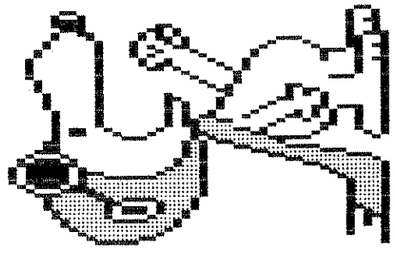
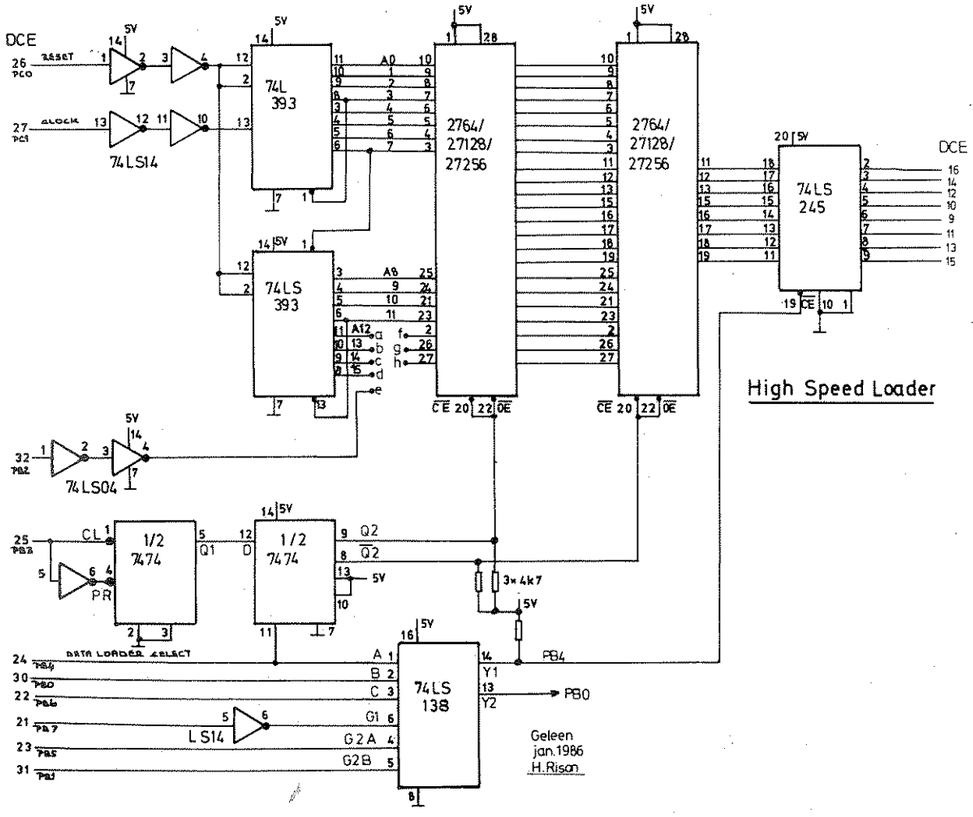


DAI NAMIC 36

DAI CLIC 9



tweemaandeljks tijdschrift sept.-okt. 86

bimestriel sept.-oct. 86

een uitgave van DAInamic VZW en IDC ASBL
une publication de DAInamic VZM et IDC ASBL
verantw. uitgever : w. hermans, mottaart 20, 3170 herselt

International

COLOFON

DAInamic verschijnt tweemaandelijks.
Abonnementsprijs is inbegrepen in de jaarlijkse contributie.
Bij toetreding worden de verschenen nummers van de jaargang toegezonden.

DAInamic redactie :

Dirk Bonné	wdw
Freddy De Raedt	Herman Bellekens
Wilfried Hermans	Frans Couwberghs
René Rens	Guido Goyvaerts
Bruno Van Rompaey	Daniël Goyvaerts
Jef Verwimp	Frank Druijff
Cedric Dufour	Willy Coremans

Vormgeving : Ludo Van Mechelen.

U wordt lid door storting van de contributie op het rekeningnr. **230-0045353-74** van de **Generale Bankmaatschappij, Leuven,** via bankinstelling of postgiro.
Het abonnement loopt van januari tot december.

DAInamic verschijnt de pare maanden.
Bijdragen zijn steeds welkom.

CORRESPONDENTIE ADRESSEN.

Redactie en software bibliotheek

Wilfried Hermans
Mottaart 20
3170 Herselt
Tel. 014/54 59 74

Kredietbank Herselt
nr. 401-1009701-46
BTW : 420.840.834

Lidgelden / Subscriptions

Bruno Van Rompaey
Bovenbosstraat 4
B-3044 Haasrode
België
tel. : 016/46.10.85

Generale
Bankmaatschappij
Leuven
nr. 230-0045353-74

Voor Nederland :

GIRO : 4083817
t.n.v. J.F. van Dunne'
Hoflaan 70
3062 JJ ROTTERDAM
Tel. : (010) 144802

Inzendingen : Games & Strategy

Frank Druijff
's Gravendijkwal 5A
NL 3021 EA Rotterdam
Nederland
tel. : 010/25.42.75

DAICLIC INFOS :

DAICLIC paraît tous les deux mois.
L'abonnement est compris dans la cotisation annuelle à IDC (du 1/1 au 31/12). A l'inscription, les numéros déjà parus dans l'année sont envoyés.

Conseil d'administration de IDC :

Président : Christian POELS, rue des Bas-Sarts 10, B-4100 SERAING Tél. : 041/37.16.06
Secrétaire : Marc VANDEMEERSCH, av. Vert Bocage 17 B-1410 WATERLOO Tél. : 02/354.13.63
Trésorier : Fabrice DULUINS, allée Tour Renard 4, B-1400 NIVELLES Tél. : 067/21.82.10
Rédaction : Christian POELS
Soumissions logiciels : Marc VANDERMEERSCH
Inscriptions, vente logiciels : Fabrice DULUINS.
(mode de paiement : voir ci-dessous)

Cotisations :

Belgique : 1000 FB virement, chèque, cash,...
Compte BBL : 371-0356842-45.
F. DULUINS et CH. POELS
ALLEE DE LA
TOUR RENARD, 4,
1400 NIVELLES
Etranger : 1100 FB par mandat postal international uniquement.

Services télématiques IDC :

MN2 Bruxelles-A :	02/242.70.08
MN2 Liège-A :	041/79.66.66
MN2 Paris-A :	1/39.71.82.91

Branches Régionales :

IDC BORDEAUX : Bruno Delannay, Rés. Acacias B+ B3, Av. de Saige, F-33600 Pessac
IDC BRUXELLES : Jacques Moens, Clos Fontaine Ducs 6, B-1310 La Hulpe
IDC CHARLEROI : Etienne Szigetvari, R. Provinciale 7, B-1361 Clabecq
IDC LIEGE : Philippe Rasquin, Rue Saivelette 89, B-4510 Saive
IDC PARIS : Philippe Casier, Rue de Paris 31ter, F-92190 Meudon

---

INHOUD - SOMMAIRE
DAINAMIC 36 DAICLIC 9

High speed data loader part 2

HIGH SPEED DATA LOADER Part 2

In de vorige publikatie heb ik geschreven hoe wij de HSL gebouwd hadden. Bij navraag in Beek bij Elektuur bleken de Eprom kaartjes nog steeds leverbaar.
Naar aanleiding van de veelvuldige vraag, wij bezitten geen DAI-VC1541 interface maar willen toch graag een HSL bouwen,kunnen jullie niet de software publiceren volgt hier het complete software programma.
Deze listing is NIET hetzelfde als wat er in onze DAI - VC1541 DOS staat,maar een op zich zelf staand programma.
Aan het einde van de listing staat "ENTRY FROM DAI POWER ON/RESET". Dit stukje wordt alleen gebruikt wanneer de interface of HSL beschikt over een hardware Power On initialisatie.Mocht U hier over beschikken,in b.v. een DCR, dan moet U dit stukje programma nog toevoegen.
Het schema voor de Power On initialisatie hardware zal ik ook publiceren.
Het is de bedoeling dat deze software in een Eprom komt welke op de "X" bus wordt geplaatst.
Bent U in het bezit van een DCR dan moet U dit programma combineren met Uw DCR software,assembleren en opnieuw in een Eprom programmeren.
Ik raad U aan hiervoor een andere Eprom te kopen en de oude niet te wissen totdat U er zeker van bent dat alles werkt.
Mocht U niet beschikken over een "X" bus interface print dan vindt U hierbij het schema hoe U er een kunt bouwen.
Dit is geen groot probleem,het gaat op een stukje Veroboard van 7x5 cm.

Mede gelet op de feiten dat Eproms van het type 2764 en 27128 momenteel goedkoper zijn dan de 2716 en 2732,
en de meeste programma's 8k byte of groter zijn,heb ik een gewijzigde HSL gemaakt.
Hiermede is het mogelijk zelf een keuze te maken uit drie Eprom type's afhankelijk van de lengte van het programma.
Hier zijn ook beperkingen aan verbonden n.l:
Een programma moet in een halve of hele Eprom passen.
U kunt maar twee Eproms gebruiken voor maximaal 4 programma's.
Gebruikt U twee 27128 dan kunt U hierin 4 programma's opbergen van elk 8k byte groot.
De situatie veranderd natuurlijk wanneer U 27256 gebruikt.
Hierin kunt U een 32k byte programma opbergen,maar deze Eproms zijn momenteel nog duur.

Beschrijving schema.

In deze versie is de prom van het type 82S23 komen te vervallen. Hierdoor is het niet meer mogelijk om een programma over meerdere Eproms te verdelen.
Hiervoor is in de plaats gekomen een 7474 welke het mogelijk maakt om d.m.v. PB3 een van de twee Eproms te selecteren.

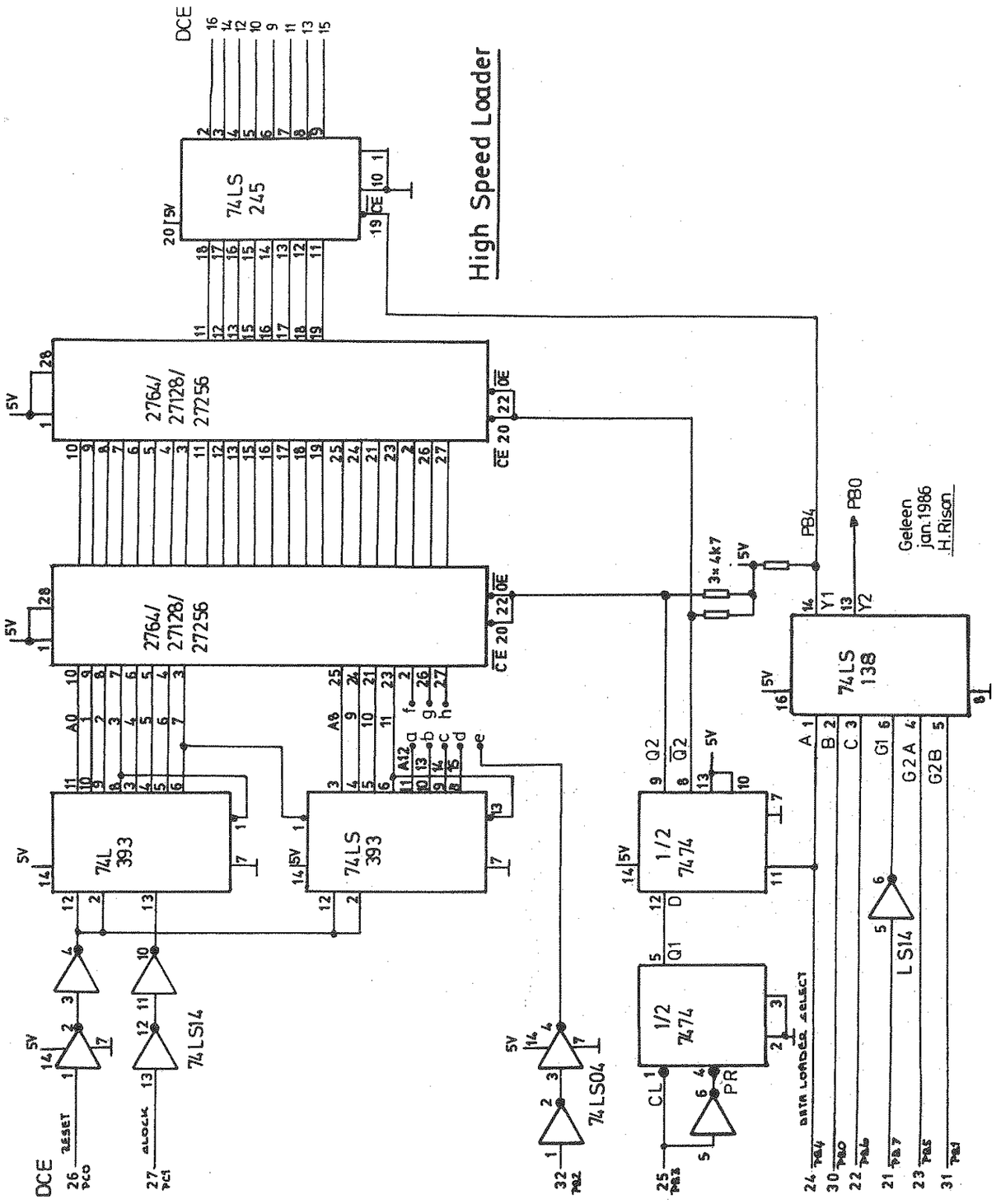
Het signaal PB2 wordt nu gebruikt om te kunnen schakelen tussen de onderste en bovenste helft van een Eprom.
Door de punten a t/m h op verschillende manieren met elkaar door te verbinden krijgen we een aantal mogelijkheden om met diverse typen van Eprom te werken.

Selectie Tabel.

Eprom type	Geheugen Inhoud	Door verbinding	Selectie commando
2764	2 x 8K	a - f	HSL en HSL2
2764	4 x 4K	e - f	HSL,HSL1, HSL2 en HSL3
27128	2 x 16K	a - f b - g	HSL en HSL2
27128	4 x 8K	a - f e - g	HSL,HSL1, HSL2 en HSL3
27256 27256	2 x 32K	a - f b - g c - h	HSL en HSL2
27256	4 x 16K	a - f b - g e - h	HSL,HSL1, HSL2 en HSL3

Ik hoop dat ik U hiermede van genoeg informatie heb voorzien om een HSL te bouwen.
Het programmeren van de Eproms moet U zelf verzorgen.
Denk aan de eerste bytes zoals beschreven in mijn vorige artikel.
De praktische realisatie van dit project laat ik aan U over.

Geleen, febr.1986
H.Rison




```
0000                    TITL    'HIGH SPEED DATA LOADER V2.0'
0000                    ;               28-01-86
0000                    ;
0000                    ;BY HENK RISON AND HEIN KOP
0000                    ;**************************
0000                    ;
0000                    ;DAI POINTERS
0000                    ;*************
0000  @=0061 MEMGIC  EQU     61H             ;Preserved GIC-mode
0000  @=0117 RDIPF   EQU     117H            ;flag set while running input
0000  @=0131 OTSW    EQU     131H            ;output switching
0000  @=0132 EFEPT   EQU     132H            ;encoding input pointer
0000  @=0134 EFECT   EQU     134H            ;encoded input count
0000  @=0135 EFSW    EQU     135H            ;encoded input switching
0000  @=01B0 FILES   EQU     1B0H            ;number following commands
0000  @=0296 INSW    EQU     296H            ;input switching
0000  @=0297 TABLE   EQU     297H            ;start address table
0000  @=0299 RSP     EQU     299H            ;rescued stackpointer
0000                    ;                    during comm.execution
0000  @=02C4 KBRFL   EQU     2C4H
0000  @=02E3 DINC    EQU     2E3H
0000                    ;DAI ROM ROUTINES
0000                    ;*****************
0000  @=C14D EXIT    EQU     0C14DH          ;POP H,D,B,PSW,RETURN
0000  @=DD45 EXIT1   EQU     0DD45H          ;POP H,PSW,CMC,RETURN
0000  @=C88F ENDCOM  EQU     0C88FH          ;jump if spec. action
0000  @=CA34 LOOKC   EQU     0CA34H          ;find instruction in table
0000  @=D1E2 INO     EQU     0D1E2H          ;if input from keyboard
0000  @=D695 MPT31   EQU     0D695H          ;output one character
0000  @=DA0B SYNERR  EQU     0DA0BH          ;run syntax error
0000  @=DAFF PMSGR   EQU     0DAFFH          ;print message
0000  @=DD5E CRLF    EQU     0DD5EH          ;print carr.return linefeed
0000  @=DDB4 INSER   EQU     0DDB4H          ;input from RS232
0000  @=DDD2 GIGNB   EQU     0DDD2H          ;get char.from line neglect
0000                    ;                    ;TAB and SPACE
0000  @=DE0D NUMER   EQU     0DE0DH          ;check for number
0000                    ;GIC CONTROLLER PPI (8255)
0000                    ;*************************
0000  @=FE00 GICPA   EQU     0FE00H          ;I/O port-A
0000  @=FE01 GICPB   EQU     0FE01H          ;I/O port-B
0000  @=FE02 GICPC   EQU     0FE02H          ;I/O port-C
0000  @=FE03 GICCMD  EQU     0FE03H          ;command word 8255 (GIC)
0000                    ;
0000                    ORG     0F000H
F000                    ;       ***************
F000                    ;ENTRY POINTS
F000                    ;*************
F000  C321F0 REMS    JMP     REM             ;Execute REM statement
F003  C329F1 CRDL    JMP     HSL             ;Command read data from EPROM
F006                    DS      1BH
F021                    ;EXECUTE REM STATEMENT
F021                    ;*********************
F021  F5     REM     PUSH PSW
F022  C5             PUSH B
F023  D5             PUSH D
F024  E5             PUSH H
F025  210A00         LXI H   0AH
F028  39             DAD SP
F029  229902         SHLD    RSP
F02C  0A             LDAX B
F02D  FEA9           CPI     0A9H
```

```
F02F  C24DC1          JNZ       EXIT          ;then pop all:ret
F032  03              INX B
F033  0A              LDAX B
F034  3C              INR A
F035  323401          STA       EFECT         ;set EFECT
F038  60              MOV H,B
F039  69              MOV L,C
F03A  223201          SHLD      EFEPT         ;BC in EFEPT
F03D  3E01            MVI A     1H
F03F  323501          STA       EFSW          ;input from string
F042  4F              MOV C,A
F043  CDA7F0          CALL      FIND          ;execute command
F046  AF              XRA A
F047  323501          STA       EFSW          ;input from keyboard
F04A  329A02          STA       29AH
F04D  0D              DCR C
F04E  CA0BDA          JZ        SYNERR        ;then "SYNTAX ERROR"
F051  C34DC1          JMP       EXIT          ;pop all:ret
F054            ;REPLACED DATA INPUT ROUTINE (DINC)
F054            ;*********************************
F054  E5        RDINC PUSH H
F055  211701          LXI H     RDIPF         ;(RDIPF)=#FF
F058  7E              MOV A,M                 ;get flag running inputs
F059  23              INX H
F05A  B6              ORA M                   ;compare with flag running
                                               programs
F05B  6F              MOV L,A                 ;00 when both 0, else FF
F05C  3A3101          LDA       OTSW          ;get output direction
F05F  E6FE            ANI       0FEH          ;mask bit 0
F061  B5              ORA L                   ;result  is 0,2 or FF
F062  E1              POP H
F063  C280F0          JNZ       CHID          ;when running something
F066  CD80F0          CALL      CHID          ;get inputs
F069  F5              PUSH PSW
F06A  FE0D            CPI       0DH           ; "CR" ?
F06C  CA8BF0          JZ        IFCR
F06F  FE09            CPI       9H            ; "TAB" ?
F071  C27EF0          JNZ       RDINC1        ;JMP else
F074  3E0C            MVI A     0CH
F076  CD95D6          CALL      MPT31         ;clear screen
F079  3E2A            MVI A     2AH
F07B  CD95D6          CALL      MPT31         ;print "*"
F07E  F1        RDINC1 POP PSW
F07F  C9              RET
F080            ;CHECK INPUT DIRECTION
F080            ;~~~~~~~~~~~~~~~~~~~~
F080  3A9602    CHID  LDA       INSW          ;get DINC flag
F083  E601            ANI       1H
F085  CAE2D1          JZ        INO           ;check for new key pressed?
F088  C3E302          JMP       DINC          ;get input from RS232
F08B            ;INPUT IS CARRIAGE RETURN
F08B            ;~~~~~~~~~~~~~~~~~~~~~~~~
F08B  35        IFCR  DCR M
F08C  F1              POP PSW
F08D  218FC8          LXI H     ENDCOM        ;jump.on carry if spec. action
F090  E3              XTHL                    ;ENDCOM on stack
F091  210000          LXI H     0H
F094  39              DAD SP                  ;get SP in HL
F095  229902          SHLD      RSP           ;save it in RSP
F098  0E01            MVI C     1H
F09A  D5              PUSH D
```



```
F09B  CDA7F0          CALL      FIND          ;find comm. and perform it
F09E  D1              POP D
F09F  E1              POP H
F0A0  AF              XRA A
F0A1  329A02          STA       29AH          ;set (029A)=0 (MAP1)
F0A4  C345DD          JMP       EXIT1         ;pop H,PSW:CMC:RET
F0A7            ;FIND INSTRUCTION IN TABLE AND PERFORM IT
F0A7            ;****************************************
F0A7  2A9702    FIND  LHLD      TABLE         ;get start addr. cmnd. table
F0AA  1E01      FIND1 MVI E     1H
F0AC  7C              MOV A,H
F0AD  B5              ORA L
F0AE  C8              RZ                      ;if zero end of cmnd. table
F0AF  C5              PUSH B
F0B0  CD34CA          CALL      LOOKC         ;find comm. in table
F0B3  DABEF0          JC        FIND2         ;when found
F0B6  C1              POP B                   ;else
F0B7  7E              MOV A,M                 ;(
F0B8  23              INX H                   ;addr. in table where
F0B9  66              MOV H,M                 ;instr.string can be found
F0BA  6F              MOV L,A                 ;)
F0BB  C3AAF0          JMP       FIND1         ;check next one
F0BE  5E        FIND2 MOV E,M                 ;(get addr.cmnd. after
F0BF  23              INX H                   ; string in DE
F0C0  56              MOV D,M                 ;)
F0C1  CDCDF0          CALL      NUCOM         ;check for more cmnds or number
F0C4  79              MOV A,C
F0C5  C1              POP B
F0C6  4F              MOV C,A
F0C7  21A7F0          LXI H     FIND
F0CA  E5              PUSH H                  ;return addr.FIND on stack
F0CB  EB              XCHG                    ;get addr.DL,command in HL
F0CC  E9              PCHL                    ;perform command
F0CD            ;CHECK FOR NUMBER AND/OR MORE COMMANDS
F0CD            ;~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
F0CD  CDD2DD    NUCOM CALL      GIGNB         ;get char. from line,neglect
                                               TAB/SPACE
F0D0  CDODDE          CALL      NUMER         ;check if char.is number
F0D3  DAD9F0          JC        NUCOM1
F0D6  3E2F            MVI A     2FH
F0D8  0D              DCR C
F0D9  0C        NUCOM1 INR C                  ;points to next char.
F0DA  D630            SUI       30H
F0DC  32B001          STA       FILES         ;store HEX val. of nr in FILES
F0DF  CDD2DD    NUCOM2 CALL     GIGNB         ;get char.neglect tab/space
F0E2  0C              INR C                   ;points to next char.
F0E3  FE3A            CPI       3AH
F0E5  C8              RZ                      ;return when ":"
F0E6  FE0D            CPI       0DH
F0E8  C8              RZ                      ;return when "CR"
F0E9  3AB001          LDA       FILES         ;get FILES
F0EC  B7              ORA A
F0ED  F2DFF0          JP        NUCOM2        ;get next char.until FILES=FF
F0F0  0E01            MVI C     1H
F0F2  117EF0          LXI D     RDINC1        ;alternative return addr.F0CC
F0F5  C9              RET
F0F6            ;INITIALISE HDL SYSTEM
F0F6            ;*********************
F0F6  F5        INIT  PUSH PSW
F0F7  C5              PUSH B
F0F8  D5              PUSH D
```

```
F0F9  E5              PUSH H
F0FA  3EC0            MVI A     0C0H          ;BAUD-rate
F0FC  3205FF          STA       0FF05H        ;set BAUD-rate register
F0FF  219DF1          LXI H     COMTAB        ;start addr.cmnd.table
F102  229702          SHLD      TABLE         ;store it in TABLE
F105  219602          LXI H     INSW          ;addr. DINC-flag
F108  3E02            MVI A     2H
F10A  B6              ORA M
F10B  77              MOV M,A                 ;set flag for(0296) OR-02
F10C  AF              XRA A
F10D  32C402          STA       KBRFL         ;reset KBRL
F110  2154F0          LXI H     RDINC         ; [
F113  22E102          SHLD      2E1H          ;new DINC into address
F116  3EC3            MVI A     0C3H          ;       02E0-02E2
F118  32E002          STA       2E0H          ; ]
F11B  21B4DD          LXI H     INSER         ; [load "JUMP DDB4" (INSER)
F11E  22E402          SHLD      2E4H          ;   into 02E3-02E5
F121  3EC3            MVI A     0C3H          ;   input from RS232
F123  32E302          STA       DINC          ; ]
F126  C34DC1          JMP       EXIT          ;POP H,D,B,PSW,RETURN
F129          ;READ DATA FROM HIGHSPEED DATA LOADER
F129          ;************************************
F129  F5      HSL     PUSH PSW
F12A  D5              PUSH D
F12B  E5              PUSH H
F12C  3AB001          LDA       FILES         ;EPROM-bank number
F12F  B7              ORA A
F130  F237F1          JP        HSL10         ;If number give
F133  3C              INR A
F134  32B001          STA       FILES         ;Else default is HSL0
F137  FE04    HSL10   CPI       4H
F139  F20BDA          JP        SYNERR        ;Syntax error if nr >4
F13C  CDA6F1          CALL      SETGIC        ;Init GIC
F13F  3AB001          LDA       FILES         ;Get EPROM nr
F142  07              RLC
F143  07              RLC                     ;Into hinibble: PB2,3
F144  C610            ADI       10H           ;Add HSL select: PB4=1
F146  3201FE          STA       GICPB         ;Address EPROM
F149  3E01            MVI A     1H            ;Reset HSL :PC0=1
F14B  3203FE          STA       GICCMD
F14E  3D              DCR A
F14F  3203FE          STA       GICCMD        ;Cancel reset: PC0=0
F152          ;LOAD MEM.ADDR. AND PRGRM LENGHT IF EPROM MOUNTED
F152          ;~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
F152  CD7FF1          CALL      HSL30         ;move character into A reg.
F155  FEFD            CPI       0FDH          ;check for empty EPROM,wrong
F157  C28FF1          JNZ       HSL40         ;coded and/or EPROM bank
F15A          ;get startaddress + length
F15A  CD7FF1          CALL      HSL30
F15D  6F              MOV L,A                 ;Lo-byte start addr.
F15E  CD7FF1          CALL      HSL30
F161  67              MOV H,A                 ;Hi-byte start addr.
F162  CD7FF1          CALL      HSL30
F165  57              MOV D,A                 ;Hi-byte lenght of program
F166  CD7FF1          CALL      HSL30
F169  5F              MOV E,A                 ;Lo-byte lenght of program
F16A          ;read data from HSL
F16A          ;~~~~~~~~~~~~~~~~~~
F16A  CD7FF1  HSL20   CALL      HSL30         ;Read byte
F16D  77              MOV M,A                 ;Store it in memory
```



```
F16E  23              INX H
F16F  1B              DCX D
F170  7A              MOV A,D
F171  B3              ORA E
F172  C26AF1          JNZ       HSL20         ;Next byte if not ready
F175          ;close HSLX
F175          ;~~~~~~~~~~
F175  210000          LXI H     0H
F178  2201FE          SHLD      GICPB         ;reset addr.and count bits
F17B  E1              POP H
F17C  D1              POP D
F17D  F1              POP PSW
F17E  C9              RET                     ;return to monitor prgrm.
F17F          ;read byte from HSL
F17F          ;~~~~~~~~~~~~~~~~~~~
F17F  3E03    HSL30   MVI A     3H
F181  3203FE          STA       GICCMD        ;Set clock: PC1=1
F184  3A00FE          LDA       GICPA         ;Get byte
F187  F5              PUSH PSW
F188  3E02            MVI A     2H
F18A  3203FE          STA       GICCMD        ;clear clock:PC1=0
F18D  F1              POP PSW
F18E  C9              RET
F18F          ;ERROR ROUTINE
F18F          ;~~~~~~~~~~~~
F18F  210000  HSL40   LXI H     0H
F192  2201FE          SHLD      GICPB         ;Disable HSL:PB+PC=0
F195  CDFFDA          CALL      PMSGR         ;print error message
F198  C7DC            DW        0DCC7H        ;"NOT AVAILABLE"
F19A  C35EDD          JMP       CRLF          ;carr.return linefeed
F19D          ;TABLE WITH COMMANDS
F19D          ;~~~~~~~~~~~~~~~~~~
F19D  03      COMTAB  DB        3H
F19E  48534C          DB        'HSL'         ; command HSL
F1A1  29F1            DW        HSL
F1A3  00              NOP
F1A4  00              NOP
F1A5  00              NOP
F1A6          ;set GIC for HSL
F1A6          ;~~~~~~~~~~~~~~~
F1A6  3E98    SETGIC  MVI A     98H
F1A8  3203FE          STA       GICCMD        ;Select GIC-mode
F1AB  326100          STA       MEMGIC        ;Keep memory
F1AE  C9              RET
F1AF          ;ENTRY FROM DAI POWER ON/RESET
F1AF          ;*****************************
F1AF          ;ORG      0F2F2H
F1AF          ;JMP      INIT        ;init Highspeed Data Loader
F1AF                    END
```

