# DAI graphics

## GASTON

Carole van Eenoo

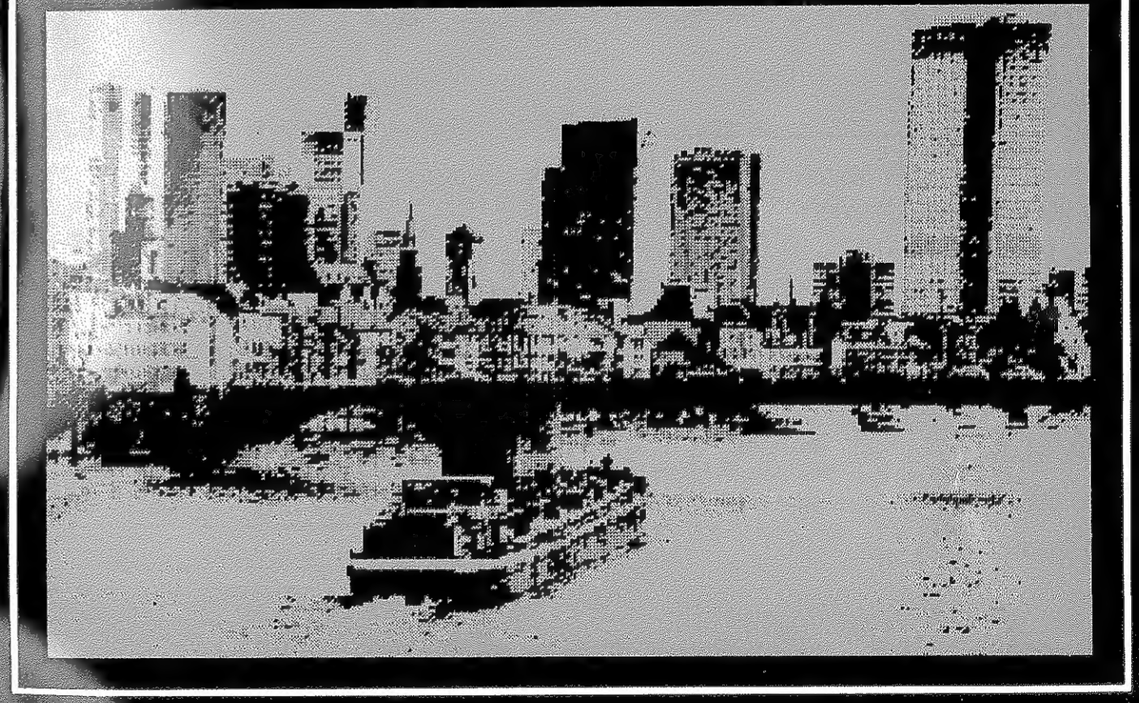
SEND YOUR DRAWINGS TO DAINAMIC


# DAI NAMIC

27

tweemaandelijks tijdschrift maart - april 1985


# FASCIMILE



## personal computer users club


een uitgave van dainamic v.z.w.
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 Govaerts |
| Bruno Van Rompaey | Daniël Govaerts |
| 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

Lidgeiden / 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

Pour la France :

DAInamic FRANCE
C. Dufour
Rue Lavoisier 9
59149 DUNKERQUE
Tel. 02866 3339

Inzendingen : Games & Strategy

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

# DAINAMIC

PERSONAL COMPUTER USERS CLUB

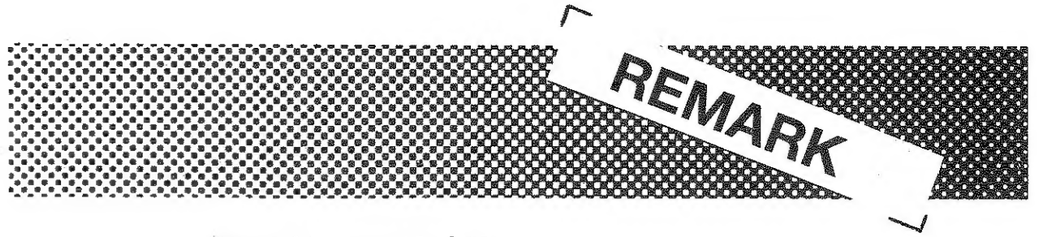
| 4 | | 3 | | 2 | | 1 | |
|---|---|---|---|---|---|---|---|
| HEX | DEC | HEX | DEC | HEX | DEC | HEX | DEC |
| 1 | 4096 | 1 | 256 | 1 | 16 | 1 | 1 |
| 2 | 8192 | 2 | 512 | 2 | 32 | 2 | 2 |
| 3 | 12288 | 3 | 768 | 3 | 48 | 3 | 3 |
| 4 | 16384 | 4 | 1024 | 4 | 64 | 4 | 4 |
| 5 | 20480 | 5 | 1280 | 5 | 80 | 5 | 5 |
| 6 | 24576 | 6 | 1536 | 6 | 96 | 6 | 6 |
| 7 | 28672 | 7 | 1792 | 7 | 112 | 7 | 7 |
| 8 | 32768 | 8 | 2048 | 8 | 128 | 8 | 8 |
| 9 | 36864 | 9 | 2304 | 9 | 144 | 9 | 9 |
| A | 40960 | A | 2560 | A | 160 | A | 10 |
| B | 45056 | B | 2816 | B | 176 | B | 11 |
| C | 49152 | C | 3072 | C | 192 | C | 12 |
| D | 53248 | D | 3328 | D | 208 | D | 13 |
| E | 54344 | E | 3584 | E | 224 | E | 14 |
| F | 61440 | F | 3840 | F | 240 | F | 15 |

**belangrijke ASCII-waarden in DAIpc**

| functie/symbool | HEX | DEC |
|---|---|---|
| back-space | 8 | 8 |
| TAB | 9 | 9 |
| linefeed | A | 10 |
| clear screen | C | 12 |
| CURSOR UP | 10 | 16 |
| CURSOR DOWN | 11 | 17 |
| CURSOR LEFT | 12 | 18 |
| CURSOR RIGHT | 13 | 19 |
| space-bar | 20 | 32 |
| Ø | 30 | 48 |
| A | 41 | 65 |
| a | 61 | 97 |
| pijltje rechts | 89 | 137 |
| pijltje links | 88 | 136 |
| pijltje boven | 5E | 94 |
| pijltje onder | 8C | 140 |
| volle blok | FF | 255 |
| verticale lijn | A | 10 |
| horizontale lijn | B | 11 |
| 6 hor. lijnen | 1D | 29 |

ASCII - HEX - ASCII CONVERSION TABLE

| LSD | MSD | 0 000 | 1 001 | 2 010 | 3 011 | 4 100 | 5 101 | 6 110 | 7 111 |
|---|---|---|---|---|---|---|---|---|---|
| 0 | 0000 | NUL | DLE | SP | 0 | @ | P | ` | p |
| 1 | 0001 | SOH | DC1 | ! | 1 | A | Q | a | q |
| 2 | 0010 | STX | DC2 | " | 2 | B | R | b | r |
| 3 | 0011 | ETX | DC3 | # | 3 | C | S | c | s |
| 4 | 0100 | EOT | DC4 | $ | 4 | D | T | d | t |
| 5 | 0101 | ENQ | NAK | % | 5 | E | U | e | u |
| 6 | 0110 | ACK | SYN | & | 6 | F | V | f | v |
| 7 | 0111 | BEL | ETB | ' | 7 | G | W | g | w |
| 8 | 1000 | BS | CAN | ( | 8 | H | X | h | x |
| 9 | 1001 | HT | EM | ) | 9 | I | Y | i | y |
| A | 1010 | LF | SUB | * | : | J | Z | j | z |
| B | 1011 | VT | ESC | + | ; | K | [ | k | { |
| C | 1100 | FF | FS | , | < | L | \ | l | \| |
| D | 1101 | CR | GS | - | = | M | ] | m | } |
| E | 1110 | SO | RS | . | > | N | ↑ | n | ~ |
| F | 1111 | SI | VS | / | ? | O | ← | o | DEL |




Herselt, april '85

Beste leden,

We ontvingen enige berichten in verband met INDATA : de distributie van de DAI zal in Belgie waargenomen worden door de firma SIDEL. Ze zullen mikken op de onderwijsmarkt, de implementatie van ELAN op de DAI zou hierbij een sterk argument moeten zijn. We hebben deze realisatie nog niet kunnen testen, we wensen de firma SIDEL alle succes.

Dit nummer biedt weer het klassieke varia aan programma's en artikelen. De presentatie is gewijzigd (verbeterd naar wij hopen?), dat zullen we wel merken aan uw reacties. Mogelijk is het interessant om over te stappen naar 2 kolommen per pagina. Dit maakt het geheel beter leesbaar en biedt ons ook de mogelijkheid om dezelfde hoeveelheid materiaal in een kleinere ruimte te publiceren. Daar we niet alle ingezonden stukken opnieuw kunnen intikken willen we voorstellen om voortaan een kolombreedte van +/- 35 karakters per lijn aan te houden.(40 opMX82).

In deze uitgave wordt slechts 1 nieuw pakket aangekondigd : PATROUILLER. Wel een programma met klasse, duidelijk nog beter dan de vorige hit van PASCAL JANIN ,'PHOENIX'.

We hebben net (nog duidelijk voelbaar) de HCC dagen Belgie achter de rug. Beslist minder bezoekers dan vorig jaar, Frank Druijff en de andere collega's vertellen hetzelfde over de computerdag in Roozendaal. Waarschijnlijk veroorzaakt de overdaad aan dergelijke manifestaties toch een beursmoeheid bij het publiek. (Of voelt het publiek een zekere moeheid in eigen beurs ??)

Jan Boerrigter en co brengen met hun DAI DOS 1541 andermaal een belangrijke bijdrage voor het DAI-gebeuren. We zien deze realisatie als een degelijk alternatief voor de DCR. Iets duurder, maar wel duidelijk meer mogelijkheden.
U kan het hele verhaal lezen op bladzijde 113 en volgende.

In de volgende uitgave vervolgt A.Beuckelaers zijn serie over de DCE-bus. In dat verband kunnen we melden dat we opnieuw een kleine voorraad hebben van blanco DCE-interface kaarten. De prijs is 1450 Bf voor de twee kaartjes (zonder componenten ).

groetjes,
tot de volgende keer

W.Hermans

# INHOUDSTAFEL

# 27







# PROGRAMMEERTECHNIEKEN

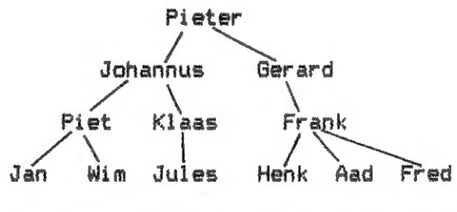
Deze keer wilde ik beginnen met een oude belofte gestand te doen. Al ruim een jaar geleden kondigde ik aan eens iets te schrijven over de functies die in de DAI-basic zitten. Dat zal dus nu gebeuren.

Volgens ons handboek , paragraaf 6.2.1.2 bezit de DAI-basic een veertigtal functies. In vergelijking tot vele andere machines is dit best een redelijk aantal. Een aantal van de functies die de DAI-basic mist zullen echter door de meeste programmeurs niet of nauwelijks gemist worden. Sterker ik ben er van overtuigd dat een groot aantal DAI-gebruikers nog nooit alle functies hebben gebruikt. Dit kan vanzelfsprekend komen door het feit dat niet iedereen alle functies nodig heeft, maar ook doordat men niet alle functies kent of niet weet hoe ze gebruikt moeten worden. In het komende zal ik trachten aan dat laatste een eind te maken, zodat dan niet gebruiken alleen komt door niet nodig hebben.

Wat is nu eigenlijk een functie ? Welnu een functie is simpel gezegd een of ander voorschrift dat aan een meegegeven argument een beeld toevoegt. Wat heet simpel gezegd zullen sommigen nu misschien verzuchten. Ik zal echter trachten de lezers die niet thuis zijn in de 'moderne wiskunde' aan de hand van wat voorbeelden toch enig inzicht in deze materie te geven.

Voorbeeld 1

Bekijk de onderstaande stamboom waarvan alleen mannelijke leden vermeld zijn.




Als functie nemen we het voorschrift
AAN zoon WORDT TOEGEVOEGD vader
We zien in de stamboom dan b.v. dat Gerard de vader is van Frank.
We zeggen dat Frank het origineel is en Gerard het beeld.
Evenzo is Johannus het beeld als we Klaas als origneel nemen.

Dit zal velen onlogisch in de oren klinken, (bij lezers onlogisch in de ogen schijnen ?) maar als we denken aan het voorschrift in kwestie is het wel logisch. Het voorschrift luidt 'aan zoon wordt toegevoegd vader' en we kunnen alleen de vader weten als aan een aantal voorwaarden is voldaan. We moeten de zoon weten waarvan de vader genoemd moet worden; deze zoon is het argument van de functie. Deze zoon zal in ons voorbeeld moeten staan omdat we er anders niets over kunnen zeggen; wiskundig: de argumenten moeten voorkomen in het domein van de functie. Het domein van de functie is dus een verzameling van argumenten waarvoor de functie gedefinieerd is. De vader die aan een zoon wordt toegevoegd is dan in deze zegswijze een beeld. De verzameling van vaders die bij de gegeven verzameling zonen (domein) komt heet het bereik. Besef wel dat hoewel Pieter biologisch wel zoon zal zijn hij toch niet in het domein zit. Eveneens is er gegeven ons voorbeeld geen vader te vinden op de onderste rij van de stamboom. Een belangrijk aspect van functies heb ik echter nog niet genoemd. Bij elke zoon hoort slechts een vader en er hoort er altijd precies een vader bij. Er zijn dus geen zonen zonder vader en er is ook geen zoon te vinden die twee of meer vaders heeft. We spreken in de wiskunde slechts van een functie als aan deze voorwaarde is voldaan. Ons voorschrift had dus ook beslist niet kunnen luiden 'aan vader wordt toegevoegd zoon' omdat dan bij bepaalde vaders niet een eenduidig antwoord is te geven en dat eist het begrip functie nu eenmaal wel.

Voorbeeld 2

Dit tweede voorbeeld is weliswaar wiskundiger maar zal toch eenvoudiger zijn te begrijpen. Als domein nemen we de getallenverzameling 0,1,2,3,4,5,6,.... van de natuurlijke getallen. Als functie neem ik: aan elk getal uit het domein wordt het getal toegevoegd dat drie groter is dan het dubbele van dat getal. We krijgen dus dat aan 2 wordt toegevoegd 7 en aan 3 wordt 9 toegevoegd. Pas op maak niet de fout die al vele middelbare scholieren hebben gemaakt : toevoegen is niet optellen maar is te vergelijken met koppelen of relateren. Het is simpel in te zien dat ook hier aan de voorwaarde voldaan wordt dat bij elk argument slechts een beeld hoort en dat dat beeld er altijd is.

Voorbeeld 3

Als derde voorbeeld wilde ik bij hetzelfde domein als bij het tweede voorbeeld als voorschrift nemen: aan elk getal uit het domein wordt toegvoegd de rest die verkregen wordt door het argument door drie te delen. Aan 7 wordt dan toegevoegd 1 en aan 11 wordt 2 toegevoegd. Pas op aan 16 wordt net als aan 7 ook 1 toegevoegd. Bij elk origineel hoort dus wel degelijk een beeld maar in tegenstelling tot voorbeeld twee hoort niet bij elk beeld een origineel. Bij voorbeeld 1 was dit trouwens ook niet zo.

Waarom deze uitleg ? Wel velen zullen problemen hebben om de functies die in het handboek genoemd worden als zodanig te herkennen. Maar met de zoeven gegeven inzichten zult U in staat zijn de functie te herkennen. Er is echter een uitzondering; n.l. PI dit is geen functie maar een standaard constante. Toch staat PI bij de functies genoemd. Als u de lijst van paragraaf 6.2.1.2 doorneemt krijgt U misschien het idee dat er nog wel een paar bijstaan die geen echte functies zijn. Hierbij denkt U dan vermoedelijk aan de functies waarbij we geen argument behoeven op te geven. We zullen echter in het vervolg zien dat de gebruikte argumenten er wel degelijk zijn al behoeven wij ze niet op te geven.

