A HAPPY 1985

from DAInamic TEAM

DAInamic software a program for e

☐	Games collection 1	400 Bfr	
☐	Games collection 2	400 Bfr	
☐	Games collection 3	400 Bfr	
☐	Games collection 4	800 Bfr	
☐	Games collection 5	400 Bfr	
☐	Games collection 6	750 Bfr	
☐	Games collection 7	750 Bfr	
☐	Games collection 8	750 Bfr	
☐	Games collection 9	750 Bfr	
☐	Games collection 10	750 Bfr	
☐	Games collection 11	750 Bfr	
☐	Games collection 12	750 Bfr	
☐	DNA assembly pack	1100 Bfr	
☐	Fast graph text	1000 Bfr	
☐	FGT applications	1000 Bfr	
☐	Toolkit 1	1000 Bfr	
☐	Toolkit 2	1000 Bfr	
☐	Toolkit 3	1000 Bfr	
☐	Toolkit 4	1000 Bfr	
☐	Toolkit 5	1000 Bfr	
☐	Primary Education 1	1000 Bfr	
☐	Math' fun 1	1000 Bfr	
☐	Secundary Education 1	1000 Bfr	
☐	Secundary Education 2	1000 Bfr	
☐	Mathematics 3	1000 Bfr	
☐	Bits & Bytes	750 Bfr	
☐	Mailing List	1000 Bfr	
☐	Graphic Tablet	1000 Bfr	
☐	Music collection 1	300 Bfr	
☐	Music collection		
☐	Music collection		
☐	DAI Tiny Pascal		
☐	English-German		
☐	DAI DEMO + Ba		
☐	Sargon Chess		
☐	Space Invaders		
☐	Tape 80-81		
☐	Newsletter 10		
☐	Newsletter 11-1		
☐	Newsletter 13-1		
☐	Centipede		
☐	Driver		
☐	Super Invader		
☐	DAI PANIC		
☐	MICRO'S-Onde		
☐	SPL Macro-ass		
☐	Taal-tape 1		
☐	Fysica 1		
☐	Familiebudget		
☐	Grafische Hulp		
☐	Acrobates		
☐	Character Gene		
☐	Fast Word Proc		
☐	PAC-MAN		
☐	DAYLAXIANS		
☐	PUZZLY		
☐	DUEL		
☐	C.L.I.O.		

FRACTALS

```
PAGE 01  --  FRACTALS SOUND

10      REM fractals with sound
15      MODE 0:PRINT CHR$(12):COLORT 0 5 0 0
20      INPUT "Geef A met 3<A<3.95";A!:PRINT
30      IF A!<3.0 THEN A!=3.0:IF A!>3.95 THEN A!=4.0
40      INPUT "DUUR VAN TONEN 0<TIJD<10";TIJD:PRINT
50      IF TIJD<0.0 THEN TIJD=0
60      IF TIJD>10 THEN TIJD=10
65      PRINT "START/STOP met spatiebalk"
70      G=GETC:IF G<>32 THEN GOTO 70
75      Z!=RND(1.0)

80      REM REPEAT UNTIL SPACE-BAR IS PRESSED
90      Z!=A!*Z!*(1.0-Z!)
100     SOUND 0 1 15 0 FREQ(1000.0*Z!)
105     WAIT TIME TIJD
110     G=GETC:IF G<>32 THEN GOTO 90
120     SOUND OFF
130     GOTO 15
```

```
PAGE 01  --  FRACTALS GRAPHIQUE

5       MODE 0:PRINT CHR$(12):COLORT 0 5 0 0
10      INPUT "Geef beginwaarde A met 1<A<4 bv 2.8 of 3.7";AINT!:PRINT
20      IF AINT!<1 OR AINT!>4 THEN GOTO 5
25      PRINT CHR$(12)
30      PRINT "Geef maximum van A met ";AINT!;:INPUT "<A<4 bv 3.99 of 3.9";
        AMAX!:PRINT
40      IF AMAX!<AINT! OR AMAX!>4.0 THEN GOTO 25
50      COLORG 0 5 10 15:MODE 6
70      SCALE!=(AMAX!-AINT!)/XMAX
80      Z!=RND(1.0)
90      FOR X=0 TO XMAX
100   1  A!=X*SCALE!+AINT!
110   1  IF A!<2.95 THEN CYCLE=2
120   1  IF A!>=2.95 THEN CYCLE=20
130   1  FOR Y=1 TO CYCLE:Z!=A!*Z!*(1.0-Z!):NEXT Y
140   1  FOR Y=1 TO CYCLE
150   2    Z!=A!*Z!*(1.0-Z!)
160   2    Z1=Z!*YMAX:DOT X,Z1 5
170   1  NEXT Y
180     NEXT X
200     END
```

FWP PATCHES 2

The advantage of the following patches is that the "K" and the "X" command can be executed now, whilst marker 01 is also present in buffer 1 (see last DAInamic, FWP TIP).

The following addresses, starting from 0959, have to be changed.

0959 **79 FE 07 C8 FE 06 C8 37 C9**

It is not necessary to change address 0966.

The above changes still leave the possibility of setting more than once the same marker, so you are more flexible, however multiple markers can introduce errors. For example in case you give a "K" command with 2 markers 07, be aware that all the text between marker 06 and the last marker 07 will be deleted !

If you don't want to have the markers more than once (without getting an error message), change the the addresses below.
The above mentioned feature is also incorporated in these patches.

0941 **0C 00**

0948 **0C 0C 00 00**

0951 **0C 0C 0C 00**

0959 **79 FE 05 C8 FE 06 C8 37 C9**

In case you prefer "start printing" always from the first line in the buffer, instead of the top of the current STEP, change the next 3 addresses.

073E **21 00 30** (**21 00 90** for buffer 2)

Note: If you ever experienced a stack-overflow condition during working with FWP it may help to change the content of address 1313 in 00.
This is a small bug in the program (1313 is not a joke !!)

Ger Gruiters 25-10-1984

HEAP ORGANISATIE

Numansdorp
29.05.84

Beste DAInamic vrienden.

Hier een klein programma, dat de vrije en gebruikte heapruimte bepaalt. De opzet was dit als subroutine in een groter programma te gebruiken om te zien hoeveel heapruimte er nog over was. Wij vinden het nog steeds een nadeel van de DAI dat de FRE functie alleen werkt op de totale programmaruimte en niet ook apart op de heap. Tijdens het uitproberen van dit programma kwamen wij tot een beter inzicht in de oranisatie van de heap. Tezamen met het artikel van Jan Boerrigter in DAInamic Nr. 7 Blz. 188 e.v. geeft dit een beter inzicht in de manier waarop de DAI met de heap omspringt. Dat dit niet altijd even economisch is zal blijken. Type het programma in zoals het is, geef RUN en RETURN en na het vraagteken een paar maal alleen RETURN, daarna bijv. een A en RETURN en een paar maal alleen RETURN. Hier uit blijkt, dat bij een INPUT de heap pas na de derde INPUT op de zelfde string wordt gereorganiseerd. Dit verbetert aanmerkelijk als we de string eerst leeg maken met A$="":INPUT A$, in regel 20. Verander het begin ook eens als volgt:

```
 5 CLEAR 1000
10 DIM A$(10)
20 FOR A=0 TO 10:INPUT A$(A):NEXT
```

en verander de GOTO op regel 2000 beurtelings in GOTO 10 en GOTO 20. Hopend dat U net als wij hierdoor een beter gebruik van de heap kunt maken volgt nu het programma:

```
*IMP INT
*   5 CLEAR #100
*  10 INPUT A$
*1000 PRINT
*1010 HEAP=PEEK(#29B)+PEEK(#29C)*256
*1020 HSIZE=PEEK(#29D)+PEEK(#29E)*256
*1030 HEND=HEAP+HSIZE
*1040 PP=PEEK(HEND-1)+PEEK(HEND-2)*256
*1050 PRINT "HEAP  ","HSIZE ","H-END ","ENDPTR"
*1060 PRINT HEAP,HSIZE,HEND,PP
*1070 PRINT " #";HEX$(HEAP)," #";HEX$(HSIZE)," #";
     HEX$(HEND)," #";HEX$(PP)
*1080 PRINT
*1090 FOR HP=HEAP TO HEND
*1100 PO= PEEK(HP+1)+PEEK(HP)*256
*1110 FH=(PO+PP) IAND #FFFF
*2000 IF FH=HEND-HP-5 THEN PRINT "BYTES LEFT: ";FH;
     "  USED: ";HSISE-FH;GOTO 10
*2010 NEXT:PRINT "NO POINTERS LEFT !!!":GOTO 5
*RUN
```

De DAI gebruikt soms wel de ruimte voor de pointers voor dat de melding OUT OF STRING SPACE wordt gegeven.
Met vriendelijke groeten:

Fredrik en Frits Chabot

```
002                     ORG   :300
003             VAR     EQU   :2EC
004             HEND    EQU   VAR+2
005             PP      EQU   HEND+2
006             HP      EQU   PP+2
007             FH      EQU   HP+2
008             *
009 0300 F3     START   DI
010 0301 C5             PUSH  B
011 0302 D5             PUSH  D
012 0303 E5             PUSH  H
013 0304 F5             PUSH  PSW
014 0305 3600           MVI   M,:0
015 0307 23             INX   H
016 0308 3600           MVI   M,:0
017 030A 23             INX   H          HET ADRES VAN DE VAR.
018 030B 22EC02         SHLD  VAR        OPSLAAN
019             * HEND BEREKENEN EN OPSLAAN VOOR LATER GEBRUIK
020 030E 2A9B02         LHLD  :29B
021 0311 EB             XCHG
022 0312 2A9D02         LHLD  :29D
023 0315 19             DAD   D
024 0316 22EE02         SHLD  HEND
025             * DE POINTER OPHALEN DIE AAN HET EIND V/D HEAP STAAT
026 0319 2B             DCX   H
027 031A 5E             MOV   E,M
028 031B 2B             DCX   H
029 031C 56             MOV   D,M
030 031D EB             XCHG
031 031E 22F002         SHLD  PP
032             * HP=PEEK(#29B ...      ( VOOR DE LOOP VAN START HEAP
033 0321 2A9B02         LHLD  :29B       TOT EIND HEAP )
034 0324 22F202         SHLD  HP
035             *DIT IS DE LOOP WAARIN GEKEKEN WORDT OF DE
036 0327 2AF202 FOR     LHLD  HP         AANGETROFFEN BYTE'S OVEREEN
037 032A 56             MOV   D,M        KOMEN MET HET ADRES WAAR ZE
038 032B 23             INX   H          STAAN
039 032C 5E             MOV   E,M        DE 2 BYTE'S IN  (DE)
040 032D 2AF002         LHLD  PP         (HL)=PP
041 0330 19             DAD   D          (HL)=(HL)+(DE)
042 0331 22F402         SHLD  FH         FH=(HL)
043 0334 EB             XCHG             (HL) <=> (DE)    [ (DE)=FH ]
044 0335 2AF202         LHLD  HP         (HL)=HP
045 0338 CD7903         CALL  SUB        (BC)=-(HL)
046 033B 2AEE02         LHLD  HEND       (HL)=HEND
047 033E 09             DAD   B          ;            [ (HL)=HEND-HP ]
048 033F 01FBFF         LXI   B,:0-:5
049 0342 09             DAD   B          ;             [ (HL)=(HL)-5 ]
050 0343 CD7903         CALL  SUB        (BC)=-(HL)
051 0346 EB             XCHG             (HL) <=> (DE)    [ (HL)=FH ]
052 0347 09             DAD   B          ;    [ (HL)=FH-(HEND-HP-5) ]
053 0348 CD8003         CALL  HL         (HL) 0 ?
054 034B CA6803         JZ    TVAR       JA  GOTO TVAR
055 034E 2AEE02 NEXT    LHLD  HEND       NEE HET VOLGENDE ADRES
056 0351 CD7903         CALL  SUB        PROBEREN
057 0354 2AF202         LHLD  HP
058 0357 23             INX   H          HP=HP+1
059 0358 22F202         SHLD  HP
060 035B 09             DAD   B
061 035C CD8003         CALL  HL         EIND HEAP AL BEREIKT ?
062 035F C22703         JNZ   FOR        NEE  NEXT
063 0362 21FFFF         LXI   H,:FFFF    JA DAN ONTBREKEN DE POINTERS
```



```
064 0365 22F402         SHLD  FH
065             *
066 0368 2AF402 TVAR    LHLD  FH
067 036B 23             INX   H
068 036C EB             XCHG
069 036D 2AEC02         LHLD  VAR
070 0370 72             MOV   M,D        FRE IN DE VAR ACHTER DE
071 0371 23             INX   H          CALLM
072 0372 73             MOV   M,E
073             *
074 0373 F1             POP   PSW        EN RETURN
075 0374 E1             POP   H
076 0375 D1             POP   D
077 0376 C1             POP   B
078 0377 FB             EI
079 0378 C9             RET
080             *
081             *
082 0379 7C     SUB     MOV   A,H        (BC)=-(HL)
083 037A 2F             CMA
084 037B 47             MOV   B,A
085 037C 7D             MOV   A,L
086 037D 2F             CMA
087 037E 4F             MOV   C,A
088 037F C9             RET
089             *
090 0380 7C     HL      MOV   A,H        (HL)=0 ?
091 0381 FE00           CPI   :0
092 0383 C0             RNZ
093 0384 7D             MOV   A,L
094 0385 FE00           CPI   :0
095 0387 C9             RET
096                 *
097 0388                END

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

FH      02F4   FOR     0327   HEND    02EE   HL      0380
HP      02F2   NEXT    034E   PP      02F0   START   0300
SUB     0379   TVAR    0368   VAR     02EC
```

```
]#P.
0300 F3 C5 D5 E5 F5 36 00 23 36 00 23 22 EC 02 2A 9B
0310 02 EB 2A 9D 02 19 22 EE 02 2B 5E 2B 56 EB 22 F0
0320 02 2A 9B 02 22 F2 02 2A F2 02 56 23 5E 2A F0 02
0330 19 22 F4 02 EB 2A F2 02 CD 79 03 2A EE 02 09 01
0340 FB FF 09 CD 79 03 EB 09 CD 80 03 CA 68 03 2A EE
0350 02 CD 79 03 2A F2 02 23 22 F2 02 09 CD 80 03 C2
0360 27 03 21 FF FF 22 F4 02 2A F4 02 23 EB 2A EC 02
0370 72 23 73 F1 E1 D1 C1 FB C9 7C 2F 47 7D 2F 4F C9

0380 7C FE 00 C0 7D FE 00 C9
```

```
   JB.BASIC V1.0
*LIST
10    CLEAR 10000
15    DIM F(10,10,10)
20    INPUT A$
30    CALLM #300,X
40    PRINT "FRE ";X
```

JEROEN DEMO

PAGE 01 -- JEROEN DEMO 7

```
10      REM ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
20      REM Jeroen Overvoorde Helmbloem 5 3068 AC Rotterdam
30      REM Telefoon 010-210426 Nederland  datum 1-10-1983
40      REM ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
50      REM ------------------------------------------------
60      REM Logo Nederlandse Omroep Stichting  NOS
70      REM ------------------------------------------------
100     MODE 6:COLORG 0 0 0 9:C=17:YM=YMAX-20:XM=XMAX:R=19
110     POKE #BFEE-13*90,#FF:POKE #BFEE-15*90,#F9
120     POKE #BFEE-209*90,#FF
200     FILL 22,YM-80 61,YM C
205     R=19:K=C:G=72:MY=YM-19:MX=92:GOSUB 1000
210     FILL 72,YM-80 111,YM-19 C
215     R=19:K=C:G=167:MY=YM-61:MX=192:GOSUB 2000:MY=YM-19:MX=141:GOSUB 3000
220     FILL 122,YM-19 167,YM-80 C:FILL 167,YM-61 211,YM C
225     R=29:K=0:MY=YM-40:MX=166:GOSUB 5000:R=20:K=C:GOSUB 5000
230     R=19:K=C:G=267:MY=YM-61:MX=292:GOSUB 2000:MY=YM-19:MX=241:GOSUB 3000
235     FILL 222,YM-19 267,YM-80 C:FILL 267,YM-61 311,YM C
240     R=29:K=0:G=222:MY=YM-71:MX=250:GOSUB 1000:G=311:MY=YM-9:MX=287:GOSUB
        4000
245     FILL 222,YM-80 279,YM-71 0:FILL 258,YM-9 311,YM 0
250     R=20:K=C:MY=YM-71:G=222:MX=250:GOSUB 1000:G=311:MY=YM-9:MX=287:GOSUB
        4000
255     FILL 222,YM-71 270,YM-80 C:FILL 267,YM-9 311,YM C
260     R=19:K=0:G=221:MY=YM-61:MX=241:GOSUB 4000:G=312:MY=YM-19:MX=292:GOSUB
        1000
265     FILL 22,YM+6 311,YM+7 23:FILL 22,YM-220 311,YM-219 23
300     FOR I=22 TO 112:DRAW 0,I+128 114,I+128 19:WAIT TIME 2:NEXT
305     FOR I=22 TO 112:DRAW I,150 I,YM 18:DRAW 114,I+128 214,I+128 19:NEXT
310     FOR I=22 TO 112:DRAW I+100,150 I+100,YM 18:DRAW 214,I+128 314,I+128 19:
        NEXT:TE=TE+1:IF TE=2 THEN GOTO 330
315     FOR I=22 TO 112:DRAW I+200,150 I+200,YM 18:DRAW 14,I+128 114,I+128 19:
        NEXT
320     GOTO 305
330     FOR I=150 TO 240:DRAW 14,I 114,I 19:DRAW 114,I-45 214,I-45 19:DRAW I+72,
        150 I+72,YM 18:NEXT
340     FOR I=195 TO 240:DRAW 114,I 214,I 19:DRAW 214,I-45 314,I-45 19:WAIT
        TIME 2:NEXT
350     FOR I=195 TO 240:DRAW 214,I 314,I 19:WAIT TIME 2:NEXT
400     WAIT TIME 200:LOAD "JEROEN DEMO 8"
1000    D=R*R:FOR Y=0 TO R:X=SQR(D-Y*Y)
1010    DRAW G,MY+Y MX+X,MY+Y K:NEXT Y
1020    RETURN
2000    D=R*R:FOR Y=0 TO R:X=SQR(D-Y*Y)
2010    DRAW G,MY-Y MX+X,MY-Y K:NEXT Y
2020    RETURN
3000    D=R*R:FOR Y=0 TO R:X=SQR(D-Y*Y)
3010    DRAW MX-X,MY+Y G,MY+Y K:NEXT Y
3020    RETURN
4000    D=R*R:FOR Y=0 TO R:X=SQR(D-Y*Y)
4010    DRAW MX-X,MY-Y G,MY-Y K:NEXT Y
4020    RETURN
5000    D=R*R:FOR Y=0 TO R:X=SQR(D-Y*Y)
5010    DRAW MX-X,MY-Y MX+X,MY-Y K:DRAW MX+X,MY+Y MX-X,MY+Y K:NEXT Y
5020    RETURN
```


LE MONITEUR BASIC

DAInamic INFO

LE MONITEUR BASIC (#C80C-#C956)
==========

1er PARTIE : GENERALITES

Cet article décrit les grandes lignes du fonctionnement des routines du moniteur BASIC du DAI. De prochains articles permettront de détailler certaines parties du moniteur. Le DAI PC FIRMWARE MANUAL sert de référence à cet article. L'organigramme simplifié ci-joint peut vous guider au long de cet article.