FOP-reset

```
;       **********************
;       ***   FOP- RESET    ***
;       **********************
;
;       door Mark Stout       datum 1986 01 17
;       Verzusteringslaan 36, B 2700 Sint-Niklaas
;
;
;Dit programma bootst een RESET na, maar vernietigd de geheu-
;gen inhoud niet. Je kan zelf twee maal een tekst op het scherm
;zetten : de eerste in de plaats van 'DAI PERSONAL COMPUTER'
;         de tweede in de plaats van 'BASIC V1.1'
;Je kan deze routine gewoon vanuit basic aanroepen door een
;CALLM#300.  Zo kan je grappige effekten bereiken in je
;programma's : de gebruiker is de wanhoop nabij omdat hij ziet
;dat er een RESET gebeurt (zeker als je als tekst 'DAI PC'
;neemt zoals bij een gewone reset), maar dan blijkt dat het
;maar een FOP-RESET is.
;Je kan het natuurlijk ook gewoon gebruiken om je programma's
;wat op te fleuren.
;
;
        ORG     300H
        PUSH B
        LXI H   0H          ;het scherm wordt opgebouwd
        RST 5               ;zoals bij een gewone reset
        DB      9H
        CALL    0C7FBH
        DCX H
        LXI D   0C7E0H
        RST 5
        DB      0H
        LXI H   TEKST1      ;hier wordt de eerste tekst
        CALL    0DAD4H      ;op het groene scherm gezet
        LHLD    2A5H        ;verdere schermorganisatie
        LXI D   97BH
        DAD D
        MVI M   5FH
        LXI D   0FFD0H
```



```
        DAD D
        CALL    0CEF9H
        MVI D   12H
NEXT    CALL    0CECFH
        DCR D
        JNZ     NEXT
        RST 1               ;wacht op een toets
        DB      15H
        LXI H   0C7EEH      ;terug naar gewone mode 0
        RST 5               ;het basicprogramma is niet
        DB      6H          ;gewist !
        LXI H   TEKST2      ;hier wordt de tweede tekst
        CALL    0DAD4H      ;op het scherm gezet
        POP B
        RET                 ;terug naar basic
;
;       *** eigen tekst   ***
;
;eerst wordt er 6 keer een CHR$(13) geprint
```



```
TEKST1  DB      0DH,0DH,0DH,0DH,0DH,0DH
;
;hier mag je uw eerste tekst zetten : die bestaat uit 2 regels
;de eerste regel moet 18 characters bevatten
;(een tekort moet je aanvullen met spaties)
;richt hierop:***'                  '***
        DB      'DATA APPLICATIONS ' ;   (voorbeeld tekst)
;opm :de volgen de tekst (spaties) kan je niet zien maar
;is noodzakelijk
        DB      '      '
;hier moet je de tweede regel van de eerste tekst zetten
        DB      '  INTERNATIONAL' ;      (voorbeeld tekst)
        DB      0DH             ;CHR$(13)
        DB      0H              ;einde eerste tekst
;
;
TEKST2  DB      0CH             ;wis scherm
;zet hier de tweede tekst      (normaal 'BASIC V1.1)
        DB      'FOP-RESET   door Mark Stout' ; (voorbeeld)
        DB      0DH,0H          ;CHR$(13)
        DB      0H              ;einde tweede tekst
        END
```

cont on p. 15

Blokeditor

GEBRUIK

Deze blokeditor stelt gebruikers van de Commodore-1541 diskettedrive in staat om ieder blok van de schijf te halen, te veranderen en weer terug te schrijven naar de schijf. De blokeditor wordt gebruikt in combinatie met de 'FDD-MONITOR'.

Op het scherm verschijnt de inhoud van het blok (256 bytes) in het linkerveld en rechts de ASCII-vertaling. Beide velden zijn via cursorbesturing eenvoudig te wijzigen. Met SH^ kan men de opeenvolgende blokken van een file oproepen.

TOEPASSINGEN

Vooral voor het bestuderen van het beheer op de schijf, en voor ingrepen in de inhoudsopgave is het programma van nut. Deze bevinden zich op spoor 18. Voorbeeld: verandering van naam van schijf of een file of van het type, het in ere herstellen van geveegde files enz.

Ook langere files kunnen op de schijf veranderd worden, maar dan riskeert men een 'checksum-error'. De checksum (het controlegetal) wordt immers over de GEHELE file berekend van iks aantal blokken. Het staat dan ook aan het eind van de data en moet bij elke verandering aangepast worden. Dat is geen eenvoudige zaak.

CHECKSUM

In principe gaat het als volgt: je voert met het startgetal #56 en het eerste byte een XOR operatie uit en schuift daarna de bits een plaats naar links (vermenigvuldig met 2). Een achtste bit komt op de nulde plaats. Weer XOR van het resultaat met het volgende byte en weer SHL 1; het resultaat is steeds de 'CHECKSUM'; enz. tot en met het laatste byte van de file. De berekening vindt plaats in een routine op #D30F. Kleine operaties, zoals het veranderen van het plaatsadres van een machinetaalprogramma kunnen met de editor zo eenvoudig uitgevoerd worden. Van deze twee bytes kan gemakkelijk 'met de hand' de checksum berekend worden.

AANPASSING

Voor andere systemen is het progamma aan te passen. De gebruikte FDD- commando's (OPEN,GET,PUT) zijn gebruikelijk en elk DOS bezit zijn eigen methode van spoor/blok- acces

```
1       REM BLOKEDITOR;gebruikt in combinatie met FDD-MONITOR
2       REM *** C.W.A.van Dijk - Kampen ***

10       CLEAR 2000:DIM X(16):COLORT 7 0 7 14:GOTO 90

20       REM LEES ASCI UIT VIDEO RAM
25       YD=(23-Y)*#86
30       AD=#BFE7-YD-X(K)*2:B=PEEK(AD):GOSUB 50:BH=B SHL 4:B=PEEK(AD-2)
           :GOSUB 50:G=BH+B:RETURN
40       REM MAAK GETAL UIT ASCI
50       IF B<#30 THEN B=-#100:RETURN
60       IF B<#3A THEN B=VAL(CHR$(B)):RETURN
70       IF B<#41 THEN B=-#100:RETURN
80       B=VAL(CHR$(B-#11))+10:RETURN

90       PRINT CHR$(12);:CURSOR 5,3:PRINT "CHAR.DEL:  ANDER VELD"
100       CURSOR 5,2:PRINT "SHIFT ^: VOLGEND BLOK"
110       CURSOR 5,1:PRINT "SHIFT ";CHR$(140);": SCHRIJF BLOK"
120       CURSOR 5,0:PRINT "TAB: NIEUW SPOOR/BLOK";
160       N=1:IN$=" "
170       OPEN4=#403:PRNT15=#40C:PUT4=#42D:GET4=#412:INP15=#41E:
           CLOSE4=#424
180       RESTORE:FOR K=1 TO 16:READ X(K):NEXT
190       DATA 2,4,7,9,12,14,17,19,22,24,27,29,32,34,37,39
200       CURSOR 5,22:INPUT "SPOOR:";IN:IF IN>35 OR IN<1 THEN 220:
           GOSUB 1920:SP$=IN$
210       INPUT "  - BLOK:";IN:IF IN<0 THEN 220:GOSUB 1920:BL$=IN$:
           PRINT :GOTO 230
220       PRINT "FOUTE INVOER":GOTO 200
230       IF SP$=" 0" THEN 200
240       OPEN$="#":CALLM OPEN4,OPEN$
250       OP$="U1 4 0"+SP$+BL$:CALLM PRNT15,OP$
260       REM VOLGEND SPOOR&BLOK
270       OP$="B-P 4 0"
280       CALLM PRNT15,OP$:CALLM GET4,IN$:GOSUB 1900:SP1$=IN$:
           CALLM GET4,IN$:GOSUB 1900:BL1$=IN$
300       REM BLOK OP SCHERM
305       IN$=" "
310       CALLM PRNT15,OP$
320       CURSOR 0,20
330       FOR I=0 TO 15:K=0:PRINT HEX$(I);" ";
340       FOR J=44 TO 59:CALLM GET4,IN$:IN=ASC(IN$):
           PRINT RIGHT$("00"+HEX$(IN),2);
350       IF CURX MOD 5=1 THEN PRINT " ";
360       X=CURX:CURSOR J,CURY
370       IF IN<32 OR IN>127 THEN PRINT ".";:GOTO 390
380       PRINT IN$;

390       CURSOR X,CURY:NEXT:PRINT :NEXT
400       CALLM CLOSE4
410       CURSOR 0,20
420       POKE #74,0:POKE #75,#FF

999       REM *** TEKENVAK RECHTS ******************
1000       CURSOR 44,20:XMN=44:XMX=59:YMN=5:YMX=20
1010       IF CURX<44 THEN 1210:GOSUB 1800:REM K BEREKENEN
1020       G=GETC:IF G=8 THEN 1180:IF G=9 THEN 1600:IF G<16 THEN 1020
1030       UIT=0:IF G>15 AND G<24 THEN ON G-15 GOSUB 1260,1280,1300,
           1340,1400,1410,1180,1180:IF UIT=0 THEN 1010:
           ON UIT GOTO 1500,1550
1040       PRINT CHR$(G);:X=CURX:CURSOR X(K),CURY:
           PRINT RIGHT$("00"+HEX$(G),2);
1042       IF X<XMX+1 THEN 1044:IF CURY>YMN THEN 1043:CURSOR XMN,YMX:
           GOTO 1010
1043       CURSOR XMN,CURY-1:GOTO 1010
1044       CURSOR X,CURY:GOTO 1010
1180       XMN=2:XMX=40:CURSOR X(K),CURY:GOTO 1210

1199       REM ***ASCII-VAK LINKS***************
1200       CURSOR 2,20:XMN=2:XMX=40:YMN=5:YMX=20
1210       IF CURX>40 THEN 1010:GOSUB 1800:REM K BEREKENEN
1215       IF CURX MOD 5=1 THEN CURSOR CURX+1,CURY
1220       G=GETC:IF G=8 THEN 1390:IF G=9 THEN 1600:IF G<16 THEN 1220
1230       UIT=0:IF G>15 AND G<24 THEN ON G-15 GOSUB 1260,1280,1300,
           1340,1400,1410,1390,1390:
           IF UIT=0 THEN 1210:ON UIT GOTO 1500,1550
1235       IF (G>47 AND G<58) OR (G>64 AND G<71) THEN 1240:GOTO 1210
1240       PRINT CHR$(G);:X=CURX:Y=CURY:CURSOR K+43,Y:GOSUB 25:
           IF G<32 OR G>127 THEN 1241:PRINT CHR$(G);:GOTO 1242
1241       PRINT ".";
```

```
1242    IF X<XMX+1 THEN 1244:IF CURY>YMN THEN 1243:CURSOR XMN,YMX:
        GOTO 1210
1243    CURSOR XMN,CURY-1:GOTO 1210
1244    CURSOR X,CURY:GOTO 1210

1250    REM CURSORBEWEGINGEN
1260    IF CURY=YMX THEN CURSOR CURX,YMN:RETURN:REM OP
1270    CURSOR CURX,CURY+1:RETURN
1280    IF CURY=YMN THEN CURSOR CURX,YMX:RETURN:REM NEER
1290    CURSOR CURX,CURY-1:RETURN
1300    IF CURX=XMN THEN 1320:REM LINKS
1310    CURSOR CURX-1,CURY:IF CURX MOD 5=1 THEN 1310:RETURN
1320    IF CURY<YMX THEN CURSOR XMX,CURY+1:RETURN
1330    CURSOR XMX,YMN:RETURN
1340    IF CURX=XMX THEN 1360:REM RECHTS
1350    CURSOR CURX+1,CURY:RETURN
1360    IF CURY>YMN THEN CURSOR XMN,CURY-1:RETURN
1370    CURSOR XMN,YMX:RETURN
1380    CURSOR X(K),CURY:RETURN
1390    XMN=44:XMX=59:CURSOR K+43,CURY:GOTO 1010
1400    UIT=1:RETURN
1410    UIT=2:RETURN

1500    REM SH CURUP: VOLGEND BLOK
1510    N=N+1:Y=20:K=1:GOSUB 20:NSP$=STR$(G):K=2:GOSUB 20:NBLK$=STR$(G)
1515    SP$=LEFT$(NSP$,LEN(NSP$)-2):BL$=LEFT$(NBLK$,LEN(NBLK$)-2)
1520    IF SP$=" 0" THEN CURSOR 33,3:PRINT "LAATSTE BLOK";:GOTO 1000
1530    CURSOR 5,22:PRINT "SPOOR:";SP$;"  -  BLOK:";BL$;"   #";
        MID$(STR$(N),1,LEN(STR$(N))-3);"   ";
1540    GOTO 240
1550    GOSUB 2000:GOTO 1000
1560    GOTO 240

1600    REM EINDE PROGRAMMA
1605    GOTO 90
1610    REM POKE #74,1:POKE #75,#5F:PRINT CHR$(12);:END

1800    REM MAAK PLAATSCODE K UIT CURSOR
1810    X=CURX:Y=CURY:IF X<44 THEN 1830
1820    K=X-43:RETURN
1830    K=0
1840    K=K+1:IF K>16 THEN 1850:IF X(K)<=X THEN 1840:K=K-1:RETURN
1850    K=16:RETURN

1900    REM ASCIL NAAR STRING
1910    IN=ASC(IN$)
1920    IN$=LEFT$(STR$(IN),LEN(STR$(IN))-2)
1950    RETURN

2000    REM VIDEO-RAM UITLEZEN
2005    CURSOR 33,3:PRINT "VIDEO-RAM UITLEZEN";
2010    CALLM OPEN4,OPEN$
2020    OP$="B-P 4 0":CALLM PRNT15,OP$
2030    FOR Y=20 TO 5 STEP -1:YD=(23-Y)*#86
2040    FOR K=1 TO 16:GOSUB 30:IF G<0 THEN 2100:IN$=CHR$(G):
        CALLM PUT4,IN$:NEXT:NEXT
2080    OP$="U2 4 0"+SP$+BL$:CALLM PRNT15,OP$
2090    OP$="IO":CALLM PRNT15,OP$:GOTO 2110
2100    CURSOR 33,1:PRINT "LEESFOUT; GEEN ASCII":CALLM CLOSE4:RETURN
2110    CALLM CLOSE4:CURSOR 33,3:PRINT SPC(26);:RETURN
```



```
---------- data-veld ----------    ---ASCII-veld--

   SPOOR:?18  - BLOK:?1

1204 C214 1241 2D44 2031 A0A0 A0A0 A0A0   .....A-D 1......
A0A0 A0A0 A000 0000 0000 0000 0000 0E00   ................
0000 C20F 0141 2D44 2032 A0A0 A0A0 A0A0   .....A-D 2......
A0A0 A0A0 A000 0000 0000 0000 0000 0500   ................
0000 C20F 0441 2D44 2033 A0A0 A0A0 A0A0   .....A-D 3......
A0A0 A0A0 A000 0000 0000 0000 0000 0400   ................
0000 C215 0741 5252 4159 2D45 4449 54A0   .....ARRAY-EDIT.
A0A0 A0A0 A000 0000 0000 0000 0000 0700   ................
0000 C314 0A41 5252 4159 4441 5441 2F4F   .....ARRAYDATA/O
A0A0 A0A0 A000 0000 0000 0000 0000 0200   ................
0000 8218 0442 4153 424F 4F54 A0A0 A0A0   .....BASBOOT....
A0A0 A0A0 A000 0000 0000 0000 0000 0100   ................
0000 C311 0042 4153 4943 4F44 452D 32A0   .....BASICODE-2.
A0A0 A0A0 A000 0000 0000 0000 0000 0C00   ................
0000 8218 0742 4C4F 4B45 4449 544F 522F   ....BLOKEDITOR/
42A0 A0A0 A000 0000 0000 0000 0000 0E00   B...............

   CHAR.DEL:  ANDER VELD                  Veel plezier ermee,
   SHIFT ↑: VOLGEND BLOK                  Kees van Dijk
   SHIFT ↓: SCHRIJF BLOK                  Watermunt 5
   TAB: NIEUW SPOOR/BLOK                  8265 EL Kampen
```

cont from p.11

```
0300 C5 21 00 00 EF 09 CD FB C7 2B 11 E0 C7 EF 00 21
0310 3D 03 CD D4 DA 2A A5 02 11 7B 09 19 36 5F 11 D0
0320 FF 19 CD F9 CE 16 12 CD CF CE 15 C2 27 03 CF 15
0330 21 EE C7 EF 06 21 6C 03 CD D4 DA C1 C9 0D 0D 0D
0340 0D 0D 0D|44 41 54 41 20 41 50 50 4C 49 43 41 54
0350 49 4F 4E 53 20|20 20 20 20 20 20|20 20 49 4E 54
0360 45 52 4E 41 54 49 4F 4E 41 4C|0D 00 0C|46 4F 50
0370 2D 52 45 53 45 54 20 20 20 64 6F 6F 72 20 4D 61
0380 72 6B 20 53 74 6F 75 74|0D 00 00
>
```

Hierboven kan je de geheugeninhoud zien ; het programma zit van #300 tot #38A. Je kan dus twee maal je eigen tekst laten afdrukken ; de eerste tekst bestaat uit twee regels. Je kan gewoon de ascii waarden van de letters op de volgende plaatsen invullen :

```
tekst 1, regel 1 van #343 tot #354     (normaal : DAI PERSONAL )
         regel 2 van #35B tot #369     (normaal :   COMPUTER   )
tekst 2          van #36D tot #387     (normaal : BASIC V1.1   )
```

Parametervergelijkingen

PARAMETERVERGELIJKINGEN.

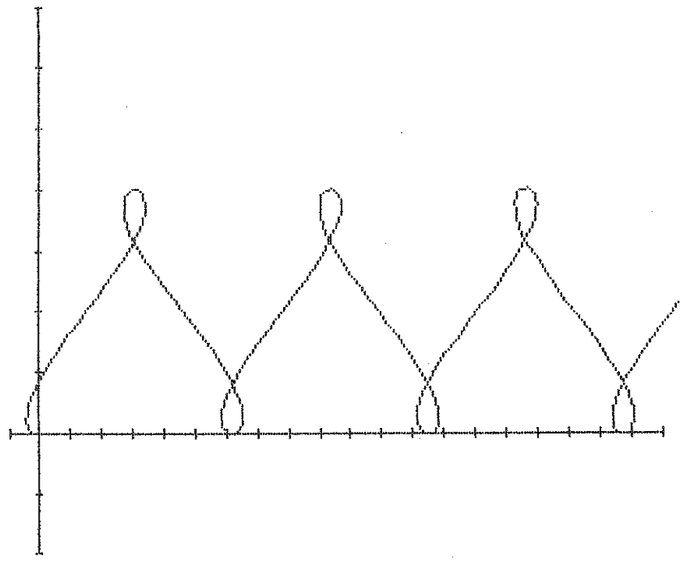
Naast de gewone funkties : Y=F(X) , waarbij Y een funktie is van X (vb.Y=3*X + 4), is er nog een andere klasse : de parametervergelijkingen.
Bij deze vergelijkingen zijn X en Y funktie van een onafhankelijk veranderlijke : de parameter T : X=F(T) en Y=F(T).
Vb. X=R*COS(T) en Y=R*SIN(T) met T:0 -> 2*PI zal een cirkel met straal R tekenen.
De 2 parametervergelijkingen moeten zelf op de regels 50 en 60 geprogrammeerd worden. Daarna RUN 1100 om het programma te doen vervolgen. Door de onder- en bovengrens van de X- en Y-as zodanig te kiezen, kan men een bepaald gedeelte van de kurve uitvergroten of slechts een bepaald gedeelte bekijken.
Hieronder volgt een uitgewerkt voorbeeld van een parametervergelijking die een hypocycloide tekent.

```
50 X=2*COS(T) + COS(2*T)
60 Y=2*SIN(T) - SIN(2*T)
```

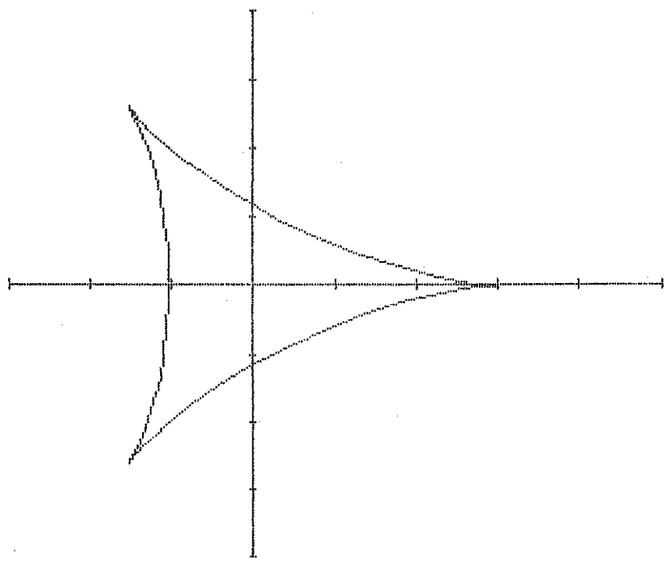
T-ondergrens : 0 X interval : -3,5
T-bovengrens : 6.3 Y interval : -4,4
stap : 0.1 eenheid van de assen : 1