De functies van het handboek kunnen in groepen onderverdeeld worden :

| Functies met tekstvariabele als argument en getalsvariabele als beeld | Functies met getalsvariabele als argument en getalsvariabele als beeld | Functies met getalsvariabele als argument en tekstvariabele als beeld | Functies schijnbaar zonder argument en getalsvariabele als beeld |
|---|---|---|---|
| ASC | ABS | CHR$ | CURX |
| LEN | ACOS | HEX$ | CURY |
| VAL | ALOG | LEFT$ | FRE |
| VARPTR | ASIN | MID$ | GETC |
| | ATN | RIGHT$ | XMAX |
| | COS | STR$ | YMAX |
| | EXP | SPC | |
| | FRAC | TAB | |
| | FREQ | | |
| | INP | | |
| | INT | | |
| | LOG | | |
| | LOGT | | |
| | PDL | | |
| | PEEK | | |
| | RND | | |
| | SCRN | | |
| | SGN | | |
| | SIN | | |
| | SQR | | |
| | TAN | | |
| | VARPTR | | |



We beginnen de groep te bespreken die het meest intrigerend is namelijk de laatste waarin schijnbaar geen argument aanwezig is. Het argument is er wel maar wordt gegeven door de machine zelf. Bij CURX en CURY is vanzelfsprekend de positie van de cursor het argument. Voor alle zekerheid herhalen we nog even CURX geeft de horizontale positie van de cursor links beginnend bij 0 en met 1 oplopend naar rechts tot maximaal 60. Ha, ha hoor ik een aantal al grinniken; er gaan maar 60 tekens op een regel en als we bij 0 beginnen is de laatste 59 en niet 60. FOUT !!!!!! D.w.z. de beredenering klopt wel maar zoals het volgende programma zal laten zien komt de CURX wel degelijk op 60. Op deze positie kunnen we echter met PRINT niet meer iets neerzetten.

```
10      REM CURX-TEST / F.H. DRUIJFF - 19850416
20      CURSOR I,3:PRINT "x"I:C=CURX
30      CURSOR 10,7:PRINT I,C
40      I=I+1
50      H=GETC:IF H=0 GOTO 50:GOTO 20
```

Ook over CURY valt nog wel iets te zeggen. U tikt in PRINT CHR$(12); en komt aldus met de cursor linksboven te staan. Maakt het nu iets uit of U nu intikt PRINT CURY of A=CURY:PRINT A ? Probeert U het zelf maar als U het antwoord wilt weten. Beseft U zich trouwens ook dat de CURY boven aan het scherm 23 is en onderaan 0 ? Maar dat U bij intikken van PRINT CURY zowel bij de onderste regel als de een na onderste regel het antwoord 0 krijgt ? De verklaring is simpel: het intikken besluiten we met een [RETURN] en komen daarmee op de volgende regel terecht, daar komt dan de prompt of eventueel het resultaat van de gegeven opdracht. Bij de opdracht PRINT CURY op regel 1 zal dus eerst naar regel 0 gegaan worden om daar weer te geven dat de CURY nu 0 is, wordt de opdracht gegeven op regel 0 wordt eerst het scherm een regel omhoog gescrolled, dan naar de nu nieuw ontstane regel 0 gegaan om dan weer vast te stellen dat CURY 0 is. Dit laatste is alleen maar een doordenkertje maar het eerste is wel degelijk iets om te onthouden. Het betekent dus dat we zeer beslist niet aan de DAI kunnen opdragen : CURSOR CURX,CURY+2:PRINT "#" ,doen we dit toch kunnen we OFF SCREEN krijgen.

Ook de XMAX en YMAX lijken zulke onschuldige 'functies'. In de eerste plaats valt er mee te delen dat zij het beeldscherm of nauwkeuriger de mode waarn het scherm staat als argument meekrijgen. In werkelijkheid gaat het in een iets andere volgorde maar dat maakt voor de theorie der functies niet uit. Bij het initialiseren van de mode worden de pointers op #94 en #95 gevuld met de correcte waarde van het aantal horizontale punten en NIET de XMAX die 1 kleiner is en wordt #96 gevuld met het aantal horizontale lijnen dus het aantal vertikale punten ofwel YMAX+1. Vragen wij naar XMAX wordt alleen gekeken naar de waarden van de adressen #94 en #95. Door de juiste POKE's kunnen wij dus de 'XMAX' veranderen. Onderstaand programma geeft hier een voorbeeld van.

```
10      REM VERANDERENDE XMAX / F.H. DRUIJFF - 19850416
20      MODE 2
30      X=20
40      DOT X,Y 21:X=X+1:IF X<XMAX GOTO 40
50      POKE #94,PEEK(#94)-2:Y=Y+1:GOTO 30
```

We kunnen trouwens ook omgekeerd spelen en de XMAX vergroten, maar pas op het

aantal punten wordt echt niet groter !!! Maar doe voor de aardigheid eens in MODE 2A een POKE #95,5 en vraag dan nog eens naar XMAX. Alles kan weer eenvoudig hersteld worden door een nieuwe MODE X opdracht.

Voor GETC is vanzelfsprekend de ingeslagen toets het argument en het resultaat de ASCII-waarde die daar bij hoort. De DAI neemt hier nogal een uitzonderingspositie in. Alle andere mij bekende machines geven niet de ASCII-code maar een karakterstring van een karakter nl die van de ingeslagen toets. De instructie voor andere machines is dan vaak INKEY$. Beide methodes bieden voordelen, als ik echter moet kiezen, kies ik voor de DAI-aanpak. De INKEY$ is handiger voor beginners en de GETC voor gevorderden. Na deze opmerking vind iedereen GETC handiger ? Maar laten we een en ander toelichten met een voorbeeld :

Wachten op indrukken van de spatiebalk

andere computer

```
100  H$=INKEY$
110  IF H$<>" " GOTO 100
```

DAI

```
100 H=GETC
110 IF H<>32 GOTO 100
      of
110 IF H<>ASC(" ") GOTO 100
      of
110 IF CHR$(H)<>" " GOTO 100
```

Ondanks de mogelijkheden om zonder te weten dat bij de spatiebalk ASCII-code 32 hoort vindt ik de getoonde linker methode toch eleganter. Maar nu een ander voorbeeld waarbij de DAI juist beter voor de dag komt.

Testen of er wel een cijfer is ingetoetst, iets anders dan een cijfer wordt hierbij domweg niet geaccepteerd.

andere computer

```
100 H$=INKEY$
110 IF ASC(H$)<48 GOTO 100
120 IF ASC(H$)>57 GOTO 100
      of
100 H=ASC(INKEY$)
en nu verder als de DAI
```

DAI

```
100 H=GETC
110 IF H<48 GOTO 100
120 IF H>57 GOTO 100
```

Bij het eerste voorbeeld kunnen we bij de DAI ook zetten 100 H$=CHR$(GETC) en dan verder als de linkerkant. Beide laatst genoemde mogelijkheden werken wel maar laten duidelijk zien dat de computer niet geeft wat wij verlangen, daar we direct beginnen te converteren.
Tot besluit van deze keer de FRE die bij de MS en MSX computers wel degelijk een argument verlangt. Bij DAI geeft hij de vrije ruimte aan tussen 'end basic' en 'bottom screen'. Ook hier kunnen we nog even aardig mee spelen door bv in een net aangezette machine even POKE #2A6,0 in te tikken. ?FRE zal U dan aangenaam verrassen maar op de duur toch tegenvallen. Een gemis in DAI-basic vind ik het ontbreken van een mogelijkheid om te weten hoeveel ruimte er nog in de heap beschikbaar is. Volgende keer verder maar eerst iets rechtzetten.

Erratum : De vorige keer beweerde ik dat er geen tijdsduurverschil is tussen FOR....NEXT en FOR....NEXT I. Dit is onjuist. Bij al mijn testen had ik zelfs de body's apart getest en daar geen verschil ontdekt en zo kwam de vergissing. Vrijwel gelijktijdig publiceerde MSX-mozaik een artikel van mij waarin juist op dat tijdsverschil werd gewezen. Afijn het gaat zonder de I zo'n 25 % sneller.

Frank H. Druijff



# SORT ALGORITHMES

Fast sort algorithmes in DBASIC.

Sorting of a string-type array is often needed in a lot of application programs. However most trivial sort algorithmes run very slow, and a quicksort algorithem is hard to program in BASIC because it is recursive.
In DBASIC a recursive quicksort procedure is very easy to program. The following listing shows a short demonstration program 'QUICKSORT 1' that can create a string-type array with random strings and sort it with the QUICKSORT procedure.

```
DBASIC V2.2
PAGE     1  QUICKSORT 1

    10       TITLE "QUICKSORT 1"
    60       PROCEDURE TIMEON :
          1    DOKE #1BE,#FFFF:
             END PROC
    70       FUNCTION TIME(DUMMY)=(#FFFF-DEEK(#1BE))*20
   100       PROCEDURE QUICKSORT N,M ARR A$
   110    1   LOCAL G$,R,S
   120    1   IF N<M THEN
          2    G$=A$((N+M)/2)
   130    2    R=N:S=M
   140    2    WHILE R<S DO
          3     WHILE A$(R)<G$ DO
          4      R=R+1:
          3     %END
   150    3     WHILE A$(S)>G$ DO
          4      S=S-1:
          3     %END
   160    3     IF R<=S THEN
          4      H$=A$(R):A$(R)=A$(S):A$(S)=H$:R=R+1:S=
          4      S-1:
          3     END IF
   170    2    %END
   180    2    QUICKSORT N,S,A$()
   190    2    QUICKSORT R,M,A$()
   200    1   END IF :
             END PROC
  1000       CLEAR 10000
  1010       INPUT ""+CHR$(8C)+"DIMENSION ";K:DIM
             AR$(K)
  1020       ON BREAK GOTO *TEST1:
             FOR I=0 TO K:
          1   FOR J=1 TO 6:
          2    AR$(I)=AR$(I)+CHR$(%41+RND(26)):
          1   NEXT:
             %NEXT
  1030       ON BREAK GOTO *CNT:PRINT CHR$(%C);"START
             SORT":TIMEON
  1040       QUICKSORT 0,K,AR$()
  1050       I=TIME(DUMMY):PRINT "SORTED IN ";I;" ms":
             PRINT
 10100       END
 60000      *TEST1
             PRINT I
 60010      *CNT
             CONTINUE
```

The procedure TIMEON and the function TIME allow a correct timing for controlling the sorting speed. As you can see in table 1 the quicksort procedure is much faster then SORT (selection sort) or BUBBLESORT when the array contains a lot of elements. Another sorting algorithmes doing not so bad is 'SHELLSORT'. This procedure is based on BUBBLESORT but it uses a variable step for comparing array elements.

```
DBASIC V2.2
PAGE     1  BUBBLESORT

   300       PROCEDURE SORT  ARR A$
   310    1   LOCAL I,DNE,DM:DM=DIM(A$,1)-1
   320    1   REPEAT
          2    DNE=1
   330    2    FOR I=0 TO DM
   340    3     IF A$(I+1)<A$(I) THEN
          4      H$=A$(I):A$(I)=A$(I+1):A$(I+1)=H$:DNE=
          4      0:
          3     END IF
   350    2    NEXT:
          1   UNTIL DNE=1:
             END PROC
```

```
DBASIC V2.2
PAGE     1  SORT

   300       PROCEDURE SORT  ARR A$
   310    1   LOCAL R,S
   320    1   FOR R=0 TO DIM(A$,1)-1
   330    2    FOR S=R+1 TO DIM(A$,1)
   340    3     IF A$(S)<A$(R) THEN
          4      H$=A$(R):A$(R)=A$(S):A$(S)=H$:
          3     END IF
   350    2    NEXT:
          1   NEXT:
             END PROC
```

```
DBASIC V2.2
PAGE     1  SHELLSORT

   300       PROCEDURE SORT  ARR A$
   310    1   LOCAL L,I,DNE,DM:DM=DIM(A$,1):L=DM
   320    1   WHILE L>1 DO
          2    L=L/2
   330    2    REPEAT
          3     DNE=1
   335    3     FOR I=0 TO DM-L
   340    4      IF A$(I+L)<A$(I) THEN
          5       H$=A$(I):A$(I)=A$(I+L):A$(I+L)=H$:
          5       DNE=0:
          4      END IF
   350    3     NEXT:
          2    UNTIL DNE=1:
          1   WEND:
             END PROC
```

array dimension

| | | 20 | 50 | 100 | 200 | 500 |
|---|---|---|---|---|---|---|
| | bubble | 5320 | 25060 | 119660 | ? | ? |
| m | sort | 3220 | 17960 | 67760 | ? | ? |
| e | | | | | | |
| t | shell | 2320 | 10500 | 27960 | 99020 | ? |
| h | | | | | | |
| o | quick 1 | 2680 | 7240 | 18900 | 47000 | 100320 |
| d | | | | | | |
| | quick | 2120 | 6500 | 14600 | 32820 | 98940 |

table 1. Time in ms for different sorting methods and various array dimensions.

As you can see in table 1, for large arrays the sorting time of the quicksort procedure (quick 1) will grow much

faster than linear with the array size. This is because requests for heap space (string variable assignments) will take a lot of time then.

All the 'swapping' of string variables will also waste a lot of valuable string space. The following quicksort procedure does not have these disadvantages.

```
 10      TITLE "QUICKSORT"
 50      PROCEDURE SWAPPTR I,J:
     1    LOCAL H:H=DEEK(I):DOKE I,DEEK(J):DOKE J,
     1    H:
         END PROC
 60      PROCEDURE TIMEON :
     1    DOKE #1BE,#FFFF:
         END PROC
 70      FUNCTION TIME(DUMMY)=(#FFFF-DEEK(#1BE))*20
100      PROCEDURE QUICKSORT N,M
110  1    LOCAL G$,R,S
120  1    IF N<M THEN
     2     G$=VAR$(N+(M-N)/4*2)
130  2     R=N:S=M
140  2     WHILE R<S DO
     3      WHILE VAR$(R)<G$ DO
     4       R=R+2:
     3      WEND
150  3      WHILE VAR$(S)>G$ DO
     4       S=S-2:
     3      WEND
160  3      IF R<=S THEN
     4       SWAPPTR R,S:R=R+2:S=S-2:
     3      END IF
170  2     WEND
180  2     QUICKSORT N,S
190  2     QUICKSORT R,M
200  1    END IF :
         END PROC
1000     CLEAR 10000
1010     INPUT ""+CHR$(#C)+"DIMENSION ";K:DIM
         AR$(K)
1020     ON BREAK GOTO "TESTI:
         FOR I=0 TO K:
     1    FOR J=1 TO 6:
     2     AR$(I)=AR$(I)+CHR$(#41+RND(26)):
     1    NEXT:
         NEXT
1030     ON BREAK GOTO "CNT:PRINT CHR$(#C);"START
         SORT":TIMEON
1040     QUICKSORT VARPTR(AR$(0)),VARPTR(AR$(K))
1050     I=TIME(DUMMY):PRINT "SORTED IN ";I;" ms":
         PRINT
10100    END
60000   "TESTI
         PRINT I
60010   "CNT
         CONTINUE
```

Instead of swapping the string variables, only the pointers to the string variables are swapped. This is possible because DBASIC knows special functions to handle (string) pointers. This quicksort procedure does not waste a single byte of heap space and it runs a lot faster for large arrays. A third advantage of this procedure is that the arrays to sort are not limited to 1 dimension. A 3 dimensional array AR$(10,10,2) can be sorted by :

```
QUICKSORT VARPTR(AR$(0,0,0)),VARPTR(AR$(10,10,2))
```

the smallest element will be in AR$(0,0,0), the next in AR$(0,0,1), AR$(0,0,2), AR$(0,1,0)...etc...



Corrections for DBASIC extensions.

1. XREF extension

As mentioned by some DBASIC users, there is a bug in the XREF extension : The cross reference will abort with a 'UNDEFNED FUNCTION' error if arrays are dimensioned with a variable dimension (i.e. DIM A$(K)). A second bug I detected recently, overwrites numeric output when a line is splitted (because it is to long).

Because DBASIC extensions are relocatable, it is not possible to 'patch' the extension by 'poking' in it. To solve the problem, you will have to adapt the XREF V2.2 SPL assembly language source on the DBASIC V2.2 cassette and recreate the XREF extension.

The changes to be made are listed in listing 1. Besides changes to the 'xref' macro, one tiny changement has to be made in the 'relocw' macro (see listing 2). The 'relocw' macro is a small program that automaticaly makes the extension relocatable and writes it in a $-type file. If you ever change one of the other extensions, please change also the 'relocw' macro as shown in listing 2.

To recreate the extension :

load the SPL macro assembler
load the XREF V2.2 source (R command)
make all the changes (see listing 1 & 2)
save the new source (WXREF V2.2)
assemble the source (A command)
position the DBASIC V2.2 cassette at the $XREF file
go in utilities (UT command)
run 'relocw' (63800 command : 3800 is ORG of 'relocw')
type space to write the exension file.

Note :

*If there is a hardware formfeed on your printer it is* better to set the printer parameter FF to TRUE (in XREF V2.2 change FF SET FALSE in FF SET TRUE).
The other printer parameters can be adapted if you use forms of different length and/or width.

2. KEY extension

The KEY extension sometimes suspends execution due to a forgotten 'di' instruction (disable interrupt). The solution is very easy : insert a 'di' at TSTKEY in the KEY V2.2 source file as shown below.

```
...
176  TSTKEY  DI
177          POP H
178          XTHL
...
```

Also change the 'relocw' macro (listing 2). The generation of the corrected KEY extension is analog to the generation of XREF.