Le moniteur BASIC peut etre considéré comme le coeur du système d'exploitation du DAI-BASIC. Sous son controle, des lignes de programme peuvent etre entrées dans le TEXT-BUFFER, des commandes directes peuvent etre éxécutées, l'éxecution d'un programme est controlée, un BREAK est géré,...

Le moniteur est appelé par le DAI après un RESET ou la mise sous tension à l'adresse #C818. D'autres adresses (#C80C, #C814 et #C823) sont aussi utilisées à la fin certaines routines (ceci sera explicité dans la 2em partie)

INITIALISATION (#C818-#C843) :

Le moniteur doit etre initialisé. Ceci signifie qu' une situation bien définie de départ est à creer. Le STACKPOINTER est initialisé, les flags controlants l' éxecution des INPUT, appel de programmes et de ss prgms, et de l'encodage des lignes stockées sont remis à zéro. Résultat: un DAI pret. Les interruptions clavier et horloge sont 'en ligne', sans lesquelles aucune entrée, ni aucune sortie sur écran n'est possible.

ENTREE A PARTIR DE L'EDIT BUFFER (#C846-#C84E) :

Le SWITCH d'encodage des entrées #0135 est controlé. Ce switch définie la provenance de l'entrée (l'information) qui doit etre encodée. Encodé signifie : traduction des lignes de BASIC (commandes directes ou lignes de programme) du BASIC 'lisible' en un BASIC codé lisible par le semi-compilateur du DAI permettant ainsi L1'execution de ces lignes. C'est dans ce code q'un programme est stocké dans le TEXT-BUFFER. Des détails sur ce pseudo-code peuvent etre trouvés dans le NEWSLETTER 11, page 196...

Si la source est l'EDIT-BUFFER, alors la saisie des informations à partir de ce buffer est préparé via #D879, et l'encodage intervient via #C867. Lorsqu'une ligne est encodée, le moniteur recommence de meme jusqu'a l'epuisement de l'edit-buffer.

A voir également sur ce sujet un précédent article 'EDITOR STORY'.

REMARQUE: L'encodage d'une 'cochonnerie indéfinie' (c.a.d. le contenu momentané du buffer d'entrée d'encodage) intervient aussi si le switch #0135 est >2 !!! Cette faiblesse du FIRMWARE s'explique sur la conviction que #0135 est toujours <=2.

ENTREE A PARTIR DU CLAVIER (#C851-#C864) :

Si la source d'entrée n'est pas l'edit-buffer, mais le clavier , le signe ('*') est

affiché à l'écran et une entrée de ligne BASIC est attendue via #DD1A. En toutes lettres: le DAI attend une ligne de commandes directes ou une ligne de programme BASIC à stocker dans le TEXT-BUFFER, à partir du clavier uniquement. La séquence de saisie ne peut s'achever que par un CR ('RETURN')
ou par un BREAK. Tous les caractères tapés sont affichés à l'écran La mémoire d'écran est alors à ce moment le seul emplacement ou existe l'information entrée. Lorsqu'un retour chariot CR est entré, le moniteur commence alors à travailler. Il saisie le premier caractère de la ligne entrée dans la mémoire d'écran, pour l'analyser.

La différence entre une ligne de commandes directes, et une ligne de programme à stocker dans le TEXT-BUFFER est la présence d'un no de ligne en tete de la ligne de programme. Et

si ce no existe, alors la ligne doit se coder dans le text-buffer. Dans le cas contraire, et son execution doit suivre immédiatement son encodage. Enfin, si le premier caractere est un CR, la ligne d'entrée est non valide , et le moniteur attend après une autre ligne.

ENCODAGE DE LIGNE DE PROGRAMME (#C867-#C86A) :

Si une ligne de programme a été entrée, elle est encodée et stockée dans le TEXT-BUFFER, à un emplacement correct; ce qui signifie dans l'ordre des numéros de ligne. Le moniteur est relancé pour saisir la nouvelle ligne. Des détails sur le processus d'encodage seront apportés dans un prochain article.

ENCODAGE DE LIGNE DE COMMANDES DIRECTES (#C86D-#C87D) :

La ligne de commandes directes est encodée via une routine d'encodage d'ordres BASIC via RST1/00. Le résultat de ce processus, le code 'semi-compilé' , est stocké dans le buffer d'entrées encodées EBUF #013E-01BD. Ce buffer de 128 octets explique l'erreur 'LINE TOO COMPLEX' qui intervient pour une ligne d'instructions encodées de plus de 128 octets

Le flag #0117 est activé (#FF), simulant ainsi un RUN de lignes de programme. Le registre BC, utilisé pour indiquer ou le moniteur est actif dans dans la ligne de commandes en cours, pointe sur le debut de la ligne dans le buffer d'entrées encodées EBUF, et la ligne de commandes est prete à s'executer.

EXECUTION DE COMMANDES DIRECTES OU DE LIGNES DE PROGRAMMES

Ce chapitre concerne aussi bien l'execution de commandes directes que celle de lignes de programme sockées dans le TEXT-BUFFER. La seule différence concerne le contenu du registre pair BC. Il pointe vers le buffer d'entrées codées EBUF pour l'execution de commandes directes.Et dans le cas de l'execution d'un programme stocké dans le buffer de texte, il pointe sur le début d'une ligne de texte dans ce buffer. Le premier caractère d'une ligne est alors saisi.Si c'est un code (NDT: Jan Boerrigter parle d'un 'token', soit un signe, une marque...) -une commande BASIC sous sa forme compilée- l'adresse de départ de la routine d'exécution de la commande BASIC correspondant à ce code est trouvée dans la table #CF02, et cette routine est executée via #C8A9. Dans le cas ou le premier octet n'est pas un code (#C8E5), il peut alors etre soit un '0', soit l'octet de longueur placé au début

d'une ligne de programme. Un '0' indique alors la fin du contenu du TEX-BUFFER,(ou un '0' est inséré après la dernière instruction), ou la fin d'une ligne de commandes directes (voir #C876). La rencontre d'un '0' relance le MONITEUR. Si le premier octet lu par le moniteur est l'octet de longueur d'une ligne de programme stockée, alors le TRACE/STEP FLAG est (#0115-#0116). Si l'un de ces flags est ACTIF, la nouvelle ligne de programme est affichée sur l'écran. Dans le cas du FLAG STEP, l'enfoncement de la touche SPACE est attendue.Si la touche 'BREAK' n'est pas enfoncée, le moniteur continue en #C87F.

LA FIN DE CERTAINES ACTIONS (#C8AA-#C8B5) :

Quand le code a été trouvé, quand l'execution de la routine est terminée, le moniteur retourne en #C88F. Il controle alors l'éventualité d'une fin spéciale de la routine BASIC en cours. Cette 'fin spéciale' es signalée par un code dans l'accumulateur A et la mise sur ACTIF du CARRY-FLAG après l'execution de la routine BASIC. Voici par exemple #DF03: 'STOP' : CY=1, A=3

Quatre actions sont possibles:

0: #C908-#C915 : Après 'LOAD' : if 'LOAD' n'est pas une commande directe, le programme chargé sera démarré immédiatement.

1: #C818 : CAN'T CONTINU :Après certaines commandes qui provoquent unemodification des pointeurs de programme, ou qui altèrent le contenu du TEXT-BUFFER le moniteur est relancé.

2: #C8C0-#C8C8 : Après un 'soft break' (BREAK provoqué par le programme),le BREAK doit etre pris en compte et traité.

3: #C8B8-#C8BD : Après la commande 'STOP' d'une ligne BASIC . L'endroit ou s'arrete le programme doit etre mémorisé et un 'STOPPED IN LINE....' est affiché.
D'autres traitements interviennent comme pour le 'soft break'

PROGRAMMEERTECHNIEKEN (NO 13, page 301)

Voyons maintenant le sujet d'aujourd'hui: les couleurs 16 à 19 d'une part et les couleurs 20 à 23 d'autres part. Je commencerais avec ces derniers, l'utilisation des couleurs 20 à 23 servant dans le dessin en mode 4 couleurs . C'est pourtant simple . Au tout debut d'un programme , nous placons l'ordre COLORG A B C D ou A B C D sont nos quatres couleurs desirees.Dans notre programme, chaque appel de la couleur 20 donnera la Couleur A ,21 la couleur B , et ainsi de suite... C'est d'ailleurs une bonne habitude de commencer par une instruction COLORG,plutot que par une instruction MODE. Nous eviterons ainsi de voir l'ecran prendre une voliente teinte jaune (provenant du precedent ordre COLORG) plutot que le noir souhaite. Dans le mode 16 couleurs,la premiere couleur de l'ordre COLORG devient la couleur de fond,et le bord 'obtient' aussi cette couleur.Ceci non pas avec un ordre FILL 0,0 XMAX,YMAX K , mais seulement avec un nouvel ordre COLORG suivi de MODE 1,3 ou 5.L'avantage de l'habitude de surtout utiliser les codes couleurs 20 à 23 est dans le fait que, pour un nouvel usager, une mauvaise combinaison de couleurs est facile à modifier. Sur mon Moniteur RGB, j'ai l'excellente habitude de prendre les couleurs 0 1 2 3,mais je comprend que l'utilisateur d'un moniteur noir et blanc me maudisse si je choisis les couleurs 4 5 8 13 !!.

MOD. DNA-FWP

Cette petite modification apportée à DNA permet de récupérer dans le buffer de FWP le résultat de la commande]#L. de DNA; ceci en vue de l'insérer dans un texte ou à tout autre usage.

Marche à suivre :

1. Lire DNA.

2. Introduire la modification par SUBSTITUTE.

3. TRES IMPORTANT : initialiser le pointeur PTRBUF (selon le buffer que l'on veut utiliser) avec les valeurs suivantes :
   BUFFER 1 -> #1175/#1176 = 00-30
   BUFFER 2 -> #1175/#1176 = 00-90

4. >G1100

5. Lire le programme que l'on veut éditer.

6. Taper la commande]#L.

7. Taper JU. et à l'adresse pointée par PTRBUF (#1175/#1176) substituer #00 (FLAG DE FIN pour buffer de FWP).

8. Remplacer le contenu de #3FF par #FF (Bypass clear buffer).

9. >G400

Ci-dessous le programme et les commentaires :

```
PAGE 01     DNA ---> BUFFER FWP

002                  PTRBUF  EQU   :1175
003                          ORG   :1EEA
004 1EEA CD5EDD              CALL  :DD5E
005 1EED 7E          LOOP    MOV   A,M
006 1EEE E67F                ANI   :7F
      Il faut masquer le bit 7 car FWP
      n'affiche pas les caractères >#7F.
007 1EF0 23                  INX   H        ************************
008 1EF1 E5                  PUSH  H        * S Y M B O L   T A B L E
009 1EF2 2A7511              LHLD  PTRBUF   ************************
010 1EF5 77                  MOV   M,A
      On garnit ici le buffer de FWP.      LOOP   1EED   PTRBUF 1175
011 1EF6 23                  INX   H
012 1EF7 227511              SHLD  PTRBUF
      On remet à jour le pointeur.
013 1EFA E1                  POP   H
      On restitue le pointeur pour DNA
      et on continue le traitement normal.
014 1EFB FE0D                CPI   :0D
015 1EFD CA061F              JZ    :1F06
016 1F00 CD7711              CALL  :1177
017 1F03 C3ED1E              JMP   LOOP
018 1F06                     END
```


NEW

FYSISCHE GEOGRAFIE en KLIMATOLOGIE

1- OMBROTHERMISCH DIAGRAM
2- KLIMAATSKLASSIFIKATIE VAN BAGNOULS
3- ANALYSE VAN POLYGONEN
4- INOEFENEN KAARTSCHAALBEREKENINGEN
5- INOEFENEN BEREKENEN VAN DE HELLINGSGRAAD
6- RELIEFSPROFIEL EN VERTIKALE OVERDRIJVING
7- KOSMISCHE KALENDER

(c) diDAIsoft MA+E.M DJ-84

FYSISCHE GEOGRAFIE

Een nieuwe collectie onderwijs-programma's , verzameld door Marc Antrop.
audio : 1000 Bfr
DCR : 1150 Bfr

D - BASIC

NEW

DBASIC version 2.2 and how to get it...

W.Coremans has released a new version of DBASIC.(see also his article on p.427-430).
A few small bugs have been corrected but what is more important, 2 beautiful extensions for DBASIC are available now:
1/ a very special format listing , with extra list of extended commands & functions, procedures,functions,labels,arrays and variables. (see sample program on p.429-430).
2/ another extension, offering true programmable function keys.

If you already own the DBASIC program and you want these new facitilies together with the new version of DBASIC, send us your cassette together with 250 Bfr for mailing and administration costs.
For compatibitily reasons, only version 2.2 will be delivered in the future.

note : we look forward to receive the first DBASIC programs!

BOEKHOUDPROGRAMMA

new!

Dit programma is bestemd voor kleine bedrijven of verenigingen, voor het vervaardigen van een boekhouding vanaf het invoeren van de administatieve gegevens tot en met de verlies- en winstrekening, balans en kapitaalsmutatie.

Een beperkte kennis van het boekhouden is voldoende om met dit programma Uw boekhouding zelf uit te voeren.

Het programma gaat uit van het principe van dubbel-boekhouden waardoor per boeking (boekstuknummer) de totalen van debet en credit aan elkaar gelijk zijn, dus in evenwicht moeten zijn, voordat die boeking in het geheugen kan worden opgeslagen.

Het programma biedt de navolgende mogelijkheden :

* In het geheugen per keer 50 boekingen op de 29 groot boekrekeningen op te slaan, compleet met datum, boekstuknummer en 50 posities voor de omschrijving voor elke boeking.

* Boekingen op zoveel grootboekrekeningen als nodig tegen te boeken.

* Vergissingen onmiddellijk en eenvoudig op het scherm te corrigeren voordat wordt doorgeboekt.

* Het saldo automatisch naar een van de de BTW rekeningen te boeken door deze als sluitpost te gebruiken

* Reeds opgeslagen gegevens te controleren en naar behoefte te wijzigen.

* Gecontroleerde gegevens op papier of op het scherm of op beide af te drukken als controlelijst of journaal.

* Overzichten per grootboekrekening te vervaardigen.

* Gegevens op tape of DCR op te slaan voor later gebruik en om verder te gaan met boeken.

* Om fouten te voorkomen is printen op papier of wegladen op tape of DCR niet mogelijk als de gegevens niet in evenwicht zijn. U moet dan eerst controleren en wijzigen.

* Op elk gewenst moment een tussentijdse proef- en saldibalans op te vragen.

* Een volledige (tussentijdse) balans met resultatenoverzicht te vervaardigen, zonder dat er iets met de andere gegevens gebeurt, anders dan transporteren. Hierdoor kan bijvoorbeeld elk kwartaal een overzicht vervaardigd worden, wat voor een goede bedrijfsvoering erg belangrijk kan zijn.

* Een automatische kapitaalsmutatie, waarbij het resultaat direct naar het kapitaal geboekt wordt.(Ook tussentijds zonder gevolgen voor de gegevens).

audio:2000
dcr :2150

RAUTEN

```
PAGE 01  --  RAUTEN

30      MODE 6:COLORG 1 1 1 1

50      REM ......... FENSTERLN .... BY ROLF SCHALL
100     REM ....... BLAU-WEISS .......
110     FILL 33,21 XMAX-33,YMAX-21 21:COLORG 1 15 15 15
112     FILL XMAX/2-36,YMAX/2-24 XMAX/2+36,YMAX/2+24 23
115     FOR X0=0 TO 1000-59 STEP 59:X=X0
120   1  FOR A=0.0 TO YMAX-20.0 STEP 7.0
130   2   IF X<7 THEN 150:X=X-8:IF X<XMAX-32 THEN GOSUB 500
140   1  NEXT A
150     NEXT X0

200     REM ....... FARBWECHSEL .......
210     R=20.0:GOSUB 600
220     FOR I=1.0 TO 20:COLORG 1 15 15 1:WAIT TIME 40-I:COLORG 1 15 1 15:WAIT
        TIME 20-I:NEXT
230     COLORG 1 15 15 1:R=23:GOSUB 600
240     FOR I=1.0 TO 20:COLORG 1 15 15 1:WAIT TIME 40-I:COLORG 15 1 15 1:WAIT
        TIME 20-I:NEXT
250     COLORG 1 15 15 1:R=20:GOSUB 600
260     FOR I=1 TO 15:COLORG 1 15 RND(15) 15:WAIT TIME 30:NEXT
270     FOR I=1 TO 20:C=RND(15):COLORG C 15 C 15:WAIT TIME 30:NEXT
280     COLORG 15 15 1 1
290     GOTO 290

500     REM ...... RAUTENDRUCK ......
520     FOR I=0.0 TO 13.0:DRAW I+X,I+A I+X+20,I+A+7 16:NEXT:RETURN

600     REM ....... RAND .......
610     FILL 0,0 XMAX,20 R:FILL 0,YMAX-20 XMAX,YMAX R
620     FILL 0,21 32,YMAX-21 R:FILL XMAX-32,21 XMAX,YMAX-21 R:RETURN
```

ERRATA : numbers up

```
470   Q=RND(4.0):IF (Q+QO) MOD 4=1 GOTO 470:GOSUB 740
510   COLORT 8 8 8 9:FOR Y=0 TO B:S$="":FOR X=0 TO B*7:S$=S$+CHR$(11):NEXT
```

80 KOLOMMEN TEKST

```
*************************************************
*                                               *
*    Wijziging: 80 kolommen tekst op de DAI     *
*                                               *
*************************************************
```

Omdat bij mij de belangstelling groot is om op korte termijn "CPM" software te gaan "draaien" op een apart systeem, waarbij de DAI als terminal moet gaan dienen. Hierbij is echter wel raadzaam dat men over een terminal beschikt met >= 80 kolommen- en >=24 rijen tekst. Als men in de toekomst op de DAI zelf "CPM" gaat "draaien" lijkt mij deze wijziging ook uiterst zinvol. Om de wijzigingen beperkt te houden zijn alle screen modes onveranderd gebleven, behalve de laagste resolutie tekst modes. Naar mijn weten wordt deze tekst modes door niemand gebruikt, zodat ik deze modes (10 kolommen tekst) omgebouwd heb naar 80 kolommen tekst, met als praktische mogenlijkheid van 24- of 25- of 26- rijen tekst. Met normaal gebruik van de DAI merkt niemand iets van de hard-ware ingreep.