Verberckmoes Filip
Meidoornlaan 16
2750 BEVEREN

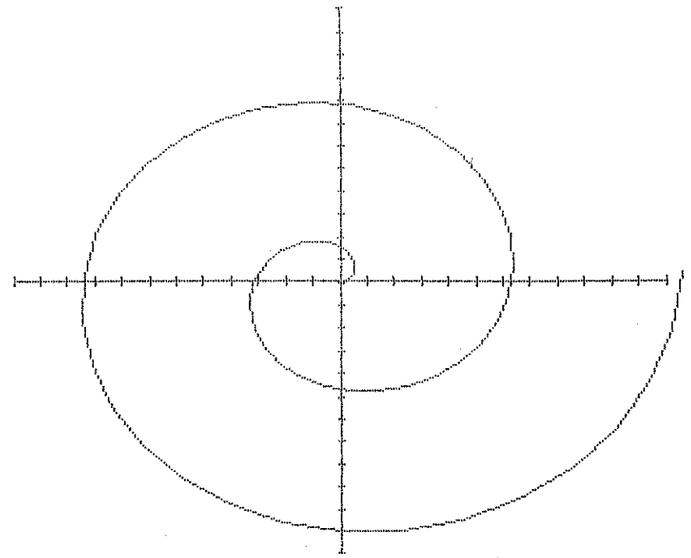

```
10    REM PARAMETERVERGELIJKINGEN:
      VERBERCKMOES F.  9/1986
20    GOTO 1000

25    REM **  TEKENROUTINE  **
30    X1=-1:Y1=-1
40    FOR T!=TMIN! TO TMAX! STEP ST!

50  1  REM ** PROGRAMMEER HIER X=F(T)
60  1  REM ** PROGRAMMEER HIER Y=F(T)
70  1  X2=K+FX!*X!:Y2=R+FY!*Y!
80  1  IF X1<0 OR X1>XMAX THEN 130
90  1  IF X2<0 OR X2>XMAX THEN 130
100 1  IF Y1<0 OR Y1>YMAX THEN 130
110 1  IF Y2<0 OR Y2>YMAX THEN 130
120 1  DRAW X1,Y1 X2,Y2 15
130   X1=X2:Y1=Y2:NEXT
140   COLORG 6 0 0 15
150   IF GETC<>32 THEN 150
160   MODE 0:PRINT CHR$(12)
170   PRINT "Keuzemenu :
      STOPPEN.....1"
180   PRINT "          NIEUWE
      FUNKTIES....2"
190   PRINT "          NIEUWE
      PARAMETERS..3"
200   PRINT :INPUT "UW KEUZE...>";KZ!
210   ON KZ! GOTO 220,1000,1100
220   END
1000  MODE 0:PRINT CHR$(12)
1010  CURSOR 10,22:FOR I!=1.0 TO 40.0:
      PRINT CHR$(25);:NEXT
1020  CURSOR 18,20:PRINT
      "PARAMETERVERGELIJKINGEN."
1030  CURSOR 10,19:FOR I!=1.0 TO 40.0:
      PRINT CHR$(25);:NEXT
1040  PRINT :PRINT :PRINT
      "Programmeer de funkties X=F(T)
      en Y=F(T)"
1050  PRINT :PRINT "op de regels 50
      en 60."
1060  PRINT :PRINT "Om te starten :
      RUN 1100"
1070  PRINT :PRINT "Als achtergrond
      van kleur verandert is het
      tekenen gedaan."
1080  PRINT :PRINT "Druk dan op de
      spatiebalk om te eindigen."
1090  END
1100  PRINT CHR$(12):LIST 41-69:PRINT
1110  INPUT "T-ONDERGRENS";TMIN!:
      PRINT
1120  INPUT "T-BOVENGRENS";TMAX!:
      PRINT
1130  INPUT "STAP";ST!:PRINT :PRINT
1140  PRINT "Assenkeuze":PRINT :INPUT
      "X-ONDERGRENS";XO!:PRINT
1150  INPUT "X-BOVENGRENS";XB!:PRINT
1160  INPUT "Y-ONDERGRENS";YO!:PRINT
1170  INPUT "Y-BOVENGRENS";YB!:PRINT

1180  INPUT "EENHEID VAN DE ASSEN";UN!

1185  REM **  TEKENEN ASSEN +
      SCHAALVERDELING  **
1190  COLORG 7 0 0 15:MODE 6
1200  XM=250:YM=250
1210  K=XM*XO!/(XO!-XB!)+50:R=
      YM*YO!/(YO!-YB!)+2
1220  FX!=XM/(XB!-XO!):FY!=
      YM/(YB!-YO!)
1230  DRAW K,2 K,YM+2 0:DRAW 50,R XM+
      50,R 0
1240  AT!=XB!-XO!:DF!=XM/AT!*UN!
1250  FOR AV!=50.0 TO XM+50.0 STEP DF!
1260  DRAW AV!,R-2 AV!,R+2 0:NEXT
1270  AT!=YB!-YO!:DF!=YM/AT!*UN!
1280  FOR AV!=2.0 TO YM+2.0 STEP DF!
1290  DRAW K-1,AV! K+1,AV! 0:NEXT
1300  GOTO 30
```



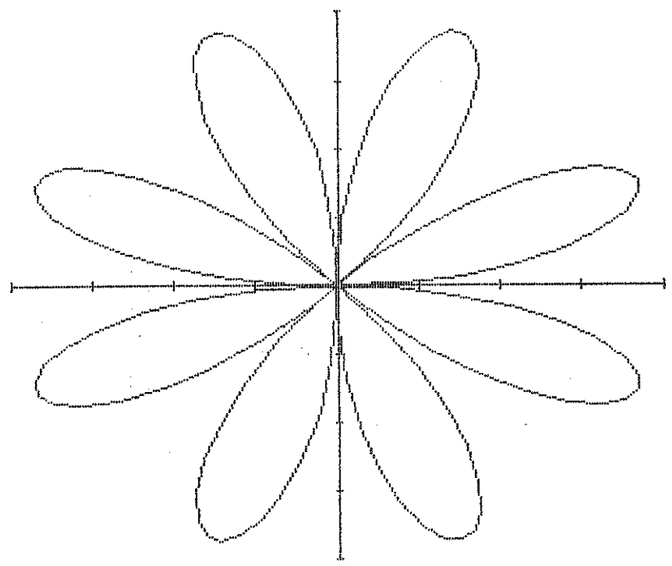
```
50     X=T-SIN(2.0*T)
60     Y=2.0-2.0*COS(T)
T : 0 -> 19.5 STEP 0.2    X(-1,20) Y(-2,7)
```



```
50     X=2.0*COS(T)+COS(2.0*T)
60     Y=2.0*SIN(T)-SIN(2.0*T)
T : 0 -> 6.3  STEP 0.1    X(-3,5) Y(-4,4)
```

```
50     X=T*COS(T)
60     Y=T*SIN(T)
T : 0 -> 12.6 STEP 0.2    X(-12,12) Y(-12,12)
```



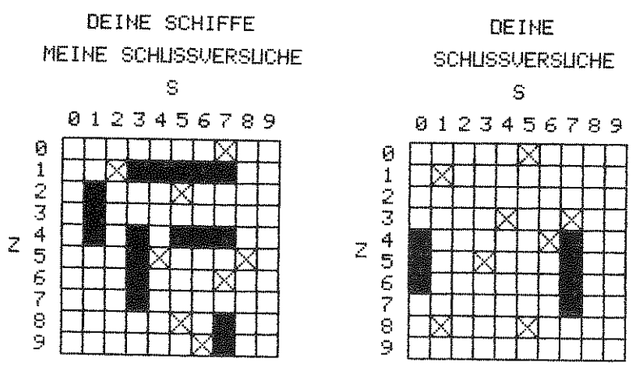
```
50     X=4.0*SIN(4.0*T)*COS(T)
60     Y=4.0*SIN(4.0*T)*SIN(T)
T : 0 -> 6.3 STEP 0.05    X(-4,4) Y(-4,4)
```

Seeschlacht

SEESCHLACHT (Ship-Battle)
==========

Dieses Programm Seeschlacht (auch Schiffeversenken genannt)
wird normalerweise mit Papier und Bleistift gespielt.
Der Sinn des Spieles ist, die versteckten Schiffe des Gegners
zu finden und zu zerstören. Dazu zielt man mit den Cursor-
Tasten und wirft dann mit SPACE eine Bombe ab. Wenn man ein
Schiff getroffen hat, dann darf man nochmals schiessen. Danach
schießt der Gegner (hier der DAI).
Nach dem Start des Programms, erscheint zunächst ein Titelbild.
Nachdem man eine Taste gedrückt hat, baut sich das Spielfeld
auf. Man kann nun seine Schiffe auf der Spielfläche verteilen.
Man benutzt dazu die Cursor und die Shift-Cursor -Tasten.
Wer alle fünf Schiffe des Gegners versenkt hat, der hat
gewonnen.

Layout des Spielfeldes :



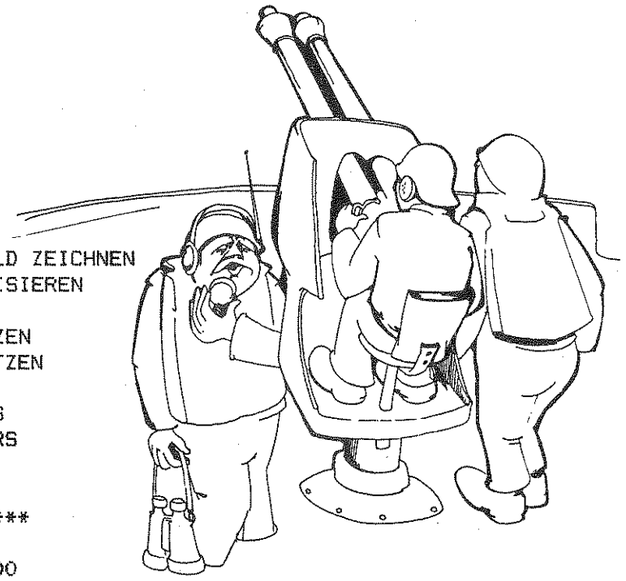
Zum Programm :

Im Programm ist ab Zeile 60000 das Unterprogramm SLOW GRAF TEXT
enthalten. Falls Sie Besitzer von FAST GRAF TEXT sind, können
Sie FGT wie folgt verwenden :

Implementation von FGT:

1. FGT laden (Adresse #300)
2. Standart-Table laden
3. HEAP über FGT legen (ca. #900)
4. SEESCHLACHT laden oder eingeben
5. Zeile 10010 löschen
6. Zeile 60000 - Ende löschen
7. CLEAR 4:UT
8. Programm saven (W29B XXXX SEESCHLACHT)

Der Aufruf von FGT ist im Programm enthalten (Zeile 10000-11000).



```
1     REM
2     REM SEESCHLACHT
3     REM
4     REM
5     REM ****** IMP INT ******
6     REM
100   CLEAR 4000:GOSUB 7000:REM TITELBILD ZEICHNEN
200   CLEAR 4000:GOSUB 8000:REM INITIALISIEREN
300   GOSUB 6000:REM SPIELFELD ZEICHNEN
400   GOSUB 5000:REM SPIELERSCHIFFE SETZEN
500   GOSUB 4000:REM COMPUTERSCHIFFE SETZEN
600   GOSUB 3000:REM SPIELBEGINN
700   GOSUB 2000:REM SCHUSS DES SPIELERS
800   GOSUB 1000:REM SCHUSS DES COMPUTERS
900   GOTO 700
1000  REM
1001  REM ***** SCHUSS DES COMPUTERS *****
1002  REM
1010  PRINT CHR$(12);:IF FLAG=1 GOTO 1300
1020  XC=RND(10):YC=RND(10)
1030  IF XC=10 OR YC=10 OR (XC+YC) MOD 2=0 GOTO 1020
1050  Z=SF(0,XC+1,YC+1):SF(0,XC+1,YC+1)=10
1060  IF Z=10 THEN R1=9:GOTO 1000
1070  PRINT "Ich schiesse auf den Punkt (";YC;" /";XC;" )"
1080  IF Z<>0 GOTO 1200
1100  DRAW 41+XC*10,141-YC*10 49+XC*10,149-YC*10 23
1110  DRAW 41+XC*10,149-YC*10 49+XC*10,141-YC*10 23
1120  GOSUB 12000:IF FLAG=1 THEN R1=9
1130  RETURN
1200  FILL 41+XC*10,141-YC*10 49+XC*10,149-YC*10 23
1210  GOSUB 13000:TR(1,Z)=TR(1,Z)-1
1212  IF RICHT=1 OR RICHT=3 THEN T3=XC:T4=YC
1213  IF RICHT=2 OR RICHT=4 THEN T1=XC:T2=YC
1220  FLAG=1:IF TR(1,Z)=0 GOTO 1400
1230  PRINT "Ich darf noch einmal schiessen !":PRINT
1240  IF FLAG=0 GOTO 1020
1300  IF RICHT=0 THEN X0=XC:Y0=YC:T1=XC:T3=XC:T2=YC:T4=YC:RICHT=INT(RND(1.99))*2
:R1=9
1310  IF RICHT=R1 THEN XC=XC+CX:YC=YC+CY:IF XC>=0 AND XC<=9 AND YC>=0 AND YC<=9
GOTO 1050
1320  RICHT=RICHT MOD 4+1:R1=RICHT:XC=X0:YC=Y0:CX=0:CY=0
1330  IF RICHT=1 THEN CX=+1:GOTO 1310
1340  IF RICHT=2 THEN CX=-1:GOTO 1310
1350  IF RICHT=3 THEN CY=+1:GOTO 1310
1360  IF RICHT=4 THEN CY=-1:GOTO 1310
1370  GOTO 1320
1400  GOSUB 14000:VSC=VSC+1:FLAG=0:RICHT=0
1410  FOR X=T1 TO T3+2:FOR Y=T2 TO T4+2
1420  SF(0,X,Y)=10:NEXT Y:NEXT X
1430  IF VSC<>5 GOTO 1230
1440  FOR Y=0 TO 9:FOR X=0 TO 9
1450  IF SF(1,X+1,Y+1) MOD 10<>0 THEN FILL 201+X*10,141-Y*10 209+X*10,149-Y*10 2
3
1460  NEXT:NEXT
1470  IF GETC<>0 GOTO 1470
1480  A1$="** D A I **":CALLM #D6DA:COLORG 1 1 1 1:GOSUB 7000
1490  GOTO 200
2000  REM
2001  REM ***** SCHUSS DES SPIELERS *****
2002  REM
```

```
2010  PRINT CHR$(12);
2020  PRINT "Bewege den Zielpunkt mit den CURSOR-Tasten !"
2030  PRINT "Schiesse mit der Leertaste !":CURSOR 0,0
2100  FILL 201+XS*10,141-YS*10 209+XS*10,149-YS*10 17
2110  G=GETC:IF G=0 GOTO 2110
2120  FILL 201+XS*10,141-YS*10 209+XS*10,149-YS*10 16
2130  IF G=32 GOTO 2200
2140  IF G=16 AND YS>0 THEN YS=YS-1
2150  IF G=17 AND YS<9 THEN YS=YS+1
2160  IF G=18 AND XS>0 THEN XS=XS-1
2170  IF G=19 AND XS<9 THEN XS=XS+1
2180  GOTO 2100
2200  Z=SF(1,XS+1,YS+1):SF(1,XS+1,YS+1)=10
2210  IF Z=10 GOTO 2500:IF Z<>0 GOTO 2400
2300  DRAW 201+XS*10,141-YS*10 209+XS*10,149-YS*10 22
2310  DRAW 201+XS*10,149-YS*10 209+XS*10,141-YS*10 22
2320  GOSUB 12000:RETURN
2400  FILL 201+XS*10,141-YS*10 209+XS*10,149-YS*10 22
2410  TR(0,Z)=TR(0,Z)-1:GOSUB 13000
2420  IF TR(0,Z)=0 THEN PRINT "Versenkt":GOSUB 14000:VSS=VSS+1:IF VSS=5 THEN WAI
T TIME 150:COLORG 1 1 1 1:GOSUB 7000:GOTO 200
2430  PRINT "Du darfst noch einmal schiessen !":WAIT TIME 50
2440  GOTO 2000
2500  PRINT "Auf diesen Punkt hast du schon geschossen !";
2510  WAIT TIME 50:GOTO 2000
3000  REM
3001  REM ***** UP. SPIELBEGINN *****
3002  REM
3010  PRINT CHR$(12):INPUT " Wie heisst du mit Vorname ";A1$:L=LEN(A1$)
3020  IF L>11 THEN PRINT :PRINT " Der Name ist zu lang !!":WAIT TIME 100:GOTO 30
00
3030  PRINT CHR$(12):PRINT A1$;" ; beginne mit dem Spiel !"
3040  IF L<8 THEN A1$=" "+A1$+" ":FOR L=L TO 8 STEP 2:A1$="*"+A1$+"*":NEXT
3050  WAIT TIME 100:PRINT CHR$(12):RETURN
4000  REM
4001  REM ***** UP. COMPUTERSCHIFFE SETZEN *****
4002  REM
4010  PRINT CHR$(12);"Ich setze meine Schiffe !";
4020  SC=1:RESTORE
4030  FOR I=1 TO 5:READ S$,S
4040  X=RND(10):Y=RND(10):G=RND(4)
4050  IF X=10 OR Y=10 OR G=4 GOTO 4040
4060  GOSUB 11000:IF OK=0 GOTO 4040
4070  NEXT I:RETURN
5000  REM
5001  REM ***** UP. SPIELERSCHIFFE SETZEN *****
5002  REM
5010  PRINT CHR$(12);
5020  PRINT "Bewege den Startpunkt des Schiffes mit den CURSOR-Tasten !"
5030  PRINT "Gebe die Richtung mit den SHIFT-CURSOR-Tasten !"
5040  X=0:Y=0:SC=0:RESTORE
5050  FOR I=1 TO 5
5060  READ S$,S:CURSOR 0,0:PRINT SPC(59);:CURSOR 0,0:PRINT S$;
5100  FILL 41+X*10,141-Y*10 49+X*10,149-Y*10 17
5110  G=GETC-16:IF G<0 OR G>8 GOTO 5110
5120  IF G>3 THEN G=G-4:GOTO 5200
5130  FILL 41+X*10,141-Y*10 49+X*10,149-Y*10 16
5140  IF G=0 AND Y>0 THEN Y=Y-1
5150  IF G=1 AND Y<9 THEN Y=Y+1
5160  IF G=2 AND X>0 THEN X=X-1
5170  IF G=3 AND X<9 THEN X=X+1
5180  GOTO 5100
5200  SC=0:GOSUB 11000:IF OK=0 GOTO 5100
```



```
5210  NEXT I:PRINT CHR$(12);
5220  RETURN
5300  DATA Fuenfer,4,Vierer,3,Dreier,2,Dreier,2,Zweier,1
6000  REM
6001  REM ***** UP. SPIELFELD ZEICHNEN *****
6002  REM
6010  MODE 6:COLORG 12 0 7 15:COLORT 12 0 0 0:MODE 6A
6020  FOR A=0 TO 100 STEP 10
6030  DRAW 40+A,50 40+A,150 0:DRAW A+200,50 A+200,150 0
6040  DRAW 40,A+50 140,A+50 0:DRAW 200,A+50 300,A+50 0
6050  NEXT A
6100  A$="0123456789      0123456789":X=42:Y=155:HS=10:C=0:F=0:GOSUB 10000
6110  A$="0 01 12 23 34 45 56 67 78 89 9":X=27:Y=141:HS=80:VS=10:GOSUB 10000
6120  A$="S S":X=87:Y=170:GOSUB 10000
6130  A$="Z Z":X=15:Y=96:GOSUB 10000
6140  A$="DEINE SCHIFFE                    DEINE":X=50:Y=200:HS=6:GOSUB 10000
6150  A$="MEINE SCHUSSVERSUCHE          SCHUSSVERSUCHE":X=30:Y=185:GOSUB 10000
6160  RETURN
7000  REM
7001  REM ***** UP. TITELBILD / SCHLUSSBILD *****
7002  REM
7005  MODE 6:MODE 6:COLORG 1 12 8 0
7010  IF A1$="" THEN A1$="SEESCHLACHT"
7015  FILL 0,50 XMAX,YMAX 12:FILL 130,40 230,60 8
7020  FOR Z=0 TO 20:DRAW 130-Z,40+Z 130,40+Z 8:NEXT Z
7030  FOR Z=0 TO 6:DRAW 230+Z/2,40+Z/2 230+Z,60 8:NEXT Z
7040  FOR Z=1 TO 3:DRAW 130+2*Z,40 XMAX,10*Z 12:NEXT Z
7100  RESTORE
7110  READ OB$:IF OB$<>"TITELBILD" GOTO 7110
7120  FOR Z=1 TO 8:READ X1,Y1,X2,Y2:FILL X1,Y1 X2,Y2 8:NEXT Z
7130  FILL 170,70 171,71 0
7140  FOR Z=1 TO 18:READ X1,Y1,X2,Y2:DRAW X1,Y1 X2,Y2 0:NEXT Z
7200  FOR BZ=0 TO LEN(A1$)-1:IF GETC=32 THEN RETURN
7210  Y=200:C=1:F=2:A$=MID$(A1$,BZ,1):X=20+15*BZ:GOSUB 7300
7220  COLORG 1 15 8 0:DRAW 170,72 X+6,Y+8 0:COLORG 1 12 8 0
7230  DRAW 170,72 X+6,Y+8 21:GOSUB 10000:GOSUB 13020:NEXT BZ
7240  GOSUB 7300:WAIT TIME 3:IF GETC=0 GOTO 7240
7250  RETURN
7300  FR=0:FOR ZR=1 TO 5:GOSUB 7400:NEXT ZR
7310  FR=12:FOR ZR=5 TO 1 STEP -1:GOSUB 7400:NEXT ZR
7320  RETURN
7400  DRAW 187+ZR,95 187+ZR,100 FR:DRAW 187-ZR,95 187-ZR,100 FR
7410  RETURN
7500  DATA TITELBILD
7510  DATA 120,60,135,65,  140,60,160,65
7520  DATA 143,65,158,70,  165,60,195,65
7530  DATA 170,65,195,70,  180,70,195,85
7540  DATA 185,85,190,90,  205,60,220,65
7600  DATA 120,62,110,64,  120,63,110,65
7610  DATA 143,67,133,72,  143,68,133,73
7620  DATA 220,62,230,67,  220,63,230,68
7630  DATA 110,60,235,60,  187,90,187,100
7640  DATA 112,58,234,58,  170,65,195,65
7650  DATA 180,70,195,70,  180,75,195,75
7660  DATA 180,77,195,77,  180,80,195,80
7670  DATA 180,75,180,80,  185,75,185,80
7680  DATA 190,75,190,80,  195,75,195,80
8000  REM
8001  REM ***** UP. INITIALISIERUNG *****
```

```
8002  REM
8010  DIM SF(1,11,11),TR(1,5):R1=8
8020  READ A$:IF A$<>"TREFFER" GOTO 8020
8030  FOR N=1 TO 5:READ M:TR(0,N)=M:TR(1,N)=M:NEXT N
8040  SOUND OFF :ENVELOPE 0 15
8050  ENVELOPE 1 9,2;12,3;15,5;12,3;9,2;0
8060  RETURN
8070  DATA TREFFER,5,4,3,3,2
10000 REM
10001 REM ** UP. FAST GRAF TEXT **
10002 REM
10010 GOTO 60000:REM BEI FGT DIESE ZEILE LOESCHEN
10020 POKE #2F0,C:POKE #2F1,F
10030 POKE #2F2,X MOD 256:POKE #2F3,X SHR 8
10040 POKE #2F4,Y:POKE #2F5,HS:POKE #2F6,VS
10050 CALLM #300,A$:RETURN
11000 REM
11001 REM ** UP. AUFSTELLUNG DER SCHIFFE **
11002 REM
11010 OK=0:F=0
11020 IF G=0 THEN X1=X:X2=X+2:Y1=Y-S:Y2=Y+2:IF Y1<0 THEN RETURN
11030 IF G=1 THEN X1=X:X2=X+2:Y1=Y:Y2=Y+S+2:IF Y2>11 THEN RETURN
11040 IF G=2 THEN Y1=Y:Y2=Y+2:X1=X-S:X2=X+2:IF X1<0 THEN RETURN
11050 IF G=3 THEN Y1=Y:Y2=Y+2:X1=X:X2=X+S+2:IF X2>11 THEN RETURN
11060 FOR K=X1 TO X2:FOR J=Y1 TO Y2
11070 IF SF(SC,K,J)<>0 THEN F=1
11080 NEXT J:NEXT K:IF F=1 THEN RETURN
11090 FOR K=X1+1 TO X2-1:FOR J=Y1+1 TO Y2-1:SF(SC,K,J)=I
11100 IF SC=0 THEN FILL 31+K*10,151-J*10 39+K*10,159-J*10 7
11110 NEXT J:NEXT K:OK=1
11120 RETURN
12000 REM
12001 REM ** UP. SOUND WASSER **
12002 REM
12010 FOR A=8000 TO 1000 STEP -100:SOUND 0 0 5 0 FREQ(A):NEXT
12020 SOUND OFF
12030 NOISE 1 15:WAIT TIME 15:NOISE OFF
12040 RETURN
13000 REM
13001 REM ** UP. SOUND TREFFER **
13002 REM
13010 FOR A=8000 TO 1000 STEP -100:SOUND 0 0 5 0 FREQ(A):NEXT
13020 FOR M0=0 TO 15:FOR M1=0 TO 22
13030 POKE #FC00,M0+M1
13040 NEXT M1:NEXT M0
13050 SOUND OFF :RETURN
14000 REM
14001 REM ** UP. SOUND VERSENKT **
14002 REM
14010 SOUND 0 0 15 0 FREQ(2000)
14020 FOR M0=1 TO 8
14030 SOUND 0 0 15 2 FREQ(1000):WAIT TIME 5
14040 SOUND 0 0 15 2 FREQ(2000):WAIT TIME 5
14050 NEXT M0:SOUND OFF
14060 RETURN
60000 REM
60001 REM ** SLOW GRAF TEXT **
60002 REM
60010 IF STARTZ$="OK" GOTO 60100
60020 DIM CARZ$(90)
60030 READ AZ$:IF AZ$<>"START Z" GOTO 60030
60040 STARTZ$="OK"
60050 FOR ZZ=32 TO 90:READ AZ$,CARZ$(ZZ):NEXT ZZ
```



```
60100 XZ=X:YZ=Y:X1Z=XZ:Y1Z=YZ
60130 FOR MZ=0 TO LEN(A$)-1
60150 TZ$=MID$(A$,MZ,1)
60160 IF ASC(TZ$)<32 OR ASC(TZ$)>94 THEN TZ$=" "
60170 GRZ$=CARZ$(ASC(TZ$))
60180 FOR NZ=0 TO LEN(GRZ$)-1 STEP 4
60200 IF MID$(GRZ$,NZ,1)="/" THEN XZ=XZ+(HS*(F+1)):GOTO 60270
60210 JZ1=XZ+VAL(MID$(GRZ$,NZ,1))*(F+1)
60220 JZ2=YZ+VAL(MID$(GRZ$,NZ+1,1))*(F+1)
60230 JZ3=XZ+VAL(MID$(GRZ$,NZ+2,1))*(F+1)
60240 JZ4=YZ+VAL(MID$(GRZ$,NZ+3,1))*(F+1)
60250 DRAW JZ1,JZ2 JZ3,JZ4 C:NEXT NZ
60270 IF XZ+HS*(F+1)>=XMAX THEN XZ=X1Z:YZ=YZ-VS*(F+1)
60280 NEXT MZ:RETURN
60600 REM
60601 REM DATEN FUER SLOW GRAF TEXT
60602 REM
60605 DATA START Z
60610 DATA BLANK ,/, ! ,/, ANFUEHR ,/
60620 DATA # ,/, $ ,/, % ,/, & ,/, ' ,/, ( ,/, ) ,/
60655 DATA * ,125616523137/
60660 DATA + ,/, KOMMA ,/, - ,/, . ,/, / ,/
60685 DATA 0 ,121612212141415252565647472727161256/
60690 DATA 1 ,214131372637/
60695 DATA 2 ,5111111212333355555565647472716 27/
60700 DATA 3 ,122121414152525353343456565757 17/
60705 DATA 4 ,4147531313141447/
60710 DATA 5 ,122121414152525454454515151717 57/
60715 DATA 6 ,121512212141415252535344441415373757/
60720 DATA 7 ,212222323 5656575717/
60725 DATA 8 ,4121211212131324244444533535252412415151616272747475656555544/
60730 DATA 9 ,113131533536564747272716161515242454/
60735 DATA : ,/, ; ,/, < ,/, = ,/, > ,/, ? ,/, @ ,/
60770 DATA A ,11155155135315373755/
60775 DATA B ,111717471444114147565655554444535352524 1/
60780 DATA C ,121616272747475612212141415 2/
60785 DATA D ,11171141415252565474717/
60790 DATA E ,1117115114441757/
60795 DATA F ,111714441757/
60800 DATA G ,121616272757122121515153534 3/
60805 DATA H ,111714545157/
60810 DATA I ,214131372747/
60815 DATA J ,1221214141525257/
60820 DATA K ,111713572451/
60825 DATA L ,11171151/
60830 DATA M ,111717353534355757 51/
60835 DATA N ,111751571652/
60840 DATA O ,121616272747475656525241412121 12/
60845 DATA P ,11171444445555565474717/
60850 DATA Q ,121616272747475656531221213131533351/
60855 DATA R ,111717474756565555444414245 1/
60860 DATA S ,122121414152525353444424241515161627274747 56/
60865 DATA T ,17573137/
60870 DATA U ,111711515157/
60875 DATA V ,1317535713313153/
60880 DATA W ,111751571133335133 4/
60885 DATA X ,111217165152575612561652/
60890 DATA Y ,16175657163434563134/
60895 DATA Z ,17575756561212111151/
```

Alternative display

AN ALTERNATIVE "DISPLAY" UTILITY

The international nature of DAInamic happily means that programmes originate from members in many different countries. Consequently, particularly with games programmes, there is often a need to change a few words of text or screened instructions, from the author's language to ones own language. That task is simple with a BASIC programme, but with a machine language programme it is difficult to find where the text is hidden unless one has the source code.