# listing 1

```
106  SCMD    SET     2B19H
107  UNMSYM  SET     1B6AH
108  LSTREP  SET     2B44H
109  LVAR    SET     2A4AH
110  ;
111  ;---rom call's---

876  BRST06  XRA A                   ;reset definition flag
877          STA     SYDFLG+1H
878  BRST07  DCX H                   ;else keep definition flag
879          DCX H
880          INX B
881          INX B
882          RET
883  ;


893  ;
894  ;---test symbol's definition---
895  ;
896  ;exit   sydflg=0ffh (init 1st)  def fn
897  ;                               def proc
898  ;                               function
899  ;                               procedure
900  ;                               label
901  ;       sydflg=0    (no  init)  else
902  ;
903  TSTSDF  CPI     TKDIM
904          JZ      PDIM
905  TSTSD1  LXI H   SYDFLG+1H
906          MVI M   0FFH
907          XCHG
908          LXI H   TKNSDF
909          CALL    SCNT81
910          JZ      SCMD
911          XCHG
912          MVI M   0H
913          JMP     SCMD
914  ;
915  ;---process dim---
916  ;
917  PDIM    MVI E   ','
918          LXI H   PARR            ;special processing arrays
919          JMP     LSTREP
920  ;
921  PARR    MVI A   0FFH            ;force symbol definition
922          STA     SYDFLG+1H
923          JMP     LVAR
924  ;
925  ;---definition tokens table---



1312 ;
1313 ;---print integer rigth justified---
1314 ;
1315 ;entry  integer in macc
1316 ;       a=number of positions
1317 ;
```

```
1318  PINTRJ  PUSH PSW
1319          CALL    SVDECB          ;save decbuf
1320          CALL    0DB5FH          ;fixed point output buf
1321          POP PSW
1322          LHLD    0C033H          ;get length of output
1323          SUB M
1324          INR A
1325  PINTR2  DCR A                   ;justifie rigth
1326          JM      PINTR5
1327          space   CALL
1328          JMP     PINTR2
1329  PINTR5  MOV A,M
1330          DCR A
1331          INX H
1332          INX H
1333          CALL    PSTRM           ;print string
1334  ;
1335  ;---retrieve decbuf---
1336  ;
1337          PUSH D
1338          LHLD    0C033H
1339          LXI D   SVAREA
1340          LDAX D
1341          MOV M,A
1342          CALL    0D172H
1343          POP D
1344          RET
1345  ;
1346  ;---save decbuf---
1347  ;
1348  SVDECB  LHLD    0C033H
1349          MOV A,M
1350          CPI     0AH
1351          RNC
1352          PUSH D
1353          XCHG
1354          LXI H   SVAREA
1355          MOV M,A
1356          CALL    0D172H
1357          POP D
1358          RET
1359  ;
1360  SVAREA  DB      0H,0H,0H,0H,0H,0H
1361          DB      0H,0H,0H,0H,0H
1362  ;
1363          ENDIF
1364          MEND
```

# 2

```
1466          LXI D   B2-B1
1467          CALL    0DE14H          ;substract assemble-offset
1468          POP H
1469          POP B
1470          POP D
1471          RET
1472  ;
1473  DUMBLK  DW      0H
1474          MEND
1475  ;
```






L'inteface DAI-Joystick présentée dans un précedent numéro 40 sucité une question qui trouve sa réponse ci-dessous.

Les Joysticks n'ont aucun rapport (Du point de vue de la conception autant que de celui de l'utilisation) avec les classiques paddles. Et ce independamment de l'ordinateur utilisé. Aussi il n'est pas envisageable de REMPLACER purement et simplement un paddle par un joystick. Chacun ayant une fonction différente.

En conclusion il faut noter que des programmes précedemment réalisé pour l'utilisation d'un paddle ne peuvent 'tourner' avec un joystick, sans y ajouter une modification logiciel.

Une coquille s'est glissée lors de la composition du texte de MEMO-DAI.

Son prix réel est de 95 FF sur cassette DCR et 70 FF sur cassette AUDIO. De plus son auteur nous a récemment fait remarquer que cette version ne tourne pas sur les basic V1.0. Il en va de meme pour le programme ZLARG.

only for BASIC V1.1!

The price of MEMODAI, as announced in a previous issue is not right. Price on DCR is 675 Bf , audio is 500 Bf.

BAD "DCR" NEWS

The tremendous rice of the price of DCR-cassettes forces us to apply new prices for DCR-versions. From now on, extra price for DCR will be 250 Bf.

GESTRUCTUREERD PROGRAMMEREN DEEL II

Het lang verwachte deel II van de handleiding door Bruno Van Rompaey is nu beschikbaar. Prijs : 1100 Bf.

SOON AVAILABLE :

TOOLKIT 7 containing : Variable precision, Graphics scroll, Zoom, MSX-extensions ...

JOB PLANNING : a program, creating your critical path of jobs that need to be done.

GAMES COLLECTION 15 : Zeeslag, Cubit and others

The SPL macroassembler documentation is now available in English. J.P. Berckmans and L.Huygh did the job . The documentation is free with any order (send your old documentation), the price is 350 Bf if sold seperately.

Still available in (very) small quantity:
FIRMWARE MANUAL : 1250 Bf
SERVICE MANUAL : 1350 Bf
DAI SCHEMATICS : 850 Bf
BEST OF DAInamic 80/81 : 500 Bf

Peter Noij

Zilstraat 27
6114 KE Susteren
tel.: 04499 - 3557

Dieteren, 17 maart 1985

mijn referentie 85/13
Onderwerp *bijgevoegde subroutine*
Bijlagen subroutine als alternatief voor het "INPUT"-statement.

L.S.

Als grootste bezwaar van het "INPUT"-statement zie ik de onbeheersbaarheid van het ingelezene. Daarnaast ervaar ik als groot bezwaar van het standaard basic het ontbreken van een mogelijkheid tot formatteren van uitvoer. Daarom zoek ik naar een alternatieve subroutine. Een eerste versie Hiervan is bijgevoegd. Ik hoop dat ook andere lezers dit probleem onderkennen en naar alternatieven zoeken. Mogelijk dat deze subroutine een hulpmiddel daarbij kan zijn danwel een eerste aanzet tot discussie. Voor op- en/of aanmerkingen vindt u bij mij een open oor.

```
50000 - 50600  inleessubroutine
               input  :  AARD$     aard in te lezen variabele (I/R/A)
                         DFLT$     defaultwaarde (wordt gebruikt ingeval
                                   enkel <RETURN> is ingebracht)
                         LENGTE%   maximale lengte in te lezen string/getal
               output :  X$        ingeleze string
                         X%        ingelezen mantisse/integer
                         X!        ingelezen real
               click  :  CL%       0 (aan) of 1 (uit)
                         GOED%     frequentie indien ingevoerde correct
                         FOUT%     frequentie indien ingevoerde niet correct
               overige gebruikte hulpvariabelen:
                         PUNT%  TEKEN%  LENGTH%  M%  Y%  EX%  I%
               50000 - 50300  selecteren correcte karakters
               50300 - 50350 correcte inlezing: printen
               50350 - 50400  backspce(s)
               50400 - 50500  inlezen en kontroleren defaultwaarde
               50500 - 50700  RETURN (einde subroutine)

     50700 - 50800  getal in 6 bytes
               50700 - 50750  getal vertalen in 6 bytes
               input  :  X%        integer/mantisse
               output :  Y$        6 bytes   1    teken
                                           2/5   getal (9 cijfers)
                                            6    exponent
               50750 - 50800  getal berekenen uit 6 bytes
               input  :  Y$        6 bytes
               output :  Y en Y!   getal berekend Y$ .

     50800 - 50900  herleiden fortrannotatie (vlgs. free format)
               input  :  X$        integer/mantisse
                         Y$        exponent (laatste byte)
               output :  Z$        fortrannotatie
               overige gebruikte variabelen:
                         Z%  Y%  EX$  EX%

     50900 - 51000  getal in een string zetten
               input  :  AARD$     I(nteger) of R(eal)
                         DEC       aantal gewenste decimalen (<0 mag)
                         X         integer/mantisse
               output :  X$        getal in fortrannotatie
               overige gebruikte variabelen:
                         X!  EX%  HULP$  HULPIE$  I%
```



# INPUT

```
100   REM ------------------------------------------------------------
110   REM subroutines       :  - inbrengen van een getal/string
120   REM .                    - vertalen getal in 6 bytes (ASCII-waarden)
130   REM .                    - vertalen ASCII-waarden in getal
140   REM .                    - fortran-notatie (free format)
150   REM .                    - getal in string
190   REM ------------------------------------------------------------
200   REM 21 februari 1985  :  versie 0
210   REM pnoij
290   REM ------------------------------------------------------------
400   REM AARD$             :  geeft karakter (A/I/R) weer van de input
410   REM GOED & FOUT & FR  :  frequentie click
420   REM CLICK             :  click uit (0) cq. aan (1)
430   REM DFLT$             :  defaultwaarde
440   REM LENGTE            :  maximale lengte in te lezen string
450   REM LENGTH            :  aantal ingelezen posities
460   REM TEKEN             :  positief(0) of negatief (1)
470   REM EX                :  exponent
500   REM X                 :  ingelezen integer
510   REM X!                :  ingelezen real
520   REM X$                :  ingelezen string (cq. mantisse)
530   REM Y                 :  mantisse uit 6 bytes
540   REM Y$                :  getal in 6 bytes
550   REM Z                 :  mantisse
560   REM Z$                :  getal in fortran-notatie
590   REM ------------------------------------------------------------
1000  PRINT CHR$(12);"subroutine :  inlezen van getallen":COLORT 9 14 9 9:CLEAR 1000
1100  REM ------------------------------------------------------------
1110  GOED=5000:FOUT=500:CL=1:REM *** init click ***
1190  REM ------------------------------------------------------------
1200  CURSOR 10,21:PRINT "aard";:AARD$="A":DFLT$="I":LENGTE=1:GOSUB 50000
1210  IF X$<>"A" AND X$<>"I" AND X$<>"R" GOTO 1200
1300  CURSOR 10,19:DFLT$="123.456":AARD$=X$
1310  IF AARD$="A" THEN LENGTE=30
1320  IF AARD$="I" THEN LENGTE=9
1330  IF AARD$="R" THEN LENGTE=10
1340  GOSUB 50000
1400  CURSOR 10,17:PRINT "X$ ingelezen string   :  ";X$:IF AARD$="A" GOTO 1800
1410  CURSOR 10,16:PRINT "X  ingelezen integer  : ";X:REM IF AARD$="I" GOTO 1500
1420  CURSOR 10,15:PRINT "X! ingelezen real     : ";X!
1500  GOSUB 50700
1510  CURSOR 10,13:PRINT "Y$ getal in 6 bytes   :  ";LEFT$(Y$,1);:FOR I=1 TO 4:PRINT ASC(MID$(Y$
,I,1));:NEXT:PRINT ASC(RIGHT$(Y$,1))-128
1520  CURSOR 10,12:PRINT "Y  mantisse uit Y$    : ";Y
1530  CURSOR 10,11:PRINT "Y! getal uit Y$       : ";Y!
1600  GOSUB 50800
1610  CURSOR 10,9:PRINT "Z$ getal (fortran)     :  ";Z$
1620  CURSOR 10,8:PRINT "Z  mantisse            : ";Z
1700  Y=X:Y!=X!:HULP$=AARD$:CURSOR 10,6:PRINT "hoeveel decimalen   ";:AARD$="I":LENGTE=1:GOSUB 5
0000:DEC=X
1710  X=Y:X!=Y!:AARD$=HULP$:LENGTE=10:GOSUB 50900
1720  IF AARD$="I" THEN CURSOR 10,5:PRINT "X  integer            : ";X
1730  IF AARD$="R" THEN CURSOR 10,5:PRINT "X! real               : ";X!
1740  CURSOR 10,4:PRINT "X$ resultaat           :  ";X$
1800  CURSOR 10,1:PRINT "nog een sessie (j/n)   : ";
1810  M=GETC
1820  IF M=ASC("j") OR M=ASC("J") GOTO 1000
1830  IF M=ASC("n") OR M=ASC("N") GOTO 1900
1890  GOTO 1810
1900  PRINT CHR$(12);
1990  END
50000 X$="":X=0:PUNT=0:LENGTH=0:TEKEN=0:PRINT "  : ";
50100 M=GETC:IF M=0 GOTO 50100
50110 IF M=13 GOTO 50500
```

```
50120 FR=FOUT
50130 IF M=ASC("+") AND TEKEN=0 GOTO 50320
50140 IF M=ASC("-") AND TEKEN=1 GOTO 50320
50150 IF M=ASC("+") OR M=ASC("-") THEN IF LENGTH>0 THEN X=M:M=22:GOSUB 50350:M=X:X=0
50160 IF M=ASC("+") THEN TEKEN=0:PRINT X$;:FR=GOED:GOTO 50320
50170 IF M=ASC("-") THEN TEKEN=1:PRINT "-";X$;:FR=GOED:GOTO 50320
50180 IF M=8 OR M=22 AND LENGTH=0 GOTO 50320
50190 IF M=8 OR M=22 THEN FR=GOED:GOSUB 50350:GOTO 50320
50200 IF LENGTH=LENGTE GOTO 50320
50210 IF AARD$="A" THEN FR=GOED:GOTO 50300
50220 IF M=ASC(".") THEN IF PUNT=1 OR AARD$="I" GOTO 50320
50230 FR=GOED
50240 IF M=ASC(".") THEN PUNT=1:GOTO 50300
50280 IF M<ASC("0") OR M>ASC("9") THEN FR=FOUT:GOTO 50320
50290 REM ------------------------------------------------------------
50300 X$=X$+CHR$(M):PRINT CHR$(M);:LENGTH=LENGTH+1
50320 IF CL=1 THEN SOUND 0 0 15 0 FREQ(FR):WAIT TIME 2:SOUND OFF
50330 GOTO 50100
50340 REM ------------------------------------------------------------
50350 X=1:IF M=22 THEN X=LENGTH+TEKEN
50360 FOR I=1 TO X:PRINT CHR$(8);:IF X=0 THEN LENGTH=LENGTH-1:Y$=RIGHT$(X$,1):X$=LEFT$(X$,LENGTH
):IF Y$="." THEN PUNT=0
50370 NEXT:IF LENGTH=0 THEN TEKEN=0
50380 RETURN
50390 REM ------------------------------------------------------------
50400 X$=DFLT$:IF AARD$="A" GOTO 50450
50410 IF LEFT$(X$,1)="-" THEN TEKEN=1:X$=RIGHT$(X$,LEN(X$)-1)
50420 FOR I=0 TO LEN(X$)-1:IF MID$(X$,I,1)="." THEN PUNT=PUNT+1
50430 NEXT:IF (AARD$="I" AND PUNT=1) OR PUNT>1 THEN FR=FOUT:X$=""
50440 IF AARD$="R" AND PUNT=0 THEN X$=X$+".0"
50450 IF FR=GOED THEN PRINT X$;:LENGTH=LEN(X$)
50480 RETURN
50490 REM ------------------------------------------------------------
50500 FR=GOED:IF LENGTH=0 THEN GOSUB 50400
50510 IF CL=1 THEN SOUND 0 0 15 0 FREQ(FR):WAIT TIME 10:SOUND OFF
50520 IF FR=FOUT GOTO 50100
50530 IF AARD$="A" THEN RETURN
50540 IF AARD$="I" THEN X$=X$+".0"
50550 LENGTH=LENGTH+2:EX=0:FOR I=0 TO LEN(X$)-1
50560 IF MID$(X$,I,1)="." THEN X$=LEFT$(X$,I)+RIGHT$(X$,LEN(X$)-I-1):EX=EX+I:I=I-1
50570 NEXT
50590 REM ------------------------------------------------------------
50600 IF LEN(X$)>1 AND RIGHT$(X$,1)="0" THEN X$=LEFT$(X$,LEN(X$)-1):GOTO 50600
50610 IF LEN(X$)>1 AND LEFT$(X$,1)="0" THEN X$=RIGHT$(X$,LEN(X$)-1):EX=EX-1:GOTO 50610
50620 IF LEN(X$)>9 THEN PRINT "  truncated to 9 positions";:X$=LEFT$(X$,9)
50630 X=VAL(X$):IF LEN(X$)>4 THEN X=INT(VAL(LEFT$(X$,LEN(X$)-4))+0.5):Y=INT(VAL(RIGHT$(X$,4))+0.
5):X=X*10000+Y
50640 X!=X*10^(EX-LEN(X$)):IF TEKEN=1 THEN X=(-X):X!=(-X!)
50650 RETURN
50690 REM ------------------------------------------------------------
50700 Y$="+":IF TEKEN=1 THEN Y$="-":X=(-X)
50710 Y$=Y$+CHR$(X MOD 256)+CHR$(X/256 MOD 256)+CHR$(X/65536 MOD 256)+CHR$(X/16777216 MOD 256)+C
HR$(EX+128)
50730 REM RETURN
50740 REM - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
50750 Y=ASC(MID$(Y$,1,1))+ASC(MID$(Y$,2,1))*256+ASC(MID$(Y$,3,1))*65536+ASC(MID$(Y$,4,1))*167772
16
50760 Y!=Y*10^(ASC(RIGHT$(Y$,1))-128-LEN(X$))
50770 IF LEFT$(Y$,1)="-" THEN Y=(-Y):Y!=-(Y!)
50780 RETURN
50790 REM ------------------------------------------------------------
50800 Z=VAL(X$):IF LEN(X$)>4 THEN Z=INT(VAL(LEFT$(X$,LEN(X$)-4))+0.5):Y=INT(VAL(RIGHT$(X$,4))+0.
5):Z=Z*10000+Y
50810 IF LEN(X$)<9 THEN X$=X$+"0":GOTO 50810
50820 Z$=LEFT$(Y$,1)+"."+X$:EX=ASC(RIGHT$(Y$,1))-128+0.5:EX$=MID$(STR$(EX),1,LEN(STR$(EX))-3)
50830 IF LEN(EX$)<3 THEN EX$="0"+EX$:GOTO 50830
50840 IF EX>=0 THEN Z$=Z$+"+"
50850 IF EX<0 THEN Z$=Z$+"-":EX=ABS(EX)
50860 Z$=Z$+EX$:IF LEFT$(Z$,1)="-" THEN Z=(-Z)
50880 RETURN
50890 REM ------------------------------------------------------------
50900 X$="":IF AARD$="I" THEN X!=X
50910 IF DEC<0 THEN X!=X!/10:DEC=DEC+1:GOTO 50910
50920 EX=0:HULP$=STR$(X!):FOR I=1 TO LEN(HULP$)-1:HULPIE$=MID$(HULP$,I,1):IF HULPIE$<>"." THEN X
$=X$+HULPIE$
50930 IF HULPIE$="." THEN EX=LEN(HULP$)-LEN(X$)-2
50940 NEXT:HULP$=LEFT$(X$,LEN(X$)-EX):X$=RIGHT$(X$,EX)
50950 IF LEN(X$)<DEC THEN X$=X$+"0":GOTO 50950
50960 IF LEN(X$)>DEC THEN X$=LEFT$(X$,DEC)
50970 IF LEN(HULP$)>LENGTE-DEC-1 THEN HULP$="":X$="":FOR I=3 TO LENGTE:X$=X$+"*":NEXT
50980 IF X!<0 THEN HULP$="-"+HULP$
50990 X$=SPC(LENGTE-LEN(HULP$)-LEN(X$)-1)+HULP$+"."+X$:RETURN
50999 REM ------------------------------------------------------------
```