Alvorens men de beslissing neemt om de DAI te gaan wijzigen moet men vast stellen dat zijn of haar machine voorzien is van snelle type dynamishe Ram geheugens. Het type Ram moet zijn: 416-2 of 4116-2 of 16k2 of dergelijke 150 nS Ram ic's. Is dit echter niet het geval dan kan men de wijziging niet uitvoeren of men moet natuurlijk de geheugen ic's vervangen door alreeds 1 van aangegeven typen. Mijn machine was al voorzien van het snelle type Ram ic's. (DAI Rev. 4)

Door de nog te omschrijven wijziging ondergaat de betekenis van de lijn control words van het video screen geheugen een kleine aanpassing:

High addr.byte MODE	7 6 5 4	Video screen mode: Voor wijziging	Na wijziging
2	0 0 0 0	88 Kol. 4 Kl. Graf.	Idem
4	0 0 0 1	176 Kol. 4 Kl. Graf.	Idem
6	0 0 1 0	352 Kol. 4 Kl. Graf.	Idem
(8)	0 0 1 1	528 Kol. 4 Kl. Graf.	Idem
???	0 1 0 0	11 Kol. 4 Kl. Tekst	88 Kol. 4 Kl. Tekst
~	0 1 0 1	22 Kol. 4 Kl. Tekst	Idem
~	0 1 1 0	44 Kol. 4 Kl. Tekst	Idem
0	0 1 1 1	66 Kol. 4 Kl. Tekst	Idem
1	1 0 0 0	88 Kol.16 Kl. Graf.	Idem
3	1 0 0 1	176 Kol.16 Kl. Graf.	Idem
5	1 0 1 0	352 Kol.16 Kl. Graf.	Idem
(7)	1 0 1 1	528 Kol.16 Kl. Graf.	Idem
~	1 1 0 0	11 Kol.16 Kl. Tekst	88 Kol.16 Kl. Tekst
~	1 1 0 1	22 Kol.16 Kl. Tekst	Idem
~	1 1 1 0	44 Kol.16 Kl. Tekst	Idem
~	1 1 1 1	66 Kol.16 Kl. Tekst	Idem

Om de wijzigingen hardware-matig beperkt te houden wordt er tijdens de nieuwe 80 (88) koloms tekst mode wordt er gebruik gemaakt van een andere character set. deze set is gebaseerd op een 6*10 matrix (spacing) in plaats van de in de DAI gebruikte 8*11 matrix (spacing). Dit houdt in dat de huidige character generator prom moet worden vervangen door 4k eprom van het type 2732(A) of 2532. In de ene helft van de prom zit dan de normale character set van de DAI, terwijl in de andere helft dan de aangepaste character set bevindt. Dit laatste wordt specifiek voor de 80 (88) koloms tekst modes gebruikt. De nieuwe character prom dient op een ic-voet geplaatst te worden in verband met de nog eventueele latere wijzigingen aan de character sets. De omschakeling van de character sets gebeurd hardware-matig.

Vanzelf sprekend moet de snelheids selectie prom (74S288 bruin ic 5) (resolutie besturing) vervangen worden door een nieuwe prom van een dergelijk type met een aangepaste inhoud (74S288 of 82S123).

Tevens wordt er een 74LS10 (3-voudig-input nand) circuit toegevoegd om het cycle-skipping process (2 uit 3 overslaan) welk specifiek wordt gebruikt voor de 80 (88) koloms tekst modes te kunnen bewerkstelligen. Het genoemde ic wordt boven op een ander ic geplaatst

De RAS timing van de geheugen chips moet worden teruggebracht van ongeveer 250 nS naar ongeveer 160 nS. Dit houdt een vervanging in van een timing condensator.

Verder moet een aantal print-sporen worden doorgesneden en moet er een aantal draadjes worden gelegd.

Kosten plaatje:

- Eventueel snellere geheugen ic's (150 nS)
  HCC Hardware service 4116-2 (24%) (: fl 182,40)
- Nieuwe character generator ic 2732(A) of 2532
  HCC Hardware service 2732 : fl 18,95
- Nieuwe snelheids selectie prom ic 74S288 of
  82S123 : fl 18,90
- Een ic t.b.v. cycle skipping 74LS10 : fl 2,50
- Enventueel ic voet t.b.v. character gen. (: fl 2,--)
- 1 weerstand + 3 condensatoren : fl 2,--
- montagedraad, soldeertin en verzendkosten : fl 12,--
- 1 a 2 avonden montage- en testwerk (: fl ???,??)

Is de machine van het snelle type Ram ic's voorzien, beschikt men over hardware ervaring, test- en meet- en programmeer apparatuur, is de wijziging goedkoop uit te voeren. Niet is meegerekend de kosten welke eventueel zijn verbonden aan de bijbehorende (firm-) (soft-) ware aanpassingen / toevoegingen.

De hiervoor gobaal aangegeven wijzigingen wil ik pas in details publiceren (ook in DAInamic.) en vervolgens vrijgeven nadat ik een aantal gewillige "slachtoffers" heb gevonden om hun of haar machines tegen kostprijs aan te laten passen om zodoende wat meer ervaring op te kunnen doen. Tevens wil ik dan van hun bevindingen op de hoogte gesteld te worden.

Voordat ik deze brief heb geschreven heb ik al enkele weken met de nieuwe tekst modes gespeeld. Hierbij is de machine niet "plat" gegaan ten gevolge van hardware fouten. Mijn aanbeveling is om een nieuwe tekst mode te definieeren, bestaande uit 80 kolommen en 26 rijen tekst, gebruik makend van de 4 kleuren mode. De naamgeving voor deze nieuwe screen mode stel ik dan voor: "MODE 9".(Al reeds aangegeven in tabel.)

Voor diegenen die de 80 (88) koloms tekst geheel software matig willen oplossen in een "MODE 8" omgeving door middel van een speciale "FGT" wil ik de door mij gebruikte character set beschikbaar stellen. De software oplossing heeft als belangrijke nadelen: Bijna 32 Kb geheugen in gebruik in plaats van ruim 4,6 Kb met de hardware oplossing ((88*2+2)*26+2*16), en zal bijzonder traag zijn in verband met het scrollen en het clear screen commando. Hierbij is praktisch slechts 24 rijen tekst mogenlijk.

De (K)TV- en video monitor interface kaarten behoeven niet te worden aangepast. Nodig is misschien wel dat de resolutie van met name de "pal color tv interface chart" verbetert moet worden als men de 80 (88) koloms tekst mode op een gewone KTV wil gebruiken. Bij de overige interface kaarten is de resolutie al groot genoeg. De wijzigingen die ik een ieder aanbeveel ten aanzien van de "pal color tv chart" heb ik al reeds omschreven in DAInamic 18. (Alleen uit te voeren, welke zijn omschreven in hoofdstukken 1 en 2.)

Ik hoop met de hiervoor omschreven 80 koloms aanpassing een kleine impuls te kunnen geven ten voordelen van de DAI personal computer, waardoor misschien de levenskans van deze overigs schitterende machine wat toeneemt.

Gaarne ontvang ik opmerkingen en suggesties ten aanzien van dit onderwerp.

Ik hoop op de volgende landelijke bijeenkomst van de DAI-GG te Utrecht het een en ander te kunnen demonstreren.

Anton Doornenbal,
Oud AA 39A,
3621 LA Breukelen.
Tel. Prive : 03462-63237
Kantoor : 035-891036

SAVEV / LOADV

SAVEV / LOADV (c) Ch. POELS 25/12/83

The instructions LOADA and SAVEA of BASIC have a few limitations which can become very troublesome if one desires to handle arrays of important dimensions, for instance in the case of files on cassettes.

Indeed, when we save an array on cassette, BASIC reorganizes all the data of the array before recording them. This reorganization requires a memory area which lies immediately behind the SYMBOL TABLE. This is a major inconvenience if our BASIC program is very big and if too little memory is left over for this operation. In that case, an error message is displayed:"OUT OF MEMORY".

Similarly, when we load an array in memory, data are first saved in bulk behind the SYMBOL TABLE and then reorganized and transferred into the HEAP. This operation can be very lenghty, and the fact that digital cassettes are used does not, unfortunately, change anything to this state of affairs.....

In order to cope with this problem, I have written a routine which saves directly on cassette the whole of the HEAP and the SYMBOL TABLE. This routine has also its drawbacks, but it can be very useful in certain cases. Indeed, it is not necessary anymore to have a free memory area available for reorganization.Moreover, the waiting time after LOADA is cancelled. These 2 reasons seemed enough to me for writing this routine. In order to be able to use it, two limitations have to be kept in mind: when loading the variables, the CLEAR and the BASIC program have to be exactly the same as when the recording of variables was done!Moreover, after loading the variables, all their values are updated (because the whole of the variables has been loaded). It is therefore advised to perform the reading of the variables at the beginning of the progam, during the initializations.

Use of the routine: saving of the variables occurs as follows: CALLM #34A,A$ (A$ being an alphanumeric variable containing the name of the file). The FILE TYPE of this file is "3".

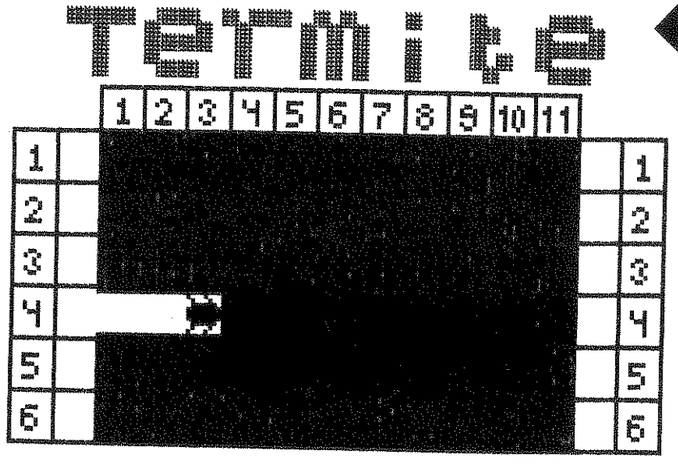
To reload the variables: CALLM #300,A$.