The ALTERNATIVE "DISPLAY" programme is an extended version of the DAI's Utility Display routine and enables text to be found quickly. It is similar to the display facility in CP/M 2.2. It allows a machine code programme in memory to be scanned, showing at the right of the screen any memory contents that fall within the ASCII range of character codes. Once found, text can be altered in Utility, using the S command to substitute appropriate ASCII codes, character by character. Remember that the replacement block of text must be exactly the same length as the original otherwise any machine code further on in the programnme would be out of step and would fail. For instance, in the example below, replacing 'Reihe 5 ' by 'Row 5 ' would require two extra spaces (#20) to keep the string length the same; 0BF8-0BFF would thus become 52 6F 77 20 35 20 20 20.

Before running the programme set the 3rd colour of COLORT with a suitably inconspicuous colour for the divider lines between the compacted bytes on the screen display; for example: COLORT 8 0 7 0. Load the machine language programme that needs to be altered, then the Alternative Display machine language programme. Run it with GB000. Next enter the start and end addresses of the programme to be scanned. The display can be halted by pressing any key, and restarted with the space-bar.

Example of display:

```
0BC0  0BE13ED9327C7D210300220203C9BE5C  ..>.2;}!.."....\
0BD0  D5BDDD1D5320502041204320452020 2D  ....S P A C E  -
0BE0  202049204E20562041204420452052 20    I N V A D E R
0BF0  5312BD50DABCD11752656968652035 20  S..P....Reihe 5
0C00  202E2E2E2020363020507 56E6B746514    ...  60 Punkte.
0C10  BCCADBBC4B1752656968652034202 02E  ....K.Reihe 4  .
0C20  2E2E2020353020507 56E6B746514BC44  ..  50 Punkte..D
0C30  DCBBC51752656968652033202 02E2E2E  ....Reihe 3  ...
0C40  2020343020507 56E6B746514BBBEDDBB    40 Punkte.....
0C50  3F1752656968652032202 02E2E2E2020  ?.Reihe 2  ...
0C60  333020507 56E6B746514BB38DEBAB917  30 Punkte..8....
0C70  5265696865203120202E2E2E20203230  Reihe 1  ...  20
0C80  2050756E6B74651450756E6B74653A20   Punkte.Punkte:
0C90  004C617365723A20486967687363 6F72  .Laser: Highscor
0CA0  653A2047616D65206F7665724E657565  e: Game overNeue
0CB0  7320537069656C20284A2F4E29203F44  s Spiel (J/N) ?D
```



```
0000                    TITL    'ALTERNATIVE "DISPLAY" UTILITY'
0000            ;Author:   Bill Read, Bristol (0272) 424290.  October 1985
0000  @=DAD4 PMSG    EQU     0DAD4H          ;Print a message
0000  @=DD60 OUTC    EQU     0DD60H          ;Output a character
0000  @=ED01 SPACE   EQU     0ED01H          ;Output a space
0000            ;
0000                    ORG     0B000H          ;NOTE: When using the SPL
B000            ; assembler, memory area #B000-#B190 is not free for this
B000            ; display programme: assembly must therefore be done with
B000            ; an offset, and the object code moved (in UTility) to #B000
B000            ; later, when SPL is no longer needed. eg: A6000 will
B000            ; assemble at #1000-#1190
B000            ;
B000            ;Start. Print instructions:
B000  2187B0            LXI H   HEADNG
B003  CDD4DA            CALL    PMSG
B006            ;Set up screen:
B006  218EB3            LXI H   0B38EH          ;Screen, bottom
B009  118600            LXI D   86H             ;Value for step to next line
B00C  3E80              MVI A   80H             ;Value for colour byte
B00E  0E18              MVI C   18H             ;Counter,24 lines
B010  0611    NXTLIN    MVI B   11H             ;Counter,17 dividers
B012  E5                PUSH H                  ; to be coloured.
B013  77      SETCB     MOV M,A                 ;Set colour byte
B014  23                INX H                   ;Step to next but
B015  23                INX H                   ; one colour byte
B016  23                INX H
B017  23                INX H
B018  05                DCR B                   ;End of this line?
B019  C213B0            JNZ     SETCB           ;Loop if not
B01C  E1                POP H                   ;Move screen address
B01D  19                DAD D                   ; up 1 line
B01E  0D                DCR C                   ;Top line reached?
B01F  C210B0            JNZ     NXTLIN
B022            ;Get two addresses from keyboard:
B022  3E09    CMND1     MVI A   9H
B024  CD60DD            CALL    OUTC            ;Change prompt symbol.
B027  0E02              MVI C   2H              ;2 addresses wanted
B029  CDDEEA            CALL    0EADEH          ;Get them on stack
B02C  0D                DCR C
B02D  F262EA            JP      0EA62H          ;Error if only 1 address
B030  D1                POP D                   ;High address in DE
B031  E1                POP H                   ;Low address in HL
B032  E5      LOOP1     PUSH H                  ; and on stack
B033  CD3AED            CALL    0ED3AH          ;Carriage return
B036  CD18ED            CALL    0ED18H          ;Print address
B039  CD01ED            CALL    SPACE
B03C  CD01ED            CALL    SPACE           ;Print 2 spaces
B03F  7E      LOOP2     MOV A,M                 ;Get contents
B040  CD1DED            CALL    0ED1DH          ; and print it
B043  23                INX H
B044  7D                MOV A,L
B045  E60F              ANI     0FH             ;End of line?
B047  CA4DB0            JZ      TEXT            ;Then show text chars
B04A  C33FB0            JMP     LOOP2           ;Else next byte
B04D            ;Print any text bytes in right hand margin:
B04D  CD01ED  TEXT      CALL    SPACE
B050  CD01ED            CALL    SPACE
```

```
B053  E1            POP H                   ;Line start again
B054  AF     LOOP3  XRA A
B055  7E            MOV A,M                 ;Get byte
B056  FE20          CPI     20H             ;If it is an
B058  DA60B0        JC      CONTRL          ; ASCII code from
B05B  FE7E          CPI     7EH             ; #20 to #7D inclusive,
B05D  DA62B0        JC      SHOWIT          ; then display it.
B060  3E2E   CONTRL MVI A   2EH             ;Substitute dot if not
B062  CD60DD SHOWIT CALL    OUTC            ; ASCII and display it.
B065  CD80ED        CALL    0ED80H          ;INX H,compare with DE
B068  DA81B0        JC      DONE            ;If end addr reached
B06B  7D            MOV A,L
B06C  E60F          ANI     0FH             ;Line end?
B06E  C254B0        JNZ     LOOP3           ; no, test next byte
B071  CDBBD6 PAUSE  CALL    0D6BBH          ;Any key pressed?
B074  DA81B0        JC      DONE            ;If BREAK
B077  B7            ORA A
B078  CA32B0        JZ      LOOP1           ;No key pressed
B07B  CDDAD6        CALL    0D6DAH          ;Wait for space,
B07E  C332B0        JMP     LOOP1           ; then do next line
B081  CD3AED DONE   CALL    0ED3AH          ;Carriage return
B084  C322B0        JMP     CMND1           ;Await next command
B087        ;
B087  0C     HEADNG DB      0CH             ;Clear screen
B088  2E2E2E        DB      '... DISPLAY ... The byte dividing lines are'
B0B3  20636F        DB      ' coloured by'
B0BF  0D            DB      0DH
B0C0  202020        DB      '                 the 3rd colour of the current
B0ED  20434F        DB      ' COLORT set.'
B0F9  0D            DB      0DH
B0FA  205479        DB      ' Type start address [space] end address'
B121  205B72        DB      ' [return].'
B12B  0D            DB      0DH
B12C  205072        DB      ' Press any key to halt the display,'
B14F  207370        DB      ' space to continue.'
B162  0D            DB      0DH
B163  205072        DB      ' Press left arrow key to escape to UTility.'
B18E  0D0D00        DB      0DH,0DH,0H
B191        END     END
```

SHORTENED VERSION. If an assembler is not available and the programme is to be entered by means of the Substitute command, the following version without the text will be suitable. The only change is to make #B088 zero and omit everything beyond.

```
B000 21 87 B0 CD D4 DA 21 8E B3 11 86 00 3E 80 0E 18
B010 06 11 E5 77 23 23 23 23 05 C2 13 B0 E1 19 0D C2
B020 10 B0 3E 09 CD 60 DD 0E 02 CD DE EA 0D F2 62 EA
B030 D1 E1 E5 CD 3A ED CD 18 ED CD 01 ED CD 01 ED 7E
B040 CD 1D ED 23 7D E6 0F CA 4D B0 C3 3F B0 CD 01 ED
B050 CD 01 ED E1 AF 7E FE 20 DA 60 B0 FE 7E DA 62 B0
B060 3E 2E CD 60 DD CD 80 ED DA 81 B0 7D E6 0F C2 54
B070 B0 CD BB D6 DA 81 B0 B7 CA 32 B0 CD DA D6 C3 32
B080 B0 CD 3A ED C3 22 B0 0C 00
```


Assembly language p. 5

PROGRAMMING THE DAI IN MACHINE AND ASSEMBLY LANGUAGES
by C W Read

Part 5 - More arithmetic and flags.

It is instructive to examine how the microprocessor's various flags are set. A few example programmes running in the DAI's Utility monitor will provide suitable demonstrations and also introduce some more arithmetic instructions.

Utility commands L (Look) and X (examine) display the flag register contents as two Hex digits. It is not easy to determine from the display which flags are set because the hex equivalent of 8 bits is shown whereas only 5 bits are flags. The states of Sign (S), Zero (Z) and Auxiliary Carry (AC) flags are indicated by the 1st hex digit of the display; the Carry (C) and Parity (P) flags by the 2nd hex digit. Here is a quick reference list that shows the significance of each hex digit:

1st digit	Flag set	2nd digit	Flag set
0	none	2	neither
1	AC	3	C
4	Z	6	P
5	Z and AC	7	C and P
8	S		
9	S and AC		

Until now addition and subtraction has been limited to operands of one byte, using the instructions ADD r, ADI d, SUB r and SUI d, where d is a single byte of data and r can be either any register or the memory whose address is in register pair HL. Those four instructions set the carry flag if a carry or borrow is generated by the arithmetic operation. When adding or subtracting a value greater than one byte, any carry or borrow from the operation on the first pair of bytes has to be added to or subtracted from the second pair of bytes. The set of instructions for adding with the carry or subtracting with the borrow are:

```
ADC r     Add register or memory and Carry to A.
ACI d     Add Immediate data and Carry to A.
SBB r     Subtract register or memory and borrow from A.
SBI d     Subtract Immediate data and Borrow from A.
```

Here is a programme, short enough for the object code to be entered with the Utility S command. It can be made to add and subtract various values to show how the flags behave.

Example 1

```
Addr-  Object      Source code          Remarks
 ess    code
                PRTADR  EQU     0ED18H  ;ROM routine: prints an address
                        ORG     400H    ;Start at #0400
0400    00      START   NOP             ;No operation
0401    013428          LXI B   2834H   ;1st number into BC
0404    111219          LXI D   1912H   ;2nd number into DE
0407    79              MOV A,C         ;LSByte of 1st number to A
0408    83              ADD E           ;Add LSB of 2nd number. Ans in A
0409    00              NOP
040A    6F              MOV L,A         ;Save result in L
```

```
040B    78              MOV A,B         ;MSB 1st to A
040C    8A              ADC D           ;Add MSB 2nd and Carry to A
040D    00              NOP
040E    67              MOV H,A         ;Move result to H
040F    CD18ED          CALL    PRTADR  ; and print what is in HL
0412    C9              RET             ;(To Utility)
0413                    END
```

Enter the code and observe the action with the Utility's Look function: initialise first with command Z3 then type L400 400 40F. The result of the addition, #4146, should be printed as a continuation (C) line, after line 040F.

You will notice that the flag (F) register shows 02 (no flags set) for the first addition, but that it changes to 16 for the second addition, showing that both Auxiliary Carry and Parity flags have been set. AC is set when there is a carry actually within a byte between the low nibble and the high nibble, in this case when the 8 and 9 were added. P is set when the result in the accumulator has an even number of 1 bits, or no 1 bits. Here is what happened after instruction MOV A,B at 040B; the bytes are shown split into two nibbles because each nibble represents one hex digit, ie: we are working in Binary Coded Decimal instead of true binary.

```
Value in accumulator              #28 = 0010 1000
Value in register D               #19 = 0001 1001
Value in carry flag                0 =         0
Add low nibbles + carry              =    1 0001   the carry from low
                                                   to high sets AC flag
Add high nibbles + aux carry         = 0100
Total                             #41 = 0100 0001   even number of 1s
                                                   in result sets P flag
```

The P and AC flags are of little use in programming. The AC flag is only used by the microprocessor (see description of DAA instruction later). Conditional branching instructions are available for the P flag but not for AC. They, with their op codes in brackets, are:

RPO (E0) Return when Parity Odd.	RPE (E8) Return when Parity Even.
JPO (E2) Jump when Parity Odd.	JPE (EA) Jump when Parity Even.
CPO (E4) Call when Parity Odd.	CPE (EC) Call when Parity Even.

In Example 1 the DAI produced an answer that was the hex sum of two hex numbers, #2834 and #1912. It assumed that those values, loaded into BC and DE, were hex. There is a microprocesor instruction which treats them as decimal values: it is DAA (op code 27).

DAA means Decimal Adjustment of Accumulator. A DAA instruction must be given after each "add" instruction. In the programme example above replace the NOPs (00) at 0409 and 040D by DAAs (27), and re-run it. The result printed will be the decimal sum of the two decimal values in BC and DE. DAA adjusts the value in the accumulator to form two Binary Coded Decimal digits. Reference is made to the AC flag because each nibble is dealt with seperately. DAA operates according to the following rules:

If the value in the least significant nibble is greater than 9, or the Auxiliary Carry flag is set, 6 is added to the nibble.

If the value in the most significant nibble is now greater than 9, or the Carry flag is set, 6 is added to the nibble.

Example 2

Replace the add instructions in Example 1 by subtract instructions (DAA works only after additions): change ADD E to SUB E, ADC D to SBB D, and delete the DAAs by substituting as follows:-

S0408 83-93 27-00 and S040C 8A-9A 27-00

Run the programme again, after Z3, with L400 400 40F. This time flag AC is set on the 1st subtraction (34-12). This was because the microprocessor performed the subtraction by adding the two's complement of the value in E to the accumulator. After instruction 79 (MOV A,C) at address 0407 the situaton is:

```
Value in accumulator              #34 =              0011 0100
Value in register E               #12 = 0001 0010
Now perform instruction SUB E:
                                                                  Flags
    (1) complement value in E          = 1110 1101                set
    (2) +1 (makes 2's complement)      = 1110 1110     1110 1110
    (3) add to accumulator                           1 0010 0010  AC,C,P
    (4) complement the carry      #22 =              0 0010 0010  AC,P
```

Note that the carry flag is also complemented. In this example a small number has been subtracted from a larger so there is no borrow, hence Carry=0.

Example 3

Now consider the more important flags, S, Z, C, and how they are affected by arithmetic operations. Restore the programme to its Example 1 state, with ADD E and ADC D instead of SUB E and SBB D. Substitute the 1st number #2834 by #75FE thus: S0402 34-FE 28-75. Now the Look exercise shows the Flag register as 13 for the 1st addition, indicating that a carry has been generated. The 2nd addition adds the carry and the remaining bytes, and the Flag register changes to 82; the 8 means that the Sign flag has been set to indicate a negative value. But, how does adding 2 positive numbers provide a negative result? The way the microprocessor recognizes negative values is by the most significant bit in the accumulator: if that is 1, no matter how it came to be so, the sign flag is set; if the bit is 0 the flag is not set. In this case our addition has produced a sum large enough to put a 1 in the most sgnificant bit. In practice this anomoly is not a problem because a programmer would not expect to put commands like, for example, a conditional "jump if minus" after an addition. It is however worth remembering that the sign flag has its peculiarities. Fig 1 shows how each flag is set or not when Example 3 is run.

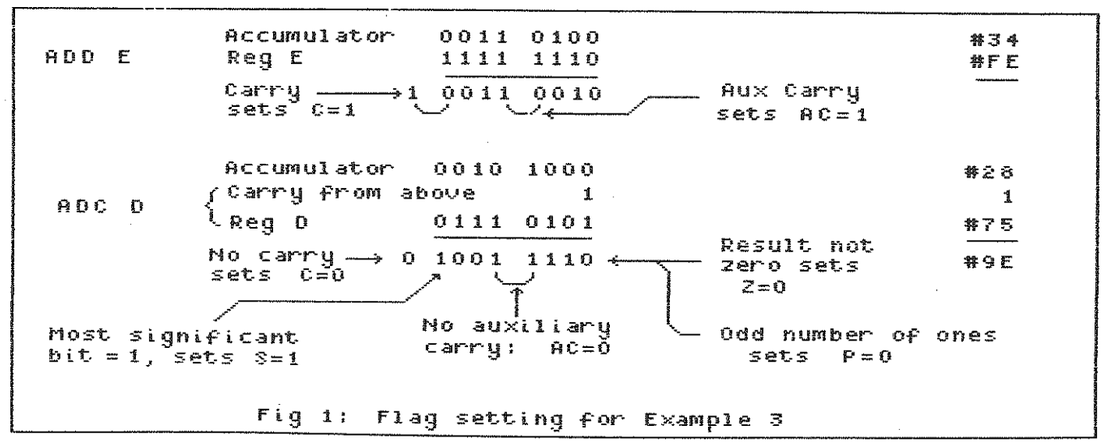


Fig 1: Flag setting for Example 3

Memory addresses are in the form of 4-bit hex numbers and it is often necessary to programme arithmetic operations on them. There is a double byte add instruction (DAD) specifically for that purpose. It adds the value in a nominated register pair to the value in the HL pair, leaving the result in HL. The mnemonics, with their op codes in brackets, are:

DAD B (09), DAD D (19), DAD H (29) and DAD SP (39).

Here is an example, using DAD D to increment a screen RAM address (held in HL). The increment, #0086 is held in DE. #86 is the number of bytes between similar positions on adjacent screen lines. 0 is put in memory at each address selected, thus drawing a vertical line from bottom to top of the screen. Call the programme from Basic with CALLM #400.

```
     Obj code          ORG     400H
0400  21A3B3           LXI H   0B3A3H   ;Bottom centre of screen.
0403  118600           LXI D   86H      ;Increment to next line.
0406  AF               XRA A           ;Zero the accumulator.
0407  0E18             MVI C   18H      ;C counts 24 screen lines.
0409  77        LOOP   MOV M,A          ;0 to screen character position
040A  19               DAD D           ;Add #86 to address in HL
040B  0D               DCR C           ;Reduce line count by 1
040C  C20804           JNZ     LOOP     ;Repeat until count is 0
040F  C9               RET             ;End of routine.
```

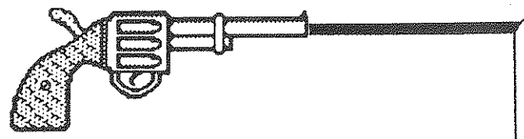
There is no corresponding "double byte subtract" instruction but the effect can be achieved by loading the nominated register pair with the two's complement of the value so that DAD r can add a negative value to HL. Here is how to convert a 4-digit value held in the DE register pair to its twos complement:-

```
Object code    Source code
    7A          MOV A,D          Move value in D to accumulator
    2F          CMA              complement it
    57          MOV D,A          and return it to D
    7B          MOV A,E
    2F          CMA              Complement E
    5F          MOV E,A
    13          INX D            Add 1 to the complemented value
```

There is a ROM routine similar to this at DE26 to DE2F inclusive. It is known by the label NEGHL and it converts a value in the HL pair to two's complement. The following section of code uses the NEGHL routine to change #86 to -#86. We can use it, with the last part of the above line drawing routine, to draw the line from top to bottom of the screen. Addressing screen colour bytes instead of data bytes, will draw a line of different width, in the 3rd colour of the COLORT set. Call this one from Basic with CALLM #410:-

```
              NEGHL   EQU     0DE26H
                      ORG     410H
0410  218600          LXI H   86H     ;Value of increment
0413  CD26DE          CALL    NEGHL   ;Make it negative and
0416  EB              XCHG            ; put it in DE.
0417  21AEBF          LXI H   0BFAEH  ;Address of screen,top centre
041A  3E01            MVI A   01H     ;Byte 01 (or your choice) into A
041C  C30704          JMP     407H    ;Jump into previous routine to
                                        put 01 into a colour byte at
                                        centre of each screen line.
```

To be continued.



International DAI Club

Chers amis,

Pour vous rappeler les merveilleux moments que vous avez connus cet été, vous trouverez à la page 51, la recette d'un délicieux cocktail au nom prédestiné. Mais n'en abusez pas car un verre ca va, trois verres bonjour les dégats ... Pour vous faire patienter jusqu'au prochaine vacances ,nous ne pouvons que vous conseiller de rester fidèle à DAIclic.

Dès à présent, nous pouvons vous signaler que DAIclic paraîtra l'année prochaine, sous réserve d'un nouveau changement de présentation, et fêtera ainsi son deuxième anniversaire.

La rédaction de la revue va également 'bouger' sous peu. En effet, la mise au point d'une revue telle que DAIclic prend énormément de temps, et les responsables actuels n'ont plus toujours le temps de s'occuper de DAIclic. L'édition d'une revue comme DAIclic peut paraître très simple pour certains, mais il n'en est rien, cela demande énormément de temps à un certain nombre de bénévoles. Pour ne vous citer qu'un exemple : savez-vous qu'un des responsables du club passe une demi-journée tous les deux mois uniquement pour expédier les revues aux membres, mais avant l'expédition il y a eu tout le travail de la rédaction. Il nous est donc difficile de faire paraître une revue comme DAIclic dans les délais prévus si l'aide de nos membres reste si faible...

Nous nous permettons une fois de plus de solliciter votre aide, ce que certains font déjà "dans l'ombre" depuis un petit temps et nous les en remerçions. Toutes les propositions seront les bienvenues, mais dès à présent sachez que nous risquons de tomber en mal d'articles palpitants !!! De plus que ceux qui nous écrivent fassent un effort de présentation...cela nous simplifie tellement la tâche. Deux petits exemples : les articles de messieurs L. Legry et R. De Lombaert que vous trouverez dans ce numéro nous sont parvenus quasiment prêts pour l'impression ! Nous ne pouvons que remercier ces personnes et encourager nos membres à faire de même, ce qui nous permettra de consacrer moins de temps à l' élaboration de la revue (impression, découpages et collages en tout genre !), temps qui sera utilisé plus intelligemment à la création de nouvelles activités DAI-istiques.

Nous vous souhaitons une agréable lecture et vous donnons rendez-vous dans deux mois.

IDC rédaction.

