROGRAM."
1210 PRINT "7 STARTING THE (NEW) BASICODE PROGRAM."
1211 PRINT "                                        TYPE SPACE";
1220 GOSUB 1500:PRINT CHR$(12)
1230 PRINT "BEFORE LOADING A BASICODE PROGRAM, THE PRESENT PROGRAM"
1240 PRINT "WILL BE SENT TO THE EDIT BUFFER (THIS TAKES SOME TIME)."
1250 PRINT "WHEN THIS IS READY THE ANNOUNCEMENT 'TYPE SPACE' APPEARS."
1260 PRINT "THEN YOU CAN START THE RECORDER AND TYPE SPACE WHEN THE "
1270 PRINT "LEADER IS HEARD.THE REST TAKES PLACE AUTOMATICALLY."
1280 PRINT "WHEN THIS ALL IS DONE AND LOADED THE FOLLOWING ANNOUNCE-"
1290 PRINT "MENTS CAN APPEAR:"
1300 PRINT "1 DATA DROPOUT ERROR (IF THERE IS A DROPOUT ON TAPE)."
1310 PRINT "2 CHECKSUM ERROR (FAILURE ON TAPE)."
1320 PRINT "3 NO TAPE ERROR (IF EVERYTHING ON TAPE IS OK)."
1330 PRINT "AT THIS MOMENT THE EDITBUFFER WILL BE EMPTIED AND AFTER"
1340 PRINT "THIS THE PROGRAM IS READY FOR USE.THERE ARE POSSIBLY"
1350 PRINT "A FEW SMALL ERRORS SUCH AS E.G. SYNTAX ERROR ,OVERFLOW ETC."
1360 PRINT "THESE WILL BE CORRECTABLE AS USUAL IN THE EDITMODE."
1370 PRINT
1380 PRINT "BEFORE SAVING THE PROGRAM, IT WILL BE MOVED INTO A BUFFER."
1390 PRINT "THIS ALSO TAKES SOME TIME (DEPENDING ON THE LENGTH OF"
1400 PRINT "PROGRAM), WHEREAFTER THE ANNOUNCEMENT 'SET RECORD ,START"
1410 PRINT "TAPE AND TYPE SPACE'APPEARS. THE SPACE KEY SHOULD NOW BE"
1420 PRINT "PRESSED IN. THE BASICODE PROGRAM THEN COMES OUT OF THE"
1430 PRINT "COMPUTER (CASSETTE INTERFACE)."
1440 GOSUB 1500
1450 PRINT CHR$(12)
1460 GOTO 1600
1500 A=GETC:IF A=0.0 THEN 1500
1501 RETURN
1600 PRINT CHR$(12)
1610 PRINT " START BASICODE ? (Y/N)"
1620 A=GETC:IF A<>89.0 AND A<>78.0 THEN 1620
1630 IF A=78.0 THEN 500
1640 PRINT "TYPE CALLM#300":END
```

```
2010    INT
2020  PRINT "MET DIT PROGRAMMA IS HET MOGELIJK PROGRAMMA'S MET ANDERE"
2030  PRINT "COMPUTERS UIT TE WISSELEN IN BASICODE FORMAAT."
2040  PRINT "HET PROGRAMMA WORDT GELADEN DOOR ACHTEREENVOLGENS IN TE"
2050  PRINT "TOETSEN :'UT','R' EN RETURN. NA HET LADEN TOETST U 'B' DAN"
2060  PRINT "IS DE COMPUTER TERUG IN DE BASIC MODE."
2070  PRINT "HET PROGRAMMA WORDT GESTART DOOR CALLM #300."
2080  PRINT "DAN IS ER KEUS UIT 7 MOGELIJKHEDEN:"
2090  PRINT
2100  PRINT "1 HET LADEN VAN EEN BASICODE PROGRAMMA: HIERBIJ WORDT"
2110  PRINT "  AUTOMATISCH HET MID$-STATEMENT GECORRIGEERD (DEZE IS BIJ"
2120  PRINT "  DE DAI AFWIJKEND VAN ANDERE COMPUTERS)."
2130  PRINT "2 WEGSCHRIJVEN VANAF REGEL 1000 VAN HET AANWEZIGE PROGRAMMA."
2140  PRINT "  OOK HIER WORDT HET MID$-STATEMENT GECORRIGEERD;BOVENDIEN"
2150  PRINT "  WORDEN DE SPATIES TUSSEN REGELNUMMER EN REGEL ONDERDRUKT."
2160  PRINT "3 HET LADEN VAN EEN BASICODE PROGRAMMA (ZONDER CORRECTIE)."
2170  PRINT "4 WEGSCHRIJVEN VAN HET BASIC PROGRAMMA (ZONDER CORRECTIE)."
2180  PRINT "5 WISSEN VAN HET AANWEZIGE BASIC PROGRAMMA EN HET"
2190  PRINT "  (OPNIEUW) INVOEREN VAN DE BASICODE SUBROUTINES."
2200  PRINT "6 HET MERGEN VAN DE BASICODE SUBROUTINES MET HET AANWEZIGE"
2210  PRINT "  PROGRAMMA."
2220  PRINT "7 HET STARTEN VAN HET (NIEUWE) BASICODE PROGRAMMA."
2230  PRINT "                                              TYPE SPACE";
2240  GOSUB 1500
2250  PRINT CHR$(12):PRINT "VOOR HET LADEN VAN EEN BASICODE PROGRAMMA,WORDT
 HET"
2260  PRINT "AANWEZIGE PROGRAMMA NAAR DE EDITBUFFER GESTUURD (DIT DUURT"
2270  PRINT "ENIGE TIJD). ALS DIT KLAAR IS VERSCHIJNT DE MEDEDELING "
2280  PRINT "'TYPE SPACE'. DAN START U DE RECORDER EN DRUKT U OP DE"
2290  PRINT "SPATIEBALK ALS U DE FLUITTOON (= LEADER) HOORT."
2300  PRINT "DE REST GEBEURT AUTOMATISCH."
2310  PRINT "ALS HET PROGRAMMA GELADEN IS VERSCHIJNT EEN VAN DL VOLGENDE
2320  PRINT "MEDEDELINGEN:"
2340  PRINT "1 DATA DROPOUT ERROR (BIJ EEN DROPOUT OP DE BAND)."
2350  PRINT "2 CHECKSUM ERROR (STORING OP DE BAND)."
2360  PRINT "3 NO TAPE ERROR (GEEN BANDFOUT)."
2370  PRINT "OP DIT MOMENT WORDT DE EDITBUFFER GELEEGD,WAARNA
2380  PRINT "PROGRAMMA GEREED VOOR GEBRUIK IS. MOGELIJK
2390  PRINT "KLEINE FOUTJES BV SYNTAX ERROR,OVERFLOW ETC.DEZE ZIJN OP
2400  PRINT "DE GEBRUIKELIJKE MANIER IN DE EDITMODE TE VERBETEREN."
2410  PRINT
2420  PRINT "VOOR HET WEGSCHRIJVEN VAN HET PROGRAMMA,WORDT DIT"
2430  PRINT "VERPLAATST NAAR EEN BUFFER. DIT DUURT ENIGE TIJD"
2440  PRINT "(AFHANKELIJK VAN DE LENGTE VAN HET PROGRAMMA),WAARNA DE"
2450  PRINT "MEDEDELING 'SET RECORD,START TAPE TYPE SPACE'VERSCHIJNT."
2460  PRINT "NU DRUKT U OP DE SPATIEBALK,WAARNA DE BASICODE UIT DE"
2500  PRINT "COMPUTER (CASSETTE INTERFACE) KOMT."
2600  PRINT "                                        TYPE SPACE";
2610  GOSUB 1500
2620  GOTO 1600



500   PRINT CHR$(12):PRINT " BASICODE"
510   PRINT
520   PRINT
530   PRINT " NEDERLANDSE UITLEG / ENGLISH EXPLANATION ? (N/E)"
540   A=GETC:IF A<>78.0 AND A<>69.0 THEN 540
550   IF A=78.0 THEN 2000
560   GOTO 1000
1000  PRINT CHR$(12)
1010  PRINT "THE BASICODE MACHINE LANGUAGE PROGRAM.":PRINT
                                                EXCHANGE PROGRAMS WITH"
```

DAInamic DEBUGGING TOOL

Dainamic Debugging Tool (DDT) by W. COREMANS

1.DESCRIPTION :

DDT is a program ment for testing machine language programs
The functions this program can handle are :

1. Putting up to 255 (default = 100) breakpoints in a mlp (Machine Language Program). This is done in an edit-like way with the possibility to change, insert or delete breakpoints
2. Testing the mlp in 3 different modes :
   -A trace mode to test the mlp step by step
   -A breakpoint mode which runs your mlp in real time until a breakpoint is reached
   -A breakpoint mode which gives you also breaks at rom-entrypoints (ex. bankswitching)
3. Ability to change all the 8080 processor's registers and flags during testing
4. Full control of the stack. You can display the stack-contents and change it in a similar way as you do with breakpoints (i.e. change, insert or delete 16-bit data, returnaddresses or anything else which is on the stack)
5. Disassemble your mlp or the firmware in any rom-bank, with an option to display bytes after RST1, RST4 and RST5 as DATA
6. A number of options concerning the display of mnemonics and or registers, tracing RST instructions or not, disable/enable interrupts a.o.

DDT is partly written in BASIC ,partly in machine language. The machine language part is placed in 2 stringarrays which are loaded and relocated by the BASIC program. This means it has no fixed place in memory, so the mlp to be tested can be located anywhere

Running DDT is very easy : first load your mlp, eventualy adapt the heap pointer and at last LOAD and RUN DDT. When your mlp is located in higher ram addresses (ex. near to the screen-ram) you don't have to change the heap pointer, so your mlp will be located in the free ram space behind the BASIC program

Breakpoints can be set at will into the mlp you want to test, but be sure the breakpoint points to the begin of an 8080 instruction. If this is confusing then just keep in mind that you may specifie all addresses displayed in a disassembling run, exept when the address is followed by DATA

Special care has been taken concerning interrupt handling. Your mlp can change the interrupt mask at will or even reset the interrupt system, DDT will always find its way back.