# IF THEN

Geachte DAInamic redaktie,

Ik wil U en vooral de lezers wijzen op een fout in de manual betreffende het IF-statement. Zoals bekend kan deze statement voorkomen in een aantal varianten:

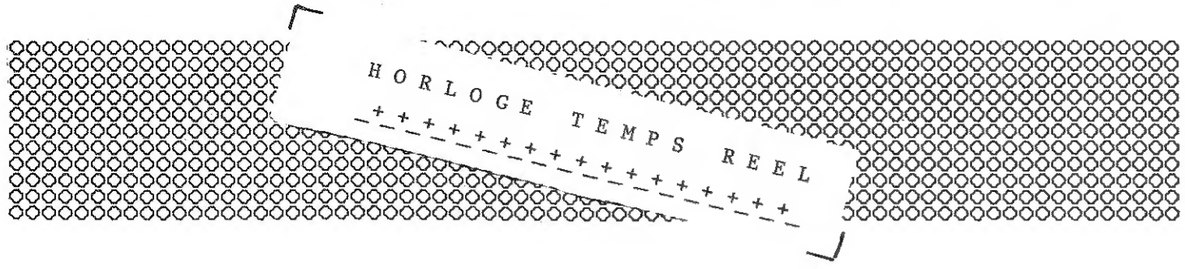
1. IF [expression] THEN [linenumber] : [statement]
2. IF [expression] GOTO [linenumber] : [statement]
3. IF [expression] THEN GOTO [linenumber] : [statement]

De varianten 1 en 2 zijn equivalent, als [expression] de waarde 'true' heeft, wordt gesprongen naar [linenumber], zoniet dan wordt [statement] uitgevoerd.
In geval 3 echter, wordt [statement] nooit uitgevoerd, ongeacht de waarde van [expression]. Er wordt slechts gesprongen naar [linenumber] indien [expression] de waarde 'true' heeft.
Dit alles geldt voor BASIC V1.1, met V1.0 heb ik geen ervaring.
In de manual staat echter op blz. 66, 6.2.6.5, example (ii), dat bij variant 2 [statement] nooit wordt uitgevoerd. Dit is dus onjuist.

Hierbij ook een programma om op een andere manier dan met vele sinus en cosinus berekeningen cirkels en ellipsen te tekenen. De remarks in de subroutine geven de mogelijkheden aan. Regels 10 t/m 80 zijn een demo'tje. Een voordeel van deze routine is dat de hoogste definitie gebruikt wordt waar het het hardst nodig is, namelijk in de lijnstukken met de grootste kromming. De routine is daardoor efficienter dan wanneer een konstante hoekverdraaiing wordt gebruikt, vooral bij lange ellipsen.

Voor de machinetaalprogrammeurs een aantal snelle 8080-routines overgenomen uit 'Electronics'.

Marc Boon
Dirklangendwstr. 2
NL-2611 JA  Delft
tel. 015-126576




Désirant adapter un programme de simulateur de vol aux instruments, j'avais besoin de créer une horloge donnant heures, minutes et secondes écoulées, cela quel que soit mon temps de réaction aux diverses manoeuvres et l'enchaînement de sous-programmes utilisés. Ne pratiquant pas l'assembleur, je devais la réaliser en BASIC.

J'explorais la memory map du n° 11 et étudiais le timer externe utilisé par l'instruction WAIT TIME et WAIT MEM, situé aux adresses # 1BE et # 1BF.

Première constatation : ces deux adresses sont chargées à # FF à la mise sous tension.

Deuxième constatation : #1BF est décrémenté de 1 pour un décrement de 256 de #1BE.

Troisième constatation : il ne faut pas utiliser WAIT dans le programme puisqu'il provoque un réensemencement de ces adresses.

La suite est simple.
D'abord affecter aux variables H, M et S leurs valeurs de départ (ex : H = 8, M = 35, S = Ø donnant une heure de début à 8h 35 mn Ø s).
Puis, dès le démarrage de la partie active du programme, par exemple en ligne 100 :

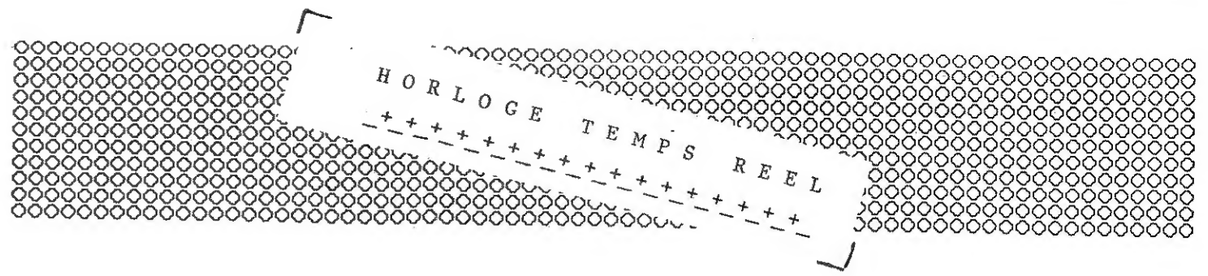
```
100  POKE # 1BF, # FF : POKE # 1BE, # FF
```

Dans le corps du programme, dès que cela est possible sans géner son déroulement et surtout sans excéder un écart de 20 mn entre deux accès, placez :

```
XXX  GOSUB 10000 (ou autre adresse)
```

.../...






```
10000  T1 =PEEK (#1BF) : T2 = PEEK (# 1BE)
10010  DTE = (255-T1) * 256 + 255 - T2
10020  SE = INT (DTE * 2E - 2)
10030  IF SE  60 THEN DS = SE : DM = Ø : GOTO 10050
10040  DM = INT (SE/6Ø) : DS = SE - DM * 60
10050  M = M + DM : S = S + DS
10060  IF S> 59 THEN S = S - 60 : M = M + 1
10070  IF M>59 THEN M = M - 60 : H = H + 1
10080  IF H>23 THEN H = Ø
10090 CURSOR 10,10 : PRINT H, M, S : REM formattez votre impression à votre
       convenance
10100  POKE # 1BF, # FF : POKE # 1BE, 248
10110  RETURN
```

Remarque : en ligne 10100 la valeur à "poker" en # 1BE dépend de la longueur du sous-programme commençant en 10000.

Cette solution n'est sans doute pas optimale mais aidera les DAIistes qui ne sont pas versés en language machine.

Question : Je rencontre de gros problèmes de précision dans l'adaptation d'un programme de calcul d'éphémérides astronomiques (les 6 chiffres significatifs insuffisants).

Quelqu'un a-t-il mis au point une routine (même assembleur) permettant de passer à 8 ou 9 chiffres significatifs ?

Est-ce possible ?

Contactez-moi.

J.P. CHEREAU
11, avenue Saint-Exúpéry
92160 ANTONY

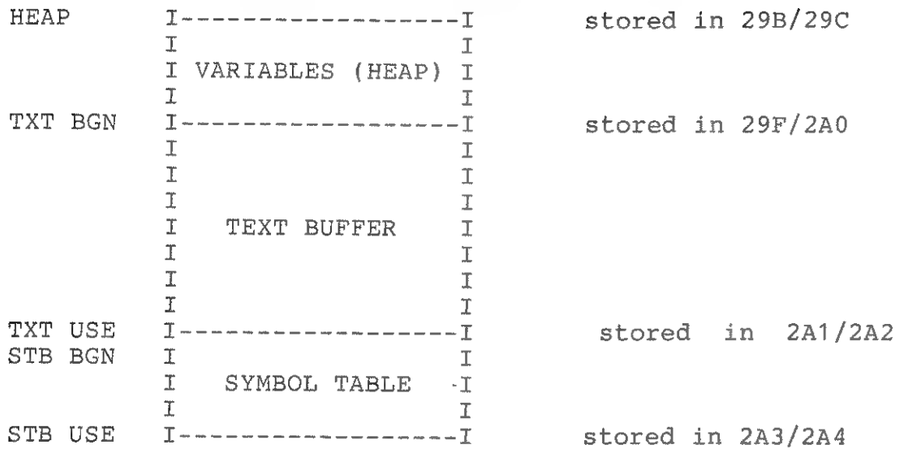
# BASIC & ML SAVE

please read ç as ] (utility CURSOR)

## SAVING OF A PROGRAM IN BASIC

### UNDER HIS MACHINE LANGUAGE FORM

THE 'SAVE' , BASIC COMMAND




The variables that the program needs are stored in the restricted area whose address is stored in 29B/29C and the size in 29D/29E.

A 'SAVE' in BASIC :
1. initialise the HEAP and the SYMBOL TABLE.
2. saves the TEXT BUFFER.
3. saves the SYMBOL TABLE.

A 'LOAD' in BASIC :
1. initialise the HEAP and the SYMBOL TABLE.
2. loads the TEXT BUFFER and stores the corresponding addresses in 29F/2A0 and 2A1/2A2.
3. loads the SYMBOL TABLE and stores the corresponding addresses in 2A3/2A4.

At the beginning,the HEAP pointer (29B/29C) is initialised at 2EC and the size is 100 (TEXTBUFFER: 2EC+100=3EC).

So,we have to save,if the BASIC program is alone,the TEXTBUFFER,the symbol table and store their addresses in 29F-2A4.
Hence,it's easier to save the program from 29F to STB USE

```
*LOAD              loading of the file
*UT
çD2A3 2A4
çAA BB
çW29F BBAA         name of the program
```

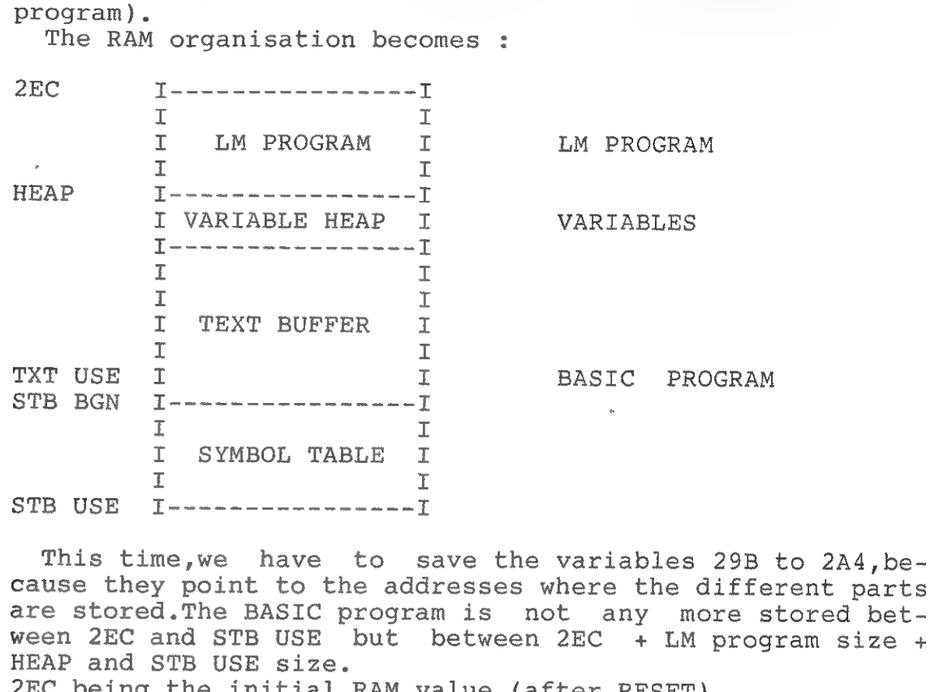
The BASIC program is then saved under his LM form.



## SAVING OF A BASIC PROGRAM +

### A LM PROGRAM UNDER A LM FORM

It becomes attractive to put together a BASIC program, which we can save under a LM form,with a LM program to load the entirety under a LM form.(f.i. FGT (300-900) + BASIC program).
The RAM organisation becomes :




This time,we have to save the variables 29B to 2A4,because they point to the addresses where the different parts are stored.The BASIC program is not any more stored between 2EC and STB USE but between 2EC + LM program size + HEAP and STB USE size.
2EC being the initial RAM value (after RESET).
** Don't forget to readjust the pointers 29F-2A4 while loading the MLP and the BASIC.

1st method: use a BOOTSTRAP LOADER (DBL).
---------- You just have to depress the BREAK key when the loading of the BASIC program is started.

2nd method: readjust the pointers.
---------- Let's take the example FGT (300-900).
(notice that FGT takes place from 29B to 900,what updates the pointers 29F to 2A4 during the loading).

so :
---

```
*UT
çR              loading of the MLP

çS29B
çEC-01
ç02-09          start of the HEAP -ç after the LM pro-
                gram (=901).
ç00-
ç01-
çEC-01
ç03-0A          start of the TEXT BUFFER that's to say
                start of the HEAP+100=901+100=A01.
çB
*LOAD           loading of the BASIC
```

** You have stored,in the RAM,the MLP+BASIC under the form above described.Now,you have to save the program from 29B to STB USE.
STB USE has the addresses 2A3/2A4.

```
so :   *UT
---    çD2A3 2A4
       çXX YY
       çB
       *              (make the cassette ready)
       *UT
       ç29B YYXX
       çB
       *
```

You have now a BASIC+LM program,saved under a LM form.
To come out:
```
*UT
çR
çB
*RUN
```

## AUTOSTART OF A BASIC AND LM PROGRAM

With a READ or a LOAD,when the command is performed,a CALL 2D4 is executed to close the file and stop the motor. 2D4 is used to go back to D445 (audio recorder) or to F31F (DCR).(JMP D445 or JMP F31F).

## AUTOSTART IN MACHINE LANGUAGE

We have to store the machine language from address 2D4. If the addresses of the MLP are,f.i.,5000 to 6000,you have to save the RAM from 2D4 to 6000,what's not very interesting,because we don't use the RAM from 2EC to 5000 !
Hence,you have to have a file starting around the 300. To get the autostart,the start of your MLP must close the file,that's to say execute a RCLOSE (CD 45D5 for the audio recorders or CD 1FF3 for the DCR).

EXAMPLE: MLP from 300 to 1000
-------- running address: 300

We can notice that the space between 2EC and 300 is free. If it wasn't,we had to write the following little program from 1001 to 1006.Load the program and then :

so,we have in 2EC :

```
2EC  CALL   D445 (or F31F)   CD45D4   RCLOSE
     JMP    300              C30003   run in 300
```

then change 2D5 :

```
çS2D4        or:
ç45-EC       ç1F-EC
çD4-02       çF3-02
```



Now,save from hex2D5 to hex1000

```
çW2D5 1000 program name
çB
*
```

Put the cassette back,then UT and R.