X-BUS

CARTE DE SORTIE DU X-BUS POUR D.A.I

FEVRIER 1986

D.A.I.C.L.I.C

Laurent LEGRY
628 Bld LAHURE
59500 DOUAI (FRANCE)

1-PRESENTATION

Peut-être l'avez vous déja remarqué, le DAI possède 512 adresses inutilisées comprises entre 0F900 et 0FAFFH.
Pensant qu'il serait utile de pouvoir les exploiter, j'ai réalisé une carte se connectant sur le X-BUS à la place de celle du TOS et permettant de sortir les différents signaux des bus, sans risque pour la machine (une erreur de cablage ayant parfois des conséquences désastreuses).
Cette réalisation isole, à l'aide de buffers, la machine de ce que vous pourriez y connecter.

2-QUELQUES COMMENTAIRES A PROPOS DU SCHEMA DE LA CARTE

a - Modification du X-BUS

Certains signaux utilisés ne se trouvent pas sur le X-BUS de votre DAI tel qu'il est décrit dans le précédent article sur l'interface CENTRONICS. C'est le cas de DRT1, DTR2, FF/, FA/, F9/ et F0.
Excéptés DTR1 et DTR2, ces signaux sont présents sur les pattes de certains circuits du DAI : l'une des figures de l'article sur l'interface CENTRONICS vous précise où les trouver. Il vous faut donc ramener le X-BUS, dans la configuration indiquée sur la figure 2, en éffectuant les modifications décrites ci-dessous.

Les broches 1,3,5,7,9,11,13,15,17,19 du connecteur X-BUS sont reliées entre-elles et mises à la masse. L'opération la plus délicate consiste à démonter la carte mère et à retirer le cuivre reliant ces pattes entre elles à l'aide d'un cutter,d'un fer à souder et d'une pompe à dessouder. A l'issue de cette manipulation, les broches 3, 5, 7, 9, 11, 13, 15, 17, 19 doivent être 'en l'air' et la broche 1 à la masse.

Cette manipulation, assez facile, doit être effectuée avec le plus grand soin. Je ne peux, bien sûr, garantir toute dégradation de votre machine qui pourrait en résulter. Il faut ensuite, à l'aide de fin fil, relier ces différentes broches aux circuits d'où sortent les signaux, de manière à ramener le X-BUS dans la configuration représentée sur la figure 2.

Votre machine peut maintenant recevoir la carte de buffers tout en conservant la compatibilité avec la carte du TOS.

En ce qui concerne DTR1 et DTR2 vous pouvez vous reporter à l'article sur l'interface CENTRONICS. Sachez cependant qu'ils ne sont pas indispensables.

b - Fonctionnement de la carte

Les adresses A0 à A11 et les différents signaux de contrôle sortent de buffers unidirectionnels 74LS541. Les signaux sont toujours présents en sortie de ces circuits, les pattes de validation (1 et 19) étant maintenues actives (reliées à la masse).

Les données transitent par un buffer bidirectionnel 74LS245. La broche indiquant la direction dans laquelle transitent les signaux (1) est reliée directement au MEMORY READ/.

Le montage que vous voyez sur la patte OUTPUT ENABLE (19) est destiné a valider les sorties des buffers, uniquement lorsqu'on adresse des circuits sur (0F000H => 0F7FFH) ou (0F900H => 0FAFFH) ou (0FF00H =>0FFFFH et seulement en lecture); cette dernière particularité est notemment utile pour la carte centronics.

Les signaux de contrôle entrant dans la machine ne passent pas par les buffers.

REMARQUE : Le 74LS245 et le 74LS541 possèdent tout deux la caracteristique interréssante d'avoir les entrées d'un coté du circuit et les sorties de l'autre, ce qui facilite le cablage.

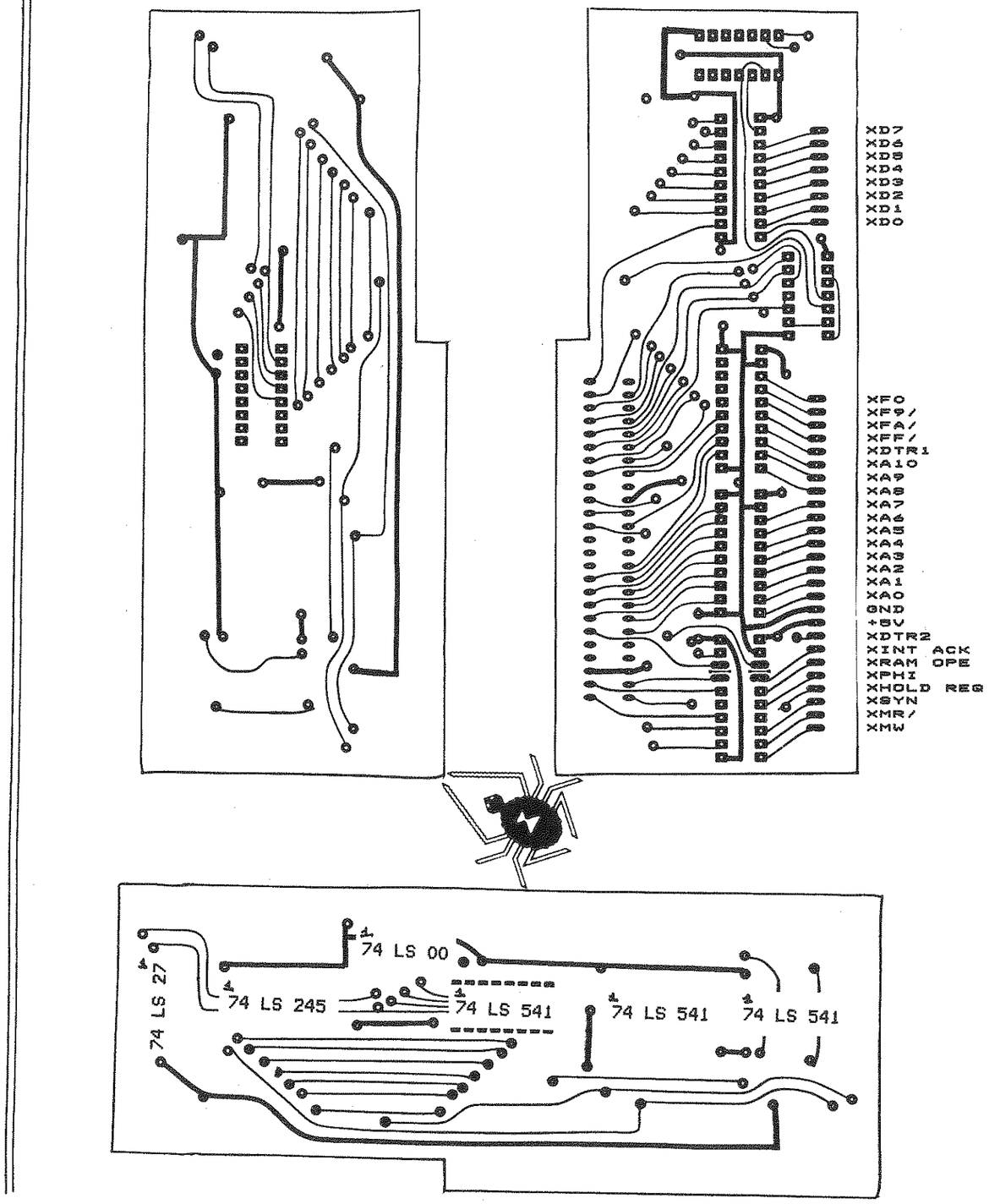
3-REALISATION

Vous trouverez ci dessous les photocopies des transparents qui m'ont servi à réaliser cette carte double face.
Si vous les reproduisez par transparence, les transferts devront être placés contre le cuivre. Cette remarque est également valable pour la carte CENTRONICS.

Le connecteur utilisé est le même que celui de la carte du TOS (c'est d'ailleurs celui que j'ai récupéré). Les composants doivent se trouver facilement chez votre revendeur habituel.
La carte terminée, il vous restera à ramener les sorties sur le connecteur de votre choix.

Cette carte remplace celle du TOS. Pour utiliser le DCR, il vous faudra réaliser la carte CENTRONICS (vous n'êtes pas obligé de mettre tous les circuits : seuls l'EPROM et le 74LS02 sont nécéssaires pour le contrôle du DCR). Attention, vous ne pouvez pas directement adapter votre vieille carte en la connectant à la sortie des buffers car elle utilise les signaux A11 à A15 du bus d'adresses.

MASQUES ET SCHEMA D'IMPLANTATION



4-CONCLUSION

A condition que la manipulation de la carte mè re de votre DAI ne vous effraie pas trop, ce montage vous donne la possibilité d'interfacer, sans risque, la machine avec pratiquement tout les périphériques dont vous avez envie.
Pour celà, vous disposez des adresses suivantes :

a - les adresses libres

Vous avez la possibilité d'ajouter des circuits :
- de OF900H à OF9FFH par l'intermédiaire du signal F9/ (actif lorsque OF9XXH est présent sur le bus d'adresses.
- de OFA00H à OFAFFH par l'intermédiaire du signal FA/.

Ces adresses ne sont utilisées, ni par la machine, ni par le contrôleur du DCR. Cependant, certaines réalisations, parues dans DAINAMIC ou d'autres revues, peuvent les utiliser.

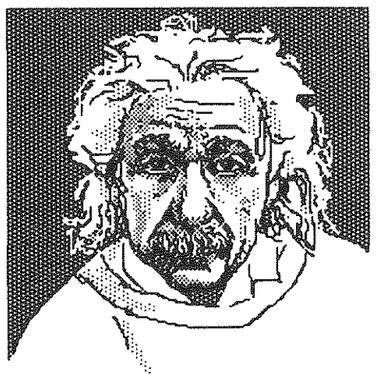
b - les adresses déjà utilisées mais auxquelles vous avez accès

Il s'agit des adresses du TOS et des adresses de OFX00H à OFXFFH en écriture.
- les adresses de OF000H à OF7FFH sont accessibles par le signal F0 (contrairement aux autres, ce signal est actif à l'état haut). Cela vous permet d'accéder aux adresses du TOS en sortie des buffers.
- pour ce qui est des autres adresses (OFX00 à OFXFF) reportez vous à l'article sur la carte CENTRONICS.

Si vous avez des remarques à formuler, à propos de ces deux réalisations ou si vous rencontrez quelques problèmes, n'hésitez pas à me le faire savoir.

Laurent LEGRY

NB - FA/ signifie Complément de FA (ou FA barre)
- X : Valeur quelconque comprise entre 0 et F



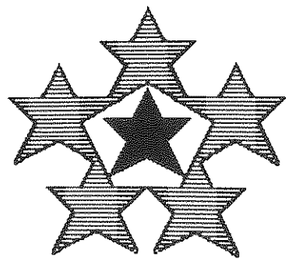
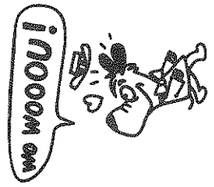


Figure 2 : NOUVELLE CONFIGURATION DU X-BUS

	GND	1 - 2	i/o	DO	
	F0	o 3 - 4	i/o	D1	
	NC	5 - 6	i/o	D2	
	F9/	o 7 - 8	i/o	D3	
	FA/	o 9 - 10	i/o	D4	
	FF/	o 11 - 12	i/o	D5	
	DTR2	i 13 - 14	i/o	D6	
	DTR1	o 15 - 16	i/o	D7	
	NC	17 - 18		NC	
	GND	19 - 20	o	Memory Write	
	RELIEE A 22	21 - 22		RELIEE A 21	
	A10	23 - 24	o	Memory Read /	
	A14	o 25 - 26	o	A11	
	A12	o 27 - 28	o	A13	
	A9	o 29 - 30	o	A15	
	A7	o 31 - 32	o	A8	
	A5	o 33 - 34	o	A6	
	A3	o 35 - 36	o	A4	
	A1	o 37 - 38	o	A2	
	A0	o 39 - 40	o	INTERRUPT ACKNOLEDGE	
	CS LOW ROM	o 41 - 42	o	CS LOWER BANK UPPER ROM	
	A12 APRES DEPART	o 43 - 44	o	CS UPPER BANK UPPER ROM	
	+5V	45 - 46		+5V	
	RAM OPERATION	o 47 - 48	o	PHI2 TTL	
	HOLD REQUEST	i 49 - 50	o	SYNC	



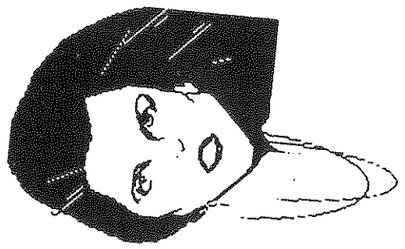
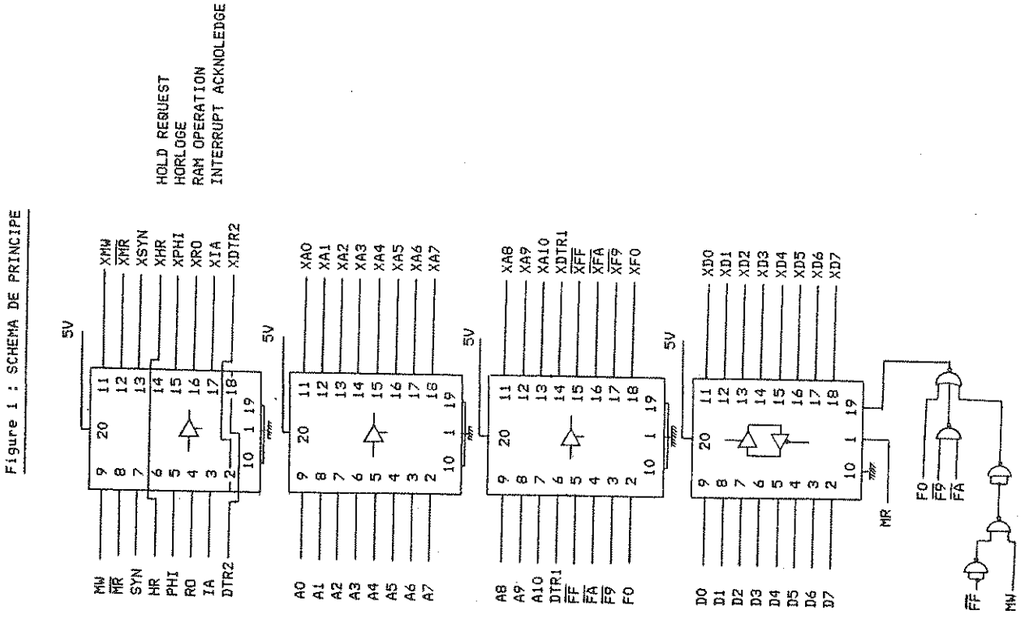


Figure 1 : SCHEMA DE PRINCIPE



TRI-RIPPLE

TRI "RIPPLE"

FL.MENCIERE (F-51 HERMONVILLE)

Voici un programme en LM qui sert à trier des données alphanumériques. J'ai utilisé la méthode du tri RIPPLE. Cette méthode consiste à comparer un mot avec le suivant. S'il est plus petit, il n'y a pas de tri et on passe au mot suivant. S'il est plus grand, dans ce cas, le deuxième mot est classé à place du premier et le premier est classé à la place du deuxième.

De cette manière, le plus grand mot sera placé en fin de tableau. Le tri se fait ensuite entre le deuxième mot (éventuellement l'ancien premier mot) et le troisième mot et ainsi de suite sans tenir compte du dernier mot qui est déjà à sa place.

Le processus se répète jusqu'à ce que tous les mots soient classés.

Pour le programme, le tableau doit être constitué de mots définis comme suit:

- Le 1er octet désigne la longueur du mot qui suit.
- Suivent les codes ASCII du mot proprement dit.
- Le dernier octet est égal à 0.

Si vous êtes curieux, vous vous serez rendu compte que cette structure des données est la même que celle utilisée dans la HEAP lorsque le DAI range un tableau de variables DIM.

Exemple: 03 44 41 49 00 ...

Long. D A I 00 ...

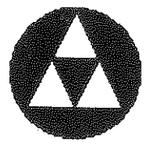
En ce qui concerne le programme, vous devez mettre l'ADRESSE de départ du tableau dans 'BETR', puis le nombre de mots à trier doit être rangé dans 'NBMOT'. Vous devez effacer les lignes qui sont suivies de '*' car elles ont été mises ici pour afficher les données dans le cas de l'exemple.

J'ai effectué un tri de 170 mots en environ 12 s.

```
0000 ;*****************************************************
0000 ;***          TRI   RIPPLE                         ***
0000 ;***   FL MENCIERE CORMICY 51220 HERMONVILLE FRANCE ***
0000 ;*****************************************************
0000 ;***  PROGRAMME DE TRI DE DONNEES ALPHANUMERIQUES   ***
0000 ;***  MISES SOUS LA FORME SUIVANTE:                 ***
0000 ;***  Long, code ASCII des lettres....., 00         ***
0000 ;***  REGROUPEES DANS UN TABLEAU DEBUTANT A         ***
0000 ;***  L'ADRESSE STOCKEE DANS 'BETR'                 ***
0000 ;***  LE NOMBRE DE MOTS COMPOSANS LE TABLEAU EST    ***
0000 ;***  STOCKE DANS 'NBMOT'.                          ***
0000 ;*****************************************************
0000 0=1B46 MOT1  EQU  1B46H
0000 0=1B44 MOT2  EQU  1B44H
0000 0=1B40 BETR  EQU  1B40H
0000 0=1B42 NBMOT EQU  1B42H
0000 0=1B43 NBDD  EQU  1B43H    ;0 (a effacer)
0000 0=DE39 HLM1  EQU  0DE39H   ;HL-=HL+H+1
0000 0=DEAF MOVE  EQU  0DEAFH
0000 0=3000 TAMPON EQU 3000H    ;ADRESSE VARIABLE
0000 ;*****************************************************
0000                    ORG   1B50H
1B50 F5                 PUSH  PSW
1B51 C5                 PUSH  B
1B52 D5                 PUSH  D
1B53 E5                 PUSH  H
1B54 3AA21B             LDA   NBMOT
1B57 4F                 MOV   C,A
1B58 0D          TRIE   DCR   C          ;NB DE TRIE
1B59 2A401B      TRIER1 LHLD  BETR       ;DEPART DU TABLEAU
1B5C 22461B             SHLD  MOT1
1B5F CD39DE             CALL  HLM1
1B62 23                 INX   H
1B63 22441B             SHLD  MOT2
1B66 CD751B             CALL  TRIDD
1B69 0D                 DCR   C
1B6A C2591B             JNZ   TRIER1
1B6D CD081C             CALL  AFFICH    ;0

1B70 E1                 POP   H
1B71 D1                 POP   D
1B72 C1                 POP   B
1B73 F1                 POP   PSW
1B74 C9                 RET
1B75 ;
1B75 C5          TRIDD  PUSH  B
1B76 3AA21B             LDA   NBMOT
1B79 D601               SUI   1H
1B7B 32421B             STA   NBMOT
1B7E F5          TRI1   PUSH  PSW        ;NB DE MOTS A TRIER
1B7F CDFB1B             CALL  CHLON
1B82 D601               SUI   1H
1B84 4F                 MOV   C,A
1B85 2AA61B             LHLD  MOT1
1B88 23                 INX   H
1B89 E5                 PUSH  H
1B8A 2AA41B             LHLD  MOT2
1B8D 23                 INX   H
1B8E EB                 XCHG
1B8F E1                 POP   H
1B90 ;
1B90 B7          TRI2   ORA   A
1B91 1A                 LDAX  D         ;LETTRE MOT2
1B92 BE                 CMP   H         ;LETTRE MOT1
1B93 CA9F1B             JZ    LESUI     ;A=M
1B96 DC8D1B             CC    SWAP      ;A<M
1B99 CDAF1B             CALL  MOTSUI    ;A>M
1B9C C3A91B             JMP   TRI3
1B9F 23          LESUI  INX   H
1BA0 13                 INX   D
1BA1 0D                 DCR   C
1BA2 C2901B             JNZ   TRI2
1BA5 CDAF1B             CALL  MOTSUI
1BA8 F1          TRI3   POP   PSW
1BA9 3D                 DCR   A
1BAA C27E1B             JNZ   TRI1
1BAD C1                 POP   B
```

```
1BAE  C9              RET
1BAF  2A&41B  MOTSUI   LHLD  MOT2
1BB2  22&61B          SHLD  MOT1
1BB5  CD39DE          CALL  HLN1
1BB8  23              INX   H
1BB9  22&41B          SHLD  MOT2
1BBC  C9              RET
1BBD  010030  SWAP    LXI   B TAMPON   ;DESTINATION
1BC0  2A&61B          LHLD  MOT1
1BC3  EB              XCHG
1BC4  2A&61B  LHLD    MOT1              ;MOT1 DANS
1BC7  CD39DE          CALL  HLN1        ;TAMPON
1BCA  CDAFDE          CALL  MOVE
1BCD  ;
1BCD  2A&61B          LHLD  MOT1
1BD0  &&              MOV   B,H
1BD1  4D              MOV   C,L
1BD2  2A&41B          LHLD  MOT2        ;MOT2 A LA
1BD5  EB              XCHG              ;PLACE
1BD6  2A&41B          LHLD  MOT2
1BD9  CD39DE          CALL  HLN1        ;DE MOT1
1BDC  CDAFDE          CALL  MOVE
1BDF  ;
1BDF  2A&61B          LHLD  MOT1
1BE2  CD39DE          CALL  HLN1
1BE5  3600            MVI   M 0H
1BE7  23              INX   H
1BE8  22&41B          SHLD  MOT2        ;MOUV.ADR.DE MOT2
1BEB  &&              MOV   B,H
1BEC  4D              MOV   C,L
1BED  210030          LXI   H TAMPON
1BF0  EB              XCHG
1BF1  210030          LXI   H TAMPON    ;MOT1 DANS
1BF4  CD39DE          CALL  HLN1        ;MOT2
1BF7  CDAFDE          CALL  MOVE
1BFA  C9              RET
1BFB  B7      CHLON   ORA   A
1BFC  2A&61B          LHLD  MOT1        ;CHERCHE LA PLUS
1BFF  7E              MOV   A,M         ;PETITE DES 2
1C00  2A&41B          LHLD  MOT2        ;LONGUEURS
1C03  BE              CMP   M
```

```
1C04  C8              RZ
1C05  D8              RC
1C06  7E              MOV   A,M
1C07  C9              RET
1C08  3E0C    AFFICH  MVI   A 0CH        ;@
1C0A  CD95D6          CALL  0D695H      ;@
1C0D  3A&31B          LDA   NBDD        ;@
1C10  2A&01B          LHLD  &ETR        ;@
1C13  F5      AFF1    PUSH  PSW         ;@
1C14  CD32DB          CALL  0DB32H      ;@
1C17  3E0D            MVI   A 0DH       ;@
1C19  E5              PUSH  H           ;@
1C1A  CD95D6          CALL  0D695H      ;@
1C1D  E1              POP   H           ;@
1C1E  F1              POP   PSW         ;@
1C1F  3D              DCR   A           ;@
1C20  23              INX   H           ;@
1C21  C2131C          JNZ   AFF1        ;@
1C24  C9              RET               ;@
1C25                  END
```



Extended Basic

EXTENDED BASIC: SUPPLEMENT

Jean DEPRAZ (IDC BORDEAUX), VILLEURBANNE (France)

Vous trouverez ci-joint:

- la procédure **MATRIANGSUP** destinée à transformer une matrice en matrice triangulaire supérieure.

- la définition de **TRACE** destinée à calculer la trace d'une matrice.

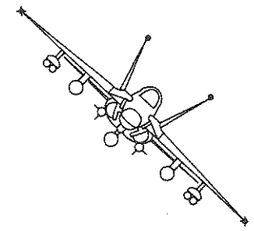
- la définition de **DETMAT** qui permet de calculer le déterminant d'une matrice.

- le programme qui utilise ce qui précède pour résoudre un système d'équations linéaires.

En modifiant comme il convient les data, on peut utiliser ce programme pour un nombre important d'inconnues; il convient également d'adapter en conséquence le CLEAR à la ligne 170.