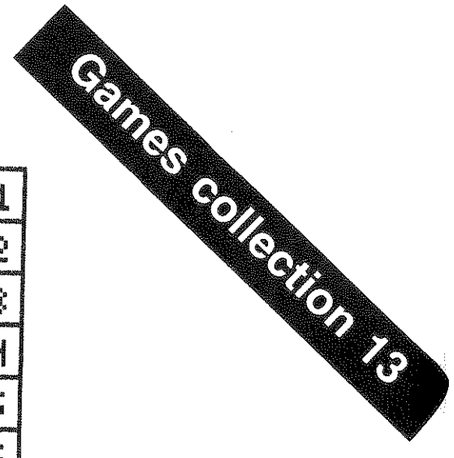
```
>D300 390
0300 C5 D5 E5 F5 F3 3A 40 00 F6 C0 32 40 00 32 06 FD
0310 5E 23 56 EB 06 33 0E FF CD CE 02 2A 9B 02 11 00
0320 F9 CD D1 02 D2 AD D2 2A A1 02 11 00 F9 CD D1 02
0330 D2 A8 D2 22 A3 02 CD D4 02 3A 40 00 E6 3F 32 40
0340 00 32 06 FD FB F1 E1 D1 C1 C9 C5 D5 E5 F5 F3 3A
0350 40 00 F6 C0 32 40 00 32 06 FD 3E 33 5E 23 56 EB
0360 CD C5 02 2A 9D 02 EB 2A 9B 02 CD C8 02 2A A3 02
0370 EB 2A A1 02 7B 9D 5F 7A 94 57 CD C8 02 CD CB 02
0380 3A 40 00 E6 3F 32 40 00 32 06 FD FB F1 E1 D1 C1
0390 C9
```



```
]#L.
PAGE 01

001                         ORG   :300
002 0300 C5                 PUSH  B          LOADV
003 0301 D5                 PUSH  D
004 0302 E5                 PUSH  H
005 0303 F5                 PUSH  PSW
006 0304 F3                 DI
007 0305 3A4000             LDA   :40
008 0308 F6C0               ORI   :C0
009 030A 324000             STA   :40
010 030D 3206FD             STA   :FD06
011 0310 5E           DLECT MOV   E,M        FILENAME POINTED BY HL
012 0311 23                 INX   H
013 0312 56                 MOV   D,M
014 0313 EB                 XCHG
015 0314 0633               MVI   B,'3'      FILE TYPE
016 0316 0EFF               MVI   C,:FF      FILE NAME PRINTED
017 0318 CDCE02             CALL  :2CE       READ FILE NAME
018 031B 2A9B02             LHLD  :29B       READ HEAP
019 031E 1100F9             LXI   D,:F900
020 0321 CDD102             CALL  :2D1
021 0324 D2ADD2             JNC   :D2AD      RUN L.E. WITHOUT NEW
022 0327 2AA102             LHLD  :2A1       READ SYMBOL TABLE
023 032A 1100F9             LXI   D,:F900
024 032D CDD102             CALL  :2D1
025 0330 D2A8D2             JNC   :D2A8      RUN LOADING ERROR
026 0333 22A302             SHLD  :2A3       END OF S. TABLE UPDATED
027 0336 CDD402             CALL  :2D4       RCLOSE
028 0339 3A4000             LDA   :40
029 033C E63F               ANI   :3F
030 033E 324000             STA   :40
031 0341 3206FD             STA   :FD06
032 0344 FB                 EI
033 0345 F1                 POP   PSW
034 0346 E1                 POP   H
035 0347 D1                 POP   D
036 0348 C1                 POP   B
037 0349 C9                 RET
038 034A C5                 PUSH  B          SAVEV
039 034B D5                 PUSH  D
040 034C E5                 PUSH  H
041 034D F5                 PUSH  PSW
042 034E F3                 DI
043 034F 3A4000             LDA   :40
044 0352 F6C0               ORI   :C0
045 0354 324000             STA   :40
046 0357 3206FD             STA   :FD06
047 035A 3E33               MVI   A,'3'      FILE TYPE
048 035C 5E                 MOV   E,M        FILE NAME POINTED BY HL
049 035D 23                 INX   H
050 035E 56                 MOV   D,M
051 035F EB                 XCHG
052 0360 CDC502             CALL  :2C5       SAVE FILE NAME
053 0363 2A9D02             LHLD  :29D       SAVE HEAP
```

```
PAGE 02

054 0366 EB              XCHG
055 0367 2A9B02          LHLD  :29B
056 036A CDC802          CALL  :2C8
057 036D 2AA302          LHLD  :2A3      SAVE SYMBOL TABLE
058 0370 EB              XCHG
059 0371 2AA102          LHLD  :2A1
060 0374 7B              MOV   A,E
061 0375 9D              SBB   L
062 0376 5F              MOV   E,A
063 0377 7A              MOV   A,D
064 0378 94              SUB   H
065 0379 57              MOV   D,A
066 037A CDC802          CALL  :2C8
067 037D CDCB02          CALL  :2CB      WCLOSE
068 0380 3A4000          LDA   :40
069 0383 E63F            ANI   :3F
070 0385 324000          STA   :40
071 0388 3206FD          STA   :FD06
072 038B FB              EI
073 038C F1              POP   PSW
074 038D E1              POP   H
075 038E D1              POP   D
076 038F C1              POP   B
077 0390 C9              RET
078 0391                 END

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

DLECT  0310
```






```
50000   SCREEN TABULATOR & EPSON 80/2 III, replaces print tab();

50010 REM E. ZAHNER, CH 8910 AFFOLTERN MARCH 25,1983
50020 REM PERMITS SCREEN TABLE WITH RS232C OFF
50030 REM PRINTER COMMANDS:
50040 REM CHR$(27);"D";CHR$(FIRST);CHR$(SECOND);..CHR$(0) TO CONCLUDE
50050 REM CHR$(9) TO MOVE PRINT HEAD TO NEXT TAB POSITION
50060 REM
50100 CLEAR 1000
50110 POKE &131,1:PRINT CHR$(12);
50120 V24=&FF85:BAUD0=&C0:BAUD1=&90:REM RS232C V24 SERIAL OUTPUT 9600 2400 BAUD
50130 WRITER=&131:SWON=0.0:SWOFF=1.0:REM PRINTER ON, OFF
50132 REM PRI= PREVIOUS STATUS WRITER
50140 REM VARIABLES
50150 REM TABUZ% NBR OF TABS.  TABULA%(..) TAB SETTINGS
50160 REM CX &CY LAST CURSOR POSITION
50170 REM TABULA$ = TAB INSTRUCTION TO PRINTER (EPSON III)
50180 REM WO = NEW TAB POSITION.
50200 GOTO 50300
50210 REM SUBROUTINES
50220 PRI=PEEK(WRITER):POKE WRITER,SWOFF:CX=CURX:CY=CURY
50230 INPUT "<RETURN> ";RETU$:PRINT :POKE WRITER,PRI:RETURN
50240 REM WIPE SCREEN, NO FORM FEED
50250 PRI=PEEK(WRITER):POKE WRITER,SWOFF:INPUT "<RETURN WHEN OK > ";RETU$
50260 PRINT CHR$(12);:CX=CURX:CY=CURY:POKE WRITER,PRI:RETURN
50300 TABU$="":TABULA$="":REM TAB POSITION EXAMPLE
50310 DATA 5,15,25,35,45,55
50320 TABUZ%=6:DIM TABULA%(TABUZ%):FOR I%=1 TO TABUZ%
50330 READ TABULA%(I%):TABU$=TABU$+CHR$(TABULA%(I%))
50360 NEXT I%:TABULA$=CHR$(27)+"D"+TABU$+CHR$(0)
50370 PRINT TABULA$;
50400 POKE V24,BAUD1:POKE WRITER,SWON
50410 PRINT TABULA$
50420 PRINT "0";:FOR I%=1 TO TABUZ%:PRINT CHR$(9);"*";
50430 NEXT I%:PRINT
50440 FOR I%=1 TO TABUZ%:PRINT CHR$(9);I%;
50450 NEXT I%:PRINT
50460 FOR I%=-1 TO TABUZ%*(-1) STEP -1:PRINT CHR$(9);I%;
50470 NEXT I%:PRINT
50480 FOR I%=1 TO TABUZ%:PRINT CHR$(9);TABULA%(I%);
50490 NEXT I%:PRINT :POKE WRITER,SWOFF
50495 LIST 50495:REM NOTE: THE TAB POS. IS IN FRONT OF THE   FIRST CHARACTER ON
THE PRINTER
50500 LIST 50500:REM NOW A TABLE SHALL BE SHOWN ON THE SCREEN
50510 GOTO 50700
50520 REM SUBROUTINES FOR SCREENTABLE
50530 CX=CURX:CY=CURY:REM RETURN:REM STORES OLD CURSOR POSITION
50540 WO=TABULA%(I%):REM GETS NEXT TAB POSITION INTO WO
50550 IF WO<CX THEN WO=CX:REM RETURN:REM OLD<NEW
50560 PRINT CHR$(9);:PRI=PEEK(WRITER)
50570 POKE WRITER,SWOFF:PRINT SPC(WO-CX);:POKE WRITER,PRI
50580 RETURN:REM IF 50530/50550 RETURNS ARE USED, 50560 COMES TWICE ??
50700 POKE WRITER,SWON
50710 PRINT TABULA$
50720 TABUZ%=6
50730 FOR I%=1 TO TABUZ%
50732 GOSUB 50530
50734 REM GOSUB 50540
50736 REM GOSUB 50560:REM NOTE 50580
50740 PRINT CHR$(124);:NEXT I%:PRINT
50750 POKE WRITER,SWOFF
50800 LIST 50800:REM INPUT TAB POSITIONS
50805 PRINT "NBR","POS","AVOID 12 or 13"
50810 INPUT "NUMBER OF TAB-STOPS ";TABUZ%:PRINT
50820 DIM TABULA%(TABUZ%):TABU$="":TABULA$="":T=0.0
50830 FOR I%=1 TO TABUZ%:PRINT I%,
50840 INPUT TABULA%(I%):PRINT
50850 IF TABULA%(I%)<T THEN 50840
50860 T=TABULA%(I%)+1.0:IF T<3.0 THEN T=3.0
50870 TABU$=TABU$+CHR$(T-1):NEXT I%:PRINT
50880 TABULA$=CHR$(27)+"D"+TABU$+CHR$(0)
50900 REM
50910 GOSUB 50240:REM WIPE
50920 PRINT "NOW INPUT A STRING SUITABLE FOR THE COLUMN WIDTH"
50930 POKE WRITER,SWON:PRINT TABULA$:STRING$=""
50940 FOR I%=1 TO TABUZ%:GOSUB 50530:PRINT CHR$(124);:NEXT I%:PRINT
50950 FOR I%=1 TO TABUZ%:GOSUB 51530:PRINT STRING$;:NEXT I%:PRINT
50960 FOR I%=1 TO TABUZ%:GOSUB 50530:PRINT I%*(-1.0);:NEXT I%:PRINT
50970 FOR I%=1 TO TABUZ%:GOSUB 50530:PRINT I%;:NEXT I%:PRINT
50980 FOR I%=1 TO TABUZ%:GOSUB 50530:I=I%:PRINT I;:NEXT:PRINT
50990 REM GOTO 50940
51000 REM GOSUB 50240:REM WIPE
51010 POKE WRITER,SWOFF
51020 DATA 10,20,30,40
51030 TABUZ%=4:TABU$="":TABULA$="":DIM TABULA%(TABUZ%)
51040 PRINT "NEW TAB SETTING";:FOR I%=1 TO TABUZ%:READ TABULA%(I%):PRINT TABULA%
(I%);:NEXT:PRINT
51050 FOR I%=1 TO TABUZ%:TABU$=TABU$+CHR$(TABULA%(I%))
51070 TABULA$=CHR$(27)+"D"+TABU$+CHR$(0)
51080 PRINT "NOW INPUT A INTEGER NUMBER 1 TO 5 DIGITS"
51085 POKE WRITER,SWON:PRINT TABULA$
51090 FOR J%=0 TO 2:FOR I%=1 TO TABUZ%:GOSUB 51900:REM INPUT
51100 PRINT CHR$(124);SPC(LEER);NBR%;:NEXT I%:PRINT CHR$(124.0)
51110 NEXT J%
51120 POKE WRITER,SWOFF:GOSUB 50240:POKE V24,BAUD0:END
51500 REM SUBROUTINE STRING INPUT
51530 CX=CURX:CY=CURY:PRI=PEEK(WRITER):POKE WRITER,SWOFF
51535 CURSOR 0,1:PRINT SPC(58);:CURSOR 25,1
51540 INPUT "STRING ";STRING$
51550 WO=TABULA%(I%):IF WO<CX THEN WO=CX
51560 CURSOR CX,CY:POKE WRITER,PRI
51570 PRINT CHR$(9);:POKE WRITER,SWOFF
51580 PRINT SPC(WO-CX);:POKE WRITER,PRI
51590 RETURN
51700 REM FORMATTING
51710 NBR%=INT(NBR)
51750 LE=7.0
51760 IF NBR%<1E5 AND NBR%>(-1E5) THEN LE=6.0
51770 IF NBR%<10000.0 AND NBR%>(-10000.0) THEN LE=5.0
51780 IF NBR%<1000.0 AND NBR%>(-1000.0) THEN LE=4.0
51790 IF NBR%<100.0 AND NBR%>(-100.0) THEN LE=3.0
51800 IF NBR%<10.0 AND NBR%>(-10.0) THEN LE=2.0
51820 RETURN
51900 CX=CURX:CY=CURY:PRI=PEEK(WRITER):POKE WRITER,SWOFF
51910 CURSOR 0,1:PRINT SPC(58);:CURSOR 20,1
51920 INPUT "INTEGER ";NBR:GOSUB 51700
51925 IF I%=TABUZ% THEN COLWID=12.0:GOTO 51940:REM LAST COLUMN =12
51930 COLWID=TABULA%(I%+1.0)-TABULA%(I%):REM COLUMNWIDTH
51940 LEER=COLWID-LE-2.0:REM 1+CHR$(24) & 1 SPACE
51970 POKE WRITER,PRI:CURSOR CX,CY
51980 PRINT CHR$(9);
51990 RETURN
51995 REM END OF PROGRAM
END PROGRAM
```

SPL
UN ASSEMBLEUR POUR LE DAI PC

LE logiciel SPL, comme disent les initiés, est un des cinq éditeurs-assembleurs de langage-machine qui « tournent » sur le DAI PC. Sans doute est-il le plus puissant, car il se veut très proche de MACRO 80, un assembleur tout à fait professionnel.

SPL est livré sous forme de cassette audio (ou micro-cassette numérique) avec plusieurs utilitaires, dont DISPLAY, un très puissant désassembleur et TRANSLATOR, qui permet de récupérer des sources écrites avec les autres assembleurs. Voilà qui est bien pratique. La notice, en français, est longue de 34 pages ; c'est dire que les commandes sont nombreuses et qu'il convient de lire et relire le texte avant de prétendre maîtriser ce logiciel. Cela étant, *SPL* offre une grande souplesse d'emploi.

Le droit à l'erreur

D'abord, sa façon de compacter le fichier-source autorise la compilation de programmes très longs, avantage décisif pour toute application « sérieuse » : on peut assembler en une seule fois jusqu'à 12 Koctets de codes-machine ! Ensuite, l'écriture de la source sous éditeur plein écran (avec tabulation automatique pour chacune des zones labels, opérandes, etc.) simplifie grandement le travail : il est possible de déplacer le texte-source dans toutes les directions, de se positionner n'importe où pour corriger.

Le premier déverminage (ou débogage) se fait en sortie d'édition : en cas d'erreur de syntaxe, *SPL* réédite la source à partir de la ligne fautive, et le curseur est remplacé par une lettre clignotante mnémonique du type d'erreur détecté. Ce système fait gagner beaucoup de temps lors de l'écriture d'un programme. Les possibilités de travail sont très grandes, et il est hors de question, en quelques lignes, de les passer toutes en revue. Les grands « classiques » sont là, avec souvent un « plus » : copie ou déplacement de paragraphes, recherche ou remplacement d'étiquettes, paramétrage de la liste (nombre de lignes par page; affichage en décimal, hexadécimal, octal, binaire, au choix, des adresses et/ou des opérandes). J'ai beaucoup apprécié la possibilité de demander un segment de liste ou d'édition par numéro de ligne ou par étiquette. Ainsi, la commande L 10 FIN déclenche-t-elle la liste de la source de la ligne 10 jusqu'à l'endroit où se trouve l'étiquette FIN. L'inconvénient d'une telle richesse est l'abondance des commandes : il vaut mieux garder le manuel près de soi pour s'y retrouver, du moins au début !

La liste 1 montre un exemple de source *SPL*, utilisant l'assemblage conditionnel, la gestion dynamique d'étiquettes, les macro-instructions, avec ou sans passage de paramètres. Ce programme est destiné à mesurer le temps que le Dai met à parcourir 255 instructions NOP (1). Les lignes 21 à 25 montrent comment expanser 255 instructions en peu de place : c'est l'assembleur qui fait la boucle en gérant dynamiquement l'étiquette COMPT.

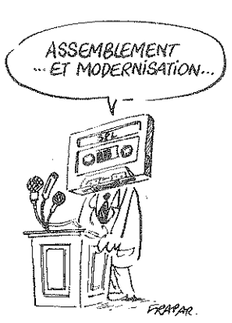
Le début de liste (lignes 9 à 14) est un exemple d'assemblage conditionnel, utile pour faire plusieurs versions d'une même source. Ici, l'étiquette VERS (version) vaut 1, le début du code-machine sera compilé à partir de 400 (hex), sinon, à partir de 500 (hex). Ceci peut être encore affiné, car *SPL* permet les tests booléens sur les étiquettes, du genre IF VERS = 1 AND LABEL = 2 OR VERS < > 5, etc. Les macro-instructions peuvent aussi passer des paramètres (exemple : store passe l'adresse RESULT dans l'étiquette RE lors de l'expansion de la macro). La ligne 2 montre comment donner des directives à l'imprimante, lors du listage de la source. Ici, on commande le passage en écriture grasse sur une Epson.

L'Assembleur n'est pas tout

L'utilitaire DISPLAY, livré avec *SPL*, ajoute des fonctions à l'assembleur : il permet d'obtenir la liste hex et ASCII d'une zone mémoire. Il désassemble aussi un code-machine en recréant une source utilisable par *SPL*, avec des étiquettes, s'il vous plaît ! Sur la liste 2, on voit comment apparaît le désassemblage de la routine « sortie série » du Dai. Remarquez les étiquettes placées aux points de branchement de la routine. DISPLAY peut désassembler *SPL* lui-même (12 Ko de codes-machines !) et le tout (code et source) tient encore en mémoire vive. Enfin,

*(1) Si vous désirez savoir combien de temps le Dai met à parcourir 255 NOP, faites, sous Basic : PRINT (#FFFF - PEEK (#300) - (PEEK (#301) *256)) *20 ; "millisecondes".*

En fait, ce n'est qu'à partir de plusieurs milliers de NOP que la décrémentation devient visible.



Liste 1 : programme-source obtenu avec SPL

```
1       ;
2               PRT     1BH,45H         ;directives imprimante
3       ;
4               TITL    ESSAI SPL
5       ;
6               PUT     "H"             ;affichage hexadecimal
7               ORG     300H
8       TEMPS   DW      0FFFFH          ;initialisation
9       VERS    SET     1H              ;version 1
10              IF      VERS=1H
11              ORG     400H
12              ELSE
13              ORG     500H
14              ENDIF
15      ;
16      TIMER   EQU     1BEH            ;horloge du DAI
17      RESULT  EQU     300H            ;tampon du resultat
18      ;
19              push                    ;appel de macro
20              time
21      COMPT   SET     0FFH            ;label gere dynamiquement
22      loop    ###                     ;pt d'entree de l'expansion
23              NOP
24      COMPT   SET     COMPT-1H        ;decrement du label
25              loop    COMPT>0H        ;boucle tant que label>0
26      ;
27              store   RESULT          ;macro avec parametre
28              pop
29              RET
30      FIN     END
31      ;
32      ;zone des macro-instructions
33      ;
34      push    MACRO
35              PUSH B
36              MEND
37      time    MACRO
38              LHLD    TEMPS
39              SHLD    TIMER
40              MEND
41      store   MACRO   RE              ;passage du parametre
42              LHLD    TIMER
43              SHLD    RE
44              MEND
45      pop     MACRO
46              POP B
47              MEND
48      ;
```

Liste 2 : exemple de désassemblage d'une routine

```
0000            ;
0000            ;
0000                    PRT     1BH,45H
0000            ;
0000                    ORG     0DD94H
DD94  F5        LHDD94  PUSH PSW
DD95  3A00FD    LHDD95  LDA     LHFD00
DD98  E608              ANI     8H
DD9A  CA95DD            JZ      LHDD95
DD9D  3AF3FF    LHDD9D  LDA     LHFFF3
DDA0  E610              ANI     10H
DDA2  CA9DDD            JZ      LHDD9D
DDA5  F1                POP PSW
DDA6  32F6FF            STA     LHFFF6
DDA9  FE0D              CPI     0DH
DDAB  C0                RNZ
DDAC  F5                PUSH PSW
DDAD  3E0A              MVI A   0AH
DDAF  CD94DD            CALL    LHDD94
DDB2  F1                POP PSW
DDB3  C9                RET
DDB4  à=FD00 LHFD00     EQU     0FD00H
DDB4  à=FFF3 LHFFF3     EQU     0FFF3H
DDB4  à=FFF6 LHFFF6     EQU     0FFF6H
DDB4                    END
```

LES COUPS D'ŒIL DE LIST

SPL, UN ASSEMBLEUR POUR DAI

l'utilitaire IMPLEM permet de paramétrer *SPL*, si les options par défaut ne vous conviennent pas, et ce, sans POKEs fastidieux.

La fonction LINK (« accrochage », lors de la compilation, de sous-routines en bibliothèque) est absente de *SPL*. Elle est remplacée par un MERGE (commande Y) qui inclut dans la source de travail des segments provenant de la mémoire de masse.

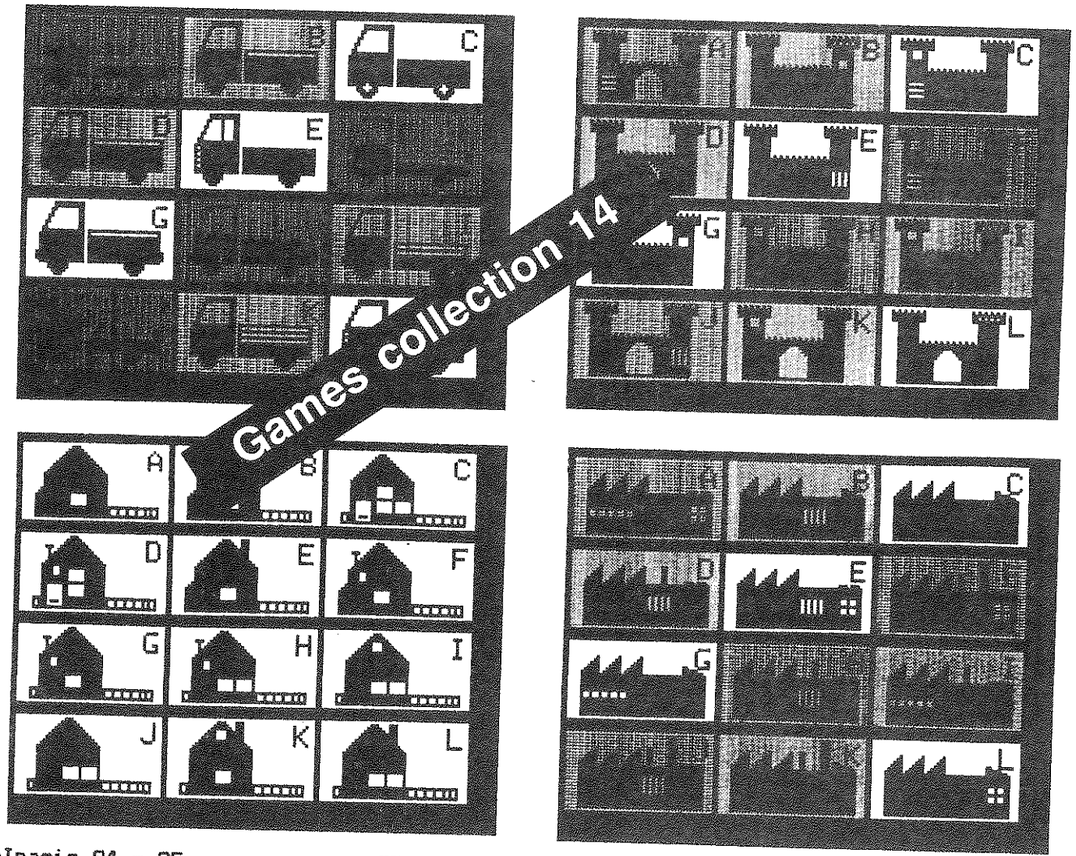
L'interactivité de *SPL* se traduit en (nombreux !) messages signalant telle ou telle anomalie : fautes de syntaxe, fautes de structure (lors de la compilation), fautes d'introduction (commande erronée), etc. De plus, à chaque retour à *SPL*, un *check sum* de la source, du code et de *SPL* lui-même vérifie l'intégrité de ce qui est en mémoire (vous savez : un programme « pas tout à fait au point » et que l'on essaie, peut occasionner des ravages sournois !).