(If your LM program is placed,f.i.,from hex5000 to hex6000,move it from hex300 to hex1300.Your little program,in hex2EC,will have to execute a MODE before starting it.)

## AUTOSTART OF A BASIC PROGRAM

The method is the same,but instead of starting the program with a CD,start it with a RUN.
Example :
Type in the following BASIC program :

```
10 MODE 0
20 PRINT CHR$(12)
30 COLORT 0 3 0 0
40 PRINT "IT'S WORKING"
```

Save it on a cassette (or DCR) and rewind it at the beginning.Shut the DAI down.(to reset all the variables).We'll write our little LM program from hex2EC to hex300.

```
çS2EC 3E 00 CD 06 D8 31 00 F9 01 FD 02 CD 1F F3 C3 92 C8 87
                              -----    -----
                                         ^
                    for audio cassettes : 45 D4
```

Update the pointers in hex29B :

```
çS29B EC-00 02-03 00-00 01-01 EC-00 03-04
çB
*NEW
*LOAD
*UT
çD2A3 2A4
ç43 04
çS2D5 1F-EC F3-02
(CAS:       45-EC D4-02)
çW29B 443 program name
```

Shut the DAI down and reload the program.

Now,if you have a MLP+BASIC program,you have to put both together and save them under an LM form.Add the LM program if you have room between hex2EC and hex300;otherwise,put this program after your LM file (LM+BASIC).Then,you have to change the address in hex2D4.It's not any more hex2EC,but the address where you've stored your little LM program.
On the other hand,in the program,there is an address (underlined) FD02 (=hex2FD).It corresponds to hex2EC + hex11 (points to hex87).
Don't forget to rectify.

Henri-Pierre LEGRY
F-59 DOUAI

---

Translated by L.J.

# ASSEMBLER BY PRACTICE

by Raymond VANLATHEM (Club CAROLODAI)

## Preamble

We don't want here to give you a theoretical account on the Assembler Language.They exist enough good books in bookshops and many magazines who approach the question more or less precisely.We wish to propose to our readers practical applications,first easy,so that they can practise on their DAI and better understand the meaning of the different instructions and their involvement.So,we hope to contribute to turn that over your mind,a way to enrich your brain.

If the utility of some routines,particularly at the beginning,is not clear for you,don't forget that they are didactic routines who connect themselves gradualy before to become real operational routines.But,any clever guy will be able to adapt these routines and to insert them in his programs.

## Introduction

A recall is maybe not unnecessary.
A microprocessor (a 8080A for the DAI) can execute instructions only in binary code,what we call "machine code" (represented by the human being under the form of hexadecimal code).As it would be hard and long to program in Machine Language (particularly for long programs),programming languages have been developped.One of these is the Assembler Language,which exists in different developments (DNA,SPL,.) Nevertheless,in each of those,each basic instruction is represented with the same mnemonic (condensed word which reminds the nature of the instruction).Each instruction in Assembler is converted in a machine code by the assembling program.

The instruction set of the 8080A has in Machine Language 244 codes on 256 possible combinations of hexadecimal codes Each of these codes forms the code of operation of a machine instruction.They are as many machine instructions as dif ferent codes of operation.Each instruction of the 8080A has one,two or three bytes.The first byte is always the code of operation of the instruction.The next possible byte(s) are a data byte or a memory address (8 or 16 bits).Among the 244 machine instructions,200 have one byte,18 two bytes and 26 three bytes.Moreover,they correspond to 78 instructions in Assembler,composed of a different mnemonic for each instruction and,contingently,of an operand with one or two elements.

**ù = hex ç = >**



It's good to remember that the 8080A has 6 main registers of one byte B,C,D,E,H and L.These can be grouped by pairs (B+C,D+E,H+L) where the first term (B,D or H) indicates the pair and contains the most significant byte,the second register containing the least significant byte.The accumulator A (one byte),through which pass the arithmetic and logical operations,and the memory location M (two bytes),whose address is pointed by the register pair H+L,are also considered as registers.We have also a stack pointer SP and a counter for the developping of the program PC (Program Counter),each with two bytes.

There is still a special register with one byte whose 5 bits only are used :they are the status or condition indicators (Flags),with a flip-flop type.They are called,in decreasing order of the used bits : the sign S,the zero Z, the auxiliary carry AC,the parity P and the carry C.The 3 unused bits of the byte are the sixth,the fourth (always 0) and the second one (always 1).Let's note that the accumulator A and the flags register can form a pair called PSW (Program Status Word),in which the accumulator is the MSB (most significant byte).

## Application No 1

**Purpose : clear memory from a start address in memory.**

DIDACTIC LISTING

| | | | |
|---|---|---|---|
| 300 | 97 | SUB A | Clear the accumulator,by cutting off the contents of A from the accumulator. |
| 301 | 00 | NOP | Has no effect;allows to reserve a memory location to intercalate an instruction without disturbing the general diagram of the routine. |
| 302 | 21 0B 03 | LXI H,:30B | Loads the memory start address in H+L (ù0B in L and ù03 in H). |
| 305 | 77 | MOV M,A | Moves the contents of A in M (whose address is in H+L). |
| 306 | 23 | INX H | Increments the contents of H+L by one (gives the next memory location address). |
| 307 | C3 05 03 | JMP :305 | Unconditional jump to memory address ù305 to execute the next move from A to M. |
| 30A | C9 | RET | Allows to go out from the routine;same as the RETURN instruction in Basic. |

COMMENTS
--------

The execution of the routine is made in UTILITY (type UT in Basic),then çG300 (with çZ3 before or not).The memory locations are cleared from ù30B (the screen is also cleared), but the DAI loops:it's normal,because nothing stops the execution of the routine.Press RESET,then *UT and display the contents of the memory from ù30A to ùB34F (page by page) by means of DISPLAY,to verify the resetting of the memory.Let's note that the memory locations ù3EA and ù3EB contain respectively ù7F and ùFF,because of the RESET.

You can also type çZ3 (necessary here) followed by çL300 300 30A,what will allow you to look at the successive states of the different registers at the end of the execution of each instruction.In this case,the executuion time is very long,but you can stop the run of the program at any time by pressing the BREAK key,and then look how far is the resetting of the memories,with DISPLAY.

Let's note that we can't misuse the NOP (00) instruction because,though this one is inoperative,she uses four clock-cycles.

EXPERIMENTS
-----------

(1) Under line 300,you can replace SUB A (97) by XRA A (AF),which gives the same result : exclusive OR between the contents of A and the accumulator.

(2) You can also replace in lines 300 & 301 SUB A and NOP (00) by MVI A,:0 (3E 00),where the data 0 (one byte) is loaded into the accumulator.Of course,the 00 (machine code for the NOP instruction) can be used for the instruction MVI A (3E).Though this last solution is more obvious than the two others,it has the disadvantage to occupy two bytes instead of one and to have 7 machine cycles against 4 for the two other solutions.

N.B.:The indicators' status are different following the resetting type of A.With SUB A,we get F (Flag)=ù56,with XRA A,F=ù46 and with MVI A,:0,F=ù02.

Application No 2
----------------

Purpose : Load data in two registers and look at their
-------  contents.

DIDACTIC LISTING
----------------

```
300 ! 21 00 13 ! LXI H,:1300 Loads the memory address in HL
303 ! 01 01 0A ! LXI B,:A01  Loads the data ùA01 (two bytes
    !          !             in B+C (ù01 in C and 0A in B).
306 ! 70       ! MOV M,B     Moves the contents of B into
    !          !             M (here at the address ù1300).
307 ! 23       ! INX H       Increments the contents of H+L
    !          !             by one.
308 ! 71       ! MOV M,C     Moves the contents of C into M
    !          !             (here  at  the address ù1301).
309 ! 00       ! NOP         No operation.
30A ! C9       ! RET         Output of the routine.
```



COMMENTS
--------

In UT,type çG300,then display the data with çD1300 1301, which gives you 0A 01.Otherwise,type çZ3 + çL300 300 30A and then çD1300 1301 to look at their contents.

EXPERIMENTS
-----------

(1) By replacing,under line 30A,the instruction RET (C9) with HLT (76),who stops the program,you will loose controll on the DAI.So,never try HLT,otherwise be very carefull !

(2) Under line 309,if you replace NOP by PUSH B (C5),you load the contents of B+C on the stack of the DAI,that's to say that you save the data on the stack.With çG300 you have no problem and çD1300 1301 will display 0A 01.

But if you type çZ3 + L300 300 30A,the DAI will be lost after the LOOK command.If you type BREAK (at time,but not too early),you can see that the value of P(next instruction 's address to be executed) who stands at the address of the last executed instruction (I=ù30A),contents the data ù0A01.Now,we had to see the value ùEA04,who gives the return address of the ROM and who stands now at the bottom of the stack (ùF8FE & ùF8FF),whose address is given by S=ùF8FE

Application No 3
----------------

Purpose : Load on the stack the contents of two regis-
-------   ters for examination.

DIDACTIC LISTING
----------------

```
300 ! 31 02 13 ! LXI SP,:1302 Loads the start address of
    !          !               the stack and modifies the
    !          !               stack pointer.Uses a stack in
    !          !               RAM instead of the initial
    !          !               stack of the DAI.Can be use-
    !          !               full in specific applicatioms
303 ! 01 01 0A ! LXI B,:A01   Loads the data ùA01 in B+C.
306 ! 00       ! NOP          No operation.Used as spare
307 ! 00       ! NOP          for easy insertions.
308 ! 00       ! NOP
309 ! C5       ! PUSH B       Saves the contents of B+C in
    !          !              the stack.
30A ! 00       ! NOP
30B ! C9       ! RET          Allows to go out from the
    !          !              routine.
```

COMMENTS
--------

Let's extend the experiment (2) of the previous application and see what happens with another stack.By typing çZ3 + L300 300 30B,we get P=ù0A01 and the DAI is lost without BREAK,as with the initial stack.

EXPERIMENTS
---

(1) By replacing the three NOP's from line 306 to 308 by LXI H,:EA04 (21 04 EA), ²loads the address ùEA04 in H+Lè and the NOP of line 30A by PUSH H (E5) ²loads the contents of H+L on the stackè,the DAI is no more lost,even with LOOK By using çD1300 1301, we get 01 0A (reverse order than in application no 2).We can notice here the fundamental importance of the address ùEA04,who has to have a correct place in the stack.

(2) Let's type çZ3 + L300 300 30B (and skip the instruction LXI SP,:1302 under line 300,so we take back the initial stack).To display the data,we have to do çDF8FC F8FD. We will find 01 0A (the address ùF8FC is given by S in the last line of LOOK,that's to say the one concerning I=ù30B).

Translated by L.J.

---

# SEARCH ROUTINE

MLP PROGRAMME - SEARCH ROM OR RAM FOR A GIVEN ADDRESS

This programme will search the DAI's memory for references to an address that you specify. When the address of a DAI routine is typed in, the programme will find and display all references to that routine. During a ROM search sections C, D and all four E banks are examined. For a RAM search it is necessary to input, when asked, the start and end addresses of the MLP or the part of RAM to be examined. Hex addresses are expected and should be entered in alphanumeric characters without # or H. eg the GETC routine is at address #D6BE; type it in as D6BE.
RST + Data calls can also be found, but as they are not in the Low byte, High byte order of addresses, the two bytes must be reversed when entering; eg RST 4 Data 0F is E70F but must be typed in as 0FE7.
The only error report is the ? as used for errors in the DAI's Utility.

Bill Read

```
     0000                    TITL      'SEARCH ROM OR RAM FOR A GIVEN ADDRESS'
0000           ;                                    Bill Read     Nov 1984
0000           ;DAI routines called:
0000  @=D6BE GETC     EQU       0D6BEH          ;Get char from keybd
0000  @=DD60 OUTC     EQU       0DD60H          ;Print character
0000  @=DAD4 PMSG     EQU       0DAD4H          ;Print message
0000  @=DE14 COMPDE   EQU       0DE14H          ;Compare DE & HL
0000  @=EADE GETADR   EQU       0EADEH          ;(3EADE)Get address from
0000           ;                                ; keybd and put on stack.
0000           ;
0000                  ORG       300H
0300  214603 START    LXI H     START$          ;Display title and
0303  CDD4DA          CALL      PMSG            ; ask ROM or RAM
0306  CDBED6 KEYBD1   CALL      GETC            ;Get reply
0309  D8              RC                        ;Abort if BREAK
030A  FE31            CPI       31H             ;Key 1 ?
030C  CA2A03          JZ        ROM
030F  FE32            CPI       32H             ;Key 2 ?
0311  CA1703          JZ        RAM
0314  C30603          JMP       KEYBD1          ;If any other key
0317  21E403 RAM      LXI H     RAM$            ;Ask addresses of
031A  CDD4DA          CALL      PMSG            ; area of search.
031D  0E02            MVI C     2H              ;2 addresses wanted
031F  CDDEEA          CALL      GETADR          ;Addresses from
0322           ;                                ; keybd on to stack
0322  3EA0            MVI A     0A0H            ;Marker for RAM
0324  320505          STA       BANK
0327  C33703          JMP       ASKADR
032A  3E20    ROM     MVI A     20H             ;Marker for
032C  320505          STA       BANK            ; unswitched ROM
032F  2100C0          LXI H     0C000H          ;Start of ROM
0332  11FFDF          LXI D     0DFFFH          ;End of unswitched ROM
0335  E5              PUSH H                    ;Save both addresses
0336  D5              PUSH D                    ; on stack.
0337  21BE03 ASKADR   LXI H     STRT2$          ;Ask address to
033A  CDD4DA          CALL      PMSG            ; be sought.
033D  0E01            MVI C     1H              ;1 address wanted
033F  CDDEEA          CALL      GETADR          ;Get address from
0342  C1              POP B                     ; keybd to BC
0343  C36004          JMP       BEGIN1
0346  0C0D0D START$   DB        0CH,0DH,0DH
0349  202020          DB        '   SEARCH ROM OR RAM FOR REFERENCES'
036C  20544F          DB        ' TO A GIVEN ADDRESS'
037F  0D0D            DB        0DH,0DH
0381  202020          DB        '         Key 1   To search ROM'
039E  0D              DB        0DH
039F  202020          DB        '         Key 2   To search RAM'
03BC  0D00            DB        0DH,0H
03BE  0D      STRT2$  DB        0DH
03BF  202020          DB        '      What address is to be sought ? '
03E3  00              DB        0H
03E4  0D      RAM$    DB        0DH
03E5  202020          DB        '      What area of the RAM is to be searched ?'
0412  0D              DB        0DH
0413  285479          DB        '(Type start address SPACE end address RETURN)
0441  00              DB        0H
0442  0D      RUN$    DB        0DH
0443  202020          DB                  Search in progress '
```

```
045E  00              DB      0H
045F  C9              RET
0460          ;
0460          ;Entry: Address to be sought in BC
0460          ;       End address of search area on top of stack
0460          ;       Start address of search area top-1 of stack
0460  214204  BEGIN1  LXI H   RUN$
0463  CDD4DA          CALL    PMSG
0466  D1              POP D                   ;End of search area
0467  E1              POP H                   ;Start of search area
0468  23      GO1     INX H
0469  CD14DE          CALL    COMPDE          ;End reached?
046C  CAAE04          JZ      FINIS1          ;Yes
046F  7E              MOV A,M
0470  B9              CMP C                   ;Lo byte wanted addr?
0471  C26804          JNZ     GO1             ;No
0474  23      CHECK   INX H                   ;Yes. Get Hi byte
0475  7E              MOV A,M
0476  B8              CMP B                   ;Hi byte wanted addr?
0477  2B              DCX H
0478  C26804          JNZ     GO1             ;No
047B  C5              PUSH B                  ;Yes. Save the wanted
047C  E5              PUSH H                  ; address and the place
047D  3E0D            MVI A   0DH             ; where it was found.
047F  CD60DD          CALL    OUTC            ;Output CarRet and
0482  3A0505          LDA     BANK            ; print space or & bank
0485  CD60DD          CALL    OUTC            ; number, then the
0488  CD9004          CALL    PRTADR          ; location found.
048B  E1              POP H
048C  C1              POP B
048D  C36804          JMP     GO1
0490          ;
0490          ;Convert address in HL to ASCII and print it:
0490  7C      PRTADR  MOV A,H                 ;1st byte to A
0491  CD9504          CALL    PA1             ;
0494  7D              MOV A,L                 ;2nd byte to A
0495  F5      PA1     PUSH PSW                ;Save HEX
0496  07              RLC                     ;Move high nibble
0497  07              RLC                     ; to low
0498  07              RLC
0499  07              RLC
049A  CD9E04          CALL    PA2
049D  F1              POP PSW
049E  E60F    PA2     ANI     0FH             ;Use only low nibble
04A0  CDA604          CALL    PA3             ;Make it ASCII
04A3  C360DD          JMP     OUTC            ;Print it
04A6  C630    PA3     ADI     30H             ;Make ASCII
04A8  FE3A            CPI     3AH             ;0 to 9 ?
04AA  D8              RC                      ;Yes
04AB  C607            ADI     7H              ;For letters A to F
04AD  C9              RET
04AE          ;
04AE  3A0505  FINIS1  LDA     BANK            ;ROM Bank flag
04B1  FE20            CPI     20H             ;C and D done ?
04B3  CAC804          JZ      BNKE0           ;Yes.Start E bank 0
04B6  FE30            CPI     30H             ; 0E bank done ?
04B8  CAE104          JZ      BNKE1
04BB  FE31            CPI     31H             ; 1E bank done ?
```





```
04BD  CAE904          JZ      BNKE2
04C0  FE32            CPI     32H             ; 2E bank done
04C2  CAF104          JZ      BNKE3           ;Start E bank 3
04C5  C30605          JMP     FINIS2          ;RAM, or all ROM done
04C8          ;Bank switching. The data byte stored in BANK has a dual
04C8          ;       purpose: it is a marker to indicate when the RAM
04C8          ;       or one of the ROM sets is in use, and it is the
04C8          ;       prefix printed with each address found in search.
04C8  3A4000  BNKE0   LDA     40H             ;POROM
04CB  E63F            ANI     3FH             ;For Bank 0
04CD  324000          STA     40H
04D0  3206FD          STA     0FD06H          ;PORO
04D3  3E30            MVI A   30H             ;Marker for E bank 0
04D5  320505  AGAIN   STA     BANK            ; and save it
04D8  11FFEF          LXI D   0EFFFH          ;Reload HL and DE with
04DB  2100E0          LXI H   0E000H          ; Ebank start & finish
04DE  C36804          JMP     GO1             ;Search next bank.
04E1  CDF904  BNKE1   CALL    BNKSW
04E4  3E31            MVI A   31H             ;Marker for E bank 1
04E6  C3D504          JMP     AGAIN
04E9  CDF904  BNKE2   CALL    BNKSW
04EC  3E32            MVI A   32H             ;Marker for E bank 2
04EE  C3D504          JMP     AGAIN
04F1  CDF904  BNKE3   CALL    BNKSW
04F4  3E33            MVI A   33H             ;Marker for E bank 3
04F6  C3D504          JMP     AGAIN
04F9  3A4000  BNKSW   LDA     40H             ;Bank switching routine
04FC  C640            ADI     40H             ;Increment bits 6,7 to
04FE  324000          STA     40H             ; move to the next bank.
0501  3206FD          STA     0FD06H
0504  C9              RET
0505  00      BANK    DB      0H              ;Flag for E bank or ROM
0506  E5      FINIS2  PUSH H
0507  212C05          LXI H   END$
050A  CDD4DA          CALL    PMSG
050D  E1              POP H
050E  CDBED6  KEYBD2  CALL    GETC
0511  D8              RC                      ;To UTility if BREAK
0512  FE52            CPI     'R'
0514  CA0003          JZ      START           ;Restart
0517  FE41            CPI     'A'
0519  C20E05          JNZ     KEYBD2          ;If any other key
051C  3A0505          LDA     BANK            ;Where was the search
051F  FEA0            CPI     0A0H            ;RAM ?
0521  CA1703          JZ      RAM             ;Yes.
0524  FE33            CPI     33H             ;ROM ?
0526  CA2A03          JZ      ROM             ;Yes.
0529  C30003          JMP     START
052C  202020  END$    DB      '   Search completed.'
0540  0D              DB      0DH
0541  202020          DB      '       Press A to search for another address.
056E  0D              DB      0DH
056F  202020          DB      '       Press R to restart the programme.'
0597  0D              DB      0DH
0598  202020          DB      '       Press BREAK to end the programme.'
05C0  0D00            DB      0DH,0H
05C2                  END
```