2. HOW TO USE :

After you have loaded your machine language program and DDT type RUN. DDT will start with loading its arrays, then it will ask you for a command :

G for GO
H or ? for HELP,
X for display and change the 8080 registers
M for changing the MODE : TRACE or BREAKPOINT
B for display and change BREAKPOINTS
S for display and change the STACK
O for changing the OPTIONS
D for DISASSEMBLING

For a detailed description of the commands and their function see part 3 COMMANDS

2.1 INITIALISATION :

The first thing you HAVE TO DO is setting the 8080 programcounter PC to the address where you would like to start your mlp
Use the X-command to do this
Also it may be needed to put some initial conditions into some registers or put some flags into the flag register to test the behaviour of the mlp
If your mlp does a setting of the STACKPOINTER on an address differing from F900H (ex. LXI SP or SPHL) then you also have to set the STACKPOINTER in DDT to this address.
If the mlp is normally started from a call instruction you may first put the returnaddress on stack. When you forget to do this DDT will detect an error when the mlp returns. This is done with the S-command

2.2 RUNNING :

Now every thing is ready to start testing your mlp. The way your mlp will be tested depends on the mode. The default mode is the TRACE mode. This means DDT will execute one 8080 instruction at a time and display the mnemonics BEFORE executing it. To start the execution you have to give the G command, to stop the execution hit the spacebar . If you also want to display the 8080 registers you will have to change the options with a O command (see part 3). The registers are displayed after the execution of the 8080 instruction.
At any moment you can change the mode in wich DDT is running. Use the M command (see part 3 . However take care : if you chose the mode 2 (breakpoints in ram) be shure you have DEFINED one ore more breakpoints in your mlp, otherwise DDT migth never come back. Chosing mode 1 (also breakpoints at EI in rom) gives no problems, DDT will come back.
You can specifie up to 100 breakpoints in your mlp. If this doesn't seem sufficient to you, you will have to EDIT linenumber 60100 and change BRPTMAX%=100 into BRPTMAX%=255 or less, and save again DDT.
THERE CAN NOT BE MORE THEN 255 BREAKPOINTS.
Mode 1 is somehow an artificial mode but very useful. Every time an EI instruction is executed DDT will take control again.

This can be very useful to see which rom routines are called by your mlp. EX. if your mlp prints some message on the screen, it uses the instructions RST 5,DATA 03 which selects rombank 2 and jumps to E003. Since the last 2 instructions of the code used to switch the banks are (at address C6FBH) EI,RET and DDT forces an interrupt after RET (= the next instruction after EI) DDT will detect a breakpoint at E003 (2). This is a general rule when bankswitching is performed. In combination with the breakpoints you defined by yourself this gives a very clear trace of the code executed.

Another very powerfull feature of DDT is the full control of the stack. If you often write assembly language programs you certenly will agree with me that the stack can cause a lot of trouble. Ex. when yo forgot to push or pop a register, or when you reverse the order of pushing and popping registers. All this can result in wrong returnaddresses and eventualy a crash of the mlp before you even noticed your mlp had started.
DDT offers you the possibility to display at any moment the full contents of the stack and to change it if needed.
When giving the S-command the information on the stack will be displayed as 16-bit data as many as there realy are on the stack (from top of stack i.e. the stackpointer in the very begin, to the actual stackpointer).
If you see that you forgot to push a register on stack just do an insert of it after chosing the correct position with the cursor keys (part 3).
When a pop has been forgotten, you can do a delete in the same way.
When some register or returnaddress should be wrong, do a change
If you want to restart your mlp from a certain point on and therefore want to set your stackpointer a number of bytes back please do a delete of the undesired part of the stack instead of changing the stackpointer with the X-command. Changing the stackpointer with the X-command will set a new top of stack, so DDT will loose the information above this new top of stack.
The maximum length of the stack that DDT can handle is 255 bytes -12 bytes to save all the registers, program counter and the stackpointer itself. If you realy need more then this you can reset the top of stack (with the X-command) on another address, but you will loose the information, so you will have to test your mlp in pieces.

The X-command is used to change the 8080 registers, the flags, the stackpointer and to initiate the program counter. The display of the registers is the same as in utilities exept that also the rombank is displayed when the program counter is between E000H and F000H. Changing the registers however is done in another, more clear way. If you change the program counter in an address between E000H and F000H you also have to give the rombank (0 to 3).

The safest way to test your mlp surely will be with an assembly listing of the mlp near you on your desk. If you don't have a listing you can disassemble the mlp (eventualy on your printer) to see where you can put meaningfull breakpoints.

3. COMMANDS

3.1 GENERAL RULES

Several commands expect numerical data. ALL the numbers you will have to enter are HEX. NUMBERS. Every time you have to enter hex. numbers the same subroutine is used. Beside hex. digits this subroutine will only accept CHAR DEL, RETURN and LEFT ARROW as valid inputs. LEFT ARROW is used to quit the input of a hex. number and the old value will be kept and displayed again

To return to command mode the spacebar is generaly used

3.2 GO

Giving the G-command forces DDT to run your mlp until a breakpoint is reached. If DDT is in trace mode (mode 3) you will have to hit the spacebar to stop tracing. To continue just give again the G-command.

3.3 HELP

If you type H or ? a menu of all the commands will be displayed on screen. To keep DDT as short as possible no further help functions are build in. Use the spacebar to return to command mode

3.4 DISPLAY/CHANGE THE 8080-REGISTERS

To display and or change the registers you have to type X as in utilities
You are asked to type the name of the register you want to change : i.e.
type A to change the accumulator, F to change the flags etc...
The cursor will be positioned and you have to enter a hex. number.
As mentioned in 3.1 you can quit the change of a register by typing left arrow. Because changing the stackpointer could destroy the system-ram, the screen ram or DDT, the value of the stackpointer will not always be accepted. A message will give you the reason why.

3.5 MODE

Typ M to change the mode. A menu of the 3 modes will be displayed.
To switch to another mode just keyin the number of the mode.
You can also use this command to see in which mode you are. An arrow is pointing to the current mode

3.6 DISPLAY/EDIT BREAKPOINTS

Typing B will show you all the breakpoints. If you have to change a breakpoint, select it using the cursor keys and type C (of change).
To add new breakpoints type I (of insert), to delete a breakpoint type D (of delete). The order of the breakpoints is not important, they don't have to be in ascending order.

3.7 DISPLAY/EDIT STACK

Typing S will show you all the stackcontents, the highest address in the top left corner of the screen. The way to change, delete or insert data on the stack is exactly the same as in display/edit breakpoints. However the order of the data on the stack is very important. So be carefull with inserting or deleting data.

3.8 OPTIONS

Type O to change the options. The following questions will be asked :
a) 'LIST MNEMONICS ?'
Answering Y(es) to this question will cause the 8080-mnemonics to be listed before executing the instruction. Answering N(o) will stop listing the mnemonics. The default is Yes.
b) 'LIST REGISTERS ?'
Answering Yes to this question will cause the 8080-registers to be listed after the execution of the instruction. Answering No will stop listing the registers. The default is No.

REMARK : Be shure to answer Yes to at least one of these questions, because else all visual information about the execution of your mlp will be lost.
c) 'TRACE RST ?'
Answering Yes to this question will trace all the instructions of a RST 1, RST 4 or RST 5 if encountered. Answering No to this question will cause DDT to execute RST x, DATA yy as one single instruction. The default is No
d) 'DO YOU WANT TO DISABLE INTERUPTS ?'
Answering Yes to this question will allow disabeling of interups if instructed to. This means you cannot return to DDT command mode between the DI and the EI instructions by hitting the spacebar. Answering No to this question will allow you to return to DDT command mode always.
The default is No.

3.9 DISASSEMBLE

When typing D, DDT will ask you the startaddress from where to disassemble and eventualy the rombank if the address is between E000H and F000H. To stop disassembling hit the spacebar. To continue answer Yes to the question 'CONTINUE [Y/N] ?', to stop anwer No.