Voilà donc un assembleur très réussi et puissant qui ne dépayserait pas un utilisateur venant de machines n'ayant rien à voir avec un micro-ordinateur. Associé à *DDT (DAInamic debugging tool,* un excellent utilitaire de mise au point), *SPL* se devrait de figurer dans la panoplie de tout Daïste féru de langage-machine.

Alain MARIATTE

Le logiciel en quelques lignes

Nom : *Assembleur 8080*
Ordinateur : *Dai PC et Dai T (version professionnelle)*
Forme : *cassette audio ou cassette numérique*
Edité et distribué par : *Dainamic Mottaart - 20 3170 Herselt Belgique*
ou : *Dainamic France - 9, rue Lavoisier, 59140 Dunkerque*
Prix public : *1 100 F belges (165 F français, environ)*
Orientation principale : *assemblage de langage-machine*
Autres orientations : *désassembleur, translateur de sources, dump-mémoire*



DIM STATEMENT

In the DAI-manual (Basic version V1.0), it is stated that the maximum dimension which can be declared is 254. But if in a program a dimension > 254 is used, no error report occurs. A DIM N(2,255) will be executed, appearently without any problems.

But is seems only without problems. As soon as the array is filled with data, the problem will show up. A value declared to N(1,0) or to N(2,0) will be found in N(0,0) too !! What happened ?

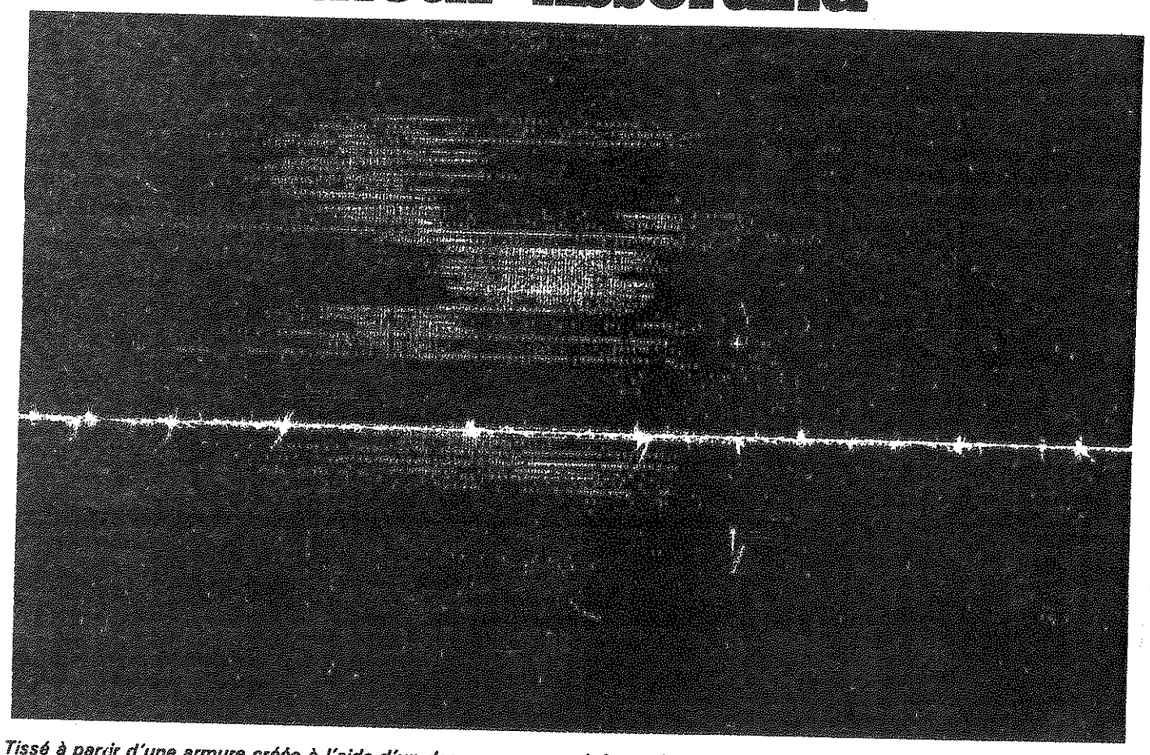
If we examine the pointer to N(1,0) via HEX$(VARPTR(N(1,0))), we will find the same VARPTR as for N(0,0) ! The DAI makes a big mess with the data stored in this array. Effectively, it has declared an array N(0,254), and the high order subscripts point to the same memory area.

The Basic version V1.1 enables the dimensioning of an array with the maximum subscript 255. So programs developped on a V1.1 machine may cause problems on a V1.0 machine !

© - Jan Boerrigter - Jan.1983

TISSUS ARMURÉS

L'ordinateur tisserand



Tissé à partir d'une armure créée à l'aide d'un des programmes informatiques, un échantillon où apparaît le jeu des fils pris et laissés.

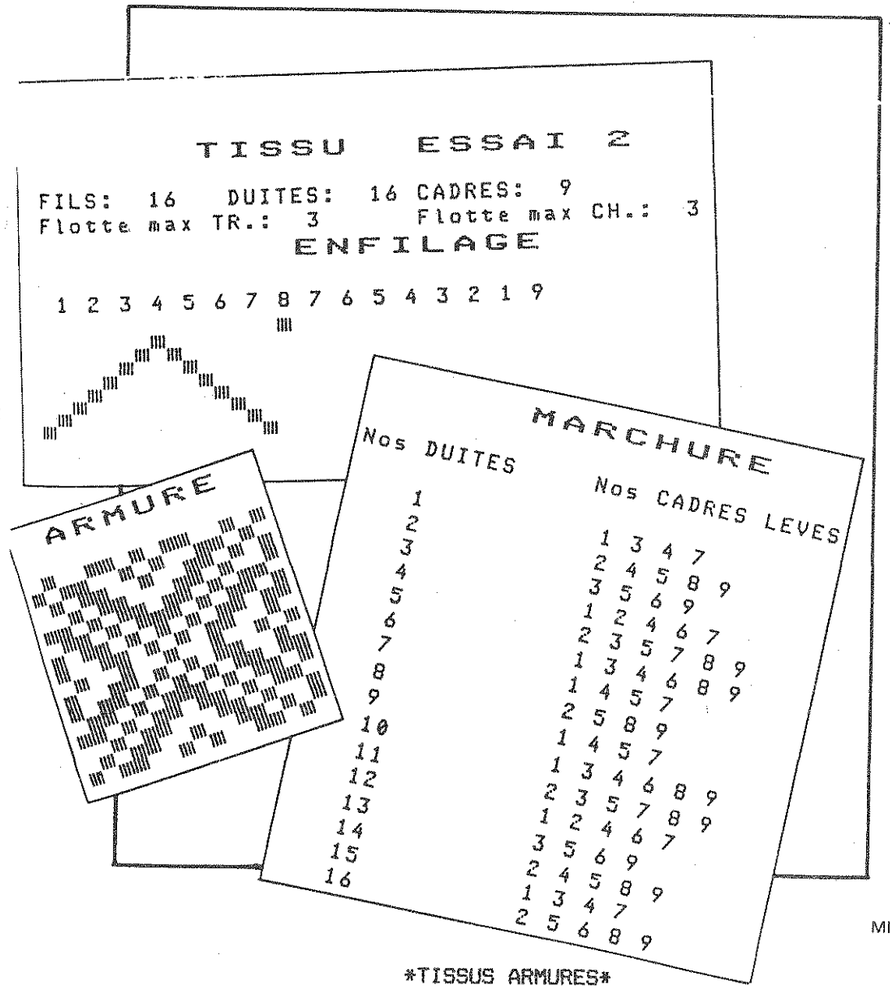
Le tissu imprimé, plus facile et moins cher à réaliser, a supplanté de nos jours le tissu armuré qui tire son relief du mode d'entrecroisement des fils. Celui-ci revient à la mode dans les haut de gamme des textiles d'ameublement.
La manière d'entrelacer fils de chaîne et fils de trame procède d'une certaine répétition. C'est ce qui a amené *Rémi Prin,* créateur textile et ingénieur informaticien, à créer des programmes de constitution d'armures. A partir des résultats obtenus, *Monique Prin* et *Rémi* tissent des échantillons que des industriels textiles vont ensuite utiliser pour leur fabrication.
Deux sortes de programmes ont ainsi été mis au point : l'un part d'un motif de base et lui fait subir toute sortes de déformations, c'est donc un programme *analytique* de traitement de texture. L'autre joue avec les paramètres constituant l'armure. Pour comprendre les manipulations réalisées, il faut connaître les opérations essentielles de tissage (voir encadré).
Rémi Prin travaille sur un micro-ordinateur type *DAI* de 48 K caractères de mémoire pour l'utilisateur. Les programmes sont écrits en langage de programmation appelé BASIC VI, sur cassette. L'unité centrale est reliée à un écran TV couleur équipé d'un câble de péritélévision. La partie édition de chaque programme est écrite pour adaptation à une imprimante graphique (de type *AXIOM* IMP 2-Q).
Appelé "Textile 3A", ce programme part donc du motif de base et comporte huit fonctions :

• Après appel de la fonction 1 "Entrée armure de base", l'opérateur définit les dimensions de l'armure (nombre de fils (1) et de duites) qui est saisie graphiquement.

• La fonction 2 permet de modifier l'armure précédente par retrait ou ajout soit de fils, soit de duites, toujours en saisie graphique, les fils de chaîne apparaissant en bleu foncé sur le rectangle bleu ciel aux dimensions de l'armure. Les laissés restent bleu clair.

• A l'appel de la fonction 3 d'analyse/édition, l'ordinateur effectue l'analyse de l'armure et édite à l'écran, ou sur imprimante au choix, les données suivantes : nombre de fils, nombre de duites, nombres de cadres nécessaires au tissage, longueur des flottés maximum en chaîne et en trame, l'enfilage, la marchure et enfin l'armure.

Rémi Prin élabore une nouvelle armure au clavier de son Daï.



Edition à l'imprimante des différentes caractéristiques d'une armure créée à l'écran.

• La fonction 4 donne l'aspect du tissu. Elle offre trois possibilités de résolution graphique, G pour 72 × 65 points, M pour 160 × 130 points et F pour 336 × 256 points. Second choix : les couleurs (4 parmi 16 disponibles). Troisième choix : la hauteur du dessin écran. Quelques répétitions de motifs étant suffisantes au début de l'élaboration d'un tissu, on peut l'afficher sur plein, moitié, quart d'écran..., etc. On peut aussi afficher une armure équilibrée (50/50) ou à chaîne élargie (ch. 100/tr. 50) ou trame élargie (ch. 50/tr. 100).

• La fonction 5 de "traitement de l'armure" offre des possibilités intéressantes de traitement du motif ou d'agrandissement du motif de base. Quatre zones de travail sont définies avec, pour chacune, onze traitements possibles dont des rotations de 90, 180 ou 270°, des symétries verticales ou horizontales, des décalages dans les quatre directions ou une inversion des pris et des laissés.
On peut revenir au début de cette fonction cinq, plusieurs fois de suite, ce qui multiplie la capacité de traiter des armures plus complexes.

• En appelant la fonction 6 "Rappel armure de base", la machine stocke deux armures en même temps sur la cassette digitale rapide (la dernière effectuée, plus l'armure de base).

• La fonction 7 permet d'agrandir la longueur d'une armure sans augmenter le nombre de lames nécessaires au tissage.

de duites (maximum 255) du motif de base qui va se répéter ? Après chaque passage en mode graphique, il est possible de modifier manuellement par insertion, suppression ou ajout de fils ou de duites (les touches M et D permettent de monter ou descendre directement en diagonale).
La sous-fonction 5 au lieu de faire subir un traitement à l'armure,

haute résolution, certaines routines ont été réalisées en langage "bas niveau" dit assembleur.
Rémi Prin devrait élaborer une version encore plus perfectionnée qui fusionnera ces deux sortes de programmes, mais elle nécessitera de posséder un double lecteur de disquettes pour augmenter les capacités mémoires insuffisantes sur cassette.

LES TISSUS ARMURÉS

La chaîne est le sens de la longueur du fil. Ce sont les fils de chaîne qui sont tendus entre les deux extrémités (ensouples) du métier à tisser. Ils passent, au milieu, par une série de cadres parallèles. Sur chaque cadre sont fixés des fils métalliques avec un œillet central où passe un fil de chaîne. Certains fils passent par les œillets du cadre n° 1, d'autres par le n° 2, etc. Un métier à tisser à main peut posséder jusqu'à 24 cadrans. Si on lève par exemple les cadres n° 1 et 5, la nappe de fils de chaîne se sépare en deux sous-nappes. Celle du dessus est constituée de tous les fils de chaîne soulevés, celle du dessous de tous les autres fils. La 1re opération de tissage consiste à passer entre ces deux nappes, une cannette, ou navette, où est enroulé le fil de trame. Ce coup de tissage (qui constitue la largeur du tissu) s'appelle une duite. On tasse ce fil de trame avec un peigne. Au second passage, on abaisse et on lève d'autres cadres au choix. Les fils de chaîne levés sont considérés comme pris. Les fils de chaîne de la nappe inférieure sont les "laissés".

On voit qu'on peut agir : sur le nombre de cadres (encore appelés lames), sur la manière d'enfiler les fils dans les cadres et enfin sur l'ordre successif des levers de cadres.

Si on a un fil pris, cinq laissés, un pris, cinq laissés, les 5 pris constituent un flotté, soit la longueur du fil de trame qui passe au-dessus d'un nombre donné de fils de chaîne (ici 5). (Certains flottés peuvent se retrouver sur l'envers ou être des flottés de chaîne.)

Suivant la répartition des pris et des laissés et compte tenu d'un enfilage donné, on va obtenir une armure bien définie, l'armure étant la représentation de l'entrelacement des fils. Cette suite de combinaisons (où si l'on veut la marche à suivre) constitue la marchure, et chacune de ces combinaisons, une marche. Chaque marche équivaut aussi à une pédale, qui dans les métiers plus perfectionnés va actionner la levée et la baisse d'un certain nombre de cadres auxquels elle est fixée par des attaches. Le bref représente le motif de l'armure.

• La fonction 8 est d'anayse-édition. Elle sort sur imprimante l'armure réalisée (voir schéma).

CRÉATION SYNTHÉTIQUE

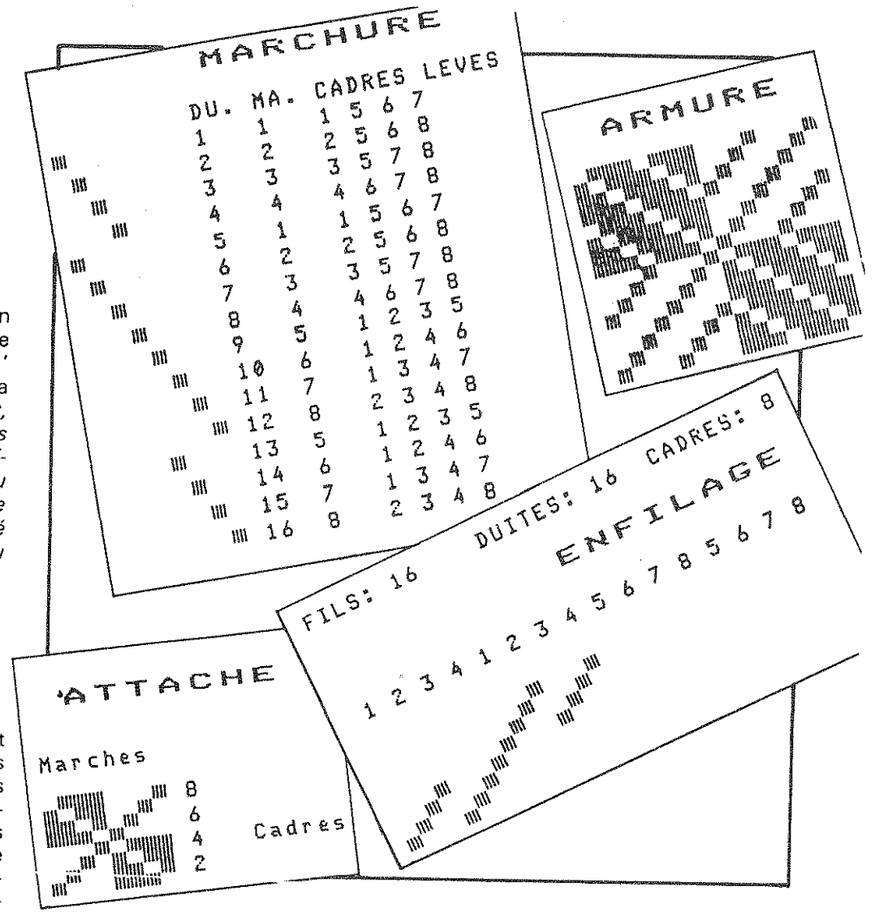
Le programme "TEXTILE 4", lui, est orienté vers la création synthétique, à partir des données de l'enfilage et de la marchure. Il possède six fonctions. La fonction 1 de "création/modification texture qui comporte sept sous-menus, pose d'abord à l'utilisateur des questions de définition de l'armure. Combien de cadres ? (maximum 24). Combien de fils ? (maximum 255). On obtient dans un premier temps l'affichage de l'enfilage et le pavé rectangulaire correspondant à la dimension choisie. Puis, combien comme dans le programme de TEXTILE 3, va agir sur le programme de tissage. Deux possibilités : traiter le tissu par groupes de fils ou par zones (cinq sont prévues). L'un des intérêts de TEXTILE 4 est de pouvoir conserver l'enfilage fixe si on le désire (toujours très long à réaliser sur les métiers à tisser) et de modifier la marchure.
Puis on revient à la fonction 5 qui permet le chargement de la texture. Une référence est attribuée au tissu. Il y a donc possibilités de constituer une bibliothèque d'armures. Pour chaque armure, la capacité mémoire nécessaire pour le stockage est donnée par le produit du nombre de fils par le nombre de duites, qui ne peut pas dépasser 12 000. Pour accroître la rapidité de l'affichage graphique en