# EXTENDED BASIC

EXTENDED BASIC is a modular library written in D-BASIC,a new language for the DAI PC.
The first module,GENLIB.D,contains thirteen new comands or instructions, which can be used in a direct mode or in a program. They are re-united in two parts.

1) SCREEN
   ******

CLS  clears the screen without wasting paper for the printer.

SPEED n controls the output speed on the screen.
n < 15 slows down the display.
n = 60 returns to the normal speed:it's necessary for the printer

NORMAL after COLORT A B C D requires the color A for the background
B for the text

INVERSE after COLORT A B C D requires the color C for the background
(=reverse video) D for the text.

WINDOW n,m defines a screen-window from the line n up to the line m.
n = 1 bottom of the screen
m = 24 top of the screen

2) MISCELLANEOUS
   *************

PRTON delivers the printer

PRTOFF blocks the printer

REPT within a program which uses GETC, allows autorepeat-keys.

SWAP A$,B$ swaps two strings

INSTR(I,X$,Y$) I is an integer constant,
searches,from the Ith position, the character Y in a string X. If no success, then INSTR equals zero.
e.g. A$="COMPUTER":B$="M"
PRINT INSTR(2,A$,B$) gives the value 3
PRINT INSTR(4,A$,B$) gives the value 0

STRING$ (I,X$) X$ is an alphanumeric constant !!
generates I-times the string X$
e.g. PRINT STRING$(4,".") gives ....

DFFTP "X","Y"
defines as real variables all variables the first character of which begins from X to Y

DFSTR the same but for strings

Two remarks: take care to keep the alphabetic order between X and Y
these instructions have the same function as the DAI commands IMP but can be used in a short first program which will load the main program.It's a simulation of Microsoft's DEFINT,DEFSTR

e.g. program INIT 10 DFFTP "A","B"
20 DFSTR "C","D"
30 LOAD "MAIN"



program MAIN 100 A=6.5:B=0.44
110 CA="YES":DA="NO

Thanks to P.JANIN for his video inverse subroutine.
The next module,MATLIB.D, will concern the matrix-calculation.

F.LEMOINE
First publication: DAICLIC/1 February 1985
Translation and adaptation for DAInamic March 1985.

---

PAGE 1 GENLIB.D

```
 2      REM ******************************************************
 3      REM EXTENDED BASIC PART 1
 4      REM GENLIB.D  GENERAL LIBRARY
 5      REM FERNAND LEMOINE RUE DU COLLEGE 34 6071 CHATELET
 6      REM CLUB CAROLODAI
 7      REM MARCH 1985
 8      REM ******************************************************
10      TITLE "GENLIB.D"
12      ON ERROR GOTO "TRAP
15      "SCREEN
        REM CLS,SPEED,INVERSE,NORMAL,WINDOW COMMANDS
20      PROCEDURE CLS :
1        PRINT CHR$(27);CHR$(68);CHR$(12);:
        END PROC
30      PROCEDURE SPEED D:
1        POKE #FF05,D:
        END PROC
40      B=#235
45      WHILE A<>#C9 DO
1        READ A
50      1    POKE B,A:B=B+1
52      WEND
54      DOKE #70,#23D:DOKE #6C,#235
56      DATA #F5,#CD,#45,2,#F1,#C3,#FD,#C6
58      DATA #F5,#CD,#45,2,#F1,#C3,#A9,#D9
60      DATA #3E,0,#32,#76,0,#C9
65      PROCEDURE INVERSE :
1        POKE #246,#FF:
        END PROC
70      PROCEDURE NORMAL :
1        POKE #246,0:
        END PROC
95      PROCEDURE WINDOW B,H
86      1    IF B<0 OR H >3 OR B=H THEN
        2     ERROR 60:
        1    END IF
88      1    LOCAL CALCB,CALCH,CALC1
90      1    CALCH=#BFEF-((23-H)*#86):CALCB=#B35F+(8*B)
92      1    CALC1=CALCB-16
94      1    DOKE #8A,CALCH:DOKE #5C,CALCB
96      1    DOKE #8E,CALC1
98      END PROC
100     "MISCELLANEOUS
        REM PRTON,PRTOFF,REPT,SWAP,INSTR,STRING$,DFFTP,DFSTR COMMANDS
110     PROCEDURE PRTON :
1        POKE #131,0:
        END PROC
120     PROCEDURE PRTOFF :
1        POKE #131,1:
        END PROC
```

```
130         PROCEDURE REPT :
        1    FOR A=#261 TO #268:
        2     POKE A,0:
        1    NEXT:
            END PROC
140         PROCEDURE SWAP  VAR X$,Y$:
        1    LOCAL HELP$
145     1    HELP$=X$:X$=Y$:Y$=HELP$:
            END PROC
150         FUNCTION INSTR(H,VAR CAR$,CHAIN$)
153     1    LG=LEN(CHAIN$):SW=0:A=H-1


DBASIC V2.2
PAGE        2   GENLIB.D


160     1    IF H<=LG THEN
        2     REPEAT
        3      D=0
165     3      IF MID$(CHAIN$,A,1)=CAR$ THEN
        4       D=A+1:SW=1
168     3      ELSE
        4       A=A+1:
        3      END IF
170     2     UNTIL SW=1 OR A=LG
172     1    END IF
175     1    FN = D:
            END FN
180         FUNCTION STRING$(H,CAR$)
182     1    LOCAL CALC$:B=1
184     1    REPEAT
        2     CALC$=CALC$+CAR$:B=B+1
186     1    UNTIL B>H
188     1    FN = CALC$:
            END FN
190         PROCEDURE DFFTP X$,Y$:
        1    XY=ASC(X$):YX=ASC(Y$)
192     1    FOR H=XY TO YX:
        2     POKE (H+#234),0:
        1    NEXT:
            END PROC
194         PROCEDURE DFSTR X$,Y$:
        1    XY=ASC(X$):YX=ASC(Y$)
196     1    FOR H=XY TO YX:
        2     POKE (H+#234),#20:
        1    NEXT:
            END PROC
198        "TRAP
            IF ERR=60 THEN
        1    PRINT "INCORRECT VALUE",:RESUME "SCREEN:
            END IF
```



# X-BUS CARD

CHOPPINET Eric
27 rue Louis Pasteur
91310 LEUVILLE SUR ORGE
FRANCE
Tel: (6) 084.60.87

Leuville le 4 mars 1985

---

Cher ami DAIiste,

Je vous remercie de l'intérêt que vous portez à la carte EXTENSION-BUS parue dans DAInamic n° 26.

Le délai de livraison de la carte nue réalisée en trous métallisés est de 6 semaines après réception de commande.

Pour l'installation de la carte, il faudra vous armer:
- d'un multimètre,
- si possible d'un oscilloscope,
- d'un grattoir pour couper 3 pistes sur la carte mère du DAI,
- d'un bon fer à souder (panne très fine !!!),
- et de fil à wrapper.

Pour la mise en service de la carte, il est prudent de le faire par étapes

*** Souder toutes les petits composants
(résistances,condensateurs,diodes,transistors.)
(connecteur X-BUS, supports circuits intégres.)

*** Apporter les modifications suivantes sur la carte mère:

1. Couper la piste arrivant à la pin 47 du X-BUS
2. Couper les pistes arrivant aux pins 22 et 21 du X-BUS
   et relier ces pistes sans passer par le X-BUS.
3. Relier IC44 pin 9 à la pin 47 du X-BUS (F0XX).
4. Relier IC45 pin 10 à la pin 18 du X-BUS (F9XX).
5. Relier IC45 pin 11 à la pin 22 du X-bus (FAXX).
6. Relier IC107 pin 1 à la pin 21 du X-BUS (reset).

*** Vérifier les modifications à l'ohmmètre.
*** Mettre votre DAI sous tension sans MEMOCOM.
*** Contrôler le + 5V sur chaque support de CI.
*** Introduire dans le bon sens les CI suivants:
- CI 5 74C00 (CMOS ou HCMOS)
- CI 6 74C42 (CMOS ou HCMOS)
- CI 3 votre EPROM DCR.

<<<< COUPER L'ALIMENTATION >>>>

*** Passer en UT. Visualiser les adresses F000-F7FF. Le contenu de la ROM apparait, si ne n'est pas le cas,vous avez mal adressé le décodeur 74C42. (SWITCH A - B - A11 ouverts)
Avec un oscilloscope,vous pouvez visualiser (F000-F7FF)
CI 44 pin 9 (carte mère)
pin 47 du X-BUS
CI 6 pin 12
CI 6 pin 4

*** Brancher votre MEMOCOM, et faire CALLM #F2F2.

Si vous utilisez votre ROM DCR, la compatibilité SOFT est totale.
A ce stade vous ne pouvez plus commuter vos EPROM à l'aide des minirupteur
Faites POKE #296,0. Maintenant modifiez la position des SWITCH, visualisez les adresses F000-F7FF.
Vous pouvez donc changer de banc mémoire mais vous n'avez plus les commandes DCR.

Vous pouvez introduire le CI 4. (INS 8154 N)

Les SWITCH ne servent qu'à l'initialisation du bon banc mémoire lors d'un départ à froid, ensuite le 8154 prend les commandes en placant les pins du port B en sortie basse impédance. (Par convention, A11 de CI 3 = 5V.)
La ROM DCR a été entièrement réécrite avec des commandes supplémentaires:

RBP Restore Basic Pointer
SBP Save Basic Pointer
IMP initialisation imprimante parallele.
( ... )

A l'initialisation le 8154 prend les commandes. A,B,A11 en basse impédance

La nouvelle ROM DCR permet toutes les fonctions:
<LOOK> <LAST> <VER> <CHECK> <LOAD> <SAVE> <CASx> <DCRx>
Il est possible que dans certains programmes on teste la présence de l'eprom DCR par controle d'un byte dans le banc mémoire F000-F7FF. Ce byte risque à présent d'être different.

Concernant le programmateur d'eprom, je tiens à votre disposition le soft correspondant pour des eproms 2732 en langage machine que vous lancerez par un CALLM #7000

** Répondez au menu **
L Lecture en #4000 - #4FFF
V Vérification
T Test de virginité
P Programmation avec données en #4000
F Fin : retour au moniteur

Attention à la tension de programmation (+ 21 V ou + 25 V). Cette tension de programmation doit etre mise au bon moment, (le soft vous previendra) et la suprime dès que le programme vous rend la main. Utiliser une alimentation +30 V; le régulateur LM137 (T1) et le potentiometre (P1) permettent d'obtenir 21 ou 25V. (Un essai sans EPROM est conseillé.)

Pour ce qui concerne l'horloge temps réel,le soft est en développement. Le CI 9 MM58174A est adressé de FAC0 à FAFF. Reportez-vous à la notice du constructeur. Le QUARTZ est de 32,768 kHz (standard horlogerie). L'horloge continue à fonctioner même en cas de coupure secteur, je compte sur vous pour me présenter des applications originales.

Un mot sur l'utilisation de la RAM de 2K.(6116)
Pour écrire dans l'espace mémoire F000-F7FF, il faut faire un POKE #10,#C9 pour éviter un STACK OVERFLOW et prendre garde que le SWITCH 'WR' soit fermé, ensuite faite POKE #10,0.
En ouvrant le SWITCH 'WR', vous protégez le contenu de la RAM en cas de coupure secteur.Cette RAM sert à lancer des programmes que vous voulez vérifier avant de les écrire en ROM et non de stockage des DATA. Pour stocker des variables, utilisez la RAM du INS 8154N.

Tous les SOFT se rapportant a cette carte ont été développés par mon ami Claude PICARD.

Vous trouverez dans les prochains numero de DAI d'autres renseignements sur cette carte.

Amicalement.

Eric CHOPPINET



# TELEX VIA DE RADIO

RTTY is de afkorting van "Radio Tele Type" of met andere woorden: Radioverschrijver. De datatransmissie bij RTTY verkeer geschiedt met diverse codes, waarbij de Baudot-kode wel een van de belangrijkste is. Voor dit soort uitzendingen is de beschreven converter dan ook bedoeld.
Bij RTTY-transmissie worden de enen en de nullen als twee verschillende frequenties uitgezonden (= FSK). Het is de taak van de converter om die twee frequenties weer terug te vertalen naar de oorspronkelijke logische nivo's. Deze techniek wordt algemeen gebruikt door zowel luister-amateurs als gelicencieerde zendamateurs, en dit op de korte-golfbanden ( <30 MHz ) alsook op de VHF-banden ( 144 MHZ ).
De opzet van een RTTY-ontvangst en- of zendketen kan er als volgt uitzien (Fig.1) : korte-golf zend-ontvanger, converter, en telex-machine. Wie geen telex heeft,maar wel een computer met scherm, kan het RTTY-signaal daarmee zichtbaar maken, op de manier zoals in figuur 2 is aangegeven.
Achter de converter is dan een 5 bits serie-naar-8 bits-parallel converter geschakeld (UARTI), waarna onze DAI de Baudot-naar-ASCII omzetting doet, en daarna de informatie op het scherm zet. Dit gebeurt natuurlijk met een computerprogramma, waarbij men moet letten op de volgende punten:

- serieel ingangssignaal, bestaande uit:
  5 databits, 1 startbit, 1 stopbit
- transmissiesneld instelbaar op:
  45 baud, 50 baud, 75 baud, 110 baud

Met de intrede van de computer is sinds kort in sommige landen ook ASCII-transmissie toegelaten, ook wel gbruikt om satellieten van de nodige besturings-programma's te voorzien. Voor amateur-gebruik is een veel gebruikte vorm van ASCII : 7 bits,1 stopbit,110 baud.
In fig.3 is het blokschema van de converter te zien.
De ingang van de converter wordt verbonden met de luidspreker-uitgang van de korte-golf onvanger. De beide tonen voor "mark" en "space" (een en nul) worden eerst versterkt, daarna begrensd en dan naar twee band-filters gevoerd, het mark en het space filter.De uitgangssignalen van de filters worden gelijkgericht en dan opgeteld door een som-versterker, die tevens als begrenzer werkt.Aan de uitgang van de schakeling is het dedekodeerde RTTY-signaal beschikbaar,dat in deze vorm direkt een telex kan sturen.
Het mark-filter is vast ingesteld op een frequentie van 1275 Hz, of naar keuze op 2125 Hz, naargelang de oude of nieuwe normen. Het space-filter kan ingesteld worden op 1445,1530,1700,2125, of tussenliggende frequenties. Afhankelijk van van de instelling van het space-filter bedraagt het frequentie-verschil (shift) tussen mark en space dan 170, 255, 425 of 850 Hz.
Voor een goede ontvangst van de meeste zenders is meestal een frequentie-verschil tussen mark en space van 425 Hz nodig.
In het schema is het mark-filter opgebouwd rond A3. Met P5 wordt de midden-frequentie ingesteld op 1275 Hz of op 2125 Hz. (1275 Hz op ingang zetten en op max. uitgangs-spanning van A3 afregelen ).