```
1     GOTO 60000
10    REM ****************************
20    REM * Dainamic Debugging Tool *
30    REM * BY W. COREMANS 08/04/83 *
40    REM ****************************
50    COMMAND$="G?HXMBSOD"
60    CURSOR 0,0:PRINT "COMMAND :  ";:CURSOR 10,0
70    G=GETC:IF G=0 THEN 70:IF G<>#D THEN PRINT CHR$(G);
80    FOR I=1 TO LEN(COMMAND$)
90    IF G=ASC(MID$(COMMAND$,I-1,1)) THEN 120
100   NEXT:CURSOR 30,0:PRINT "type H or ? for HELP";
110   SOUND 1 0 15 0 FREQ(300.0):WAIT TIME 5:SOUND OFF :GOTO 60
120   CURSOR 0,0:PRINT SPC(60);
130   CURSOR XCUR,YCUR:ON I GOSUB 1000,2000,2000,3000,4000,5000,6000,7000,8
000
140   XCUR=CURX:YCUR=CURY:GOTO 60
1000  REM ************************
1010  REM * PUTS THE BREAKPOINT  *
1020  REM * CODE IN ROUTINE$ AND *
1030  REM * GO UNTIL BREAKPOINT  *
1040  REM ************************
1050  REM ---IMPOSSIBLE TO TRACE RST 0---
1060  IF PC<8 THEN MESSAGE$="PLEASE INITIATE THE PROGRAM COUNTER!":ANSW$="
":LINE=0:GOSUB 56000:RETURN
1070  REM ---GET INSTRUCTION---
1080  PNTR=PC:GOSUB 54000:REM ---TEST BREAKPOINT---
1090  IF MDE1BPFLAG=1 THEN PC=PC+1:IF BRPTFLAG=1 THEN PC=PC-1:MDESET=2:GOSU
B 59000
1100  POINTER=PC+1:CALLM GETBYTE,POINTER:OPERANDLOW=BYTE:POINTER=PC+2:CALLM
 GETBYTE,POINTER:OPERANDHIGH=BYTE
1110  IPTR=PC:IF BRPTFLAG=1 THEN IPTR=CODEPTR1:PC=POK(6.0)
1120  CALLM GETBYTE,IPTR:INSTRUCTION=BYTE
1130  IF MDE=3 THEN PC=POK(6):NOPEI=#FB
1140  LENGTH=1:IF INSTRUCTION<>#FF THEN LENGTH=ASC(MID$(LENGTH$,INSTRUCTION
,1))
1150  REM ---DISPLAY EVT. MNEMONIC---
1160  IF LSTFLAG=1 THEN IF LSTMNEMFLAG=1 THEN GOSUB 55000
1170  IF MDE=3 THEN IF INSTRUCTION=#CF OR INSTRUCTION=#E7 OR INSTRUCTION=#E
F THEN IF TRACERSTFLAG=1 THEN LENGTH=2:MDESET=2:GOSUB 59000:GOSUB 58010
1180  REM ---DON'T EXECUTE EI OR DI---
1190  IF INSTRUCTION=#F3 THEN INSTRUCTION=0:INTSAV=PEEK(#5F):POKE #5F,0
1200  IF INSTRUCTION=#FB THEN INSTRUCTION=0:POKE #5F,INTSAV IOR #40
1210  FOR I=POK(6) TO POK(8)+2:POKE I,0:NEXT
1220  POKE POK(7)+1-LENGTH,NOPEI
1230  POKE POK(7)+2-LENGTH,INSTRUCTION
1240  POKE POK(7)+3-LENGTH,OPERANDLOW
1250  POKE POK(7)+4-LENGTH,OPERANDHIGH
1260  POKE POK(8),#C7
1270  RST0FLAG=1:RSTCALLFLAG=0
1280  IF INSTRUCTION=#CD OR INSTRUCTION IAND #C7=#C4 OR INSTRUCTION IAND #C
7=#C7 THEN RSTCALLFLAG=1:IF MDE<>3 THEN POKE POK(7)+1-LENGTH,#FB
1290  PCNEXT=POINTER+LENGTH-2
1300  IF MDE<>3 THEN IF MDESET=2 THEN POKE POK(8),#C3:POKE POK(8)+1,PCNEXT
MOD 256:POKE POK(8)+2,PCNEXT SHR 8
1310  SPOLD=SP
1320  GOSUB 50000:REM ---VARIABLES INTO MLPSTACK$---
1330  CALLM ROUTINEPTR:REM ---GO UNTIL BREAKPOINT---
1340  MDESET=MDE:GOSUB 59000:FFB=(PEEK(#85) SHL 8)+PEEK(#84)
1350  IF RSTCALLFLAG=1 THEN POKE MLPSTACKPTR+12,PCNEXT MOD 256:POKE MLPSTAC
KPTR+13,PCNEXT SHR 8
1360  GOSUB 51000:REM ---REGISTERS TO VARIABLES---
1370  IF PC=POK(8) OR PC=POK(8)+1 THEN PC=PCNEXT
1380  IF MDE=2 THEN PC=PC-1
1390  IF MDE=1 THEN PC=PC-1:CALLM GETBYTE,PC:MDE1BPFLAG=1:IF BYTE<>#C7 THEN
1400  REM ---TEST IF INTERUPTED---
1410  IF MDE=3 AND SPOLD-SP>2 AND RSTCALLFLAG=0 THEN MDESET=2:GOSUB 59000:R
ST0FLAG=1:GOTO 1320
1420  IF MDE=3 AND SP>TOPOFSTACK THEN SP=TOPOFSTACK:PC=0:MESSAGE$="---ERROR
---STACK OVERFLOW; automatic reset!":ANSW$=" ":LINE=0:GOSUB 56000
1430  LSTFLAG=0:IF RSTCALLFLAG<>1 OR MDE=3 THEN LSTFLAG=1:IF LSTREGFLAG=1 T
HEN GOSUB 53000
1440  IF MDE=1 OR MDE=2 THEN RETURN
1450  IF PEEK(#5F) IAND #40=0 THEN 1000
1460  G=GETC:IF G=#20 THEN RETURN
1470  GOTO 1000
2000  PRINT CHR$(12)
2010  CURSOR 10,20:PRINT "G-GO"
2020  CURSOR 10,19:PRINT "H/?-HELP"
2030  CURSOR 10,18:PRINT "X-DISPLAY/CHANGE REGISTERS CONTENTS"
2040  CURSOR 10,17:PRINT "M-MODE"
2050  CURSOR 10,16:PRINT "B-ADD/DELETE/LIST BREAKPOINTS"
2060  CURSOR 10,15:PRINT "S-LIST STACK CONTENTS"
2070  CURSOR 10,14:PRINT "O-OPTIONS (about list and trace)"
2080  CURSOR 10,13:PRINT "D-DISSASSEMBLE"
2090  POKE #75,#20
2100  CURSOR 0,0:PRINT "Spacebar to continue";
2110  G=GETC:IF G<>#20 THEN 2110
2120  PRINT CHR$(12):POKE #75,#5F
2130  RETURN
3000  REM ****************************
3010  REM * DISPLAY/CHANGE REGISTERS *
3020  REM ****************************
3030  GOSUB 53000:SPOLD=SP
3040  XCUR=CURX:YCUR=CURY+1
3050  CURSOR 0,0:PRINT "CHANGE REGISTER :              spacebar to RETURN";
3060  CURSOR 18,0
3070  G=GETC:IF G=0 THEN 3070:IF G<>#D THEN PRINT CHR$(G);
3080  IF CHR$(G)=" " THEN 3230
3090  FOR I=0 TO LEN(REG$)-1
3100  IF G=ASC(MID$(REG$,I,1)) THEN 3120
3110  NEXT:SOUND 1 0 15 0 FREQ(400):WAIT TIME 5:SOUND OFF :GOTO 3050
3120  CURSOR 2+I*5,YCUR:IF I=9 THEN CURSOR CURX+2,CURY
3130  LNNUM=2:IF I>7 THEN LNNUM=4
3140  HEXNUM=S(I):GOSUB 52000:REM INPUT A HEXNUMBER
3150  NUM$=HEX$(HEXNUM)
3160  FOR J=1 TO LNNUM:IF LEN(NUM$)<>LNNUM THEN NUM$="0"+NUM$:NEXT
3170  PRINT NUM$;:IF I=8 THEN GOSUB 3800:IF CHR$(G)<>"Y" THEN 3120
3180  S(I)=HEXNUM:IF I<>9 THEN 3050
3190  IF S(9)<#E000 OR S(9)>#EFFF THEN 3050
3200  XCUR=CURX:CURSOR 0,0:PRINT SPC(60);:CURSOR 0,0:PRINT "ROMBANK : ";
3210  HEXNUM=MLPBANK SHR 6:CURSOR XCUR,YCUR:PRINT " (";HEX$(HEXNUM);")";:CU
RSOR CURX-2,CURY:LNNUM=1:GOSUB 52000
3220  IF HEXNUM>3 THEN 3200:MLPBANK=(MLPBANK IAND #3F) IOR (HEXNUM SHL 6):G
OSUB 54000:GOTO 3050
3230  A=S(0):F=S(1):B=S(2):C=S(3):D=S(4):E=S(5)
3240  H=S(6):L=S(7):SP=S(8):IF PC<>S(9) THEN PC=S(9):PCNEXT=PC
3250  IF SP<>SPOLD THEN TOPOFSTACK=SP:GOSUB 57000
3260  CURSOR 0,0:PRINT SPC(60);:CURSOR 0,YCUR-1:RETURN
3270  RETURN
3800  LINE=0
3810  IF (HEXNUM>=#C000 AND HEXNUM<#F802) OR (HEXNUM>#F900) THEN MESSAGE$="
IGNORED (out of ram)":ANSW$=" ":GOSUB 56000:RETURN
3820  IF HEXNUM>FFB AND HEXNUM<#C000 THEN MESSAGE$="IGNORED (screen ram)":A
NSW$=" ":GOSUB 56000:RETURN
3830  IF HEXNUM>=HEAP AND HEXNUM<=STBUSE THEN MESSAGE$="IGNORED (destroying
 BASIC)":ANSW$=" ":GOSUB 56000:RETURN
3840  IF HEXNUM<SYSRAM THEN MESSAGE$="IGNORED (destroying system ram)":ANSW
$=" ":GOSUB 56000:RETURN
3850  MESSAGE$="ARE YOU SHURE [Y/N]?":ANSW$="YN":GOSUB 56000:RETURN
```

```
4000  PRINT CHR$(12)
4010  CURSOR 10,20:PRINT "1-BREAKPOINTS at EI instruction also"