Comme vous l'avez déjà compris, tout ce qui précède n'est compatible qu'avec DBASIC. C'est un complément à EXTENDED BASIC de F. LEMOINE paru dans le DAICLIC numéro 2 page 70.


```
    1         DEF PROC MATREAD  ARR A!
    2    1     FOR I=1 TO DIM(A!,1)
    3    2      FOR J=1 TO DIM(A!,1)
    4    3       READ A!(I,J):
         2      NEXT J:
         1     NEXT I
    5         END PROC
    6         DEF PROC MATPRINT  ARR A!
    7    1     FOR I=1 TO DIM(A!,1):
         2      PRINT :
         2      FOR J=1 TO DIM(A!,2)
    8    3       PRINT TAB((J-1)*7);A!(I,J):
         2      NEXT:
         1     NEXT
    9         END PROC
   10         PROCEDURE MATRIANGSUP  ARR A!
   15    1     FOR R=1 TO N-1:
         2      FOR E=N TO R+1 STEP -1:
         3       IF A!(R,E)=0.0 THEN 65
   20    3       FOR C=E-1 TO R STEP -1
   25    4        IF A!(R,C)=0.0 THEN 45
   30    4        U!=-A!(R,E)/A!(R,C)
   35    4        FOR I=R+1 TO N:
         5         A!(I,E)=A!(I,E)+U!*A!(I,C):
         4        NEXT
   40    4        GOTO 65
   45    3       NEXT C
   50    3       FOR I=1 TO N:
         4        F!=A!(I,R):A!(I,R)=-A!(I,E):A!(I,E)=F!
   55    3       NEXT:
         3       GOTO 70
   65    2      NEXT E
   70    1     NEXT R
   75         END PROC
   80         DEF FN DETMAT!( ARR A!)
   85    1     LOCAL I,D!:D!=1.0
   90    1     FOR I=1 TO DIM(A!,1)
   95    2      D!=D!*A!(I,I)
  100    1     NEXT
  105    1     FN = D!
  110         END FN
  120         PROCEDURE TITRE I
  130    1     CURSOR 5,21:
         1     WHILE I<>0 DO
         2      PRINT CHR$(127);:I=I-1:
         1     WEND
  140         END PROC
  145         REM ****  PROGRAMME  PRINCIPAL  ****
  150         PRINT CHR$(12):PRINT "  RESOLUTION D'UN SYSTEME D'EQUATIONS
              LINEAIRES"
  160         TITRE 40
  170         CLEAR 5000
  200         PRINT :PRINT :PRINT "TAPEZ EN 210 LES ELEMENTS DE LA MATRICE"
  205         PRINT "ET EN 220 LES VALEURS DES 2d MEMBRES"
  206         PRINT "FAITES EDIT 210-220"
  207         PRINT "TAPEZ ENSUITE RUN 230":STOP
  210         DATA 1,8,2,-4,-5,12,-24,4,6,-13,-4,10,12,5,-14,-32,-44,12
  215         DATA 10,-12,46,-42,20,-17,1,5,6,-7,8,-13,-12,40,23,-45,81,10
  220         DATA 12,46,13,-46,52,82
  230         PRINT :PRINT :INPUT "NOMBRE D'INCONNUES:";N
  240         DIM A!(N,N),B!(N):GOSUB 2000
```



```
250         FOR J=1 TO N:
      1      READ B!(J):
            NEXT
260         IF A!(N,N)=0.0 GOTO "CORR
270         GOSUB 1000
280         DE!=D!:PRINT :PRINT "DETERMINANT:";DE!:PRINT
290         IF DE!=0 THEN
      1      PRINT "INDETERMINE":
            ELSE
      1      GOTO "SUITE:
            END IF
300         REM LECTURE DONNEES
310        "SUITE
            FOR J1=1 TO N:
      1      RESTORE:GOSUB 2000
320   1      FOR I=1 TO N:
      2       A!(I,J1)=B!(I):
      1      NEXT
330   1      GOSUB 1000
340   1      PRINT :PRINT "X";J1;"=";D!/DE!
350         NEXT J1
360         END
715         DEF FN TRACE!( ARR A!)
720   1      LOCAL I,T:T=0
725   1      FOR I=1 TO DIM(A!,1)
730   2       T!=T!+A!(I,I)
735   1      NEXT
740   1      FN = T!
745         END FN
750         PROCEDURE MATRIANGSUP  ARR A!
755   1      FOR R=1 TO N-1:
      2       FOR E=N TO R+2 STEP -1:
      3        IF A!(R,E)=0.0 THEN 810
760   3        FOR C=E TO R+1 STEP -1
765   4         IF A!(R,C)=0.0 THEN 785
770   4         U!=-A!(R,E)/A!(R,C)
775   4         FOR I=R+1 TO N:
      5          A!(I,E)=A!(I,E)+U!*A!(I,C):
      4         NEXT
780   4         GOTO 810
785   3        NEXT C
790   3        FOR I=1 TO N:
      4         F!=A!(I,R):A!(I,R)=-A!(I,E):A!(I,E)=F!
800   3        NEXT:
      3        GOTO 820
810   2       NEXT E
820   1      NEXT R
825         END PROC
827         DEF FN DETMAT!( ARR A!)
830   1      LOCAL I,D!:D!=1.0
835   1      FOR I=1 TO DIM(A!,1)
840   2       D!=D!*A!(I,I)
845   1      NEXT
850   1      FN = D!
855         END FN
1000        REM CALCUL DETERMINANT
1010        MATRIANGSUP A!():D!=DETMAT!(A!())
1020        RETURN
2000        MATREAD A!()
2010        RETURN
```


S.P.L.-Display

System Programming Language:

Le S.P.L. est un assembleur 8080 pour l'ordinateur DAI. Il permet aux utilisateurs d'appliquer les définitions 'MACRO' et l'assemblage conditionnel.

Les définitions 'MACRO' rendent possible, dans un programme en assembleur, la désignation d'un groupe d'instructions par un mot.

L'assemblage conditionnel permet d'adapter rapidement un programme pour plusieurs buts. (ex. imprimante série ou parallèle)

MISE EN MARCHE:

Chargement du S.P.L.:

Il suffit de passer en **UTILITY** —> **UT** ret
de **lire** le programme —> **R** ret

Démarrer S.P.L.:

Reset de UTILITY —> **Z3** ret
S.P.L. —> **G8500** ret

Sur L'écran apparait le titre du S.P.L. et une adresse de départ est demandée. Cette adresse indique le début de la portion mémoire que S.P.L. va utiliser pour stocker le programme en assembleur. L'adresse de départ doit être au maximum *81F0 cette adresse est donnée sous forme hexadécimale et est suivie de 'ret'.

Vous pouvez maintenant entrer les commandes de S.P.L.

Pour quitter S.P.L. on utilise les commandes:

U retour en **UTILITY.**
u retour en **BASIC.**

Pour redémarrer à partir du mode UT, faire Z3 'ret' G8500 'ret'
BASIC CALLM *8500 'ret'

Alors apparait à la place de la demande d'adresse un menu:

S.P.L. : System
code source : Program
table symboles : Labels

intact ou non : OK ou BAD

Pour les différentes commandes, se référer au manuel d'utilisation du S.P.L.

DISPLAY.

Rem.: DISPLAY est un désassembleur.

DISPLAY est un auxiliaire du S.P.L. spécialement conçu pour le décodage des programmes en langage machine.

Par l'assemblage de DISPLAY, le set de commandes est étendu de quelques intructions de désassemblage. Elles ne génèrent pas de sortie visible, mais elles produisent un fichier source S.P.L., qui à l'aide des commandes S.P.L. peut être adapté.

Avantages de DISPLAY : grande rapidité de désassemblage.
produit un fichier source.

DISPLAY est un désassembleur orienté listing. Une liste de commandes précises a été établie, liste dans laquelle est indiqué si une région précise de la mémoire doit être traduite en code machine, en ASCII, ou en quelque chose d'autre.

L'une des caractéristiques les plus intéressantes du désassembleur est de générer automatiquement des labels.

Quelques commandes de la liste:

. début de la zone à désassembler
désassemble en langage machine
= caractères ASCII 8 bits (MSB 1 ou 0)
, nombres de 8 bits

Quelques commandes de désassemblage:

. montre la liste
) vide la liste
. adr désassemble à l'aide de la liste
% adr1 adr2 liste le contenu des mémoires comprises entre adr1 et adr2.

Guy PIETTE
Février '86
Club CAROLODAI pour DAICLIC.

---

* REFROIDISSEMENT DU DAI *

Le temps de calcul très long que demande la construction d'une image de l'ensemble de Mandelbrot, m'a conduit à m'intéresser au refroidissement du DAI.

Une solution très simple et très sure, consiste, sans faire subir au DAI aucune des opérations barbares déjà décrites dans la revue (trépanation, etc), à simplement déposer sur la grille noire d'évacuation de chaleur, un petit ventilateur à cage d'écureuil, qui aspire l'air chaud sortant par la grille.

Le modèle que j'utilise provient de récupération de matériel de conditionnement d'air, ne consomme que douze watts et est très silencieux. L'efficacité est excellente, au point que le boitier du DAI reste à la température ambiante, quelle que soit la dureé de fonctionnement.

Ce type de ventilateur se trouve également chez les detaillants de composants électroniques (refroidissement de racks).

La forme rectangulaire de la bouche d'aspiration se prete très bien à cette application (éventuellement avec l'aide d'un peu de carton et de plastique adhésif).

Par sécurité, il vaut mieux laisser l'interrupteur du DAI allumé et alimenter les deux appareils à partir d'une prise commune munie d'un interrupteur.

L. Laurent
B-1180 Bruxelles

DAISTAR

TEST DU MODE I DU DAI-STAR

DAISTAR est un nouveau système d'extension pour le **DAI** pc qui perme d'utiliser ce dernier à un haut niveau. Il fonctionne avec deux systèmes d'exploitation: **MODE II** (un **CP/M** étendu que nous étudierons dans un prochain article, et **MODE I** qui est un **DAI-DOS** performant.

Fiche technique:

Deux unités de 800K (1,6Mb au total): double face, double densité, 80 pistes de 10 secteurs de 1024 octets OU deux unités de 1200K (2 Mb au total): double face, double densité, 77 pistes de 16 secteurs de 1024 octets: il s'agit d'un formattage identique à celui des disquettes 8" d'**IBM** !

Remarquons que les deux formats sont parfaitement compatibles via le **MODE II** et que les secteurs, à la différence du **KEN-DOS**, sont écrits alternativement sur la face supérieure, et sur la face inférieure. Ce test a été réalisé sur un système 2 x 1,2 Mb mais les commandes sont identiques sur les deux versions.

Occupation de la mémoire:

Le **DOS** complet occupe 1,8 K de RAM entre *2F2 et *A36 au chargement. Le **TINYDOS**, plus réduit, n'occupe que 320 octets et peut etre placé n'importe où.

Vitesse:

- temps de chargement du DOS: 6 secondes,
- BACKUP d'une disquette 1,2 Mb: 3 minutes,
- chargement d'une image en mode 6: 4,2 s,
- création-ouverture-écriture de 10000 enregistrements de 7 caractères, puis fermeture du fichier: 97 s.

Une comparaison avec le test du DOS 3.0 de **PRODATA** (**DAICLIC** 3, p. 72, août 85) permet de mesurer la différence de vitesse.

Caractéristiques particulières:

Utilisation possible d'un clavier extérieur, de lecteurs de cassettes, d'un **MDCR**, d'une imprimante série et d'autres périphériques sur le **DCE-BUS**. Allocation dynamique: réutilisation immédiate de l'espace disponible sur la disquette. Gestion de fichiers à accès séquentiel et direct (4 fichiers simultanément). Possibilité de réduire le **DOS** en **TINYDOS**, et de le déplacer en mémoire en perdant cependant certaines commandes. Gestion des erreurs. Autostart de programmes **BASIC**. Interface permettant d'atteindre le **DOS** depuis l'assembleur. Compatibilité totale

- entre les deux formats de disquettes,
- avec tous les programmes et fichiers utilisables sur cassette audio, ex: **FWP**, **SPL**, **FGT**. Seuls doivent être adaptés les programmes protégés ou utilisant toute la RAM,
- avec les **BASIC V1.1** et **V1.2**,
- avec la directory, les fichiers séquentiels et directs,
  les utilitaires du **CP/M**.
- les nouvelles commandes s'ajoutent directement aux anciennes sans aucun **CALLM**, **POKE**, **REM** ou **DOS**!

Type de fichiers admis:

- **BAS** programme en **DAI-BASIC** (anciennement 0),
- **MLP** programme en langage machine (anciennement 1),
- **ARR** tableaux (anciennement 2),
- **PIC** copie d'écran graphique,
- **TXT** fichier texte séquentiel,
- **RND** fichier à accès direct,
- **TPx** pour d'autres type exemple TPe pour **SPL**.

Liste des commandes:

- **TINY** version réduite et relogeable du **DOS** permettant uniquement les chargements et les sauvetages (très utile pour le programmes **MLP** et pour les longs programmes graphiques, car ce **TINYDOS** peut être écrasé sans danger.)

- **DIR**, **DIRA**, **DIRB** pour afficher une directory.

- **A:,B:** pour sélectionner un drive.

- **ERA** (suppression), **REN** (nouveau nom), **COPY**, **BACKUP**.

Remarque: le formattage se fait en **MODE II**.

- **FLOPPY**, **MDCR**, **CASS** pour choisir la mémoire de masse.

- **LOAD**, **SAVE**, **LOADA**, **SAVEA**, **R**, **W** comme auparavant.

- **DLOAD**, **DSAVE** pour les programmes **MLP**.

- **PLOAD**, **PSAVE** pour les copies d'écran.

- **/** pour le chargement et l'exécution automatique d'un programme **BASIC**.

- **CREATE**, **OPEN**, **CLOSE**, **PRINT** (écriture), **INPUT** (lecture), **EOF** (fin de fichier) pour les fichiers séquentiels.

- **CIN**, **COUT** pour allouer les périphériques:
  entrée: clavier, **RS232**, fichier(s), **MLP**.
  sortie: écran, écran + **RS232**, fichier(s), **MLP**, pas d'affichage.

- **CREATE**, **OPEN**, **CLOSE**, **EOF**, **PUT** (écriture), **GET** (lecture) **SEEK** (positionnement de la clé) **POS** (détermination de la clé), **SIZE** (longueur du fichier): gestion de fichiers à accès direct. La longueur de l'enregistrement varie au gré de l'utilisateur entre 1 et 1024 octets; de plus il n'y a ni conversion, ni manipulation intermédiaire ni de déclaration de **FIELD** à effectuer avant de sauver une donnée sur disquette. Cette gestion de fichier est plus simple qu'en **BASIC MICROSOFT** et plus fiable qu'avec le **DOS 3.0** de **PRODATA**. De plus la variable **ERROR%** , lors de la fermeture du fichier, permet d'intercepter des erreurs éventuelles, et de les corriger.

- **READS**, **WRITES** pour lecture ou écriture directe d'un secteur.

En conclusion, je n'ai pas trouvé de lacunes ou de défauts importants dans le système **DAISTAR** (peu de place occupée par rapport au **DOS 3.0** de **PRODATA**, bonne gestion de fichiers par rapport au **KEN-DOS**), seulement des imperfections faciles à corriger. D'abord au niveau de la directory: pas d'effacement préalable de l'écran, ni possibilité d'interrompre le listing ou de l'envoyer sur imprimante. Ces défauts sont en partie

rattrapés par une présentation claire et complète en quatre colonnes, ainsi que par l'utilisation de la commande **DIR** du **CP/M**.

Ensuite quelques détails: l'ajout d'une commande **LOCK / UNLOCK** pour protéger les fichiers contre une mauvaise manipulation, et la redéfinition de certains messages d'erreurs pas assez précis.

Enfin, l'utilisation d'un **TINYDOS** en mémoire vive rend relativement complexe le transfert de programmes **MLP** depuis **DCR** ainsi que leur maniement: directory, lecture et exécution d'un ensemble **MLP** + **BASIC** (bootstrap loader).

L'écriture d'un utilitaire permettant une manipulation automatique s'impose, ce qui n'est heureusement pas difficile.

En conclusion: un mot pour résumer le système **DAISTAR**: professionnalisme. Cela se marque aussi bien au niveau de la puissance, de la fiabilité et de la facilité d'utilisation, qu'au niveau de la présentation extérieure: boitier solide et esthétique, manuel clair et complet avec les adresses nécessaires (125 pages en anglais). Le prix qui varie entre 40000 et 80000 FB selon les versions (1 ou 2 drives, 800 ou 1200 K, 4 ou 6 MHz) peut paraître élevé, mais il correspond à celui d'autres lecteurs pour le **DAI**. Et en plus vous recevez un **CP/M** puissant et complet.

F.LEMOINE
Mars 1986
Club CAROLODAI pour DAICLIC

Quelques remarques au sujet de l'article "MICROPROCESSEURS PART 1" paru dans DAICLIC no. 2

(J. DEPRAZ, VILLEURBANNE (F))

1) Pour la conversion décimale-binaire, j'utilise le procédé suivant très pratique: on multiplie successivement par 5 et on ne retient de chaque opération que le chiffre le plus à droite qui ne peut être qu'un 0 ou un 5. On remplace les 5 par des 1 et on les garde dans l'ordre obtenu.

Ex:

```
      69
     x 5
    ----
     34 5
    x 5
    ----
    17 0
   x 5
   ----
    8 5
  x 5
  ----
   4 0
  x 5
  ----
   2 0
 x 5
 ----
  10
  1000101
```

2) Page 48, ligne 2, je lis "la soustraction binaire n'existe pas". Cette affirmation est inexacte. TOUTES les opérations existent dans tous les systèmes quelle que soit la base. IL SUFFIT POUR LES FAIRE DE BIEN COMPLETER LES RETENUES.

Base 2

```
Addition                  11          retenues
                        --------
                        00010011                    19D
                      + 00001011                  + 11D
                      ----------                  -----
                        00011110          total     30D

Soustraction          10101101                      17
                      - 01111111                    6
                      ----------                    ----
                        11111100          retenues  11
                      ----------
                        00101110          résultat
```

Multiplication - AUCUNE DIFFICULTE

```
Division            110001 ! 101
                    - 101    +----
                    --------   1001,11...
                      001001
                    -   101
                    --------
                       01000
                     -  101
                     -------
                        0110
                     -  101 etc...
```

Opérations en base 16

```
Addition              1 111     retenues
                      ------
                      6A7CBD
                      4B65AC
                      ------
                      B5E269    résultat

Soustraction          854372
                    - 69ACBF
                    --------
                      11111     retenues
                      ------
                      1B96B3    résultat
```

3) Complément à 2

Pour FACILITER LA SOUSTRACTION, on passe en complément à 2. C'est un simple artifice de calcul.

En base décimale, on aurait pu de même utiliser le complément à 10 pour effectuer les soustractions mais on est habitués à les faire directement.

Exemple - base 10

```
Soustraction directe          4562
                            - 1387
                            ------
                              3175

Complément à 10     1387  =>  complément à 9         =>  8612
                              complément à 10 (+1)   =>  8613

la soustraction est remplacée par une addition
                              4562
                            + 8613
                            ------
     on écarte ce chiffre:  1 3175  =>  3175
```

Exemple base 2

```
          17                                        00010001
        - 6  =>  complément à 2:                  + 11111011
                   00001100                       ----------
                 +        1                      1 00001011
                 ----------
                   11111010
```

(il y a une faute dans le calcul page 49 ligne 5)

4) Dans les opérations logiques, le OU indiqué est le OU "inclusif". Il existe aussi le OU "exclusif" très important.

```
1 XOR 1 = 0          (l'un ou l'autre mais pas les 2)
1 XOR 0 = 1
0 XOR 1 = 1
0 XOR 0 = 0
```

5) Soustraction en complément à 16

```
Effectuer           854372
                  - 69ACBF

        69ACBF  ->  complément à 15  ->  965340
                    complément à 16 (+1) 965341

                    854372
                  + 965341
                  --------
    à négliger  1 1B96B3    résultat: 1B96B3
```

6) Une curiosité - la base 36

Elle utilise les chiffre de 0 à 9 et toutes les lettres de l'alphabet. Supposons que la phrase:

I love my DAI

soit écrite dans cette base; si on la convertit en système décimal, cela donnera:

18 1012010 826 17226 !!

à vous de choisir pour mieux exprimer ce que vous ressentez !

DAI QUI RIT: EXCLUSIVITE DAICLIC !

PROCESSEUR ARITHMETIQUE AMD 9511

Claude Picard, Châlon sur Saône (F)

Les fonctions mathématiques du Basic sont appelées par une instruction **RST 4** suivie d'un octet qui définit l'offset dans la table des branchements des routines mathématiques situées en début de **ROM 1** (math package). Si le **MATH CHIP FLAG** (*D4) contient 0, il n'y a pas de fonctionnement avec l'**AMD 9511**, et la table des branchements débute à l'adresse *E000 (**ROM 1**). Si *D4 contient *7B (= 123 en décimal), l'**AMD 9511** est actif, et les branchements se font dans la seconde table, qui débute en *E07B.

Autrement dit:

```
RST  4
DB   3H
```

branche sur *E003 si *D4 vaut 0 ou sur *E07E si *D4 vaut 123 car le calcul s'opère de la facon suivante:

*E000 + byte d'offset + contenu de *D4

Le byte d'offset est forcément un multiple de 3 car la table est de la forme: **JMP LLHH**, donc chaque branchement s'écrit sur 3 octets.

BYTE D'OFFSET POUR LES ROUTINES MATHEMATIQUES:

Routine appelée	Byte
addition flottante	0 (hex.)
soustraction flottante	3
multiplication flottante	6
division flottante	9
chargement	C
récupération	F
sauvegarde ABCD dans accu. flottant	12
récupère accu. flottant dans ABCD	15
ABS flottant	18
changement de signe flottant	1B
INT	1E
FRAC	21
élévation à la puissance	24
log. népérien	27
exponentielle	2A
log. décimal	2D
10 puissance x	30
racine carrée	33
sinus	36
cosinus	39
tangente	3C
arcsinus	3F
arccosinus	42
arctangente	45
FPT -> INT	48
INT -> FPT	4B
INT addition (4 octets)	4E

INT soustraction (4 octets)	51
INT multiplication (4 octets)	54
INT division (4 octets)	57
modulo	5A
valeur absolue entière	5D
INT changement de signe	60
AND entre 2 entiers	63
OR entre 2 entiers	66
XOR entre 2 entiers	69
NOT d'un entier	6C
SHL d'un entier	6F
SHR d'un entier	72

REMARQUE: l'accu. flottant est constitué soit des adresses *D5, *D6, *D7, *D8, soit du sommet de la pile de l'**AMD 9511**.

QUELQUES ADRESSES DE SOUS-PROGRAMMES DE GESTION DE L'**AMD9511**

(Ces sous-programmes se trouvent en **ROM 1**)

- E53B: test si l'**AMD9511** est occupé. Dès qu'il est libre, son statut est examiné. Si erreur, émission d'un message d'erreur, sinon, retour à l'appelant.
- E55F: charge l'**AMD9511** avec le contenu de A, B, C, D si pas d'erreur à l'opération précédente.
- E588: charge l'**AMD9511** avec le contenu des mémoires (HL), (HL+1), (HL+2), (HL+3) et détecte les erreurs sur l'opération précédente.
- E599: récupère le sommet de la pile du 9511 et le met en (HL), (HL+1), (HL+2), (HL+3).
- E56F: récupère A, B, C, D du sommet de la pile du **9511** et restauration de ce sommet (car lecture destructive; voilà qui fait penser à FORTH !).
- E527: transfert de la variable pointée par (HL) dans le **9511**. Envoi de la commande spécifiée par l'octet suivant le CALL 0E527h dans le registre de commande du **9511**.
- E535: transfert de l'octet de commande qui suit le CALL 0E535h dans le registre de commande du **9511**, dès que celui-ci a exécuté sans erreur l'instruction précédente. Ce sous-programme permet l'imbrication des commandes sur une même donnée.
- E9DB: transfert de A dans (HL), B dans (HL+1), C dans (HL+2), D dans (HL+3). A la fin, HL pointe sur (HL+4).
- EA0E: transfert de (HL), (HL+1), (HL+2), (HL+3) dans B,C,D,E.
- EA16: met à 0 l'accu. flottant et les 4 mémoires pointées par HL.
- ECCC: charge le commande pointée par HL dans le **9511**.
- ECD2: envoie un NOP au **9511**.
- ED83: teste si B,C,D,E est nul ou non et positionne le flag Z (B,C,D,E=0 --> Z=1)
- ED95: prend le statut du sommet de la pile et le place dans l'accu. A.

DAIQUIRI

Voici les ingrédients de ce délicieux cocktail au nom prédestiné :

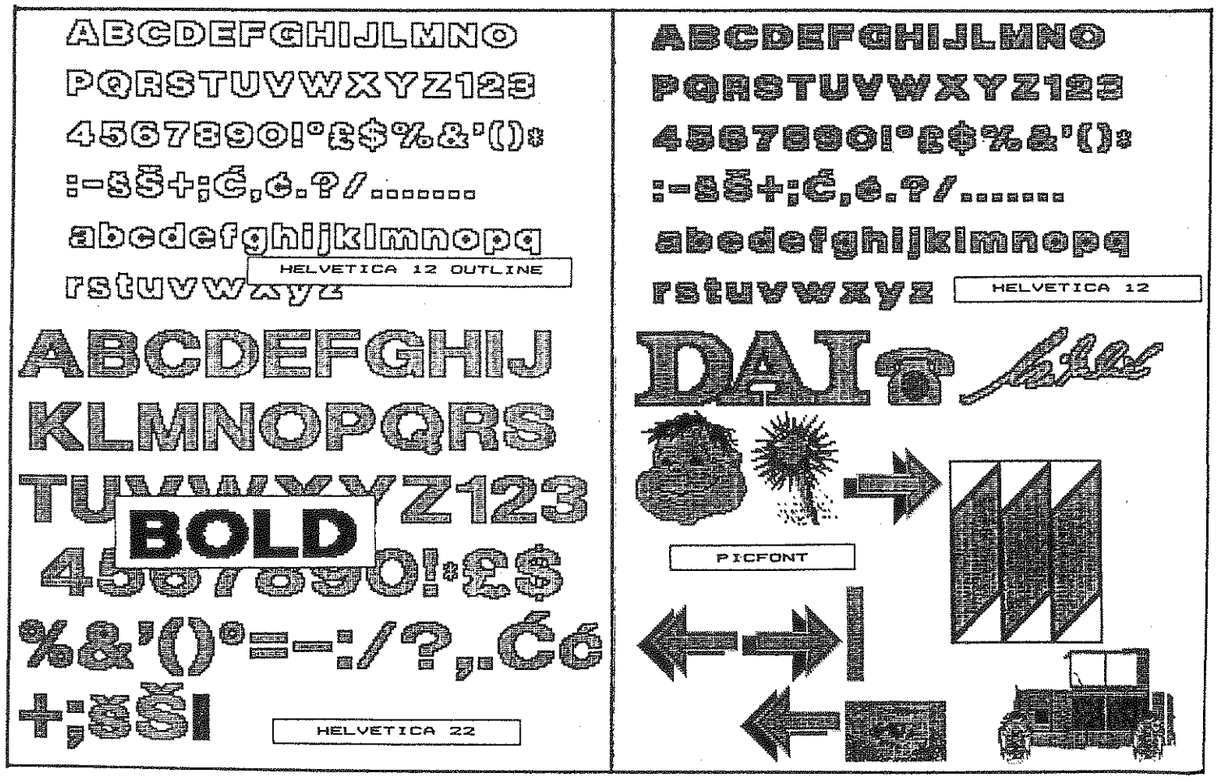
1/4 jus de citron
3/4 Rhum
1 petite cuillère de sucre en poudre

Préparer dans le mixer avec de la glace pillée et servir dans une goutte basse (traduisez 'verre de forme cylindrique et légèrement pansue')(on en apprend des choses dans DAIclic !).