Het space-filter is opgebouwd rond A2. Met behulp van de draai-schakelaar kan men de band-frequentie omschakelen.
De trim-potmeters worden daarbij ingesteld op max. uitgangs-spanning van A2 bij respektievelijk 1445, 1530, 1700 en 2125 Hz.
De uitgangen van A2 en A3 kan men aansluiten op de X- en Y-ingang van een oscilloskoop. De converter is optimaal ingesteld als op het scherm van de skoop een ellips-kruis zichtbaar is (Fig. 4).
Met de dubbele schakelaar S1a-b kan men omschakelen tussen "normale" ontvangst en "inverse" ontvangst-mode.
Het schema van de FKS-modem is van belang voor de zend-amateurs, en is een gekende beproefde schakeling. De beide opgewekte tonen worden afgeregld met de 10K trimmers en mogen maximaal 5 herts afwijken van de toegelaten frequenties. De uitgang FSK-out wordt aangesloten op de microfoon-ingang van de zender.
Het uitgangssignaal wordt beperkt met de 10k trimmer. Men kan ook kiezen tussen normaal of inverse werking.

# DE UARTI-INTERFACE

De UARTI-interface is een universeel programmeerbare , serie naar parallel omzetter en terug. Door de schakeling onder te brengen op een klein opsteek-printje , en doordat bij de meesten de rekenchip AMD9511 ontbreekt, kunnen we dankbaar gebruik maken van de lege soket. Geen nood : de rekenchip kan eventueel nog . extra op het printje ondergebracht worden.
De UARTI-interface is in de eerste plaats ontwikkeld voor het omzetten van 5-bits BAUDOT + 1 1/2 stopbit,serieel naar 8-bits ASCII parallel, hetgeen gebruikt wordt bij TELEX-verkeer tussen radiostations of telefoonlijnen.
Er is echter nog heel wat meer mee te doen; als voorbeeld : aansturen van een printer, plotter, voice synthesizer, 2'de RS 232 interface, modem,telex .....enz.
Aangezien de RS 232 interface in de DAI slechts gedeeltelijk in snelheid programmeerbaar is, en helemaal NIET in het aantal bits, moest deze weg ingeslagen worden.
De interface wordt actief wanneer deze in een bepaalde MODE wordt gezet d.m.v. een controlewoord. Dit woord bepaalt het aantal te verwerken bits, het aantal stopbits en de parity-bit.
Om te kunnen werken worden 2 adressen in het geheugen gebruikt : FA00 en FA02, waarin men kan lezen of schrijven hetgeen er zich afspeelt in de interface of wat er moet gebeuren.
Het adres FAxx wordt in de DAI afgetapt en is reeds gedecodeerd aanwezig op pin 11 van ic 45 (74LS155),en wordt door DAI niet gebruikt. Samen met A1 op de math-chip soket kan FA00 en FA02 gedecodeerd worden. RD en WR zijn voor het lezen of schrijven naar de interface.
De snelheid waarmee gewerkt wordt, wordt bepaald door het geven van een sound commando (v.b.: SOUND0 0 15 0 FREQ(BAUD X 16).



De sound generator wordt aangesloten met een klemmetje op de kathode van diode D35 of pin 10 van de 8253.
De in- en uitgang van de interface gaat via een 3-aderig (stereo) snoer naar buiten de DAI en eindigt in een normale 'stereo' bus-stekker.
Het eigenlijke werk wordt gedaan door een UART LSI-ic, afkorting van: Universal Asynchronous Receiver/Transmitter en bestaat in vele merken (v.b.:AY-3-1015D, COM8017, HS 6402) en zijn meestal pin-compatibel, en enkel-5-volts uitvoering.
Om de werking te begrijpen is het nodig de data-sheet van de betreffende ic te bestuderen, hetgeen hier te ver zou voeren binnenin dit artikel.
Het is voldoende enkele gegevens te vermelden om het geheel te laten werken (zie schema).
Pinnen 5 tot 12 en 26 tot 33 zijn verbonden met de DATA-bus. Hierop komt de te verzenden of te ontvangen byte, of ook het controlewoord en de status waarin de UART zich bevindt.
Pinnen 17 en 40 krijgen een frequentie aangeboden die 16 maal de snelheid is waarmee men wil werken (zie v.b. SOUND boven).
Pin 16 leest de status via de data-lijnen van de UART.
Pin 18 leest het te ontvangen karakter van de data-lijnen.
Pin 20 is de serie ingang van de te ontvangen data.
Pin 21 is algemene reset; ligt aan massa.
Pin 23 verstuurt een karakter op de data-bus naar de UART.
Pin 25 is de serie uitgang van het te verzenden karakter.
Pin 34 leest het controlewoord in via de data-bus en zet de UART in een bepaalde mode.

Volgende gegevens kunnen gelezen worden op de data-bus:

D0= Transmitter Buffer Empty;is 1 wanneer nieuw karakter mag verstuurd worden.
D1= Parity Error;is 1 wanneer de ontvangen parity niet overeen komt met de ingestelde parity.
D2= Framing Error;is 1 wanneer het ontvangen karakter geen geldig stop-bit heeft.
D3= Over-Run;is 1 indien de ontvangst-snelheid niet overeen komt met de ingestelde snelheid.
D4= Data Available;gaat naar 1 indien een ontvangen karakter mag gelezen worden van de data-bus.
v.b.: PRINT PEEK(#FA00 IAND 16)= Print status Data Available

Het samenstellen van het controlewoord:

Dit bestaat uit een binair getal, geladen via de data-bus, en heeft tot doel het hoog of laag maken van een aantal lijnen die de UART in een bepaalde mode brengen:
D0= een 1 geeft een even parity, 0 een oneven parity
D1= Nummer Bits 1: NB2 NB1 Bits/Karakter
D2= Nummer Bits 2: 0 0 5
0 1 6
1 0 7
1 1 8
D3= Nummer stopbits; 0= 1stopbit 1= 2stopbits of 1 1/2
D4= No Parity; 1= geen parity 0= wel parity-bit
v.b. tabel:

| D4 | D3 | D2 | D1 | D0 | |
|---|---|---|---|---|---|
| 1 | 1 | 0 | 0 | 1 | = #19 = 5 bits,1 1/2 stopbit |
| 1 | 0 | 1 | 1 | 1 | = #17 = 8 bits,1stp-bit,NP |
| 1 | 0 | 1 | 0 | 1 | = #15 = 7 bits,1stp-bit,NP |

Dus: POKE#FA02,#17 zet de UART in 8 bits,1stopbit,No Parity

Verdere gegevens voor het versturen en ontvangen van data kan men vinden in het programma 'TELEX IN BASIC'.
Een voorbeeld van de 'lay out' vindt U in dit nummer.
Gemakkelijkheidshalve is deze print ook verkrijgbaar, doch alleen als komplete unit, en dit tegen kostprijs van de onderdelen (= 1000 Bfr.).
Ook andere printen voor de ontvangst van RTTY zijn te verkrijgen. Gelieve hiervoor nader contact op te nemen met mij, op nr. 03/7770676.

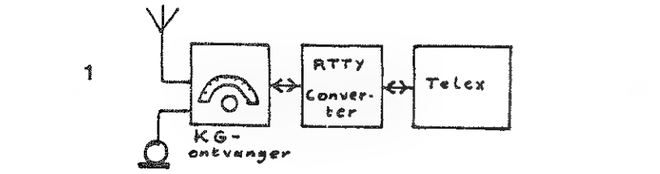
DE DAUW ARMAND
WALLENHOF 93
B-2770 NIEUWKERKEN-WAAS

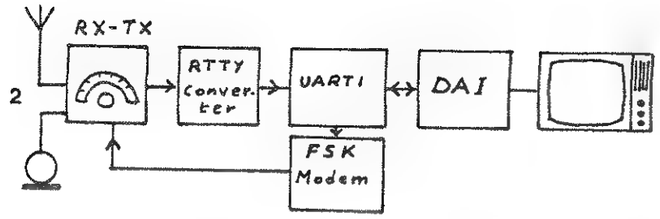
```
1     REM PROGRAM WORKS ONLY WITH UARTI INTERFACE
2     REM * * * *  T E L E X  I N  B A S I C  * * * *
3     MODE 0:PRINT CHR$(12):COLORT 7 0 7 7:GOSUB 1000
4     GOTO 900
5     WAIT MEM #FA02,16:D=PEEK(#FA00)
6     IF D=27.0 THEN P=0.0:GOTO 5
7     IF D=31.0 THEN P=1.0:GOTO 5
8     IF GETC=9 THEN 900
9     PRINT CHR$(B(P,D));:GOTO 5
10    WAIT MEM #FA02,16:PRINT CHR$(PEEK(#FA00));:IF GETC=0 GOTO 10
100   REM * RECEIVE BAUDOT *
111   BD!=45.45:M!=66.0
112   GOSUB 5000
114   GOTO 5
200   REM * RECEIVE ASCII *
900   PRINT "   CONTOL MODE :":PRINT
901   PRINT "   RECEIVE BAUDOT  <B>        SEND BAUDOT  <b>"
902   PRINT "   RECEIVE ASCII   <A>        SEND ASCII   <a>"
903   PRINT "   RECEIVE TOR     <T>        SEND B-TEST  <X>"
904   PRINT "   CHANGE SPEED    <S>        SEND A-TEST  <x>"
909   PRINT :PRINT :INPUT "Select -->";X$
910   IF X$="B" THEN 100
911   IF X$="A" THEN 3000
912   IF X$="S" THEN GOSUB 2000
950   GOTO 100
1000  REM * INITIALYSE *
1001  PRINT "DATA INLEZEN....."
1002  CLEAR 500
1004  DIM B(1.0,31.0)
1010  P=1.0:REM * LETTERS *
1015  SOUND 0 0 5 0 FREQ(45.45*16.0)
1020  POKE #FA02,#19:REM BAUDOT 5-BITS
1070  FOR X=0.0 TO 31.0:READ A:B(0.0,X)=A:NEXT X
1075  FOR X=0.0 TO 31.0:READ A:B(1.0,X)=A:NEXT X
1080  PRINT
1099  RETURN
1100  DATA 0,#33,0,#2D,#20,#27,#38,#37,#0D,#24,#34,#27,#2C
1105  DATA #21,#3A,#28,#35,#2B,#29,#32,#23,#36,#30,#31,#39
1107  DATA #3F,#26,0,#2E,#2F,#3B,#20
1110  DATA 0,#45,#20,#41,#20,#53,#49,#55,#0D,#44,#52,#4A,#4E
1115  DATA #46,#43,#4B,#54,#5A,#4C,#57,#48,#59,#50,#51,#4F
1117  DATA #42,#47,0,#4D,#58,#56,0
```

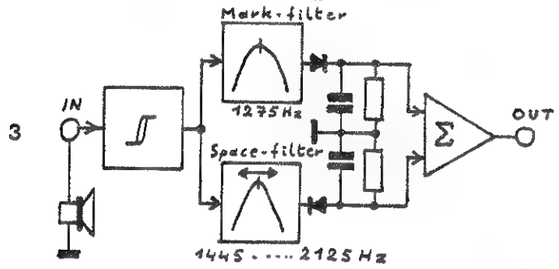


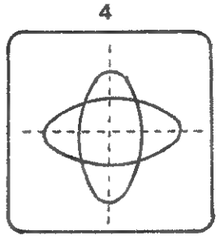
```
2000  REM * CHANGE SPEED *
2010  PRINT CHR$(12):ENVELOPE 0 15
2011  PRINT "TYPE : WPM : RATE  : CHAR/SEC :BIT LENGHT :   STOP  : OP/MIN
2012  PRINT "-------------------------------------------------------------"
2013  PRINT " <1> : 60  : 45 BD :    6     : 22 msec   : 31 msec : 368"
2014  PRINT " <2> : 66  : 50 BD :    6,6   : 20 msec   : 30 msec : 400"
2015  PRINT " <3> : 75  : 57 BD :    7,5   : 18 msec   : 25 msec : 460"
2016  PRINT " <4> : 100 : 75 BD :   10     : 13,5 msec : 19 msec : 600"
2017  PRINT " <5> : 110 :       :          :           :          :    "
2018  PRINT " <6> : 300 :       :          :           :          :    "
2019  PRINT
2020  INPUT "Select speed <NR>";S!:IF S!>6.0 THEN 2020:PRINT
2030  IF S!=1.0 THEN BD!=45.45:GOTO 2050
2031  IF S!=2.0 THEN BD!=50.0:GOTO 2050
2032  IF S!=3.0 THEN BD!=75.0:GOTO 2050
2033  IF S!=4.0 THEN BD!=100.0:GOTO 2050
2034  IF S!=5.0 THEN BD!=110.0:GOTO 2050
2035  IF S!=6.0 THEN BD!=300.0:GOTO 2050
```

```
2050  SOUND 0 0 5 0 FREQ(BD!*16.0)
2100  RETURN
3000  REM * ASCII RECEIVING *
3001  INPUT "WELKE BAUD-SNELHEID (110)";BD
3003  SOUND 0 0 5 0 FREQ(BD*16.0)
3004  POKE #FA02,#17:REM ASCII 8-BIT 1ST-BIT
3005  GOSUB 5000
3010  GOTO 10
5000  COLORT 7 0 7 7:PRINT CHR$(12):POKE #75,95
5002  POKE #8A,#D7:POKE #8B,#BD:POKE #BDD6,#CB
5004  CURSOR 5,22:PRINT "Mode  : ":CURSOR 20,22:PRINT "Case  : "
5005  CURSOR 35,22:PRINT "Char  : "
5006  CURSOR 5,21:PRINT "Speed : ":CURSOR 20,21:PRINT "S-O-S : "
5007  CURSOR 35,21:PRINT "PTT   : "
5008  CURSOR 12,21:PRINT BD!:CURSOR 28,21:PRINT "OFF"
5009  CURSOR 43,21:PRINT "OFF"
5010  PRINT "                < To escape press ' TAB' >"
5012  POKE #BF47,M!
5014  CURSOR 0,19
5020  RETURN
```