UN DÉLAI DE RÉALISATION RAMENÉ DE 5 A 1

Autre programme assez spectaculaire : un parcours circulaire utilise un algorithme de centrage qui va se structurer peu à peu. Par exemple, un point sur dix est fixé à chaque passage jusqu'à ce que l'algorithme soit complètement figé. Pour que l'œil discerne cet ordonnancement, il faut une rigueur mathématique stricte.
"En utilisant l'ordinateur, explique Rémi Prin, *le créateur de tissus est moins prisonnier du schéma classique des armures. Et la vitesse d'exécution est appréciable."* Un bon technicien peut arriver au même résultat

TISSUS ARMURES

MICRO PROGRAMMES



mais ses crayonnés lui prennent un temps fou. C'est ainsi qu'un autre programme "écossais-rayures" réduit le délai de réalisation dans la proportion de 5 à 1. *"Cependant,* reconnaît-il, *l'informatique n'est pas tout : on réalise aussi un travail plastique indépendant de l'utilisation du micro-ordinateur. Il y a toute une recherche de matière due à la densité des fils qui décale l'aspect visuel du tissu."*

DU TEXTILE A LA CRÉATION ARTISTIQUE

Le grand danger, selon lui, serait qu'industriels et créateurs textiles récupèrent l'outil informatique dans l'état actuel sans essayer d'en développer les potentialités. Avec des moyens appropriés, il serait possible d'effectuer des recherches approfondies sur le langage textile, en remettant en cause les classifications traditionnelles des armures très rigoureuses. Lorsqu'il s'agissait, autrefois, de tisser des étoffes d'armures compliquées, il fallait bien connaître les modes d'enfilage et de flottement précis pour que le tissu tienne bien. De trop grands flottés, ou des irrégularités importantes dans la répartition des points d'aiguillage (l'endroit où les fils se croisent) peuvent faire que le tissu poche ou soit peu solide. Et cela, l'ordinateur ne le dit pas. Tout au moins pas encore, car il serait possible de réaliser des programmes sur la densité à respecter. Sur les extensions envisageables de l'utilisation du micro, Rémi Prin est intarissable : *"L'idéal serait d'effectuer des recherches approfondies au niveau de la combinatoire, de l'armure et du langage textile. Remettre en cause les classifications, ce qui implique une recherche informatique sur les réseaux des pris et des laissés. Un livre édité en 1938 à Lyon écrit par Brandon et Guillet traite de* "La méthode des initiales". *C'est l'approche du tissage à lames à l'aide d'un formalisme mathématique. Il permet de relier une forme continue d'une courbe que l'on désire obtenir dans le tissu et le caractère discontinu du tissage ; des contours analytiques à des réseaux de tissage. Peu de gens maîtrisent encore bien cette méthode. Il y aurait là des travaux informatiques approfondis à envisager."*
Rémi Prin rêve de recherches pluridisciplinaires dont le textile serait le nœud. Il a animé, en mars 1982, un stage de *"Micro-informatique en création textile"* au Centre Art et Industrie de Tourcoing. Un second est prévu à Chambéry en avril 1983 auprès d'un groupe de créateurs textiles de Savoie et d'Isère. Une semaine, sur les deux que dure le stage, sera réservée à un travail plastique autour de la technique de l'ikate (2) pour laquelle Rémi Prin a également créé un autre type de programme.
Préoccupations identiques pour une démonstration réalisée par Philippe Dujardin et Rémi Prin lors des trois journées "Micro-informatique et création textile", qui ont eu lieu aux Filteries DMC les 12, 13 et 14 janvier dernier. Organisées par la filothèque DMC et l'association Textile / Art / Langage, ces rencontres de créateurs et d'industriels ont permis d'envisager les développements possibles de ces recherches.

Dans l'industrie, il existe bien quelques systèmes informatiques, mais trop lourds et donc incomplètement exploités, ou qui sont utilisés pour déchiffrer des motifs existants et les reproduire. En somme un travail de compilation et non de création. C'est pourquoi, en France, n'existe-t-il quasiment pas de programme exploitable par les tisserands-artisans eux-mêmes. Rémi Prin a voulu combler ce manque avec ses différents programmes qu'il continue d'améliorer et qui coûtent 1 200 F chacun.
Quant à l'enseignement de l'utilisation du micro, il est dommage que seuls des rares stages ponctuels soient mis sur pied à l'heure actuelle. L'outil informatique au service du textile pourrait d'abord être utilisé avec profit dans les écoles d'arts appliqués en section textile.

Micheline DOMANCICH

(1) Fils : il s'agit des fils de chaîne quand ce n'est pas précisé.
(2) Ikate : les fils sont teints avant tissage. Sur chaque fil, il existe une fragmentation des couleurs par réserve. Le programme permet d'élaborer des contours des zones à colorier à partir de ces contours et de les mélanger.

TOP SECRET

JEAN GUERARD 6,rue du Parc 92190 MEUDON
tél: 626.34.18

Il vous est sûrement déjà arrivé de vouloir protéger un programme du piratage. Si l'on parvient à protéger un programme du BREAK, rien n'empêche de faire une copie intégrale de la cassette et d'utiliser ainsi son contenu.
Voici une routine qui laissera sur la cassette une suite d'octets sans signification, car ils sont codés à l'aide d'une clé, que vous serez -bien sûr- seul à connaitre.
Petite explication: pour coder un octet de programme, on lui fait subir la fonction XOR (OU exclusif) avec la clé.
Voici les caractéristiques de la fonction XOR:

0 XOR 0 -> 0
0 XOR 1 -> 1
1 XOR 0 -> 1
1 XOR 1 -> 0

On voit bien que, pour obtenir 0 ou 1, il existe *deux* possibilités différentes. Ainsi codé, le programme est donc inviolable, pour qui ne possède pas la clé.
La fonction XOR a ceci de sympathique que, pour retrouver le programme initial, il suffit de lui refaire passer l'encodage avec la clé. Autrement dit, appliquer la clé deux fois laisse le programme intact.

Marche à suivre pour coder:
.RESET
.LOAD pgm
.UT puis R (pour charger l'utilitaire de codage)
.G B000 (pour le lancer)
.ENTRER LA CLE, puis <RETURN>
.SAVE pgm basic codé

C'est tout ! Essayez donc de lister ensuite, pour voir !

Pour décoder un programme crypté, la marche à suivre est exactement la même. Il est -bien sûr- inutile de refaire le SAVE.

Pour vous rassurer quant à la sécurité d'encodage, sachez que l'utilitaire efface toute trace de la clé après codage ou décodage. De plus, avec 100 caractères au clavier, il existe 100^n clés possibles de n lettres !
Il faudrait une patience séculaire au pirate qui tenterait d'essayer les 10^32 clés de 1 à 16 lettres.

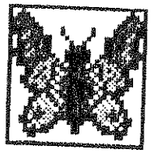
SUITE

```
B04E            ;*
B04E            ;*------ messages -----
B04E            ;*
B04E  5C      MESSG   DB      5CH             ;;nbr caracteres
B04F  0C0D            DB      0CH,0DH
B051  202050          DB      '  Pour coder ou decoder '
B069  6C6520          DB      'le programme en memoire,'
B081  0D              DB      0DH
B082  202065          DB      '  entrez la cle'
B091  206465          DB      ' de decryptage:'
B0A0  0D              DB      0DH
B0A1  202020          DB      '       >>> '
B0AB            END     END
```



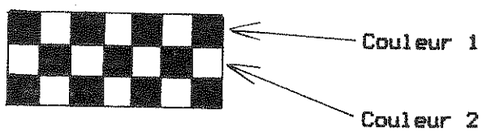
```
0000
0000            ;****************************************
0000            ;*          T O P - S E C R E T         *
0000            ;*     Pour proteger vos programmes     *
0000            ;*    (c) 1984 Jean GUERARD pour l'OI   *
0000            ;****************************************
0000            ;*
0000                    ORG     0B000H          ;;implantation
B000            ;*
B000            ;*-------- presentation -------
B000            ;*
B000  214EB0            LXI H   MESSG           ;;adr message
B003  CD32DB            CALL    0DB32H          ;;aff. message
B006            ;*
B006            ;*------ entree de la cle -----
B006            ;*
B006  à=F800 CLE        EQU     0F800H          ;;stockage cle
B006  0100F8            LXI B   CLE             ;;debut de la cle
B009  CDBBD6 GET        CALL    0D6BBH          ;;attente d'un
B00C  CA09B0            JZ      GET             ;;caractere.
B00F  EF                RST 5                   ;;affichage de
B010  03                DB      3H              ;;ce caractere.
B011  02                STAX B                  ;;stockage
B012  FE08              CPI     8H              ;;)decr. pointeur
B014  C21BB0            JNZ     TEST            ;;)si DELCHAR
B017  0B                DCX B                   ;;)est pressee.
B018  C309B0            JMP     GET
B01B  03      TEST      INX B
B01C  FE0D              CPI     0DH             ;;fin de la cle
B01E  C209B0            JNZ     GET
B021            ;*
B021            ;*--- preparation des adresses ---
B021            ;*
B021  2AA302            LHLD    2A3H            ;;fin de prgm
B024  EB                XCHG                    ;; dans DE
B025  21EC03            LXI H   3ECH            ;;debut de prgm
B028            ;*
B028            ;*--- codage / decodage du programme ---
B028            ;*
B028  0100F8 DEBUT      LXI B   CLE             ;;debut de la cle
B02B  0A     MASQUE     LDAX B                  ;;1 elt de la cle
B02C  AE                XRA M                   ;;<== C'EST ICI
B02D  77                MOV M,A
B02E  23                INX H
B02F  CD14DE            CALL    0DE14H          ;;HL=DE?
B032  CA3FB0            JZ      FIN             ;;codage termine
B035  0A                LDAX B                  ;;) si la cle est
B036  FE0D              CPI     0DH             ;;) finie,on
B038  CA28B0            JZ      DEBUT           ;;) recommence;
B03B  03                INX B                   ;;sinon,element
B03C  C32BB0            JMP     MASQUE          ;;suivant
B03F            ;*
B03F            ;*-- fin :effacer cle,retour au basic --
B03F            ;*
B03F  2100F8 FIN        LXI H   CLE             ;;debut de la cle
B042  3E80              MVI A   80H             ;;
B044  3600    EFFACE    MVI M   0H              ;;on efface la
B046  23                INX H                   ;;zone reservee
B047  3D                DCR A                   ;;a la cle,
B048  C244B0            JNZ     EFFACE          ;;puis
B04B  C3A0C7            JMP     0C7A0H          ;;retour au basic
```

TRAMES



L'utilisation des trames est trés en vigueur dans le dessin industriel, la photo ... et aussi l'informatique. En effet c'est un procédé qui, utilisé à bon escient sur votre DAI permet d'accroitre ses possibilitées

D'une part, une trame simple peut servir à fusionner deux couleurs. C'est, si elle est assez fine, au travers de cette trame que sera crée l'impression d'une troisiéme couleur , mixage des deux couleurs de bases de la trame. Ainsi une trame Rouge et Bleu donnera un effet Violet.
Le motif retenue pour la trame est le suivant :



C'est le quadrillage le plus fin que l'on puisse obtenir avec deux couleurs. Car c'est la finesse qui crée l'illusion c'est aussi pourquoi on emploiera , de préference, un mode graphique haute résolution tel que les modes 5,6,7 ou 8. A ce moment la trame devient moins percetible à l'oeuil et le fondu des couleurs est plus réaliste.

Le programme ci-dessous utilise les critéres sus-citée pour afficher les 120 couleurs secondaires, par l'utilisation de toutes les combinaisons possibles de trames bicolores.

IMPINT

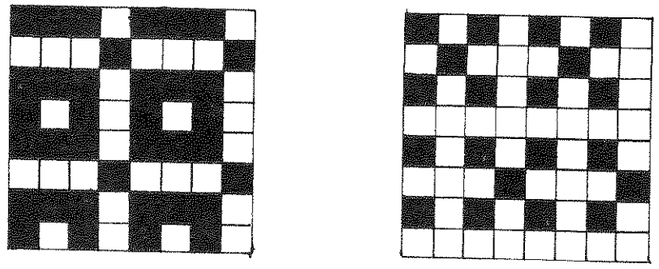
```
1       REM ************************
2       REM ** TRAMES - OCTOBRE 84 **
3       REM ************************
10      MODE 5:DIM A(15)
20      FOR I=0 TO 15 : READ A(I) : NEXT
30      FOR Y=0 TO 15
40      FOR X=0 TO 15
50      COL=A(Y)+16*A(X)
60      AD=#BC21+X*4-Y*1350
70      A=#55
80      FOR J=0 TO 15
90      FOR I=0 TO 3 STEP 2
100     POKE AD-J*90+I,A
110     POKE AD-J*90+I-1,COL
120     NEXT
130     A= INOT (A) IAND #FF
140     NEXT:NEXT:NEXT
200     DATA 0,4,8,15,14,10,3,6
210     DATA 12,1,9,5,13,11,2,7
220     REM                              C.D.
```


Le tramage des couleurs peut aussi se réaliser à partir de trois couleurs de bases, mais les choix des couleurs ainsi que celui de la trame devront etre judicieux afin d'avoir un mixage convenable.

D'autre part, une trame peut servir à representer une valeur ou une matiere. L'effet recherché est alors differents de celui de la fusion. Ici le fait que la trame soit visible ou non importe peu, ce qui compte c'est l'effet qui s'en dégage. On peut dés lors, dans les modes 'pauvres' en couleurs, tels les modes 6 & 8, faire appel à ces trames pour donner de trés nombreux effets differents.

Le programme suivant démontre mieux qu'un long discour ces effets de textures. Les trames employées ont été déssinées dans un cadre de 8x8, point par point.

Par exemple :



IMPINT

```
1       REM ************************
2       REM ** TRAMES - OCTOBRE 84 **
3       REM ************************
10      CLEAR 500:DIM A(11,7)
20      COLORG 0 1 9 12:MODE 6
30      FOR M=0 TO 11:FOR N=0 TO 7:READ A(M,N):NEXT:NEXT
40      FOR I=#BFEC TO #8000 STEP -720
50      FOR J=0 TO 68 STEP 2
60      FOR K=0 TO 7:POKE I-90*K-J,A(J/6,K):NEXT
70      NEXT:NEXT
1000    DATA #AA,#44,#AA,#00,#AA,#11,#AA,#00
1010    DATA #55,#EE,#55,#BB,#55,#EE,#55,#BB
1020    DATA #00,#55,#00,#AA,#00,#55,#00,#AA
1030    DATA #44,#40,#5F,#40,#04,#F5,#04,#44
1040    DATA #88,#44,#22,#11,#11,#22,#44,#88
1050    DATA #99,#66,#99,#66,#99,#66,#99,#66
1060    DATA #EE,#AA,#EE,#00,#BB,#AA,#BB,#00
1070    DATA #AA,#55,#AA,#55,#AA,#55,#AA,#55
1080    DATA #00,#77,#00,#BB,#00,#DD,#00,#EE
1090    DATA #AA,#99,#55,#66,#AA,#99,#55,#66
1100    DATA #EE,#11,#EE,#AA,#EE,#11,#EE,#AA
1110    DATA #FF,#FF,#FF,#FF,#FF,#FF,#FF,#FF
1120    REM                              C.D.
```

Vous voyez donc que meme avec deux couleurs seulement il est possible de representer :un mur puis une route, une veste et des chaussures ...! avec les trames les plus adaptées, et à chaque matiére sa trame.

Pour obtenir des resultats encore plus beau, vous pouvez mixer les deux methodes et créer des trames de matiéres multicolores. Ces méthodes sont trés interessante dans les mode 6 & 8 car il est trés facile d'y dessiner mais le nombre de couleurs est réduits à 4 (Par lignes).

Enfin pour ceux qui n'aiment pas les Pokes un procédé rapide d'obtention de trames multicolores est le tracé d'une diagonale sur deux. La trame obtenue ainsi est exactement la meme que celle decrite au début. C'est ce que démontre aussi ce dernier programme qui affiche les 6 trames bicolores possibles en mode 6.

IMPINT

```
1       REM ************************
2       REM ** TRAMES - OCTOBRE 84 **
3       REM ************************
10      COLORG 0 1 9 12:MODE 6
20      FILL 40,0 295,255 21
30      FOR I=40 TO 295 STEP 2
40      DRAW I,0 295,295-I 22:DRAW 335-I,255 40,I-40 22
50      NEXT
60      FILL 50,10 150,110 20
70      FOR I=50 TO 150 STEP 2
80      DRAW I,10 150,160-I 21:DRAW 50,I-40 200-I,110 22
90      NEXT
100     FILL 50,245 150,145 21
110     FOR I=50 TO 150 STEP 2:DRAW I,145 150,295-I 23:NEXT
120     FILL 285,10 185,110 22
130     FOR I=185 TO 285 STEP 2:DRAW I,10 285,295-I 23:NEXT
140     FILL 285,245 185,145 23
150     FOR I=185 TO 285 STEP 2:DRAW I,145 285,430-I 20:NEXT
160     COLORG 6 10 14 3
170     REM
                                                        C.D.
```

N'hesitez pas à me faire parvenir vos plus belles réalisations à partir des trames, ainsi que les trames de matiéres que vous avez construites. Bon 'tramages' !

Cédric DUFOUR

TRUCS

Voici deux petits trucs pour le DAI qui vous seront (je l'espère) très utiles.

1> Comment charger un programme BASIC en mémoire sans effacer celui qui est déjà en mémoire ?

C'est très simple : Vous devez tapez :

```
1 : LOAD <return>   (REM: Chargement du 1er programme)
2 : EDIT <return>   (REM: Ne surtout rien changer dans le programme !!!)
3 : BREAK + BREAK   (REM: Vous avez bien lu : taper BREAK puis BREAK)
4 : NEW <return>    (REM: OUI,j'ai bien dit 'NEW')
5 : LOAD <return>   (REM: Chargement du 2eme programme)
6 : POKE 309,2 <return>
7 : LIST <return> ==> Les deux programmes sont en mémoire !!!!
```

```
= N.B : Il faut évidement qu'aucun des numéros de lignes ne soient  =
=       identiques dans les deux programmes !!!)                    =
```

2> Comment supprimer une partie d'un programme ?