CHARACTER GENERATOR

1. INTRODUCTION.

Une présentation de ce programme a été faite dans la revue DAInamic 83-19.La cassette contenant le CHARACTER GENERATOR est accompagnée par une notice qui en donne le mode d'emploi en MANUEL et en AUTO. Toutefois, aucun renseignement n'est fourni en ce qui concerne la manière de modifier la couleur du texte ni comment créer ou modifier des caractères. Ceci est un inconvénient car,si ce programme contient certains caractères yougoslaves, il ne s'y trouve pas les caractères spécifiques à la langue française tels que les lettres accentuées. Le but de la présente notice est d'apporter des éléments complémentaires nécessaires pour permettre ces ajoutes ainsi que celle de pictogrammes. Elle décrit les modes MANUEL et AUTO avec,en plus,un mode PROGRAMMABLE. Ce dernier est très utile pour la création de génériques vidéo par exemple. Le tableau ci-dessous, extrait de la revue DAInamic déjà citée,illustre les polices de caractères et les pictogrammes contenus à l'origine dans la cassette.



2. CHARGEMENT.

La cassette contient:

1.CHARGEN V1.0	(BASIC)
2.HELVETICA 22	(1E70-53D7)
3.HELVETICA 12	(1E70-314E)
4.PICFONT 0/9+M	(1E70-42DC)

Ces programmes permettent de générer au choix deux polices de caractères et des pictogrammes.Le CHARGEN se charge par une commande LOAD.Il faut passer en UT et,par une commande R,charger la police désirée. Le retour au BASIC se fait en tapant B.Les caractères HELVETICA occupent 22 ou 12 paires de lignes TV en hauteur.Un caractère est contenu dans une matrice dont les dimensions maximales sont 256*256 points (MODE 6). Il faut prendre soin de ne pas déborder de l'écran:les octets excédentaires risquent de modifier le LINE CONTROL et de détruire la page. Chaque caractère peut être positionné avec précision sur l'écran par déplacement du curseur spécifique à ce programme.Les caractères sont à espacement proportionnel.
Il est facile d'avoir des caractères pleins (BOLD) ou seulement leur contour (OUTLINE).
La couleur du fond, du contour, du caractère ainsi que du curseur sont modifiables par COLORG.
Si la police HELVETICA 22 a été chargée, un programme de démonstration s'exécute automatiquement.Il suffit de pousser sur la touche SPACE pour l'arrêter. Il y a le choix entre un mode MANUEL,un mode AUTO et un mode PROGRAMMABLE.
L'action sur la touche SPACE fait passer en mode MANUEL.

3. MODE MANUEL.

Ce mode permet de faire apparaitre lettre après lettre par actions sur le clavier.Les touches ont pour fonctions:

↓ ↑ ← →	les 4 flèches du curseur déplacent celui-ci de la largeur d'un caractère.
SHIFT...	chaque flèche accompagnée de SHIFT déplace le curseur d'un point.
CHAR.DEL	recul et effacement du dernier caractère.
TAB	efface le curseur (une autre touche le fait réapparaitre).
TAB C	efface l'écran.
TAB H	HOME sans effacer l'écran.
TAB L	efface depuis le curseur jusqu'à la fin de la ligne.
TAB X	efface depuis le curseur jusqu'à la fin de la page.
TAB P	place le curseur au dernier TAB (par défaut:dans le tiers inférieur de l'écran).
TAB T	place un nouveau TAB en effaçant le précédent.
TAB E	relief ON/OFF (pas utilisable en mode AUTO).
TAB S	marque le début de l'enregistrement en mode AUTO.
TAB R	marque l'enregistrement de chaque ligne (si TAB S a été actionné préalablement).
TAB A	passage en mode AUTO:l'écran s'efface,le curseur s'éteint et la première ligne apparait.

Il est possible d'utiliser différentes polices de caractères.
Lorsque la première partie du texte est écrite:
- appuyez sur la touche TAB
- appuyez sur la touche BREAK
- placez la cassette au début du fichier souhaité
- passez en UT
- R pour charger
- B pour revenir au BASIC
- RUN pour lancer le programme.

Attention: si la police HELVETICA 22 est sélectionnée, le programme de démonstration recommence en détruisant le texte précédent.Pour éviter ceci,il suffit d'effacer la ligne 40 du programme CHARGEN.

4.MODE AUTO.

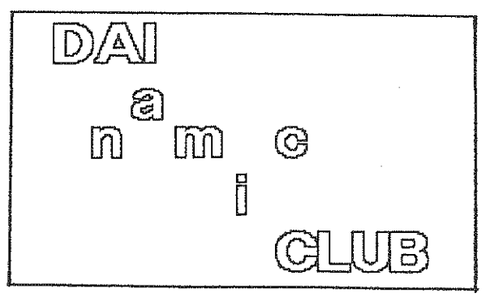
Les touches ont pour fonctions:

↑	efface l'écran.
←	efface l'écran et affiche la ligne précédente.
SPACE	efface l'écran et affiche la ligne suivante.
SHIFT ←	affiche la ligne précédente.
SHIFT →	affiche la ligne suivante.
R	répète la ligne (utilisé généralement après ↑).
T	efface l'écran et affiche la première ligne.
B	efface l'écran et affiche la dernière ligne.
F	retour au mode MANUEL avec effacement des lignes en mémoire.
L	retour au mode MANUEL et permission d'écrire après les lignes déjà en mémoire.
S	retour au mode MANUEL et affichage de la dernière ligne (les autres sont effacées) . Il est possible d'écrire après cette ligne)

Note:si vous utilisez les caractères minuscules,vérifiez que toutes les commandes sont données en MAJUSCULES.

Exemple:

Réalisez le titre:

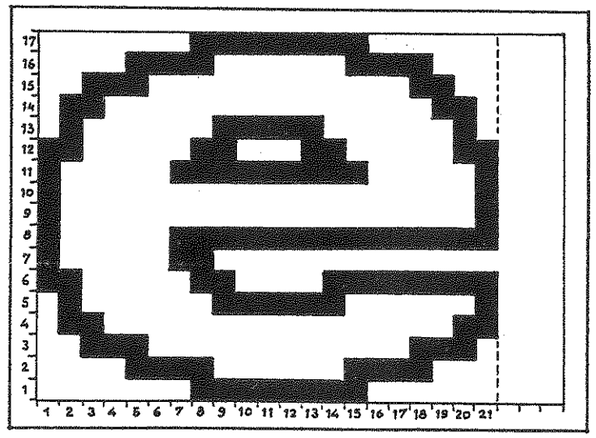


- * LOAD chargez le programme CHARGEN.
- * UT passez en UTILITY.
- < R lire HELVETICA 22.
- < B retour au BASIC.
- * RUN lancez le programme.
- SPACE l'écran s'efface et le curseur apparait.
- ↓ → → positionne le curseur sur l'écran.
- TAB S début de l'enregistrement (n'est tapé qu'une seule fois).
- DAI écriture de la première ligne.
- TAB R enregistrement de la première ligne.
- ← ← ↓ ↓ positionne le curseur au début de la deuxième ligne.
- n↑a↓m↓i↑c écriture de la deuxième ligne.
- TAB R enregistrement de deuxième ligne.
- ← ← ↓ ↓ positionne le curseur au début de la troisème ligne.
- CLUB écriture de la troisième ligne.
- TAB R enregistrement de la troisième ligne.
- TAB A appel du mode AUTO.
- utilisez les touches du paragraphe 4.

5.MODE PROGRAMMABLE.

5.1.PRINCIPE DU DESSIN.

Un caractère est contenu dans une matrice ayant comme dimensions 256 * 256 points. Le dessin se fait en mode 6 (donc à 4 couleurs puisque c'est un mode pair). Prenons comme exemple la lettre minuscule e :



L'origine de la matrice est le coin inférieur gauche.Le nombre de points en largeur et le nombre de points en hauteur de la matrice est exprimé en hexadécimal.Pour notre exemple:
largeur 21 points soit 15 en hexadécimal
hauteur 17 points soit 11 en hexadécimal
La couleur de chacun des points contenus dans la matrice est définie par les quatre registres de l'instruction COLORG 1 2 3 4:
- la couleur écrite en 1 donne la couleur du fond.
- la couleur écrite en 2 donne la couleur du contour du caractère.
- la couleur écrite en 3 donne la couleur du curseur.
- la couleur écrite en 4 donne la couleur intérieure du caractère.

Par exemple:

	COLORG	fond	contour	curseur	intérieur
avec contour et intérieur	0 5 10 15	noir	vert	orange	blanc
rien que le contour	15 0 10 15	blanc	noir	orange	blanc
contour et intérieur de même couleur pas de curseur	15 0 10 0	blanc	noir	orange	noir
	0 5 0 15	noir	vert	noir	blanc

Les quatre couleurs sont codées:

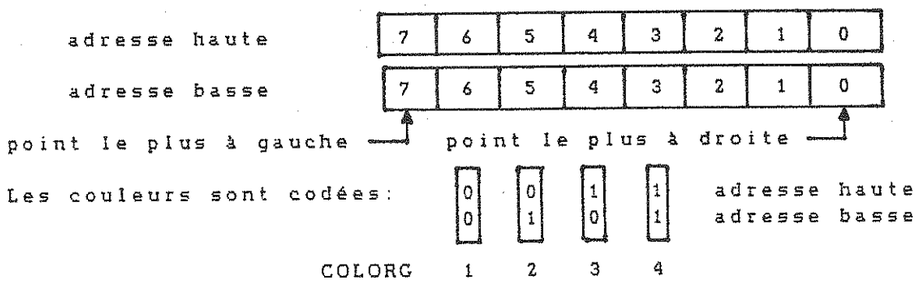
- 1 : fond
- 2 : contour du caractère
- 3 : curseur
- 4 : intérieur du caractère

0 0
0 1
1 0
1 1

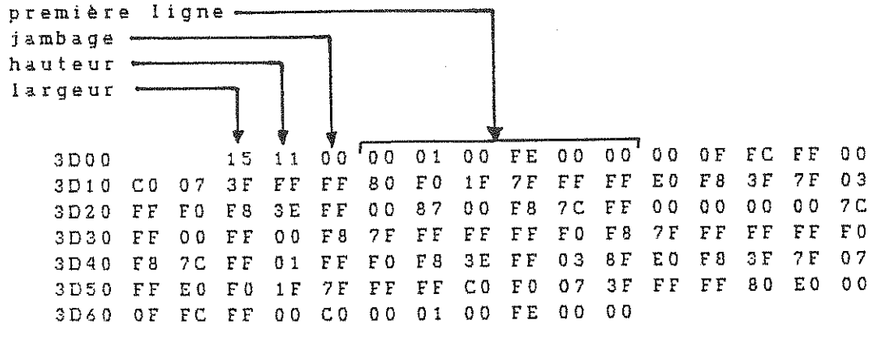
Certaines lettres minuscules ayant un jambage descendant,un octet permet de donner la dimension de ce jambage.
Le codage d'une lettre se fait en donnant:

la largeur - la hauteur - le jambage - les octets de couleur.

Pour notre exemple,le premier octet contiendra 15,le deuxième 11, le troisième 0.Les suivants définissent le caractère.
Pour les 21 points en largeur, il faut utiliser 3 octets,soit 24 bits: les bits excédentaires ne sont pas dessinés sur l'écran. Chaque ligne nécessite donc 6 octets pour coder les 4 couleurs En effet, en mode 4 couleurs,les deux bits qui définissent la couleur sont répartis dans deux octets consécutifs. Le point le plus à gauche est le bit 7 et le point le plus à droite est le bit 0.



D'où le codage de la lettre e :

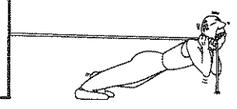


Il faut 17 * 6 = 102 octets pour cet exemple.

Les majuscules de la police HELVETICA 22 ont une hauteur de 16 hexadécimal soit 21 points et les minuscules ont une hauteur de 11 hexadécimal soit 17 points. La lettre g a une hauteur de 16 hexadécimal dont 05 hexadécimal pour le jambage (16-05: nous retrouvons la hauteur 11).

Le tableau ci-après donne l'adresse de chaque lettre de cette police:

	#20C2	@ *	#3077	` *	#4CC5
!	#4846	A	#20C7	a	#3B22
"	#53B4	B	#217A	b	#3B8B
#	#3644	C	#2201	c	#3C12
$	#36D1	D	#238F	d	#3C7B
%	#453E	E	#2442	e	#3D02
&	#37F7	F	#24C9	f	#3D6B
'	#38B2	G	#2550	g	#3DC6
(	#38E1	H	#2603	h	#3E4D
)	#3950	I	#268A	i	#3ED4
*	#48FC	J	#26B9	j	#3F03
+	#39BF	K	#2740	k	#3F72
,	#3A40	L	#27F3	l	#3FF9
-	#3A68	M	#287A	m	#4028
.	#3A59	N	#292D	n	#40B3
/	#3770	O	#29B4	o	#411C
0	#312A	P	#2A67	p	#4185
1	#31B1	Q	#45F1	q	#420C
2	#320C	R	#2AEE	r	#4293
3	#3293	S	#2B75	s	#42DA
4	#331A	T	#2BFC	t	#4343
5	#33A1	U	#2C83	u	#439A
6	#3428	V	#2D0A	v	#44D5
7	#34AF	W	#46B4	w	#49AF
8	#3536	X	#2DBD	x	#4403
9	#35BD	Y	#4793	y	#4A3A
:	#4988	Z	#2E70	z	#446C
;	#4957	[*	#2FD2	{ *	#4D4C
< *	#2EF7	\	#20C2	| *	#4DF1
=	#3AB9	] *	#4BC3	} *	#4C44
> *	#4B42	^	#22B4	~ *	#4AC1
?	#4875	_	#20C0	⊞	#53D7

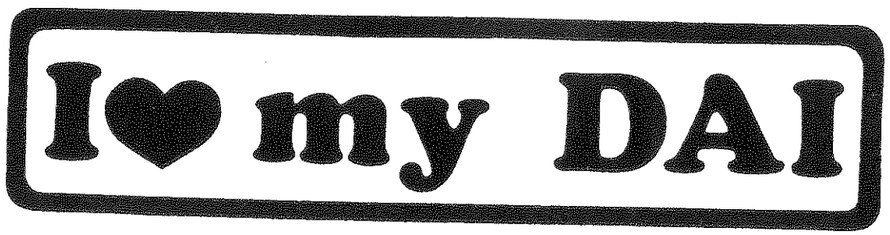


5-2.MODE PROGRAMMABLE

Nous voulons par exemple réaliser un générique vidéo se déroulant comme suit:
- écran bleu clair pendant 2 secondes
- apparition du texte DAInamic en lettres blanches bordées de noir
- après une seconde, apparition d'une bande bleu foncé derrière le texte
- ensuite,chaque lettre se détache en relief
- pour terminer, une deuxième ligne de texte apparait et affiche CLUB sans relief

Le programme est donné ci-après.Il est évident que,pour qu'il soit exécutable,il faut charger la police HELVETICA 22.

```
10    REM ..................................GENERIQUE
20    COLORG 12 0 12 15:MODE 6  ............INITIALISATION
30    WAIT TIME 100  .......................ATTENTE 2 SEC
40    E%=1  ................................RELIEF
50    X%=70:Y%=143  ........................COORDONNEES
60    A$="DAInamic"  .......................TEXTE 1ERE LIGNE
70    GOSUB 210
80    WAIT TIME 50  ........................ATTENTE 1 SEC
90    POKE #88BC,#CC  ......................BANDE DE COULEUR
100   POKE #A69E,#C1
110   E%=10  ...............................RELIEF
120   X%=70:Y%=143  ........................COORDONNEES
130   A$="DAInamic"
140   GOSUB 210
150   E%=1  ................................RELIEF
160   X%=140:Y%=105  .......................COORDONNEES
170   A$="Club."  ..........................TEXTE 2E LIGNE
180   GOSUB 210
190   WAIT TIME 200  .......................ATTENTE 4 SEC
200   GOTO 10  .............................ON RECOMMENCE !
210   REM ..................SORTIE DE CHAQUE CARACTERE
220   FOR I%=1 TO LEN(A$)
230   A%=ASC(MID$(A$,I%-1,1))
240   FOR J%=1 TO E%
250   GOSUB 340
260   CALLM #1ED0
270   X%=X%-1:Y%=Y%+1
280   NEXT J%
290   X%=X%+E%:Y%=Y%-E%
300   X1%=PEEK(#1FF7)
310   X%=X%+X1%+XX%
320   NEXT I%
330   RETURN
340   REM ..................EMISSION DE CHAQUE CARACTERE
350   DY%=255-Y%
360   CB%=#BFEF-DY%*90
370   IF X%<0 THEN X%=0
380   DX%=X% MOD 8
390   AB%=CB%-4-2*(X%/8)
400   LB%=AB% MOD 256
410   HB%=AB%/256
420   POKE #1FFC,A%
430   POKE #1FFD,DX%
440   POKE #1FFE,LB%
450   POKE #1FFF,HB%
460   RETURN
```



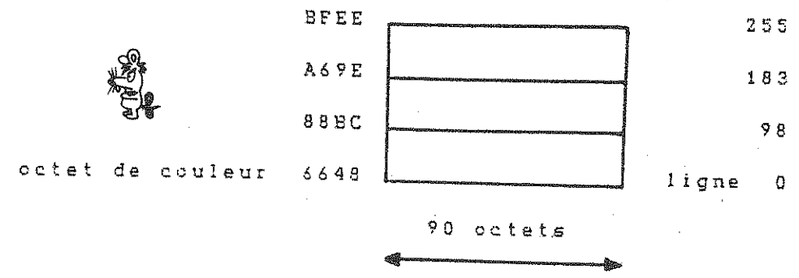
5-3.DESSIN D'UNE BANDE DE COULEUR.

Nous disposons de couleurs supplémentaires en écrivant directement dans l'octet de controle de couleur des lignes de la mémoire écran. En MODE 6,il ya 256 lignes numérotées de 0 à 255 de bas en haut La ligne 0 de la mémoire écran contient:

- 2 octets de controle	6648	6649
- 2 octets pour le bord	6646	6647
- 84 octets pour la ligne	65F2	6645
- 2 octets pour le bord	65F0	65F1

Les 84 octets pour la ligne permettent de coder les 336 points en 4 couleurs (2 bits par point), soit 336 * 2 / 8 octets.
L'écart entre 2 lignes est dès lors de 90 octets (en décimal). L'octet de couleur de la ligne 0 est en 6648 (H). L'octet de couleur de la ligne 255 est en BFEE (H). Le tableau de la page suivante donne l'adresse de l'octet de couleur de chacune des lignes. La couleur s'écrit #C.

#C0 noir	#C8 gris	
1 bleu vif	9 bleu marine	
2 pourpre	A orange	(10)
3 rouge vif	B mauve	(11)
4 violet	C bleu clair	(12)
5 vert	D vert clair	(13)
6 brun clair	E jaune	(14)
7 brun moutarde	F blanc	(15)



La couleur générale du fond est donnée par le COLORG.
Si nous voulons un fond bleu clair: COLORG 12 0 12 15
En écrivant dans l'octet de couleur d'une ligne, l'écran prend cette couleur de cette ligne jusqu'à la ligne 0.
Si nous voulons dessiner une bande de couleur bleu vif de la ligne 183 à la ligne 98,il suffit d'écrire:

```
POKE #88EC,#CC  couleur bleu clair  0 - 98
POKE #A69E,#C1  couleur bleu vif   98 - 183
```

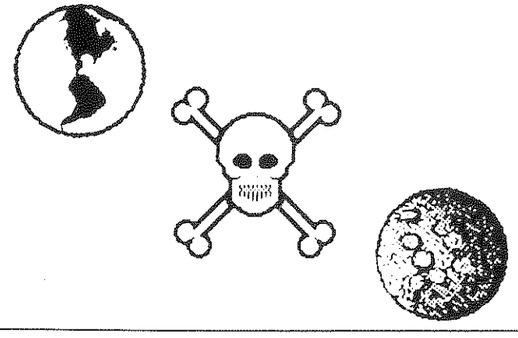
La couleur de 183 à 255 reste celle du COLORG

0	6649	64	7CC8	128	9348	192	A9C8
1	66A2	65	7D22	129	93A2	193	AA22
2	66FC	66	7D7C	130	93FC	194	AA7C
3	6756	67	7DD6	131	9456	195	AAD6
4	67B0	68	7E30	132	94B0	196	AB30
5	680A	69	7E8A	133	950A	197	AB8A
6	6864	70	7EE4	134	9564	198	ABE4
7	68BE	71	7F3E	135	95BE	199	AC3E
8	6918	72	7F98	136	9618	200	AC98
9	6972	73	7FF2	137	9672	201	ACF2
10	69CC	74	804C	138	96CC	202	AD4C
11	6A26	75	80A6	139	9726	203	ADA6
12	6A80	76	8100	140	9780	204	AE00
13	6ADA	77	815A	141	97DA	205	AE5A
14	6B34	78	81B4	142	9834	206	AEB4
15	6B8E	79	820E	143	988E	207	AF0E
16	6BE8	80	8268	144	98E8	208	AF68
17	6C42	81	82C2	145	9942	209	AFC2
18	6C9C	82	831C	146	999C	210	B01C
19	6CF6	83	8376	147	99F6	211	B076
20	6D50	84	83D0	148	9A50	212	B0D0
21	6DAA	85	842A	149	9AAA	213	B12A
22	6E04	86	8484	150	9B04	214	B184
23	6E5E	87	84DE	151	9B5E	215	B1DE
24	6EB8	88	8538	152	9BB8	216	B238
25	6F12	89	8592	153	9C12	217	B292
26	6F6C	90	85EC	154	9C6C	218	B2EC
27	6FC6	91	8646	155	9CC6	219	B346
28	7020	92	86A0	156	9D20	220	B3A0
29	707A	93	86FA	157	9D7A	221	B3FA
30	70D4	94	8754	158	9DD4	222	B454
31	712E	95	87AE	159	9E2E	223	B4AE
32	7188	96	8808	160	9E88	224	B508
33	71E2	97	8862	161	9EE2	225	B562
34	723C	98	88BC	162	9F3C	226	B5BC
35	7296	99	8916	163	9F96	227	B616
36	72F0	100	8970	164	9FF0	228	B670
37	734A	101	89CA	165	A04A	229	B6CA
38	73A4	102	8A24	166	A0A4	230	B724
39	73FE	103	8A7E	167	A0FE	231	B77E
40	7458	104	8AD8	168	A158	232	B7D8
41	74B2	105	8B32	169	A1B2	233	B832
42	750C	106	8B8C	170	A20C	234	B88C
43	7566	107	8BE6	171	A266	235	B8E6
44	75C0	108	8C40	172	A2C0	236	B940
45	761A	109	8C9A	173	A31A	237	B99A
46	7674	110	8CF4	174	A374	238	B9F4
47	76CE	111	8D4E	175	A3CE	239	BA4E
48	7728	112	8DA8	176	A428	240	BAA8
49	7782	113	8E02	177	A482	241	BB02
50	77DC	114	8E5C	178	A4DC	242	BB5C
51	7836	115	8EB6	179	A536	243	BBB6
52	7890	116	8F10	180	A590	244	BC10
53	78EA	117	8F6A	181	A5EA	245	BC6A
54	7944	118	8FC4	182	A644	246	BCC4
55	799E	119	901E	183	A69E	247	BD1E
56	79F8	120	9078	184	A6F8	248	BD78
57	7A52	121	90D2	185	A752	249	BDD2
58	7AAC	122	912C	186	A7AC	250	BE2C
59	7B06	123	9186	187	A806	251	BE86
60	7B60	124	91E0	188	A860	252	BEE0
61	7BBA	125	923A	189	A8BA	253	BF3A
62	7C14	126	9294	190	A914	254	BF94
63	7C6E	127	92EE	191	A96E	255	BFEE