4020  CURSOR 10,19:PRINT "  (for ROM,real time)"
4030  CURSOR 10,18:PRINT "2-BREAKPOINTS specified by user"
4040  CURSOR 10,17:PRINT "  (only RAM,real time)"
4050  CURSOR 10,16:PRINT "3-TRACE (virtual time)"
4060  CURSOR 10,11:PRINT "spacebar to RETURN"
4070  LINE=22-2*MDE:CURSOR 9,LINE:PRINT CHR$(9);:CURSOR 0,0:PRINT "MODE :
";
4080  CURSOR 7,0:G=GETC:IF G=0 THEN 4080:IF G<>#D THEN PRINT CHR$(G);
4090  FOR I=1 TO LEN(MDE$)
4100  IF G=ASC(MID$(MDE$,I-1,1)) THEN 4120
4110  NEXT:SOUND 1 0 15 0 FREQ(400):WAIT TIME 10:SOUND OFF :GOTO 4070
4120  CURSOR 9,LINE:PRINT " ";:ON I GOTO 4130,4140,4150,4160
4130  MDE=1:MDESET=1:GOSUB 59000:GOTO 4070
4140  MDE=2:MDESET=2:GOSUB 59000:GOTO 4070
4150  MDE=3:MDESET=3:GOSUB 59000:GOTO 4070
4160  PRINT CHR$(12):RETURN
5000  REM ***************************
5010  REM * DISPLAY/EDIT BREAKPOINTS *
5020  REM ***************************
5030  PRINT CHR$(12):I=0:LINE=1
5040  REF1=BRPTHIGHPTR:REF2=BRPTLOWPTR:REF3=CODEPTR:MAX=BRPTNBR
5050  CURSOR 0,0:PRINT "TYPE ";CHR$(9);",";CHR$(140);",";CHR$(136);",";CHR$
(140);",[C]hange,[I]nsert,[D]elete,spacebar to RETURN";
5060  GOSUB 5900
5070  LNNUM=4
5080  G=GETC:IF G=0 THEN 5080
5090  FOR J=1 TO LEN(CHANGE$)
5100  IF G<>ASC(MID$(CHANGE$,J-1,1)) THEN NEXT J:GOTO 5080
5110  ON J GOTO 5120,5140,5150,5170,5250,5180,5360,5440
5120  IF I>=MAX THEN 5080:I=I+1:IF CURX=52 THEN CURSOR 2,CURY-1
5130  CURSOR CURX+5,CURY:GOTO 5080
5140  IF I>=MAX-9 THEN 5080:I=I+10:CURSOR CURX,CURY-1:GOTO 5080
5150  IF I<1 THEN 5080:I=I-1:IF CURX=7 THEN CURSOR 57,CURY+1
5160  CURSOR CURX-5,CURY:GOTO 5080
5170  IF I<10 THEN 5080:I=I-10:CURSOR CURX,CURY+1:GOTO 5080
5180  REM ***INSERT***
5190  IF MAX>=LEN(CODE$)-2 THEN SOUND 1 0 15 0 FREQ(400):WAIT TIME 5:SOUND
OFF :GOTO 5080
5200  PARAM=((((REF1+I) SHL 8)+1) SHL 8)+((MAX-I+1) MOD 256):CALLM MOVE,PAR
AM
5210  PARAM=((((REF2+I) SHL 8)+1) SHL 8)+((MAX-I+1) MOD 256):CALLM MOVE,PAR
AM
5220  PARAM=((((REF3+I) SHL 8)+1) SHL 8)+((MAX-I+1) MOD 256):CALLM MOVE,PAR
AM
5230  POKE REF1+I,0:POKE REF2+I,0
5240  MAX=MAX+1:GOSUB 5900
5250  REM ***CHANGE***
5260  IF I=MAX THEN SOUND 1 0 15 0 FREQ(400.0):WAIT TIME 5:SOUND OFF :GOTO
5080
5270  HEXNUM=(PEEK(REF1+I) SHL 8)+PEEK(REF2+I):IF HEXNUM<>0 THEN POKE HEXNU
M,PEEK(REF3+I)
5280  GOSUB 52000:IF HEXNUM>=#C000 AND HEXNUM<#F800 OR HEXNUM>=#F900 THEN H
EXNUM=0:SOUND 1 0 15 0 FREQ(400.0):WAIT TIME 5:SOUND OFF :GOTO 5270
5290  IF HEXNUM<SYSRAM AND HEXNUM<>0 THEN MESSAGE$="IGNORED (destroying sys
tem ram)":ANSW$=" ":GOSUB 56000:GOTO 5270
5300  IF HEXNUM>=HEAP AND HEXNUM<STBUSE THEN MESSAGE$="IGNORED (destroying
BASIC)":ANSW$=" ":GOSUB 56000:GOTO 5270
5310  IF HEXNUM>FFB AND HEXNUM<#C000 THEN MESSAGE$="IGNORED (screen ram)":A
NSW$=" ":GOSUB 56000:GOTO 5270
5320  PNTR=HEXNUM:BRPTNBR=MAX:GOSUB 54000:IF BRPTFLAG=1 THEN MESSAGE$="IGNO
RED (already exists)":ANSW$=" ":GOSUB 56000:GOTO 5270
5330  POKE REF2+I,HEXNUM MOD 256:POKE REF1+I,HEXNUM SHR 8:POKE REF3+I,PEEK(
HEXNUM):IF HEXNUM<>0 THEN POKE HEXNUM,#C7
5340  NUM$=HEX$(HEXNUM):FOR II=1 TO LNNUM:IF LNNUM<>LEN(NUM$) THEN NUM$="0"
+NUM$:NEXT
5350  PRINT NUM$;:CURSOR CURX-4,CURY:GOTO 5080
5360  REM ***DELETE***
5370  IF I=MAX THEN SOUND 1 0 15 0 FREQ(400.0):WAIT TIME 5:SOUND OFF :GOTO
5080
5380  MESSAGE$="ARE YOU SURE [Y/N]?":ANSW$="YN":GOSUB 56000:IF CHR$(G)="N"
THEN 5080
5390  HEXNUM=(PEEK(REF1+I) SHL 8)+PEEK(REF2+I):IF HEXNUM<>0 THEN POKE HEXNU
M,PEEK(REF3+I)
5400  PARAM=((((REF1+I+1) SHL 8)+#FF) SHL 8)+((MAX-I+1) MOD 256):CALLM MOVE
,PARAM
5410  PARAM=((((REF2+I+1) SHL 8)+#FF) SHL 8)+((MAX-I+1) MOD 256):CALLM MOVE
,PARAM
5420  PARAM=((((REF3+I+1) SHL 8)+#FF) SHL 8)+((MAX-I+1) MOD 256):CALLM MOVE
,PARAM
5430  MAX=MAX-1:GOSUB 5900:GOTO 5080
5440  PRINT CHR$(12):BRPTNBR=MAX:RETURN
5900  X=7+5*(I MOD 10):Y=22-I/10:CURSOR X,Y:FOR II=I TO MAX STEP 1
5910  IF II MOD 10=0 THEN CURSOR 0,CURY:PRINT II+1.0;TAB(7);
5920  IF II=MAX THEN WORK$="    ":GOTO 5950
5930  WORK$=HEX$((PEEK(REF1+II) SHL 8)+PEEK(REF2+II))
5940  FOR J=1 TO 4:IF LEN(WORK$)<>4 THEN WORK$="0"+WORK$:NEXT J
5950  PRINT WORK$;" ";:IF II MOD 10.0=9.0 THEN PRINT
5960  NEXT II:CURSOR X,Y:RETURN
6000  REM **********************
6010  REM * DISPLAY/EDIT STACK *
6020  REM **********************
6030  PRINT CHR$(12):I=0:LINE=1
6040  REF=MLPSTACKPTR+11+TOPOFSTACK-SP:MAX=TOPOFSTACK-SP
6050  CURSOR 0,0:PRINT "TYPE ";CHR$(9);",";CHR$(140);",";CHR$(136);",";CHR$
(140);",[C]hange,[I]nsert,[D]elete,spacebar to RETURN";
6060  GOSUB 6900
6070  LNNUM=4
6080  G=GETC:IF G=0 THEN 6080
6090  FOR J=1 TO LEN(CHANGE$)
6100  IF G<>ASC(MID$(CHANGE$,J-1,1)) THEN NEXT J:GOTO 6080
6110  ON J GOTO 6120,6140,6150,6170,6230,6180,6290,6350
6120  IF I>=MAX THEN 6080:I=I+2:IF CURX=52 THEN CURSOR 2,CURY-1
6130  CURSOR CURX+5,CURY:GOTO 6080
6140  IF I>=MAX-18 THEN 6080:I=I+20:CURSOR CURX,CURY-1:GOTO 6080
6150  IF I<2 THEN 6080:I=I-2:IF CURX=7 THEN CURSOR 57,CURY+1
6160  CURSOR CURX-5,CURY:GOTO 6080
6170  IF I<20 THEN 6080:I=I-20:CURSOR CURX,CURY+1:GOTO 6080
6180  REM ***INSERT***
6190  IF MAX>=242 THEN SOUND 1 0 15 0 FREQ(400):WAIT TIME 5:SOUND OFF :GOTO
 6080
6200  MAX=MAX+2:SP=SP-2
6210  PARAM=((((REF-I+1) SHL 8)+2) SHL 8)+((I+1) MOD 256):CALLM MOVE,PARAM
6220  REF=REF+2:POKE REF-I-1,0:POKE REF-I,0:GOSUB 6900
6230  REM ***CHANGE***
6240  IF I=MAX THEN SOUND 1 0 15 0 FREQ(400.0):WAIT TIME 5:SOUND OFF :GOTO
6080
6250  HEXNUM=(PEEK(REF-I) SHL 8)+PEEK(REF-I-1)
6260  GOSUB 52000:POKE REF-I-1,HEXNUM MOD 256:POKE REF-I,HEXNUM SHR 8
6270  NUM$=HEX$(HEXNUM):FOR II=1 TO LNNUM:IF LNNUM<>LEN(NUM$) THEN NUM$="0"
+NUM$:NEXT
6280  PRINT NUM$;:CURSOR CURX-4,CURY:GOTO 6080
6290  REM ***DELETE***
6300  IF I=MAX THEN SOUND 1 0 15 0 FREQ(400.0):WAIT TIME 5:SOUND OFF :GOTO
6080
6310  MESSAGE$="ARE YOU SURE [Y/N]?":ANSW$="YN":GOSUB 56000:IF CHR$(G)="N"
THEN 6080
6320  MAX=MAX-2:SP=SP+2:PARAM=((((REF-I+1) SHL 8)+#FE) SHL 8)+((I+1) MOD 25
6)
6330  CALLM MOVE,PARAM
6340  REF=REF-2:GOSUB 6900:GOTO 6080
```

```