C'est également très simple : Vous devez taper :

```
1 : LOAD <return>          (REM: Charger un programme en mémoire)
2 : EDIT x-y <return>      (REM:les lignes de x à y sont celles que vous
                                désirez garder en mémoire.)
3 : BREAK + BREAK          (REM : sortir de l'éditeur)
4 : NEW <return>           (REM : efface le programe BASIC en mémoire)
5 : POKE 309,2 <return>    (REM: Fait passer la valeur du BUFFER d'édition
                                en mémoire)
6 : LIST <return> (REM : Dans la plupart des cas : OH ! ca marche !!!)
```

```
= N.B : Ce deuxième petit truc est nettement moins sur que le premier,il =
=       ne vaut mieux pas l'employer tout le temps !!!                   =
```

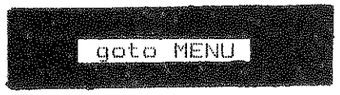
... A Bientot.....Marc Vandermeersch.....

SPECIAL LISTING

```
*********************************************************************************
***                                                                           ***
***      COMMENT INTRODUIRE UN CHR$() DANS UN  LISTING   ? ? ?                ***
***                                                                           ***
*********************************************************************************
```

Plusieurs d'entres-vous se sont, sans doute, déjà demandés comment on plaçait des caractères graphiques (= CHR$<>)dans un listing.

ex : 10 REM
20 REM
30 REM



En fait, il s'agit simplement de placer un CHR$(x) dans un"REM".
Pour plus de facilités, nous allons toujours travailler avec le CHR$(127) qui est comme tout le monde le sait, un petit carré noir.

Pour parvenir à entrer ce petit carré noir dans un "REM", il faudrait changer la valeur d'une touche du clavier (nous prendrons la touche 'A') par ce CHR$(127).

Voici la marche à suivre : il suffit de taper :

```
1 : UT                          <return> (REM: aller en UTILITY)
2 : ME8C5 E9C5 5000             <return> (REM: la mémoire clavier est
                                         maintenant placée en #5000
3 : B                                    (REM: retour au BASIC)
4 : POKE#2A7,0:POKE#2A8,#50 <return> (REM: Les 2 'POKE's en 1 seule ligne
                                         sinon on perd le controle du
                                         clavier)
```

Maintenant, nous avons donc déplacé l'espace mémoire du clavier de manière à ce que cet espace mémoire commence à partir de l'adresse hexadécimale #5000

Maintenant, la valeur ASCII de la touche 'A' se trouve à l'adresse #5011 (valeur ASCII de la touche 'B' = #5012 etc...)

Nous pouvons donc remplacer la valeur ASCII de la touche 'A' par un simple petit 'POKE'.

Si on tape "POKE#5011,127",la valeur de la touche 'A' sera le CHR$(127) soit le petit carré noir.(Pour ceux qui ne me croient pas, enfoncer la touche 'A' et vous verrez)

Voilà, avec cela, nous savons donc faire de beaux cadres.

Petite liste de CHR$() intérréssant pour ce truc:

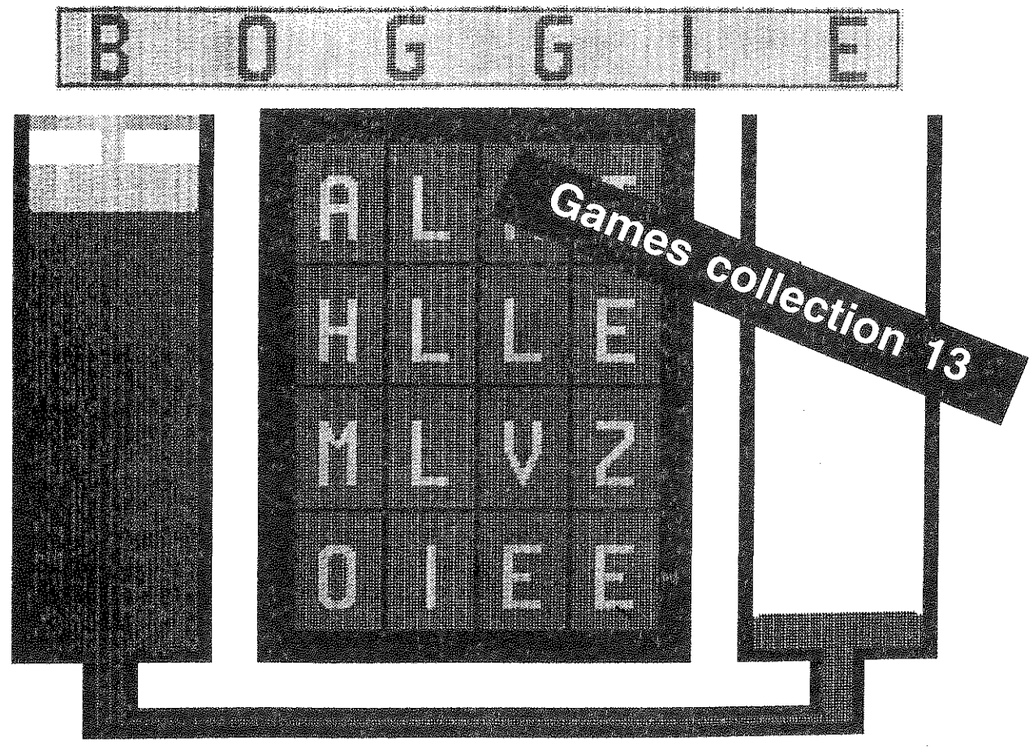
```
CHR$(29) = petites lignes verticales
CHR$(10) = ligne verticale
CHR$(11) = ligne horizontale
CHR$(30) = petite carré d'une autre couleur
CHR$(12) = vous ne verrez rien si vous poussez si la touche 'A' en mode
           commande, si vous passer en éditeur et que vous tapez 'A',vous
           verrez apparaitre une petite flèche vers le bas.
```

Si vous placez ce CHR$(12) dans un listing, (ex 1000 REM 'CHR$(12),vous verrez que dès que que vous demanderez à l'ordinateur un "LIST" de votre programme, il imprimera les lignes normalement à l'écran mais dès que il rencontre notre petite flèche vers le bas (CHR$(12), l'écran va s'effacer et l'ordinateur reprendra la suite du listing en haut de l'écran.

!!! Une chose à ne pas oublier : pour remettre la valeur ASCII de 'A'!!!
!!! sur la touche 'A',vous devz taper POKE #5011,#41 !!!

Il y a encore bien d'autres petits trucs à trouver avec ces CHR$()...

.....A Bientot.....Marc Vandermeersch.....



UTILITAIRE "DESSIN" SEMI-GRAPHIQUE 16 COULEURS

Cet utilitaire permet de "dessiner" en mode texte 16 couleurs en utilisant n'importe quel caractère semi-graphique du DAI .

Le caractère sera determiné par 3 conditions :

a)La couleur de son fond
b)La couleur du caractère
c)Son code ASCII

Le programme demande au départ sur combien de lignes vous voulez dessiner,la couleur de ces lignes (couleur fond du dessin) et enfin la couleur du cadre qui entoure le dessin. Il suffit ,si vous ne voulez pas de cadre, de le mettre de la couleur des lignes .

Vous aurez besoin d'un paddle 3 dimensions et d'un peu d'imagination !!!
PDL(0) et PDL(1) déplacent le caractère sur l'écran .
Appuyer sur le bouton du paddle fixera le caractère sur l'écran .
Mettre PDL(2) sur 255 change le caractère à afficher .
Le programme demande la redetermination du caractère
(Ne pas oublier de remettre PDL(2) sur 0)

Une fois le dessin terminé , appuyer sur BREAK,faire PRINT DEP$,taper UT et W DEP$ BFEF DESSIN.

Un exemple montrant l'utilisation de ce programme
Supposons que le dessin terminé aille de #B790 a #BFEF
Taper UT , MB790 BFEF 1000,W1000 185F dessin .
Il est alors possible d'afficher ce dessin trés rapidement grace au sous programme langage machine suivant

```
                        ORG     300H
0300    C5              PUSH B
0301    110010          LXI D   1000H
0304    216018          LXI H   1860H
0307    0190B7          LXI B   0B790H
030A    CD4FDE          CALL    0DE4FH
030D    C1              POP B
030E    C9              RET
                        END
```

Taper ce programme et faire CALLM#300

Cet utilitaire et le programme en langage machine ont servi à "dessiner" le terrain de jeu de la forteresse de ZLARG
Amusez vous bien et utilisez vos dessins dans vos jeux !!!

Patrick Pedelaborde

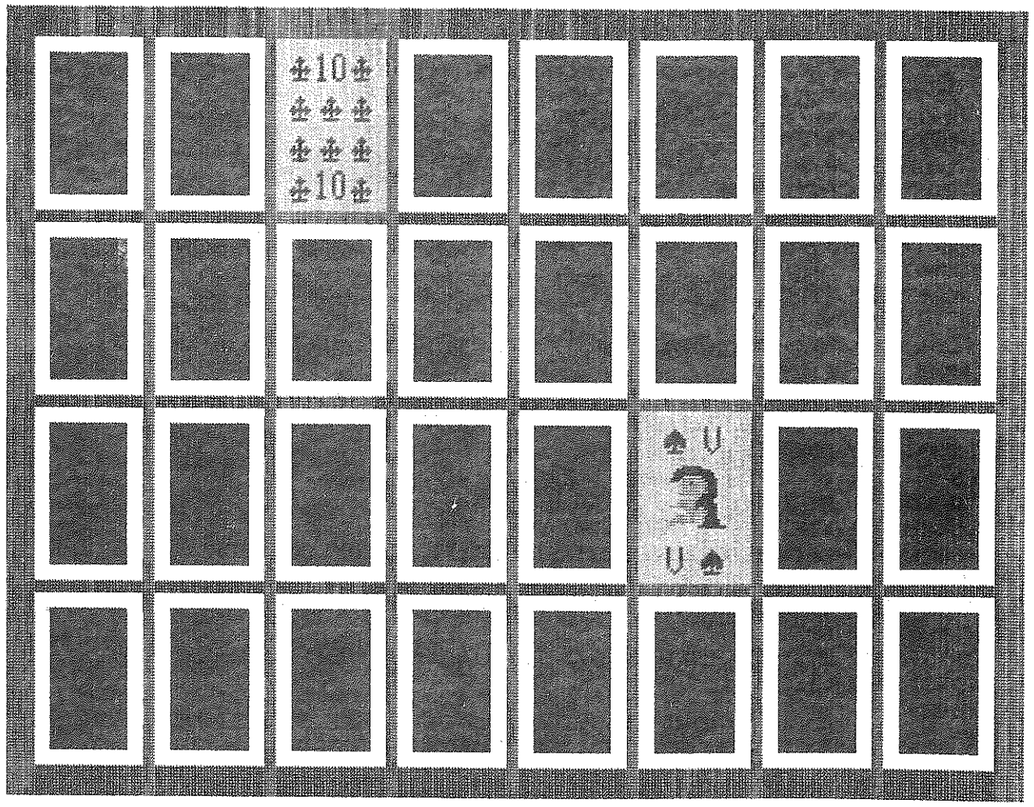
```
2   REM ****************************
3   REM * UTILITAIRE SEMI-GRAPHIQUE *
4   REM ****************************
5   REM
6   REM IMPINT
7   REM
10  MODE 0:PRINT CHR$(12):COLORT 8 0 0 0:POKE #75,95
20  CURSOR 2,20:INPUT "NOMBRE DE LIGNES (MAX20/MIN3)";NL:PRINT
30  IF NL<3 OR NL>20 THEN 20
40  CURSOR 2,19:INPUT "COULEUR DU FOND (DE 0 A 15)";CDF:PRINT
50  IF CFD<0 OR CFD>15 THEN 40
60  CURSOR 2,18:INPUT "COULEUR DU CADRE (DE 0 A 15)";CDC:PRINT
70  IF CDC<0 OR CDC>15 THEN 50
80  CURSOR 20,15:PRINT "C'EST PARTI !!!":WAIT TIME 50:PRINT CHR$(12);
90  DEP$=HEX$(#BFEF-NL*#86+1)
197 REM
198 REM ON MET LES LIGNES EN MODE 16 COULEURS
199 REM
200 FOR I=0 TO NL-1:POKE #BFEF-I*#86,#FA:NEXT
297 REM
298 REM DESSIN FOND ET CADRE AVEC COULEURS DEMANDEES
299 REM
300 FOR I=65 TO 3 STEP -1:A=#BFEF-(NL-1)*#86-I*2:POKE A,#FF:POKE A-3,CDC*16+CDC:NEXT
310 FOR K=1 TO NL-2
320 FOR I=65 TO 64 STEP -1:B=#BFEF-(NL-1-K)*#86-I*2:POKE B,#FF:POKE B-3,CDC*16+CDC:NEXT
330 FOR I=63 TO 5 STEP -1:B=#BFEF-(NL-1-K)*#86-I*2:POKE B,#FF:POKE B-3,CDF*16+CDF:NEXT
340 FOR I=4 TO 3 STEP -1:B=#BFEF-(NL-1-K)*#86-I*2:POKE B,#FF:POKE B-3,CDC*16+CDC:NEXT
350 NEXT
360 FOR I=65 TO 3 STEP -1:A=#BFEF-I*2:POKE A,#FF:POKE A-3,CDC*16+CDC:NEXT
397 REM
398 REM RESERVATION FENETRE
399 REM
400 FEN=#BFEF-NL*#86:POKE #8A,FEN MOD 256:POKE #8B,FEN SHR 8
497 REM
498 REM QUESTIONS
499 REM
500 CURSOR 0,23-NL:INPUT "COULEUR DU FOND (0 a 15)";CF:PRINT
510 IF CF<0 OR CF>15 THEN PRINT CHR$(12);:GOTO 500
520 CURSOR 0,22-NL:INPUT "COULEUR DU CARACTERE (0 a 15)";CC:PRINT
530 IF CC<0 OR CC>15 THEN 520
540 INPUT "CODE  ASCII DU CARACTERE (0 a 255)";ASCII
550 IF ASCII<0 OR ASCII>255 THEN 540
597 REM
598 REM DEPLACEMENT DU CARACTERE ET DESSIN
599 REM
600 LIGNE=#BFEF-(NL-1)*#86+INT(PDL(1)*(NL+1)/255)*#86
610 POSCAR=LIGNE-INT(PDL(0)/3.9)*2
620 X1=PEEK(POSCAR):Y1=PEEK(POSCAR-3)
630 X2=ASCII:Y2=CC*16+CF
640 POKE POSCAR,X2:POKE POSCAR-3,Y2:WAIT TIME 5
650 POKE POSCAR,X1:POKE POSCAR-3,Y1:WAIT TIME 5
660 IF PDL(2)>200 THEN PRINT CHR$(12);:GOTO 500
670 IF PEEK(#FD00) IAND #20=0 THEN 600
680 X1=X2:Y1=Y2:GOTO 650
```

```
10    REM ................................................
20    REM ..................... WEGWEISER ...................
30    REM ................................................
40    REM ..................... (c) 1984 by Rolf Schall ....
50    REM ................................................
100   REM ............... Erklaerung ........................
110   COLORT 8 0 0 0:MODE 0:PRINT CHR$(12)
120   PRINT TAB(16);"Wegweiserspiel V2.0":PRINT
130   PRINT "Aufgabe: Der gelbe Fleck ist durch ein Labyrinth von Raeumen"
140   PRINT "unterschiedlicher Farbe zu lenken."
150   PRINT "Folgen Sie mit Hilfe der  Cursortasten den Pfeilen  bis  zum"
160   PRINT "durch ein  Kreuz  gekennzeichneten Zielpunkt.  Versuchen Sie"
170   PRINT "danach, zum Ausgangsort mit moeglichst wenigen Schritten zu-"
180   PRINT "rueckzufinden.  Wenn Sie einen Raum verlassen haben,  werden"
190   PRINT "die Wegweiser weggewischt."
200   PRINT "Wenn Sie aufgeben wollen,  druecken Sie SPACE.  Es erscheint"
210   PRINT "dann der Grundriss des Labyrinth mit allen Durchgaengen."
220   PRINT :PRINT "Darin bedeuten:"
230   PRINT "  Schwarzer Fleck: Anfangs- und Zielpunkt"
240   PRINT "      Roter Fleck: Endpunkt"
250   PRINT "    Gruener Fleck: Ihr Standort"
260   PRINT "  Schwarze Punkte: Hinweg"
270   PRINT "    Weisse Punkte: Ihr Weg"
280   PRINT :PRINT TAB(16);"Bitte einen Moment warten..."
300   REM .......... Initialisierung ........................
310   CLEAR 2000:DIFFICULTY%=12:WIDTH%=10:HEIGHT%=8:DIM STATUS%(WIDTH%-1,HE
IGHT%-1),PASSAGE%(3)
320   DIM XDIR%(3),YDIR%(3),XBEG%(3),YBEG%(3),OPPOSITE%(3),XMOV%(3),YMOV%(3
)
330   YDIR%(0)=1:YDIR%(1)=-1:XDIR%(2)=-1:XDIR%(3)=1
340   OPPOSITE%(0)=1:OPPOSITE%(1)=0:OPPOSITE%(2)=3:OPPOSITE%(3)=2
350   FOR I%=0 TO 3:XMOV%(I%)=XDIR%(I%)*8:YMOV%(I%)=YDIR%(I%)*8:NEXT
360   GOSUB 1500:REM ........ Grundriss und Wegberechnung ..
370   PRINT TAB(16);"Spiel startet mit SPACE":CALLM #D6DA
380   REM ............. Init. der Bildkonstanten .............
390   MODE 6:COLORG 8 1 2 14
400   XMID%=XMAX/2+1:YMID%=YMAX/2+1
410   XS%=XMID%+8:YS%=YMID%+8
420   XBEG%(0)=XMID%-16:XBEG%(1)=XMID%+8:XBEG%(2)=XMAX-31:XBEG%(3)=32
430   YBEG%(0)=32:YBEG%(1)=YMAX-31:YBEG%(2)=YMID%-16:YBEG%(3)=YMID%+8
440   FILL 0,0 XMAX,YMAX 1:FILL 8,8 XMAX-8,YMAX-8 8
450   FILL XMID%-16,YMID%-16 XMID%+16,YMID%+16 8
460   REM ............. Spielbeginn ........................
470   RX%=STARTX%:RY%=STARTY%:STATUS%=STATUS%(RX%,RY%) IOR #8000
480   STATUS%(RX%,RY%)=STATUS%
490   GOSUB 1010:GOSUB 1200
600   REM ............... INKEY-Routine .....................
610   KEY%=GETC:IF KEY%=32 THEN 2000:IF KEY%<16 OR KEY%>19 THEN 610
620   KEY%=KEY%-16
630   FILL XS%,YS% XS%+7,YS%+7 8
640   XS%=XS%+XMOV%(KEY%):YS%=YS%+YMOV%(KEY%)
650   IF SCRN(XS%+4,YS%+4)<>8 THEN XS%=XS%-XMOV%(KEY%):YS%=YS%-YMOV%(KEY%)
660   FILL XS%,YS% XS%+7,YS%+7 14
670   IF XS%>7 AND XS%<XMAX-7 AND YS%>7 AND YS%<YMAX-7 THEN 610
800   REM ............... Wechsel des Raumes ................
810   DIRECTION%=KEY%:IF DESTINATION%=1 THEN STPS%=STPS%+1
820   RX%=(WIDTH%+RX%+XDIR%(DIRECTION%)) MOD WIDTH%:RY%=(HEIGHT%+RY%+YDIR%(
DIRECTION%)) MOD HEIGHT%
830   STATUS%=STATUS%(RX%,RY%):GOSUB 1000
840   IF STATUS% IAND #2400=#400 THEN GOSUB 1200
850   IF STATUS% IAND #100>0 THEN GOSUB 1300
860   IF STATUS% IAND #200>0 THEN GOSUB 1400
870   XS%=XBEG%(DIRECTION%):YS%=YBEG%(DIRECTION%)
880   FILL XS%,YS% XS%+7,YS%+7 14
```



```
890   STATUS%(RX%,RY%)=STATUS% IOR #2000:REM Raum betreten
900   GOTO 610
910   REM ................. Zurueckgefunden .................
920   PRINT "Schritte hin:";ROOMS%,"Zurueck";STPS%-1
930   END
1000  REM ................. Neuen Raum zeichnen .............
1010  OPPOSITE%=OPPOSITE%(DIRECTION%):COLORG 8 STATUS% IAND #F 2 14
1020  FOR I%=0 TO 3:PASSAGE%(I%)=(STATUS% SHR I%+4) IAND 1:NEXT
1030  FILL XMID%-16,YMAX XMID%+16,YMAX-7 21-PASSAGE%(0)
1040  FILL XMID%-16,0 XMID%+16,7 21-PASSAGE%(1)
1050  FILL 0,YMID%-16 7,YMID%+16 21-PASSAGE%(2)
1060  FILL XMAX,YMID%-16 XMAX-7,YMID%+16 21-PASSAGE%(3)
1070  FILL XMID%-8,YMID%-8 XMID%+8,YMID%+8 20
1080  RETURN
1200  REM ................ Wegweiser zeichnen ..............
1210  WAY%=(STATUS%(RX%,RY%) SHR 11) IAND 3
1220  WX%=XMOV%(WAY%):WY%=YMOV%(WAY%)
1230  DRAW XMID%-WX%,YMID%-WY% XMID%+WX%,YMID%+WY% 23
1240  DRAW XMID%-WY%/2,YMID%-WX%/2 XMID%+WX%,YMID%+WY% 23
1250  DRAW XMID%+WY%/2,YMID%+WX%/2 XMID%+WX%,YMID%+WY% 23
1260  RETURN
1300  REM ................ Start erreicht ..................
1310  FILL XMID%-4,YMID%-4 XMID%+4,YMID%+4 22:FILL XMID%-3,YMID%-3 XMID%+3,
YMID%+3 20
1320  IF DESTINATION%=1 THEN 910
1330  RETURN
1400  REM ................ Ende erreicht ...................
1410  DRAW XMID%-4,YMID%-4 XMID%+4,YMID%+4 22:DRAW XMID%+4,YMID%-4 XMID%-4,
YMID%+4 22
1420  DESTINATION%=1
1430  RETURN
1500  REM ................ Hauptdurchgang ..................
1510  STARTX%=RND(WIDTH%):STARTY%=RND(HEIGHT%)
1520  RX%=STARTX%:RY%=STARTY%:OPPOSITE%=5:ROOMS%=RND(DIFFICULTY%)
1530  STATUS%(RX%,RY%)=#100
1540  FOR I%=0 TO ROOMS%
1550  DIRECTION%=RND(4):IF DIRECTION%=OPPOSITE% THEN 1550:OPPOSITE%=OPPOSIT
E%(DIRECTION%)
1560  STATUS%(RX%,RY%)=STATUS%(RX%,RY%) IOR #400 IOR (DIRECTION% SHL 11) IO
R (16 SHL DIRECTION%)
1570  RX%=(WIDTH%+RX%+XDIR%(DIRECTION%)) MOD WIDTH%:RY%=(HEIGHT%+RY%+YDIR%(
DIRECTION%)) MOD HEIGHT%
1580  STATUS%(RX%,RY%)=STATUS%(RX%,RY%) IOR (16 SHL OPPOSITE%)
1590  NEXT I%:STATUS%(RX%,RY%)=STATUS%(RX%,RY%) IOR #200
1600  XEND%=RX%:YEND%=RY%
1800  REM ............. Durchgaenge & Farben ..............
1810  FOR I%=0 TO WIDTH%-1:FOR J%=0 TO HEIGHT%-1
1820  COL%=RND(16):IF COL%=8 THEN 1820
1830  DIRECTION%=RND(4):STATUS%(I%,J%)=STATUS%(I%,J%) IOR COL% IOR (16 SHL
DIRECTION%)
1840  RX%=(WIDTH%+I%+XDIR%(DIRECTION%)) MOD WIDTH%:RY%=(HEIGHT%+J%+YDIR%(DI
RECTION%)) MOD HEIGHT%
1850  STATUS%(RX%,RY%)=STATUS%(RX%,RY%) IOR (16 SHL OPPOSITE%(DIRECTION%))
1860  NEXT J%:NEXT I%
1870  RETURN
2000  REM ................. Plan ............................
2010  MODE 3
2020  FOR I%=0 TO WIDTH%-1:FOR J%=0 TO HEIGHT%-1:XI%=I%*16:XJ%=J%*16
2030  FILL XI%,XJ% XI%+15,XJ%+15 STATUS%(I%,J%) IAND #F
2040  FILL XI%+2,XJ%+2 XI%+13,XJ%+13 8
2050  NEXT J%:NEXT I%
2060  FOR I%=0 TO WIDTH%-1:FOR J%=0 TO HEIGHT%-1:XI%=I%*16:XJ%=J%*16:STATUS
%=STATUS%(I%,J%)
2070  IF STATUS% IAND 16>0 THEN FILL XI%+6,XJ%+14 XI%+10,XJ%+15 8
2080  IF STATUS% IAND 32>0 THEN FILL XI%+6,XJ% XI%+10,XJ%+1 8
2090  IF STATUS% IAND 64>0 THEN FILL XI%,XJ%+6 XI%+1,XJ%+10 8
```

```
2100  IF STATUS% IAND 128>0 THEN FILL XI%+14,XJ%+6 XI%+15,XJ%+10 8
2110  IF STATUS% IAND #400>0 THEN DOT XI%+8,XJ%+6 0
2120  IF STATUS% IAND #2000>0 THEN DOT XI%+8,XJ%+10 15
2130  NEXT J%:NEXT I%
2140  XI%=STARTX%*16:XJ%=STARTY%*16
2150  FILL XI%+2,XJ%+2 XI%+12,XJ%+12 0:REM ................ START
2160  XI%=XEND%*16:XJ%=YEND%*16
2170  FILL XI%+2,XJ%+2 XI%+12,XJ%+12 2:REM ................ ENDE
2180  XI%=RX%*16:XJ%=RY%*16
2190  FILL XI%+2,XJ%+2 XI%+12,XJ%+12 5:REM ............ STANDORT
2200  GOTO 2200
3000  REM ..................................................
3010  REM ............... BIT-Belegung STAT ..................
3020  REM .....  0- 3 : Farben ...........  4- 7 : Tueren ...
3030  REM .....     8 : Start ............     9 : Ende .....
3040  REM .....    10 : Weg ............... 11-12 : Richtung .
3050  REM .....    13 : Vom Spieler betretenes Feld .........
3060  REM ..................................................
```