BASIC : Morpion 10 x 10

```
2   REM JEU DU MORPION SUR 10X10 CASES
4   REM AUTEUR : GILSON FELICIEN
5   REM COPYRIGHT L'ORDINATEUR INDIVIDUEL
6   REM .                02-1985
7   REM
8   MODE 0:COLORT 0 10 0 0
9   GOTO 30: REM .      PROLOGUE DU JEU
10  PRINT CHR$(12): REM .      BOUCLE DES PARTIES
12  GOSUB 600: REM .      PROLOGUE DE LA PARTIE
14  GOSUB 700: REM .      PARTIE
15  PRINT :INPUT "          Voulez-vous rejouer  O/N ";R$
16  R1$=LEFT$(R$,1):IF R1$<>"O" AND R1$<>"N" THEN PRINT CHR$(12):PRINT
"Repondez par OUI ou NON svp.":WAIT TIME 100:GOTO 15
18  IF R$="O" GOTO 10
20  GOSUB 900
25  END
30  PRINT CHR$(12):WAIT TIME 50
50  POKE &29B,&BC:POKE &29C,&2:CLEAR 5000:DIM MLP(20,0):RESTORE:I%=0
60  READ A$:IF A$<>0.0 THEN POKE &2BE+I%,A$:I%=I%+1:GOTO 60
70 DATA &F5,&C5,&E5,&23,&23,&46,&23,&4E,&2A,&9E,&2,&C3,&6,&3,&CD,&39, &DE,
&7E,&B7,&3E,&4,&CA,&F5
80  DATA &D9,&E5,&CD,&1F,&3,&E1,&DA,&FC,&2,&3E,&4,&C2,&F5,&D9
90  DATA
&22,&24,&1,&3E,&FF,&32,&23,&1,&E1,&C1,&F1,&C9,&23,&7E,&B8,&C0,&23,&7E,&B9,&
C9,0
100 REM .            PROLOGUE DU JEU
120 DIM A(99,0),B(99,0),V(21,0),CO%(9,0)
130 DIM XJ%(8,0),YJ%(6,0),XJH%(8,0),YJH%(8,0)
200 PAS%=19:COEF%=190:CORR%=76
260 FOR J%=0 TO 21
270 V(J%)=0.0
280 NEXT
285 PTR%=320:CALLM 750,PTR%
290 FOR J%=0 TO 4
300 READ V(J%)
310 NEXT
320 DATA .01,.03,.5,10,10000
325 PTR%=360:CALLM 750,PTR%
330 FOR J%=5 TO 20 STEP 5
340 READ V(J%)
350 NEXT
360 DATA .1,2,100,1000000
370 FOR I%=1 TO 9
380 CO%(I%)=COEF%*I%-CORR%:NEXT
490 R%=0
550 GOSUB 7000
590 GOTO 10
600 GOSUB 3000
610 REM PROLOGUE DE LA PARTIE
630 FOR J%=0 TO 99
640 A(J%)=0.0:B(J%)=0.0
650 NEXT
660 N%=0:F%=0
670 RETURN
700 REM PARTIE
710 PRINT CHR$(12):CURSOR 12,2:INPUT "Voulez-vous commencer ";R$
720 R1$=LEFT$(R$,1):IF R1$<>"O" AND R1$<>"N" THEN PRINT :PRINT "
Repondez par OUI ou NON svp.":WAIT TIME 100:GOTO 710
730  IF R$="O" THEN PRINT CHR$(12):GOTO 770
735  PRINT CHR$(12)
740 REM BOUCLE
750  GOSUB 5000:GOSUB 1000
760  IF F%<>0.0 GOTO 800
770  GOSUB 5000:GOSUB 2000
780  IF F%=0 GOTO 740
800 REM FIN DE LA PARTIE
810  PRINT CHR$(12)
830  IF F%=1 THEN GOSUB 10000
840  IF F%=(-1) THEN GOSUB 20000
850  IF F%<>1 AND F%<>(-1) THEN GOSUB 30000
860  CURSOR 12,2:PRINT "Nous avons joue cette partie en";N%;" coups ..."
870  RETURN
900 REM EPILOGUE DU JEU
905  PRINT CHR$(12)
910  CURSOR 12,1:PRINT "Au revoir et... quand vous voudrez !!!"
920  WAIT TIME 150:PRINT CHR$(12)
930  RETURN
1000 REM LE PROGRAMME JOUE
1030 FOR J%=0 TO 99:B(J%)=0.0:NEXT
1050 CURSOR 12,3:PRINT "Je joue ..."
1090 C%=1:J1%=0:J2%=5:K1%=0:K2%=90:GOSUB 1500
1100 C%=9:J1%=4:J2%=9:K1%=0:K2%=50:GOSUB 1500
1110 C%=10:J1%=0:J2%=9:K1%=0:K2%=50:GOSUB 1500
1120 C%=11:J1%=0:J2%=5:K1%=0:K2%=50:GOSUB 1500
1170 S=-1.0
1180 Q=0.0
1200 FOR J%=0 TO 99
1210 IF A(J%)<>0.0 OR B(J%)<S GOTO 1230
1220 IF B(J%)>S THEN C%=J%:S=B(C%):Q=1.0
1225 IF B(J%)<=S THEN Q=Q+1.0:IF Q*RND(R%)<1.0 THEN C%=J%
1230 NEXT
1235 CURSOR 24,3:PRINT "
1240 CURSOR 24,3:PRINT "en";C%:WAIT TIME 15
1250 A(C%)=5.0
1260 GOSUB 4500
1270 N%=N%+1
1280 IF N%=100 OR (S=0 AND N%>1.0) THEN F%=2
1290 IF S>=V(20,0) THEN F%=1
1300 RETURN
1500 REM EXPLORATION DES QUINTUPLETS
1520 CA4%=4*C%
1530 FOR J%=J1% TO J2%
1540 FOR K%=K1% TO K2% STEP 10
1550 A%=J%+K%:B%=A%+CA4%:Z%=0
1570 FOR P%=A% TO B% STEP C%:Z%=Z%+A(P%):NEXT
1580 Q=V(Z%)
1590 IF Q<>0.0 THEN FOR P%=A% TO B% STEP C%:B(P%)=B(P%)+Q:NEXT
1600 NEXT
1610 NEXT
1620 RETURN
2000 REM L'ADVERSAIRE JOUE
2005 CURSOR 12,2:PRINT "                    "
2010 CURSOR 12,1:PRINT "                    "
2020 CURSOR 12,2:INPUT "Qu'elle case jouez-vous ";A%
2030 IF A%<0 OR A%>99 THEN CURSOR 12,1:PRINT "Attention de 0 a 99
svp.":WAIT TIME 120:GOTO 2005
2040 IF A(A%)<>0.0 THEN CURSOR 12,1:PRINT "Cette case est deja occupee
!!!":WAIT TIME 100:GOTO 2005
2060 A(A%)=1.0
2070 GOSUB 4000
2080 N%=N%+1
2100 IF N%=100 THEN F%=2
2110 IF B(A%)>=V(4,0) THEN F%=-1
2220 RETURN
3000 COLORT 8 10 0 0:COLORG 8 0 14 1:MODE 6A:C1%=8:C2%=0:C3%=14:C4%=1
3010 DIM XB%(4,0),YB%(4,0),XH%(4,0),YH%(4,0)
3020 FILL 70,12 268,210 C2%
3025 PTR%=3040:CALLM 750,PTR%
3030 FOR I%=1 TO 4:READ XB%(I%),YB%(I%),XH%(I%),YH%(I%):NEXT
3040 DATA 4,0,6,10,2,1,8,9,1,2,9,8,0,4,10,6
3050 FOR Y%=191 TO 20 STEP -19
3060 FOR X%=78 TO 249 STEP 19
3070 FOR I%=1 TO 4:FILL X%+XB%(I%),Y%+YB%(I%) X%+XH%(I%),Y%+YH%(I%)
C1%:NEXT
3080 NEXT:NEXT
3085 REM . Affichage des chiffres
3087 PTR%=3110:CALLM 750,PTR%
3090 FOR I%=1 TO 12
3100 READ XL%,YL%
3110 DATA
57,193,63,193,63,174,63,155,63,136,63,117,63,98,63,79,63,60,63,41,63,22,61,3
3120 DRAW XL%+1,YL% XL%+3,YL% C2%:DRAW XL%+1,YL%+6 XL%+3,YL%+6
C2%:DRAW XL%,YL%+1 XL%,YL%+5 C2%:DRAW XL%+4,YL%+1 XL%+4,YL%+5 C2%
3130 NEXT
3135 PTR%=3160:CALLM 750,PTR%
3140 FOR I%=1 TO 2
3150 READ XL%,YL%
3160 DATA 58,174,101,3
3170 DRAW XL%,YL% XL%+2,YL% C2%:DRAW XL%+1,YL% XL%+1,YL%+6 C2%:DOT
XL%,YL%+5 C2%
3180 NEXT
3185 PTR%=3210:CALLM 750,PTR%
3190 FOR I%=1 TO 2
3200 READ XL%,YL%
3210 DATA 57,155,119,3
3220 DRAW XL%,YL% XL%+4,YL% C2%:DRAW XL%+1,YL%+1 XL%+4,YL%+4
C2%:DRAW XL%+1,YL%+6 XL%+3,YL%+6 C2%:DOT XL%,YL%+5 C2%:DOT
XL%+4,YL%+5 C2%
3230 NEXT
3235 PTR%=3260:CALLM 750,PTR%
3240 FOR I%=1 TO 2
3250 READ XL%,YL%
3260 DATA 57,136,138,3
3270 DRAW XL%,YL% XL%+3,YL% C2%:DRAW XL%,YL%+6 XL%+3,YL%+6
C2%:DRAW XL%+1,YL%+3 XL%+4,YL%+6 C2%
```

```
3280 DRAW XL%+1,YL%+3 XL%+3,YL%+3 C2%:DRAW XL%+4,YL%+1 XL%+4,YL%+2
C2%
3290 NEXT
3295 PTR%=3320:CALLM 750,PTR%
3300 FOR I%=1 TO 2
3310 READ XL%,YL%
3320 DATA 57,117,157,3
3330 DRAW XL%,YL%+2 XL%+4,YL%+2 C2%:DRAW XL%+3,YL% XL%+3,YL%+6
C2%:DRAW XL%,YL%+3 XL%+2,YL%+5 C2%
3340 NEXT
3345 PTR%=3370:CALLM 750,PTR%
3350 FOR I%=1 TO 2
3360 READ XL%,YL%
3370 DATA 57,98,176,3
3380 DRAW XL%+1,YL% XL%+3,YL% C2%:DRAW XL%,YL%+4 XL%+3,YL%+4
C2%:DRAW XL%,YL%+6 XL%+4,YL%+6 C2%
3390 DRAW XL%+4,YL%+1 XL%+4,YL%+3 C2%:DOT XL%,YL%+1 C2%:DOT XL%,YL%+5
C2%
3400 NEXT
3405 PTR%=3430:CALLM 750,PTR%
3410 FOR I%=1 TO 2
3420 READ XL%,YL%
3430 DATA 57,79,195,3
3440 DRAW XL%+1,YL% XL%+3,YL% C2%:DRAW XL%+1,YL%+3 XL%+3,YL%+3
C2%:DRAW XL%+2,YL%+6 XL%+3,YL%+6 C2%
3450 DRAW XL%,YL%+1 XL%,YL%+4 C2%:DRAW XL%+4,YL%+1 XL%+4,YL%+2
C2%:DOT XL%+1,YL%+5 C2%
3460 NEXT
3465 PTR%=3490:CALLM 750,PTR%
3470 FOR I%=1 TO 2
3480 READ XL%,YL%
3490 DATA 57,60,214,3
3500 DRAW XL%+2,YL% XL%+2,YL%+2 C2%:DRAW XL%+2,YL%+3 XL%+4,YL%+5
C2%:DRAW XL%,YL%+6 XL%+4,YL%+6 C2%
3510 NEXT
3515 PTR%=3540:CALLM 750,PTR%
3520 FOR I%=1 TO 2
3530 READ XL%,YL%
3540 DATA 57,41,233,3
3550 DRAW XL%+1,YL% XL%+3,YL% C2%:DRAW XL%+1,YL%+3 XL%+3,YL%+3
C2%:DRAW XL%+1,YL%+6 XL%+3,YL%+6 C2%
3560 DRAW XL%,YL%+1 XL%,YL%+2 C2%:DRAW XL%,YL%+4 XL%,YL%+5
C2%:DRAW XL%+4,YL%+1 XL%+4,YL%+2 C2%:DRAW XL%+4,YL%+4 XL%+4,YL%+5
C2%
3570 NEXT
3575 PTR%=3600:CALLM 750,PTR%
3580 FOR I%=1 TO 2
3590 READ XL%,YL%
3600 DATA 57,22,252,3
3610 DRAW XL%+1,YL% XL%+2,YL% C2%:DRAW XL%+1,YL%+3 XL%+3,YL%+3
C2%:DRAW XL%+1,YL%+6 XL%+3,YL%+6 C2%
3620 DRAW XL%,YL%+4 XL%,YL%+5 C2%:DRAW XL%+3,YL%+1 XL%+4,YL%+2
C2%:DRAW XL%+4,YL%+2 XL%+4,YL%+5 C2%
3630 NEXT
3700 PTR%=3740:CALLM 750,PTR%
3710 FOR I%=1 TO 7
3720 READ XJ%(I%),YJ%(I%),XJH%(I%),YJH%(I%)
3730 NEXT
3740 DATA 0,8,20,12,1,6,19,14,2,4,18,16,3,3,17,17,4,2,16,18,6,1,14,19,8,0,12,20
3900 RETURN
     ------------------------------------------------------------
4000 REM .Le partenaire joue
     ------------------------------------------------------------
4005 CHI%=A%
4010 IF CHI%<10 THEN Y%=191:X%=CORR%+CHI%*PAS%:GOTO 4300
4020 IF CHI%<20 AND CHI%>9 THEN Y%=172:X%=CHI%*PAS%-CO%(1):GOTO 4300
4030 IF CHI%<30 AND CHI%>19 THEN Y%=153:X%=CHI%*PAS%-CO%(2):GOTO
4300
4040 IF CHI%<40 AND CHI%>29 THEN Y%=134:X%=CHI%*PAS%-CO%(3):GOTO
4300
4050 IF CHI%<50 AND CHI%>39 THEN Y%=115:X%=CHI%*PAS%-CO%(4):GOTO
4300
4060 IF CHI%<60 AND CHI%>49 THEN Y%=96:X%=CHI%*PAS%-CO%(5):GOTO 4300
4070 IF CHI%<70 AND CHI%>59 THEN Y%=77:X%=CHI%*PAS%-CO%(6):GOTO 4300
4080 IF CHI%<80 AND CHI%>69 THEN Y%=58:X%=CHI%*PAS%-CO%(7):GOTO 4300
4090 IF CHI%<90 AND CHI%>79 THEN Y%=39:X%=CHI%*PAS%-CO%(8):GOTO 4300
4100 IF CHI%>89 THEN Y%=20:X%=CHI%*PAS%-CO%(9)
4300 FOR I%=1 TO 4:FILL X%+XB%(I%),Y%+YB%(I%) X%+XH%(I%),Y%+YH%(I%)
C3%:NEXT
4310 PRINT CHR$(12):RETURN
     ------------------------------------------------------------
4500 REM .L'ordinateur joue
     ------------------------------------------------------------
4505 CHI%=C%
4510 IF CHI%<10 THEN Y%=191:X%=CORR%+CHI%*PAS%:GOTO 4800
4520 IF CHI%<20 AND CHI%>9 THEN Y%=172:X%=CHI%*PAS%-CO%(1):GOTO 4800
4530 IF CHI%<30 AND CHI%>19 THEN Y%=153:X%=CHI%*PAS%-CO%(2):GOTO
4800
4540 IF CHI%<40 AND CHI%>29 THEN Y%=134:X%=CHI%*PAS%-CO%(3):GOTO
4800
4550 IF CHI%<50 AND CHI%>39 THEN Y%=115:X%=CHI%*PAS%-CO%(4):GOTO
4800
4560 IF CHI%<60 AND CHI%>49 THEN Y%=96:X%=CHI%*PAS%-CO%(5):GOTO 4800
4570 IF CHI%<70 AND CHI%>59 THEN Y%=77:X%=CHI%*PAS%-CO%(6):GOTO 4800
4580 IF CHI%<80 AND CHI%>69 THEN Y%=58:X%=CHI%*PAS%-CO%(7):GOTO 4800
4590 IF CHI%<90 AND CHI%>79 THEN Y%=39:X%=CHI%*PAS%-CO%(8):GOTO 4800
4595 IF CHI%>89 THEN Y%=20:X%=CHI%*PAS%-CO%(9)
4800 FOR I%=1 TO 4:FILL X%+XB%(I%),Y%+YB%(I%) X%+XH%(I%),Y%+YH%(I%)
C4%:NEXT
4900 WAIT TIME 50:RETURN
5000 X1%=296:Y1%=101:FILL 17,100 40,125 C1%
5020 FOR I%=1 TO 7
5060 FILL X1%+XJ%(I%),Y1%+YJ%(I%) X1%+XJH%(I%),Y1%+YJH%(I%) C3%:NEXT
5100 RETURN
6000 X2%=18:Y2%=101:FILL 295,100 318,125 C1%
6020 FOR I%=1 TO 7
6030 FILL X2%+XJ%(I%),Y2%+YJ%(I%) X2%+XJH%(I%),Y2%+YJH%(I%) C4%:NEXT
6100 RETURN
7000 PTR%=7060:CALLM 750,PTR%
7010 COLORG 0 0 0 0:MODE 3A:WAIT TIME 25
7020 FOR I%=1 TO 116
7030 READ X%,Y%
7032 IF X%>103 THEN COL%=12:GOTO 7050
7034 IF X%>95 THEN COL%=10:GOTO 7050
7036 IF X%>87 THEN COL%=14:GOTO 7050
7038 IF X%>80 THEN COL%=11:GOTO 7050
7040 IF X%>71 THEN COL%=13:GOTO 7050
7042 IF X%>63 THEN COL%=2:GOTO 7050
7044 IF X%>55 THEN COL%=15:GOTO 7050
7046 IF X%>47 THEN COL%=1
7050 DOT X%,Y% COL%:WAIT TIME 6:NEXT
7060 DATA 76,54,96,53,88,53,81,56,100,55,56,54,72,56,50,54,64,53,74,53
7070 DATA 48,51,106,50,98,53,68,50,58,50,82,53,82,50,60,55,96,51,90,56
7080 DATA 52,53,74,56,48,56,104,50,60,53,72,50,92,54,66,56,96,56,72,53
7090 DATA 89,56,90,50,56,53,48,54,99,52,82,54,64,56,67,53,100,52,72,54
7100 DATA 100,50,58,56,52,50,97,55,64,51,100,56,75,56,83,50,92,52,91,56
7110 DATA 48,50,72,52,108,50,88,52,59,56,96,55,65,53,92,55,81,50,98,54
7120 DATA 49,55,56,51,100,54,64,50,83,56,52,56,88,55,66,53,68,55,56,55
7130 DATA 82,56,105,50,52,55,88,51,60,51,100,53,67,51,82,51,72,51,96,50
7140 DATA 48,53,89,50,73,56,107,50,65,56,96,54,64,55,82,52,56,52,75,53
7150 DATA 51,55,91,50,72,55,52,51,67,56,88,54,48,55,96,52,57,50,60,54
7160 DATA 82,55,64,52,57,56,48,52,92,51,64,54,76,55,52,54,68,54,59,50
7170 DATA 100,51,60,52,92,53,66,52,52,52,73,53
7180 WAIT TIME 50
7190 FOR X%=48 TO 108
7200 DOT X%,47 14:WAIT TIME 2:NEXT
7300 WAIT TIME 100
7500 CURSOR 50,1:PRINT "F.Gilson"
7600 WAIT TIME 120:PRINT CHR$(12)
7610 CURSOR 18,3:PRINT "A L I G N E Z - E N 5"
7620 PRINT "   Horizontalement, verticalement ou obliquement"
7630 PRINT :PRINT "                    Vous jouez contre l'ordinateur";
7640 WAIT TIME 200
8000 RETURN
10000 CURSOR 12,3:PRINT "Je gagne. Alors ? ..... perseverez !!!"
10050 WAIT TIME 150
10100 RETURN
20000 CURSOR 12,3:PRINT "Pas mal, vous savez vous y prendre."
20050 WAIT TIME 150
20100 RETURN
30000 CURSOR 12,3:PRINT "Vous voyez, je ne gagne pas"
30010 WAIT TIME 50
30100 CURSOR 12,3:PRINT "Encore un petit effort ... pour me battre !!!"
31000 RETURN
```



Nouveaux tarifs DCA *Hard*

Gamme Exteriodai

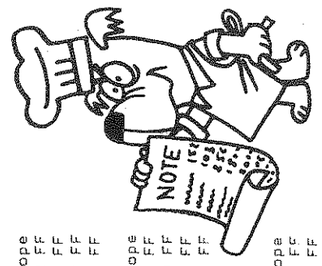
* Cable de liaison	France	Europe
Prix club VI	88 FF	94 FF
Prix ext. VI	96 FF	102 FF
* Ananum	France	Europe
Prix club VI	460 FF	475 FF
Prix ext. VI	495 FF	510 FF
Prix club VK	470 FF	485 FF
Prix ext. VK	505 FF	520 FF
* Daidev	France	Europe
Prix club VI	187 FF	202 FF
Prix ext. VI	206 FF	221 FF
Prix club VK	197 FF	212 FF
Prix ext. VK	216 FF	231 FF
* SecteurDai	France	Europe
Prix club VI	403 FF	419 FF
Prix ext. VI	444 FF	470 FF
Prix club VK	413 FF	429 FF
Prix ext. VK	454 FF	480 FF

Autre gamme

* Electrofus	France	Europe
Prix club VI	56 FF	62 FF
Prix ext. VI	62 FF	68 FF
Prix club VK	61 FF	67 FF
Prix ext. VK	67 FF	73 FF
* Daispot	France	Europe
Prix club VI	350 FF	365 FF
Prix ext. VI	385 FF	400 FF
Prix club VK	360 FF	375 FF
Prix ext. VK	395 FF	410 FF
* Interface Joystick	France	Europe
Prix club VI	35 FF	45 FF
Prix ext. VI	39 FF	49 FF
Prix club VK	40 FF	50 FF
Prix ext. VK	44 FF	54 FF

Nous sommes désolé de cette augmentation mais l'avons calculé afin de maintenir les prix stables pendant au moins un an. Néanmoins je pense que, comparé aux kits du commerce, les notres sont accessibles et de bonne qualité. Je vous parait peut être vendeur de tapis mais je suis entièrement ouvert aux propositions alléchantes sur les composants.

Ici s'arrête ma dernière intervention en tant que responsable IDC, trésorier, bidouilleur, secrétaire (et j'en passe et des meilleures pour en laisser un peu aux autres) de DCA. Aussi au revoir et à bientôt (c'est promis, je vous envois une carte postale de Brest !!!).



DCA infos

Tout d'abord de mauvaises nouvelles. Comme annoncé dans le dernier bulletin de liaison interne à DCA je m'en vais pour trois ans (snif) à Brest dans le cadre de mes études. Aussi les rôles sont-ils redistribués au sein du club. Ne m'envoyez plus rien SVP car votre demande risque d'être très fortement retardé. Je reviendrais en effet tous les deux mois environ sur Paris...

Pascal Hertzog 2 avenue Jean Moulin
94350 Villiers sur Marne
(1) 43.05.74.78
* Problèmes divers avec votre Dai (réparation notamment).
* Softthèque DCR et audio.
* DCA *hard*.
* Achats de fournitures de tout ordre.

Luc D'Arantès 10 D. Rés. Ste Marie
Chemin Ste Marie
30200 Bagnols/Cèze
66.89.57.60
serveur : DALU
* Trésorerie (tous les chèques doivent être dorénavant à son nom).
* Relation avec la Belgique (achat pgm, revue, etc).
* Regroupement articles.
* Softthèque VC1541.

Jean Guérard 6, rue du Parc
92190 Meudon
(1) 46.26.34.18
* Gestion du serveur (pseudonyme : PHILEAS).

Alexis Ferrero 57, Bld St Michel
75005 Paris
(1) 43.25.86.14
* Edition des fiches.

A présent je voudrais vous prier de bien vouloir nous excuser pour le retard qu'a pris l'envoi des réponses au demandes de certains d'entre vous. Retard causé par les vacances et le surcroit de travail du secrétaire.

D'autre part la branche hard du club se développe de plus en plus et nous ne pouvons que nous en féliciter puisque son but est la diffusion en kit des interfaces. A ce propos, ceux d'entres vous qui ont commandé des VK ou VI sont-ils satisfait des notices livrées avec les cartes ? N'hésitez pas à nous faire part des améliorations que vous jugez nécessaire dans la forme ou le contenu de celles-ci. Nous avons aussi besoin de schéma, voire d'idées, pour compléter la gamme du club. Les prix de celle-ci ont d'ailleurs augmenté pour les raisons suivantes :

* les frais posteux
* les prix des composants
* la sous-traitance des circuits imprimés afin de leur assurer une qualité optimale.

Petites Annonces

* A vendre DAI + floppy 2x320K (nov. 84) + poignées + magnéto + très nombreuses docs + centaine programmes + 15 disquettes + revues 75000 Fb à débattre (11000 FF) achetés 140000 Fb. Vends imprimante TANDY DMP105 compatible DAI, 12600 Fb (1800 FF) début 86, état neuf. S'adresser à Dominique Guillot, 8 rue Diderot, F-42300 Roannes.

* A vendre, accessoires pour DAI :

- Carte Xbasic complète (avec Qsave et nouv. possibilités)	5999,-	FB
- Drive 1541 état neuf avec cables et manuel	7999,-	FB
- +/- 1700 pgms sur 75 disk + 1 gros classeur de modes d'emplois de programmes. (format DISK VC1541)	8999,-	FB
- Paddle INDATA de précision (2 pieces)	1499,-	FB
- Firmware manual original	599,-	FB
- Interface pour imprimante Centronics (Epson,...)	999,-	FB
- Livre "Assembleur 8080 sur DAI"	599,-	FB
- Série des revues du club francais DCF	599,-	FB
- Livre "TOS désassemblé"	499,-	FB
- Livre "Hardware du 8080"	999,-	FB
- Schémas électroniques du DAI	499,-	FB
- Toute la série des DAInamic Germany + des DAInamic Belgium	3998,-	FB
- Paddle 3 boutons (pour logiciels graphiques)	499,-	FB
- 22 cassettes DCR	1999,-	FB

Possibilité de prix spéciaux par lot. Christian Poels (IDC)

* A vendre :

- ROM's BASIC V1.2 (cfr DAIclic 3) avec adaptateur DAI	1400,-	FB
- Lot de 25 disquettes bourrées de programmes (quasiment tout ce qui existe sur le DAI) au format KENDOS 800K	3750,-	FB
- Cassettes DCR PHILIPS certifiées digitales bourrées de programmes. La pièce : 200,- FB boite de 6 cas. :	1000,-	FB
- Interface SERIE - PARALLELE Mikroshop originale	3000,-	FB
- Table de mixage HI-FI STEREO JB system NEW SA-101, 10 entrées, master, pré-écoute, talk-over, mini-equalizer 2x6 canaux. Idéal pour écouter de la musique tout en travaillant sur le DAI !!!	9000,-	FB

Contacter Marc Vandermeersch (IDC)

* A vendre :

- Souris montée et cablée pour le DAI	2000,-	FB
- Imprimante EPSON MX-82	12000,-	FB
- Interface joystick de type SPECTRAVIDEO (cf DAICLIC)	400,-	FB
- Cassettes DCR certifiées digitales (bourrées de pgms)pièce	200,-	FB
- DAI (ancien clavier) avec sortie vidéo en supplément	14000,-	FB
- Memocom MDCR + TOS (1an)	13000,-	FB
- Paddle 3 potentiomètres avec bouton event	500,-	FB
- Paddle manche-à-balai + 1 potentiomètre + event	600,-	FB
- Firmware manual de Boerighter	600,-	FB

Contactez Fabrice Duluins, 4 Allée Tour Renard, B-1400 NIVELLES

Remarque : Ces petites annonces gratuites pour les abonnés sont exclusivement réservées à des propositions entre particuliers sans objectifs commerciaux et relatives à l'informatique. DAIclic se réserve le droit de refuser une annonce sans avoir à fournir de justification.