6900   X=7+5*((I MOD 20)/2):Y=22-I/20:CURSOR X,Y:FOR II=I TO MAX STEP 2
6910   IF II MOD 20=0 THEN CURSOR 0,CURY:PRINT HEX$(TOPOFSTACK-II);TAB(7);
6920   IF II=MAX THEN WORK$="    ":GOTO 6950
6930   WORK$=HEX$((PEEK(REF-II) SHL 8)+PEEK(REF-II-1))
6940   FOR J=1 TO 4:IF LEN(WORK$)<>4 THEN WORK$="0"+WORK$:NEXT J
6950   PRINT WORK$;" ";:IF II MOD 20.0=18.0 THEN PRINT
6960   NEXT II:CURSOR X,Y:RETURN
7000   REM ***********
7010   REM * OPTIONS *
7020   REM ***********
7030   X=CURX:Y=CURY
7040   CURSOR 0,0:PRINT "LIST MNEMONICS ? ";:GOSUB 7900:LSTMNEMFLAG=ANSWER
7050   CURSOR 0,0:PRINT "LIST REGISTERS ? ";:GOSUB 7900:LSTREGFLAG=ANSWER
7060   CURSOR 0,0:PRINT "TRACE RST ? ";:GOSUB 7900:TRACERSTFLAG=(ANSWER+1) M
OD 2
7070   CURSOR 0,0:PRINT "DO YOU WANT TO DISABLE INTERUPTS ? ";:GOSUB 7900:PO
KE POK(10)+13,0:IF ANSWER=0 THEN POKE POK(10)+13,#C4
7080   CURSOR X,Y:RETURN
7900   G=GETC:IF G=0 THEN 7900:IF G<>#D THEN PRINT CHR$(G);
7910   IF CHR$(G)="Y" THEN ANSWER=1:GOTO 7940
7920   IF CHR$(G)<>"N" THEN CURSOR CURX-1,CURY:GOTO 7900
7930   ANSWER=0
7940   WAIT TIME 10:CURSOR 0,0:PRINT SPC(60);:RETURN
8000   REM ***************
8010   REM * DISASSEMBLE *
8020   REM ***************
8030   X=CURX:Y=CURY:BANKOLD=MLPBANK:CURSOR 0,0:PRINT "DISASSEMBLE FROM 0000
";:LNNUM=4
8040   CURSOR 17,0:HEXNUM=0:GOSUB 52000:ADDRES=HEXNUM
8050   IF HEXNUM<#E000 OR HEXNUM>=#F000 THEN 8080
8060   HEXNUM=MLPBANK SHR 6:CURSOR 21,0:PRINT " (";HEX$(HEXNUM);")";:CURSOR
CURX-2,CURY:LNNUM=1:GOSUB 52000
8070   IF HEXNUM>3 THEN 8060:MLPBANK=(MLPBANK IAND #3F) IOR (HEXNUM SHL 6)
8080   CURSOR X,Y
8090   PNTR=ADDRES:GOSUB 54000:INSTRUCTION=BYTE
8100   LENGTH=1:IF BYTE<>#FF THEN LENGTH=ASC(MID$(LENGTH$,BYTE,1))
8110   POINTER=ADDRES+1:CALLM GETBYTE,POINTER:OPERANDLOW=BYTE
8120   POINTER=POINTER+1:CALLM GETBYTE,POINTER:OPERANDHIGH=BYTE
8130   GOSUB 55000:IF INSTRUCTION=#CF OR INSTRUCTION=#E7 OR INSTRUCTION=#EF
THEN LENGTH=2:GOSUB 58020
8140   ADDRES=ADDRES+LENGTH
8150   G=GETC:IF CHR$(G)<>" " THEN 8090:MESSAGE$="CONTINUE [Y/N]?":ANSW$="YN
":LINE=0:GOSUB 56000:IF CHR$(G)="Y" THEN 8090
8160   MLPBANK=BANKOLD:RETURN
50000 REM *********************************
50010 REM * VARIABLES INTO MLPSTACK$       *
50020 REM * FORMAT:SP/DE/BC/AF/HL/PC/rest *
50030 REM *********************************
50040 WORK=VARPTR(MLPSTACK$)
50050 MLPSTACKPTR=(PEEK(WORK+1) SHL 8)+PEEK(WORK)+1
50060 POKE MLPSTACKPTR,(SP-12) MOD 256:POKE MLPSTACKPTR+1,(SP-12) SHR 8
50070 POKE MLPSTACKPTR+3,D:POKE MLPSTACKPTR+2,E
50080 POKE MLPSTACKPTR+5,B:POKE MLPSTACKPTR+4,C
50090 POKE MLPSTACKPTR+7,A:POKE MLPSTACKPTR+6,F
50100 POKE MLPSTACKPTR+9,H:POKE MLPSTACKPTR+8,L
50110 POKE MLPSTACKPTR+10,PC MOD 256:POKE MLPSTACKPTR+11,PC SHR 8
50120 RETURN
51000 REM *********************************
51010 REM * REGISTERS TO VARIABLES         *
51020 REM * FORMAT:SP/DE/BC/AF/HL/PC/rest *
51030 REM *********************************
51040 WORK=VARPTR(MLPSTACK$)
51050 MLPSTACKPTR=(PEEK(WORK+1) SHL 8)+PEEK(WORK)+1
51060 SP=(PEEK(MLPSTACKPTR+1) SHL 8)+PEEK(MLPSTACKPTR)+12
51070 D=PEEK(MLPSTACKPTR+3):E=PEEK(MLPSTACKPTR+2)
51080 B=PEEK(MLPSTACKPTR+5):C=PEEK(MLPSTACKPTR+4)
51090 A=PEEK(MLPSTACKPTR+7):F=PEEK(MLPSTACKPTR+6)
51100 H=PEEK(MLPSTACKPTR+9):L=PEEK(MLPSTACKPTR+8)
51110 PC=(PEEK(MLPSTACKPTR+11) SHL 8)+PEEK(MLPSTACKPTR+10)
51120 RETURN
52000 REM ******************
52010 REM * INPUT A HEXNUM *
52020 REM ******************
52030 XPOS=CURX:NUM$=HEX$(HEXNUM):FOR II=1 TO LNNUM:IF LEN(NUM$)=LNNUM THEN
 52040:NUM$="0"+NUM$:NEXT
52040 PRINT NUM$;:CURSOR XPOS,CURY
52050 NUM$="#0"
52060 G=GETC:IF G=0 THEN 52060
52070 IF G=#12 THEN CURSOR XPOS,CURY:RETURN
52080 IF G=#D THEN 52150
52090 IF G=8 THEN IF CURX=XPOS GOTO 52050:PRINT CHR$(G);:NUM$=LEFT$(NUM$,LE
N(NUM$)-1):GOTO 52060
52100 IF CURX=XPOS+LNNUM THEN 52140
52110 FOR II=1 TO LEN(HEXNUM$)
52120 IF G=ASC(MID$(HEXNUM$,II-1,1)) THEN PRINT CHR$(G);:NUM$=NUM$+CHR$(G):
GOTO 52060
52130 NEXT
52140 SOUND 1 0 15 0 FREQ(400):WAIT TIME 5:SOUND OFF :GOTO 52060
52150 STP=VARPTR(NUM$)
52160 POKE #291,PEEK(STP):POKE #292,PEEK(STP+1):POKE #123,0
52170 READ HEXNUM:CURSOR XPOS,CURY:RETURN
53000 REM **********************
53010 REM * DISPLAY REGISTERS *
53020 REM **********************
53030 S(0)=A:S(1)=F:S(2)=B:S(3)=C:S(4)=D:S(5)=E
53040 S(6)=H:S(7)=L:S(8)=SP:S(9)=PC
53050 FOR I=0 TO LEN(REG$)-1:PRINT MID$(REG$,I,1);"=";
53060 NUM$=HEX$(S(I))
53070 LNNUM=2:IF I>7 THEN LNNUM=4
53080 FOR J=1 TO LNNUM
53090 IF LEN(NUM$)<>LNNUM THEN NUM$="0"+NUM$:NEXT
53100 PRINT NUM$;" ";:NEXT I:IF S(9)>=#E000 AND S(9)<#F000 THEN PRINT "(";H
EX$(MLPBANK SHR 6);")";
53110 PRINT :RETURN
54000 REM **********************
54010 REM * TEST IF BREAKPOINT *
54020 REM **********************
54030 BRPTFLAG=0:CALLM GETBYTE,PNTR:IF BYTE<>#C7 THEN RETURN
54040 IF BRPTNBR=0 THEN RETURN
54050 J=PNTR SHR 8
54060 FOR II=1 TO BRPTNBR:WORK=ASC(MID$(BRPTHIGH$,II-1,1))
54070 IF J-WORK>1 THEN 54100
54080 BRPT=(WORK SHL 8)+ASC(MID$(BRPTLOW$,II-1,1))
54090 IF BRPT=PNTR THEN 54110
54100 NEXT II:RETURN
54110 BRPTFLAG=1:CODEPTR1=CODEPTR+II-1:CALLM GETBYTE,CODEPTR1:RETURN
55000 REM *********************
55010 REM * DISPLAY MNEMONICS *
55020 REM *********************
55030 WORK$=HEX$(POINTER-2):FOR II=1 TO 3:IF LEN(WORK$)=4 THEN 55040:WORK$=
"0"+WORK$:NEXT
55040 PRINT WORK$;TAB(8);MNEM$(INSTRUCTION);TAB(18);:IF INSTRUCTION=#FF THE
N 55100:WORK$=MID$(REG2$,INSTRUCTION,1)
55050 IF WORK$<>" " THEN PRINT WORK$;:GOTO 55100
55060 IF LENGTH=1 THEN 55100
55070 IF LENGTH=2 THEN WORK$=HEX$(OPERANDLOW):IF LEN(WORK$)=2 THEN 55090:WO
RK$="0"+WORK$:GOTO 55090
55080 WORK$=HEX$((OPERANDHIGH SHL 8)+OPERANDLOW):FOR II=1 TO 3:IF LEN(WORK$
)=4 THEN 55090:WORK$="0"+WORK$:NEXT
55090 PRINT WORK$;"H";
55100 PRINT :RETURN
```

```
56010 REM * PRINT MESSAGE *
56020 REM ******************
56030 X=CURX:Y=CURY:CURSOR 0,LINE:PRINT SPC(60);:CURSOR 0,LINE:JJ=(PEEK(#79
) SHL 8)+PEEK(#78)-1:POKE JJ,#C0+((PEEK(#7D) IXOR #F) MOD #10):COLOLD=PEEK(
#7C) IOR #40