Games collection 13

DBASIC part 4

DBASIC V2.2

1. A new version of DBASIC

In using DBASIC V2.1 you may have noticed the presence of some bugs, specially in error-trapping or extension-evaluation.
All the errors which have been detected and communicated to me have been corrected in DBASIC V2.2. Besides this corrections, some new statements have been implemented. Some of them will be mentioned later on in this article.

2. New extensions

Besides a new version of the DBASIC language, a few extensions are available. One of them generates a structured listing and/or cross-reference : this extension will be the main subject of this article.
Another extension allows you to define and use function keys. See PFK and PFK$ in the program listing.

3. Structured listing and cross-reference

The program listing on the next pages is an example of the output produced by the $XREF extension. By simply typing $LIST a structured listing of the complete program, followed by a cross-reference is produced.
Another command, XREF, will skip the structured listing and will directly output a cross-reference.

3.A Structured listing

As can be seen in the program listing, DBASIC program lines are spread out on several listing lines to enable correct indentation of :

- .iteration structures (ex. FOR/NEXT, REPEAT/UNTIL, WHILE/WEND).
- .selection structure (ex. IF/ELSIF(new in DBASIC V2.2)/ELSE/END IF).
- .definition structures (ex. PROCEDURE(DEF PROC)/END PROC, FUNCTION(DEF FN)/END FN).

A counter also shows the level of indentation.

Special care has been taken for some details :

- .line numbers are printed rigth justified
- .keywords, symbols or text will not be splitted at the end of the line
- .the title will be truncated if it is to long
- .pages are numbered

3.B cross-reference

A cross-reference could be a very usefull help in debugging your software. For clarity the DBASIC cross-reference groups all symbols into 5 different classes :

```
      .extended commands & functions..(X option)
      .procedures.....(P option)
      .functions.....(F option)
      .labels.....(L option)
      .arrays.....(A option)
      .variables.....(V option)
```

Special care has been taken for some details :

- for functions,arrays and variables the type is indicated
- for functions and arrays the () indicates the necessity of arguments
- line numbers are risth justified
- the first line number indicates the line where the symbol is defined (ex. procedures,functions,labels,arrays) or used the first time

3.C General syntax

The general syntax is :

```
$LIST[ first line][ last line][,title][;options]    and
 XREF[ first line][ last line][,title][;options]   with
```

```
first line : line number from where to start the structured listing
last line  : line number where to end the structured listing
title      : the heading on each page
options    : options for the cross-reference output
```

```
ex.  $LIST 100 500,"TEST PROGRAM";"FP"
```

This command asks for a structured listing starting with line number 100 up to line number 500 with heading TEST PROGRAM and produces a cross-reference of functions and procedures

3.D Default values

```
default range   : complete program
default header  : the string expression following the DBASIC TITLE
                  statement. If the program does not contain a TITLE state-
                  ment the default header is an empty string.
                  (note : the TITLE statement is also used as a default for
                  SAVE)
default options : "XPFLAV" i.e. first print the cross-reference of all
                  extended commands & functions, then the cross-reference
                  of all procedures...and so on...
```

4. Other DBASIC aprovements

Besides error correction and implementation of new statements, I have implemented extended functions in DBASIC V2.2 (see PFK$). The execution time of extended commands and functions has been diminished quite a bit due to the addition of an extra compilation pass. For more information see the documentation of DBASIC V2.2 and the extension's documentation.

Willy Coremans


```
DBASIC V2.2
PAGE      1   MANIPULATE PROGRAMMABLE FUNCTION KEYS


   10         ON BREAK GOTO "TRAPBREAK
   20         TITLE "MANIPULATE PROGRAMMABLE FUNCTION KEYS"
   30         REM to enable/disable function keys--type PFKON/PFKOFF
   40         REM to use function keys--type <ctrl> and 0 to 9 at the same time
  100         PROCEDURE HOME :
          1    PRINT CHR$(12);:
              END PROC
  200         PROCEDURE PFKTOARRAY  ARR A$:
          1    LOCAL I
  210     1    FOR I=0 TO 9:
          2     A$(I)=PFK$(I):
          1    NEXT
  220         END PROC
  300         PROCEDURE ARRAYTOPFK  ARR A$:
          1    LOCAL I
  310     1    FOR I=0 TO 9:
          2     PFK I,A$(I):
          1    NEXT
  320         END PROC
  400         PROCEDURE CLRPFK :
          1    LOCAL J
  410     1    FOR J=0 TO 9:
          2     PFK J,"":
          1    NEXT
  420         END PROC
  500         FUNCTION PRNTPFK$(J):
          1    LOCAL I:HOME
  510     1    FOR I=0 TO J:
          2     PRINT PFK$(I):
          1    NEXT
  520     1    FN = "":
              END FN
 1000        "START
              DIM AR$(9)
 1005         REPEAT
          1    HOME
 1010     1    CURSOR 10,20:PRINT "1--ARRAY TO FUNCTION KEYS"
 1020     1    CURSOR 10,19:PRINT "2--FUNCTION KEYS TO ARRAY"
 1030     1    CURSOR 10,18:PRINT "3--CLEAR FUNCTION KEYS"
 1035     1    CURSOR 10,17:PRINT "4--PRINT FUNCTION KEYS & END PROGRAM"
 1040     1    CURSOR 0,0:PRINT "SELECT CHOICE ";
 1050     1    REPEAT
          2     I=GETC-ASC("0"):
          1    UNTIL I>0 AND I<5
 1055     1    CURSOR 14,0:PRINT I;:WAIT TIME 10
 1060     1    IF I=1 THEN
          2     ARRAYTOPFK AR$()
 1070     1    ELSIF I=2 THEN
          2     PFKTOARRAY AR$()
 1075     1    ELSIF I=3 THEN
          2     CLRPFK
 1080     1    ELSE
          2     PRINT PRNTPFK$(9)
 1090     1    END IF
 1100         UNTIL I=4
 2000         END
 3000        "TRAPBREAK
              CONTINUE
```

extended commands & functions :

PFK	310	410
PFK$	210	510

procedures :

ARRAYTOPFK	300	1060	
CLRPFK	400	1075	
HOME	100	500	1005
PFKTOARRAY	200	1070	

functions :

PRNTPFK$()	500	1080

labels :

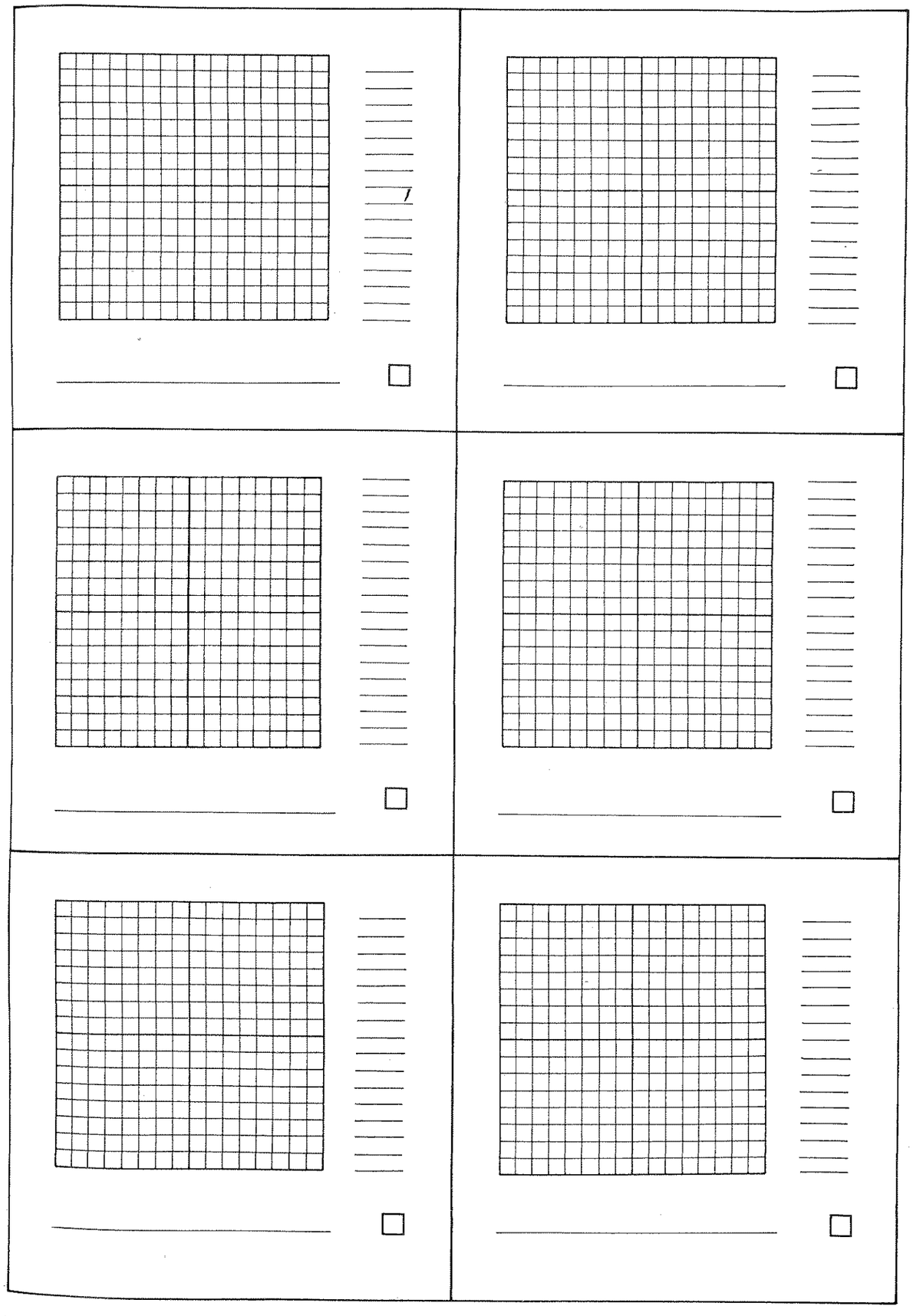
START	1000
TRAPBREAK	3000 10

arrays :

AR$()	1000	1060	1070	
A$()	200	210	300	310

variables :

I%	200	210	300	310	500	510	1050	1055	1060
	1070	1075	1100						
J%	400	410	500	510					



16 x 16 design