56040 PRINT MESSAGE$;" ";:IF LINE<>0 THEN POKE JJ-#86,COLOLD
56050 G=GETC:IF G=0 THEN 56050:FOR III=1 TO LEN(ANSW$):IF G=ASC(MID$(ANSW$,
III-1,1)) THEN 56060:NEXT:GOTO 56050
56060 PRINT CHR$(G);:WAIT TIME 10:CURSOR 0,LINE:POKE JJ,COLOLD:PRINT SPC(60
);:CURSOR X,Y:RETURN
57000 REM ********************
57010 REM * NEW TOP OF STACK *
57020 REM ********************
57030 TOSHIGH=TOPOFSTACK/256:TOSLOW=TOPOFSTACK MOD 256
57040 POKE POK(3),TOSLOW:POKE POK(3)+1,TOSHIGH
57050 POKE POK(12),TOSLOW:POKE POK(12)+1,TOSHIGH
57060 RETURN
58000 REM ---LIST DATA IF RST 1,4 OR 5---
58010 IF LSTMNEMFLAG=0 THEN RETURN
58020 WORK$=HEX$(POINTER-1):FOR II=1 TO 3:IF LEN(WORK$)<>4 THEN WORK$="0"+W
ORK$:NEXT
58030 PRINT WORK$;TAB(8);"DATA";TAB(18);:WORK$=HEX$(OPERANDLOW):IF LEN(WORK
$)=1 THEN PRINT "0";
58040 PRINT WORK$;"H":RETURN
59000 REM ---CHANGE MODE---
59010 MOPEI=0:IF MDESET=3 THEN MOPEI=#FB
59020 IF MDESET=1 OR MDESET=3 THEN POKE POK(5)-1,#32:POKE POK(5),#F9:POKE P
OK(5)+1,#FF:RETURN
59030 IF MDESET=2 THEN FOR I=-1 TO 1:POKE POK(5)+I,0:NEXT:RETURN
60000 REM ********************
60010 REM * INITIALISATION *
60020 REM ********************
60030 CLEAR #1100:PRINT CHR$(12);"INITIALISATION/READING DATA":POKE #5F,#C4
:MLPSTACK$=     255):BASICSTACK$=SPC(255)
60040 MDE$=      ":DIM       "G$="AFBCDEHLSP":HEXNUM$="1234567890ABCDEF":MLP
BANK=PEEK(#40):XCUR=
60050 HEAP=(PEEK(#29C) SHL 8)+PEEK(#29B):STBUSE=(PEEK(#2A4) SHL 8)+PEEK(#2A
3):SYSRAM=#2EC:FFB=(PEEK(#85) SHL 8)+PEEK(#84)
60060 LSTFLAG=1:LSTMNEMFLAG=1:LSTREGFLAG=0:TRACERSTFLAG=0:DIFLAG=0
60070 CHANGE$=CHR$(19)+CHR$(17)+CHR$(18)+CHR$(16)+"CID "
60080 DIM A$(0),MNEM$(255):LOADA A$ "ROUTINE$":ROUTINE$=A$(0):LOADA A$ "MOV
E$":MOVE$=A$(0):LOADA A$ "LENGTH$":LENGTH$=A$(0)
60090 LOADA MN"   MNE  A":LOADA A$ "REG2$":REG2$=A$(0):DIM A$(0)
60100 BRPTMAX=100:BRPTHIGH$=SPC(BRPTMAX):BRPTLOW$=SPC(BRPTMAX):CODE$=SPC(BR
PTMAX):BRPTNBR=0
60110 BRPTHIGHPTR=((PEEK(VARPTR(BRPTHIGH$)+1)) SHL 8)+PEEK(VARPTR(BRPTHIGH$
))+1
60120 BRPTLOWPTR=((PEEK(VARPTR(BRPTLOW$)+1)) SHL 8)+PEEK(VARPTR(BRPTLOW$))+
1
60130 CODEPTR=((PEEK(VARPTR(CODE$)+1)) SHL 8)+PEEK(VARPTR(CODE$))+1
61000 REM ****************************
61010 REM * INITIALISATION ROUTINES$ *
61020 REM ****************************
61030 DIM POK(16):I=0:ROUTINEPTR=(PEEK(VARPTR(ROUTINE$)+1) SHL 8)+PEEK(VARP
TR(ROUTINE$))+1
61040 MLPSTACKPTR=(PEEK(VARPTR(MLPSTACK$)+1) SHL 8)+PEEK(VARPTR(MLPSTACK$))
+1
61050 BASICSTACKPTR=(PEEK(VARPTR(BASICSTACK$)+1) SHL 8)+PEEK(VARPTR(BASICST
ACK$))+1
61060 RST0FLAG=1:RST0FLAGPTR=VARPTR(RST0FLAG)+3
61070 TOPOFSTACK=#F900:SP=TOPOFSTACK
61080 MOVE=(PEEK(VARPTR(MOVE$)+1) SHL 8)+PEEK(VARPTR(MOVE$))+1
61090 GETBYTE=MOVE+#25:BYTE=0
61100 FOR II=0 TO LEN(MOVE$):POINTER=MOVE+II
61110 IF PEEK(POINTER)=0 AND PEEK(POINTER+1)=0 THEN POK(I)=POINTER:I=I+1:II
61120 NEXT II
61130 MLPBANK=PEEK(40) IAND #3F:MLPBANKPTR=VARPTR(MLPBANK)+3:BANKHIGH=MLPBA
NKPTR SHR 8:BANKLOW=MLPBANKPTR MOD 256
61140 GBYTE=VARPTR(BYTE)+3:POKE POK(1),GBYTE MOD 256:POKE POK(1)+1,GBYTE/25
6
61150 POKE POK(0),BANKLOW:POKE POK(0)+1,BANKHIGH:I=0
61160 FOR II=0 TO LEN(ROUTINE$):POINTER=ROUTINEPTR+II
61170 IF PEEK(POINTER)=0 AND PEEK(POINTER+1)=0 THEN POK(I)=POINTER:I=I+1:II
=II+1
61180 NEXT II
61190 RST0=POK(8.0)+3.0
61200 POKE #62,RST0 MOD 256:POKE #63,RST0/256
61210 MLPHIGH=MLPSTACKPTR/256:MLPLOW=MLPSTACKPTR MOD 256
61220 BASHIGH=BASICSTACKPTR/256:BASLOW=BASICSTACKPTR MOD 256
61230 POKE POK(0),BASLOW:POKE POK(0)+1,BASHIGH
61240 POKE POK(1),MLPLOW:POKE POK(1)+1,MLPHIGH
61250 POKE POK(2),MLPLOW:POKE POK(2)+1,MLPHIGH
61260 POKE POK(4),BANKLOW:POKE POK(4)+1,BANKHIGH
61270 POKE POK(9.0),RST0FLAGPTR MOD 256:POKE POK(9.0)+1.0,RST0FLAGPTR SHR 8
61280 POKE POK(10),(POK(10)+5) MOD 256:POKE POK(10)+1,(POK(10)+5)/256
61290 POKE POK(11),BANKLOW:POKE POK(11)+1,BANKHIGH
61300 POKE POK(13),MLPLOW:POKE POK(13)+1,MLPHIGH
61310 POKE POK(14),BASLOW:POKE POK(14)+1,BASHIGH
61320 POKE POK(15),BASLOW:POKE POK(15)+1,BASHIGH
61330 POKE POK(10)+13,#C4
61340 GOSUB 57000:MDE=3:MDESET=3:GOSUB 59000
65000 PRINT CHR$(12):GOTO 10
```

```
900   REM FGT GOTISH TESTPROGRAMMA P.PADMOS
910   PRINT CHR$(12):MODE 6:COLORG 7 7 15 15:COLORT 8 0 0 0
915   CC=15:X1=1:Y1=YMAX-20:DF=0:ID=23:SP=9
920   REM .    1                                  2
            3
921   PF=0:T$="D it testprogram m a drukt F .G .T . program m a's af op een
 printer m et ":GOSUB 1000
922   PF=1:T$="G R A F T A X .                    A ls het schera  is vol
geschreven,    ":GOSUB 1000
923   PF=1:T$="m oet m en naar het M L P  op F 0 0, w aarna m en het schera
  opnieuw  kan":GOSUB 1000
924   PF=1:T$="volschrijven.                      D e w aarde I D  kom t
het best uit, ":GOSUB 1000
925   PF=1:T$="als deze op (2 3 ) w ordt gehouden.  I .v.m . de breedte van
 de cijfers    ":GOSUB 1000
926   PF=1:T$="hoofdletters en kleine letters (m )  ":GOSUB 1000:GOSUB 2000
927   PF=0:T$="en (w ) kan S P  niet kleiner zijn   dan (9 )!!!, en een spa
tie erachter. ":GOSUB 1000
931   PF=1:T$="D e w aarde D F  m oet op (0 ) w ord-en gehouden, w ant een
grotere        ":GOSUB 1000
932   PF=1:T$="w aarde m aakt geen m ooie letters encijfers.
              ":GOSUB 1000
933   PF=1:T$="H oe kan ik dat verhelpen ?
              ":GOSUB 1000
934   PF=1:T$="V riendelijke G roeten
              ":GOSUB 1000
935   PF=1:T$="          P .P ada os                ":GOSUB 1000:GOSUB 2000
936   PF=0:T$="          v.d.M eerw eg 1 0 A                3 2 3 3  B X
 O osvoorne    ":GOSUB 1000
937   PF=1:T$="          tel. 0 1 8 1 5 - 2 3 6 6   ":GOSUB 1000:GOSUB 2000
990   END
995   REM
1000  REM FGT SUB
1010  C=FC*#40+CC:F=FF*#80+PF*#40+ZF*#20+VF*#10+DF
1020  POKE #2F0,C:POKE #2F1,F
1030  POKE #2F2,X1 MOD 256:POKE #2F3,X1/256:POKE #2F4,Y1
1040  POKE #2F5,SP:POKE #2F6,ID
1050  CALLM #300,T$:RETURN
2000  REM SCR SUB
2010  REM CHANGE LINE 2020 IN [ 2020 RETURN ] for no hardcopy
2020  CALLM #F00:MODE 6:RETURN
```

GOTHISCH FGT

Dit testprogramma drukt FGT programma's af op een printer met GRAFTAX.
Als het scherm is volgeschreven, moet men naar het MLP op F00, waarna men het scherm opnieuw kan volschrijven.
De waarde ID komt het best uit, als deze op (23) wordt gehouden.
I.v.m. de breedte van de cijfers hoofdletters en kleine letters (m)

```
10    CLEAR 256:GOTO 1000:REM PAINT  / F.H. DRUIJFF 12/82
20    PRINT "Move blinking dot. If inside field to color press space ."
30    GOSUB 200:IF H<>32 GOTO 30
40    IF SCRN(X,Y)<>G THEN X=X-1:GOTO 40:REM FIND LEFT BORDER
50    POKE #328,W:POKE #30C,G:FOR I=0 TO 3:POKE #344+I*16,G:NEXT
60    X=X+1:POKE #335,Y:POKE #337,X MOD 256:POKE #338,X/256
70    CALLM #330:END
200   C=SCRN(X,Y):DOT X,Y 23:H=GETC:DOT X,Y C:IF H=0 GOTO 200
210   IF H<16 GOTO 270:IF H>23 GOTO 270:V=H/20*9+1
220   ON H MOD 4 GOTO 240,250,260
230   Y=Y+V:IF Y<YMAX GOTO 270:Y=YMAX:GOTO 270
240   Y=Y-V:IF Y>0 GOTO 270:Y=0:GOTO 270
250   X=X-V:IF X>0 GOTO 270:X=0:GOTO 270
260   X=X+V:IF X<XMAX GOTO 270:X=XMAX
270   RETURN
1000  G=10:W=5:COLORG 0 W G 15:MODE 4A:REM INITIALISATION
1010  INPUT "Vaste/Eigen tekening ";VE$:PRINT
1020  IF VE$="E" GOTO 1100:X=33:Y=33
1030  READ P,Q:IF P+Q=0 GOTO 20:DRAW P,Q X,Y 22:X=P:Y=Q:GOTO 1030
1100  GOSUB 200:IF H=13 GOTO 20
1110  IF H<>32 GOTO 1100:IF F=0 THEN DOT X,Y 22:F=1:P=X:Q=Y
1120  DRAW X,Y P,Q 22:P=X:Q=Y:GOTO 1100
5000  DATA 147,12,147,45,90,45,80,31,80,60,120,70
5010  DATA 85,70,75,60,66,45,45,55,33,33,0,0
```

PARAMETERS FOR PAINT I

```
POKE H328,W : W= paintcolor (color 5 in demo)
POKE H30C,G : G= border color (color 10 in demo)
POKE H335,Y : Y= vertical coordinate
POKE H337, X MOD 256 :  horizontal coordinate
POKE H338, X/256     :  horizontal coordinate
CALLM H330 : call PAINTroutine.
```

```
10    CLEAR 1000:REM PAINT DEMO / F.H. DRUIJFF 4/83
20    QW=#305:DIM A(6),B(6):FOR H!=0 TO 2*PI STEP PI/3.0
30    A(I)=83.0*COS(H!):B(I)=83.0*SIN(H!):I=I+1:NEXT
40    G=RND(15)+1:K=RND(15)+1:IF K=G GOTO 40:L=R      +1
50    IF L=G OR L=K GOTO 40:MODE 6:COLORG 0 G         1:N=170
60    FOR I=0 TO 5:FOR J=I TO 5:DRAW M+A(I),N-        (J),N+B(J) G:NEXT:NEX
T
70    READ X,Y:V=1-V:W=22+V:IF X=0 GOTO 10
80    POKE QW+5,X MOD 256:POKE QW+6,X SHR 8:POKE QW+7,Y
90    POKE QW+3,G:POKE QW+4,W:CALLM QW:GOTO 70
100   DATA 161,135,191,153,191,183,161,193,141,183
110   DATA 131,152,128,125,146,124,156,114,176,124
120   DATA 193,134,213,144,213,165,213,175,213,195
130   DATA 193,205,177,209,161,229,141,209,131,209
140   DATA 111,209,111,179,111,159,111,129,0,0
```

PARAMETERS FOR PAINT II - routine

```
POKE H30A, X MOD 256 : horizontal coordinate
POKE H30B, X SHR 8   : horizontal coordinate
POKE H30C, Y         : vertical coordinate
POKE H308, G         : G = border color
POKE H309, W         : W = paintcolor
```

```
2000  POKE #131,1
2010  MODE 0:PRINT CHR$(12):COLORT 9 14 9 9
2020  CURSOR 15,23:PRINT "C A S S E T T E L I S T  IX.I"
2030  POKE #8A,#CB:POKE #8B,#BC:POKE #BD50,#C8:POKE #BCCA,#D0
3000  CLEAR 10000:DIM A$(90),B$(50,1),TYP$(255):GOSUB 65000
3010  FOR A=0 TO 255:TYP$(A)="USR":NEXT
3020  READ FBYT$,FTYP$:IF FTYP$="END" THEN 4000
3030  TYP$(ASC(FBYT$))=FTYP$:GOTO 3020
4000  CURSOR 0,22:PRINT "PRINTER:"
4010  PRINT "[1] MX  80 III"
4020  PRINT "[2] MX  82 III"
4030  PRINT "[3] MX 100 III":CURSOR 8,22
4040  TYPE=GETC:IF TYPE<49 OR TYPE>51 THEN 4040:CURSOR 15,70-TYPE:PRINT "*"
5000  CURSOR 20,22:PRINT "PHOTOFORMAT:"
5010  CURSOR 20,21:PRINT "[1] 13*10 cm "
5020  CURSOR 20,20:PRINT "[2] 15*10 cm ":CURSOR 32,22
5030  FRMT=GETC:IF FRMT<>49 AND FRMT<>50 THEN 5030:CURSOR 33,70-FRMT:PRINT
"*"
5040  FRML=29:IF TYPE=50 THEN CDL=104:INL=28:BK$="+-------":BKL=6:GOTO 5060
5050  CDL=88:INL=23:BK$="+------":BKL=5
5060  FRMT=FRMT-49:CDL=CDL+CDL*2/13*FRMT
5070  CDH=CDL/2-1:TITLL=25+4*FRMT
6000  CURSOR 40,22:PRINT "driveno  ?";
6010  DCR=GETC:IF DCR<48 OR DCR>51 THEN 6010:PRINT CHR$(DCR)
6020  PRINT :DCR=DCR-48:POKE #11F,DCR
6030  GOSUB 40000
7000  CURSOR 40,21:INPUT "DCR casno";NO$:PRINT
7010  IF LEN(NO$)>4 THEN 7000
8000  CURSOR 40,20:INPUT "Date      ";D$:D$=SPC(CDH-LEN(D$)-8)+"update "+D$+
" "
8010  B$(47,0)=D$:B$(47,1)=D$
8020  CURSOR 20,19:INPUT "Title ";A$(0):PRINT
8030  IF LEN(A$(0))>TITLL THEN 8020
9000  CALLM #F000:REM REW
9010  FOR A=1 TO INL:LINE$=LINE$+"-":NEXT
9020  LINE$=BK$+"+"+LINE$+"+"+LINE$+"+"+LINE$+CHR$(8)+"+"
9030  FOR A=1 TO CDL:KADER$=KADER$+"-":NEXT:KADER$="+"+KADER$+"+"
9040  DIST$=SPC(INL)+CHR$(124)
9050  SS=2:S=3:NO=1:NR=1:SIDE=0
9060  CALLM #F003,LAST
9070  B$(0,1)=A$(0):B$(0,0)=A$(0):PRINT
9080  IF LEN(A$(0))>INL THEN A$(0)=LEFT$(A$(0),INL)
9090  B$(1,0)="DCR "+NO$+" A":B$(2,0)=""
9100  B$(1,1)="DCR "+NO$+" B":B$(2,1)=""
9110  A$(1)="*** DCR "+NO$+" ***"
9120  A$(2)="<<<<<<<<<< A >>>>>>>>>>"
9130  IF FRMT=1 THEN A$(2)="<<"+A$(2)+">>"
10000 REM ##################################################
10010 REM
10020 REM ---GET A NAME---
10030 CURSOR 0,1:TRY=0:PRINT
10040 LAST=0:CALLM START:IF LAST=0 THEN 10230
10050 REM ------------------------------LAST FILE SIDE 1
10060 IF PEEK(#B463)<>32 THEN TRY=TRY+1:IF TRY<5 THEN 10040
10070 SIDE=SIDE+1:NR=1:SS=2:IF SIDE>1 THEN RET=2:GOTO 10170
10080 PRINT "turn cassette,type space (Return= end)"
10090 GOSUB 60000:GOSUB 40100
10100 A=GETC:IF A=13 THEN RET=1:GOTO 10150
10110 IF A<>32 THEN 10100
10120 RET=2:CALLM #F000:REM REW
10130 GOTO 10030
10150 A$(S)="    NO CHECK DONE !!!!":B$(2,1)=A$(S)
10160 REM ------------------------------LAST FILE SIDE 2
10170 PKD=0:PRINT "Printing Mikro-format ( Y-N ) ?";
10180 IF RET=2 THEN POKE #FE03,#98:POKE #FE02,2:POKE #FE01,4*DCR+3:GOTO 101
```


SUPER DCR-LIST

```
10185 GOSUB 40000:REM REW
10186 GOSUB 60000
10187 A=GETC:IF A=#59 THEN PKD=1:PRINT "Y":GOTO 10190
10188 IF A=#4E THEN PRINT "N":GOTO 10190
10189 GOTO 10187
10190 PRINT "Printing Foto-format ( Y-N ) ?";:A=GETC:A=GETC:A=GETC
10191 GOSUB 60000
10192 A=GETC:IF A=#59 THEN PKD=PKD+10:PRINT "Y":GOTO 10195
10193 IF A=#4E THEN PRINT "N":GOTO 10195
10194 GOTO 10192
10195 PRINT "Tape space to start printing";
10196 GOSUB 60000:A=GETC:IF A<>32 THEN 10196
10200 PRINT :PRINT :GOTO 20000
10210 REM
10220 REM ---GET NUMBER---
10230 A$(S)=MID$(STR$(NO),1,LEN(STR$(NO))-3)+"-"
10240 IF LEN(A$(S))<3 THEN A$(S)="0"+A$(S)
10250 B$(SS,SIDE)=MID$(STR$(NR),1,LEN(STR$(NR))-3)+"-"
10260 IF LEN(B$(SS,SIDE))<3 THEN B$(SS,SIDE)="0"+B$(SS,SIDE)
10270 REM
10280 REM ---READ NAME---
10290 H$="":FOR A=#B3EB TO #B463 STEP 2:IF PEEK(A)=#20 THEN NEXT
10300 FOR B=#B463 TO A STEP -2
10310 PB=PEEK(B):PB=PB IAND #7F:IF PB<32 OR PB=127 THEN PB=46
10320 H$=H$+CHR$(PB):NEXT:T$=LEFT$(H$,1)
10330 HH$=RIGHT$(H$,LEN(H$)-1)
10340 HH$=T$+"-"+TYP$(ASC(T$))+"-"+HH$
10350 A$(S)=A$(S)+H$:B$(SS,SIDE)=B$(SS,SIDE)+HH$
10360 LNGTH=INL:IF S>2*FRML THEN LNGTH=INL-1
10370 IF LEN(A$(S))>LNGTH THEN A$(S+1)=SPC(3)+RIGHT$(A$(S),LEN(A$(S))-LNGTH
):A$(S)=LEFT$(A$(S),LNGTH):S=S+1:GOTO 10360
10380 IF LEN(B$(SS,SIDE))<CDH-1 THEN 10430
10390 B$(SS+1,SIDE)=SPC(9)+RIGHT$(B$(SS,SIDE),LEN(B$(SS,SIDE))-CDH+2)
10400 B$(SS,SIDE)=LEFT$(B$(SS,SIDE),CDH-2):SS=SS+1:GOTO 10380
10410 S=S+1
10420 REM ---NEXT FILE---
10430 CALLM #F000:REM SKIP1
10440 SS=SS+1:S=S+1:NO=NO+1:NR=NR+1
10450 IF S<73 THEN 10030:POKE #131,1
20000 REM ##################################################
20010 E$=CHR$(27):A$=CHR$(1):OFF$=CHR$(0)
20020 UNDERLON$=E$+"-"+A$:UNDLOFF$=E$+"-"+OFF$:CONDNSOFF$=CHR$(18)
20030 CONDNSON$=CHR$(15):LINSPC5$=E$+"A"+CHR$(5):LINSPC10$=E$+"A"+CHR$(10)
20040 SPRSCRON$=E$+"S"+A$:SUPSCROFF$=E$+"T"+A$
20050 H$=CHR$(124):DIST1$=RIGHT$(DIST$,LEN(DIST$)-1)
20060 BK1$=H$+SPC(LEN(BK$)-1):ENLARGON$=E$+"W"+A$:ENLARGOFF$=E$+"W"+OFF$
20070 REM ---PRINT HEADER---
20080 POKE #131,0
20085 IF PKD=0 OR PKD=10 THEN 30000
20090 PRINT CONDNSON$;LINSPC5$;SPRSCRON$;LINE$
20100 PRINT BK1$;H$;UNDERLON$;A$(0);UNDLOFF$;
20110 PRINT SPC(INL-LEN(A$(0)));H$;DIST$;DIST1$
20120 PRINT BK1$;H$;DIST$;DIST$;DIST1$
20130 REM
20140 REM --- PRINT TITLES---
20150 FOR A=1 TO FRML:IF A MOD 20<>5 THEN 20170
20160 PRINT H$;" ";NO$;SPC(BKL-LEN(NO$)-1);:GOTO 20200
20170 IF A MOD 20<>4 THEN 20190
20180 PRINT BK$;:GOTO 20200
20190 PRINT BK1$;
20200 PRINT H$;A$(A);SPC(INL-LEN(A$(A)));
20210 PRINT H$;A$(A+FRML);SPC(INL-LEN(A$(A+FRML)));
20220 PRINT H$;A$(A+2*FRML);SPC(INL-1-LEN(A$(A+2*FRML)));H$
20230 NEXT
20240 REM
```

```
2025:  _M --- END ---
30004 REM ------------------------------------PHOTOCARD
30005 IF PKD=0 THEN 30120
30010 PRINT SUPSCROFF$;LINSPC10$;CONDNSON$;KADER$
30015 IF PKD=1 THEN 30120
30020 FOR SIDE=0 TO 1
30030 PRINT H$;SPC(CDL);H$:PRINT CONDNSOFF$;
30040 PRINT H$;ENLARGON$;B$(0,SIDE)
30050 PRINT SPC(CDL#2/7-LEN(B$(1,SIDE)));B$(1,SIDE)
30060 PRINT ENLARGOFF$;CONDNSON$;
30070 PRINT H$;SPC(CDL);H$
30080 FOR A=2 TO 24:PRINT H$;SPC(2);B$(A,SIDE);
30090 PRINT SPC(CDH-LEN(B$(A,SIDE)));B$(A+23,SIDE);
30100 PRINT SPC(CDH-LEN(B$(A+23,SIDE)));H$
30110 NEXT:PRINT KADER$:NEXT SIDE
30120 CALLM #F006:END
40000 POKE #FE03,#98:POKE #FE02,1:POKE #FE01,4*DCR+3:REM REW
40010 POKE #3E,INT7 MOD 256:POKE #3F,INT7 SHR 8:POKE #3B,#3E:REM INT7
40020 RETURN
40100 A$(S)=""
40110 A$(S+1)="<<<<<<<<<< B >>>>>>>>>>"
40120 IF FRNT=1 THEN A$(S+1)="<<"+A$(S+1)+">>"
40130 S=S+2
40140 RETURN
60000 ENVELOPE 1 15,5;11,5;8,10;7,15;6,20;5,20;3,30;0
60010 ENVELOPE 0 15,3;3,3;
60020 FOR I=0 TO 2:SOUND 0 1 15 0 FREQ(1046):WAIT TIME 5
60030 SOUND 1 1 15 0 FREQ(1175):WAIT TIME 5:NEXT
60040 SOUND 2 1 15 0 FREQ(1760):WAIT TIME 30
60050 WAIT TIME 50:SOUND OFF :RETURN
65000 START=#B200:ADRES=START:INT7=START+#64
65010 POKE #2A5,0:POKE #2A6,#B2
65020 READ BYTE:IF BYTE<#100 THEN POKE ADRES,BYTE:ADRES=ADRES+1:GOTO 65020
65030 IF BYTE=#FFF THEN RETURN
65040 READ LBYTE,HBYTE:POKE ADRES,BYTE IAND #FF
65050 LBYTE=LBYTE+(START IAND #FF):CARRY=LBYTE SHR 8
65060 POKE ADRES+1,LBYTE IAND #FF:HBYTE=HBYTE+(START SHR 8)+CARRY
65070 POKE ADRES+2,HBYTE:ADRES=ADRES+3:GOTO 65020
65080 REM NEW LOOK#B200 /ENDCHECK INT7 #B264
65090 DATA #F5,#C5,#D5,#E5,#21,#00,#00,#01
65100 DATA #FF,#00,#CD,#5E,#DD,#1CD,#13,#00
65110 DATA #C3,#74,#F7,#F5,#5E,#23,#AF,#57
65120 DATA #CD,#48,#F3,#1C2,#18,#00,#F1,#1CD
65130 DATA #25,#00,#C3,#F8,#F1,#C5,#CD,#8F
65140 DATA #D9,#1CD,#2F,#00,#C3,#62,#F4,#CD
65150 DATA #B4,#F3,#FB,#CD,#BB,#F3,#F3,#1C2
65160 DATA #4F,#00,#CD,#79,#F3,#06,#0F,#CD
65170 DATA #94,#F4,#1C2,#32,#00,#06,#96,#CD
65180 DATA #94,#F4,#E6,#B0,#C2,#D5,#F3,#3E
65190 DATA #01,#32,#B0,#01,#CD,#D6,#F5,#2A
65200 DATA #20,#01,#36,#FF,#21,#08,#00,#39
65210 DATA #F9,#C3,#4D,#C1,#F5,#3A,#00,#FE
65220 DATA #E6,#22,#EE,#02,#C4,#1F,#F3,#F1
65230 DATA #C3,#A9,#D9,#FFF
65240 DATA 0,BAS,1,OBJ,2,ARR,#,DNA,&,SPL,$,PAS, ,END
```