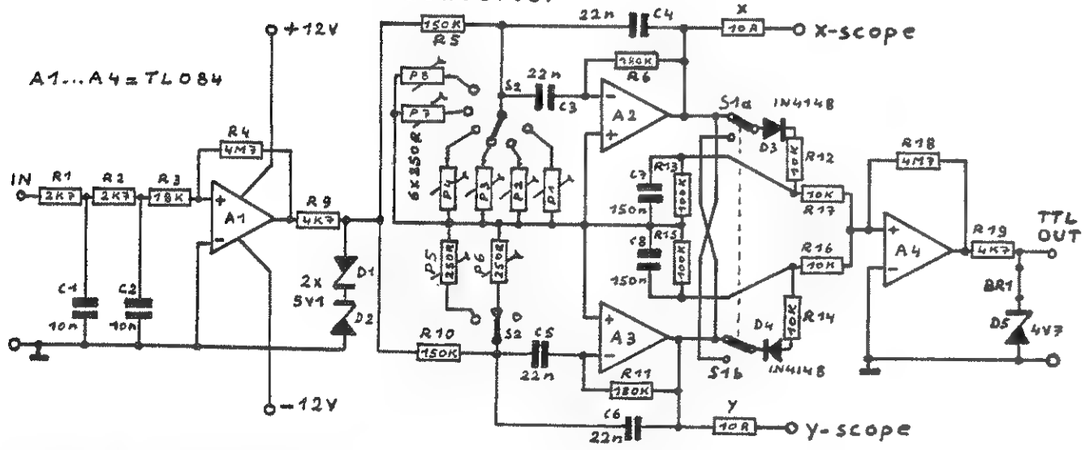


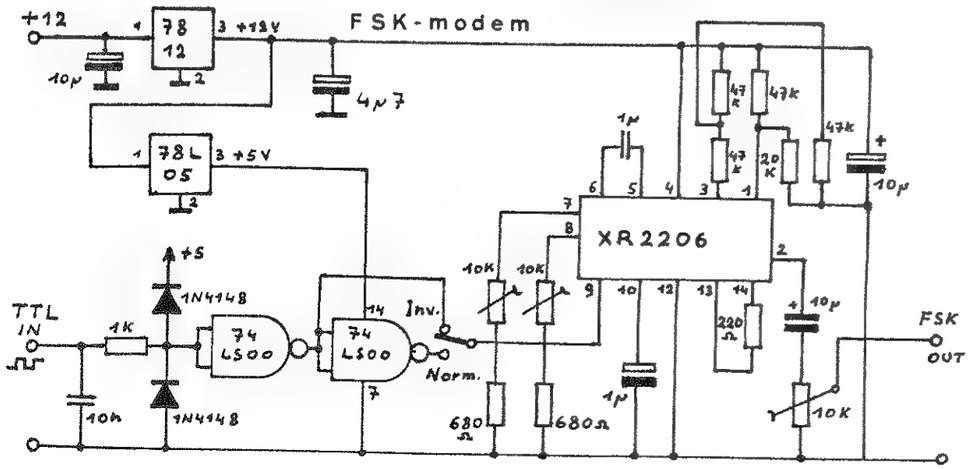




RTTY Converter

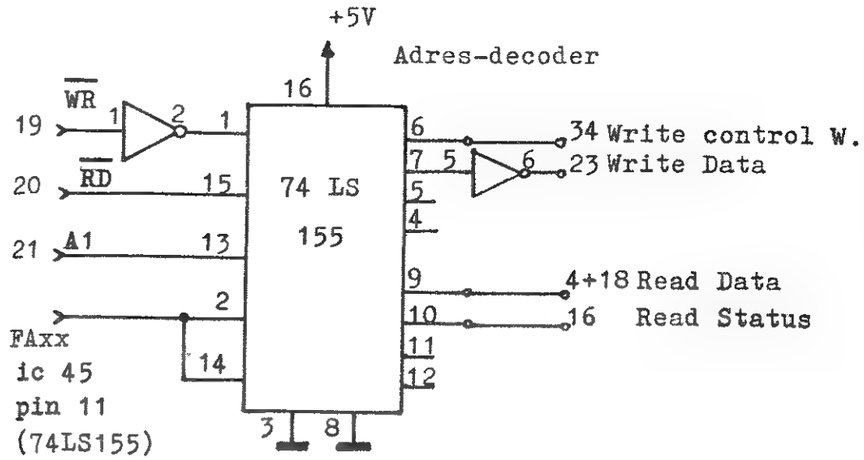


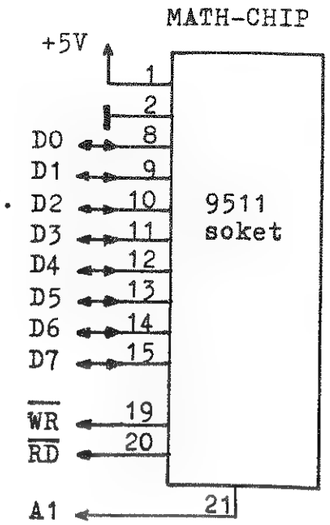

FSK-modem

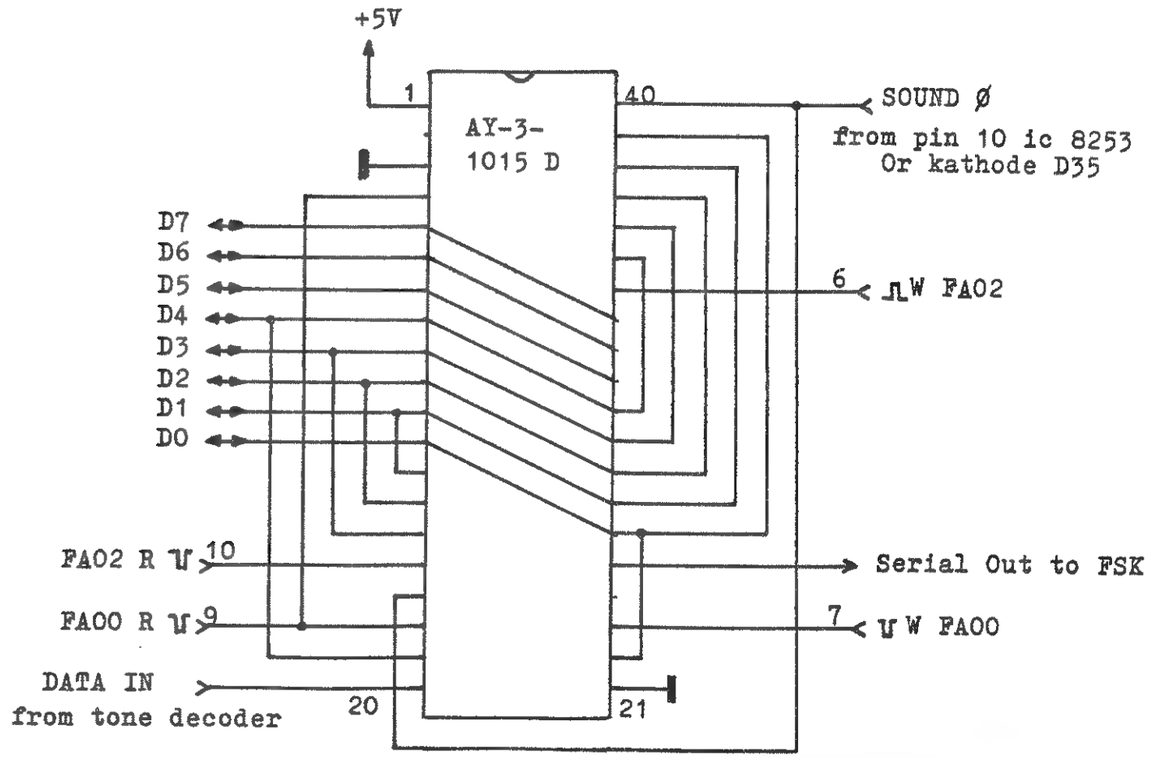





UART1 - INTERFACE



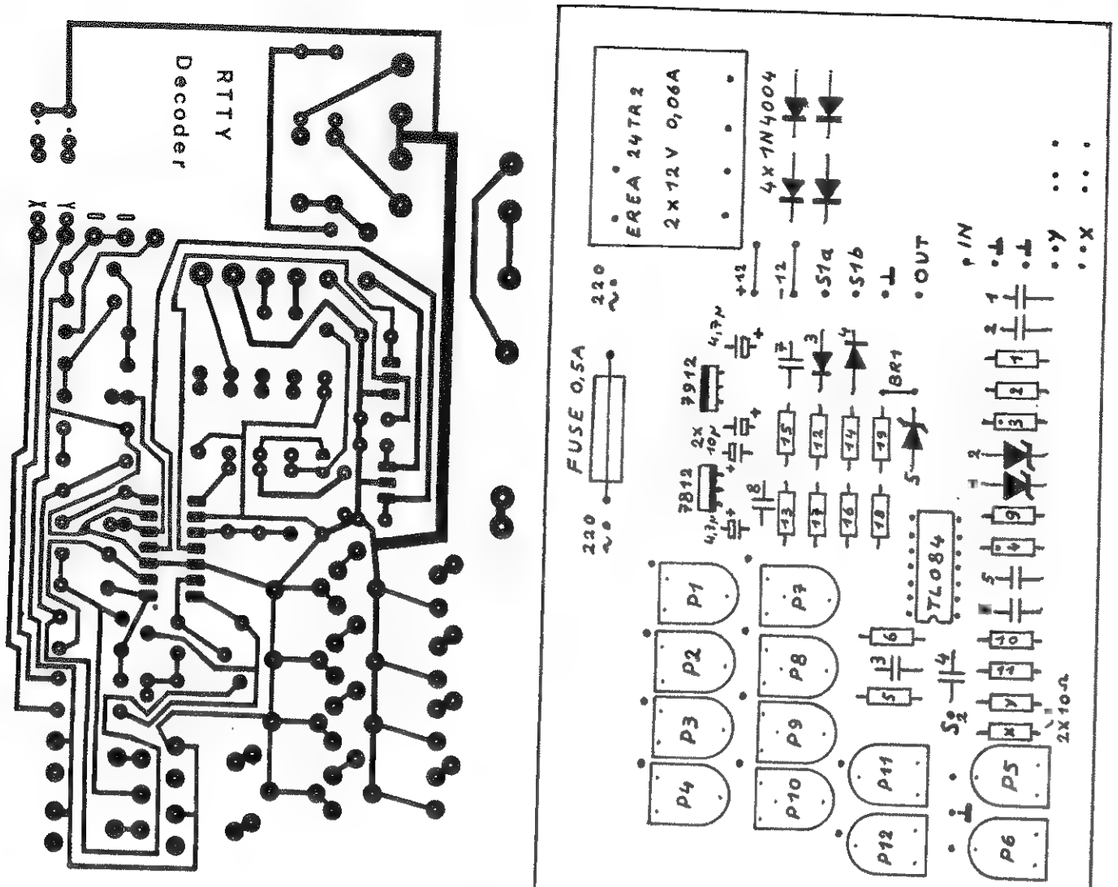

Adress range:

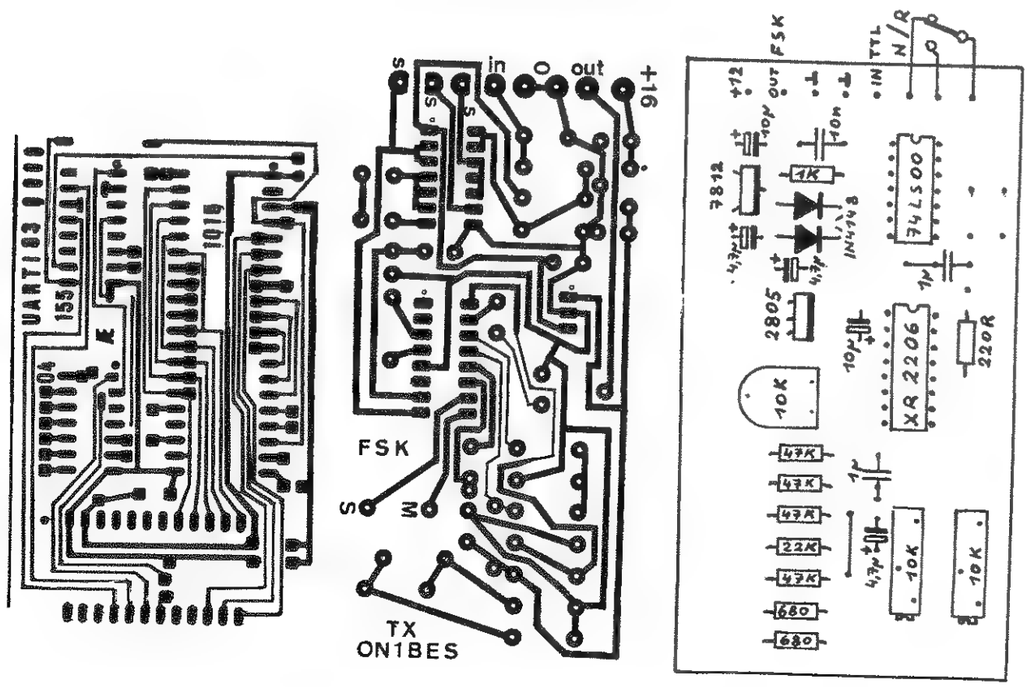
FA00 Read (Peek) data receiver
FA00 Write(Poke) data in transmitter buf.
FA02 Read status UART
FA02 Write control word in UART

Controlword: 25 Dec.= 5 bits,1 1/2 stopbit

Status UART:

FA00 IAND 1 = Send Buffer empty
FA00 IAND 2 = PE - error
FA00 IAND 4 = FE - error
FA00 IAND 8 = OR - error
FA00 IAND 16 = Data rec. ready






# DAI-DOS 1541

## OPERATING SYSTEM FOR

## DISK DRIVE VC1541
## DIGITAL RECORDER
## PARALLEL PRINTER
## HIGH SPEED LOADER

### GENERAL:

This article is an introduction on the DAI-DOS 1541, which is announced in Newsletter 26.

Up to 4 VC1541 disk drives are supported by the operating system, together with 4 digital MDCR recorders and 2 audio-cassette recorders.

The operating system is contained in one 2764 EPROM, using the address areas F000-F7FF and F900-F9FF. The address area FA00-FAFF is used for an additional 6116 RAM.

The software is fully compatible with the existing DAI BASIC versions V1.0 and V1.1. No addresses in the RAM area 0000-BFFF are used for the disk operating system, except the ones already used by the MDCR software.
For addressing purposes, the DCE-RWC concept is used.

The Memocom MDCR software is also compatible, as long as it is used from BASIC, or under UTILITY is addressed only via the entrypoints as given in the MDCR manual.

Also software for driving a parallel printer and a high speed data loader is implemented, together with several additional commands.

### MEMORY MAP:

The address range F000-F7FF is used by several switched ROM banks:

Bank 0: General routines for the floppy disk drive (FDD), the digital cassette recorder (DCR), the parallel printer, the extra commands and all housekeeping routines.
Bank 1: The floppy disk operating system.
Bank 2: The operating systems for the digital recorder and the High Speed Data Loader.
Bank 3: Free for own use, except F100-F1FF.

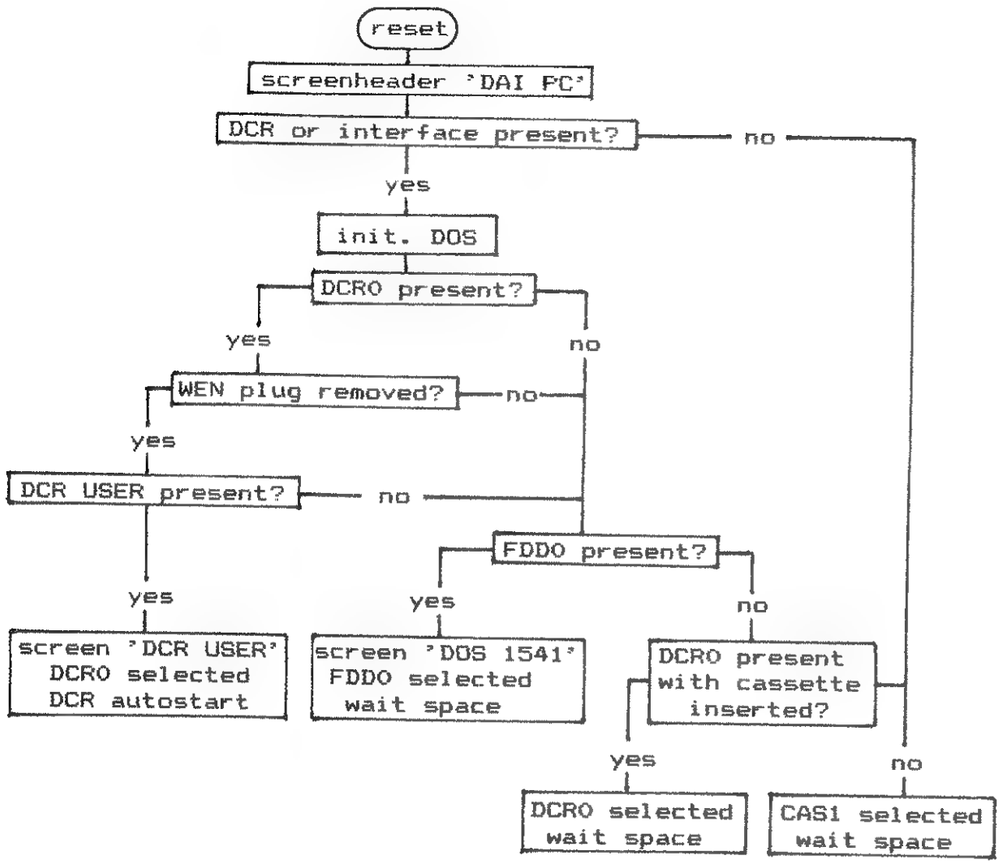
The addresses F900-F9FF are used for several general purpose routines. This address range is located in bank 3 of the EPROM.

A parallel printer can be connected to the 26 pins connector

on the interface. The high speed data loader must be connected directly to a connector on the flat cable.

POWER-ON:
---

When the DAI is switched on, or when the 'reset'-button is pressed, the following happens:




COMMANDS:
---

All DAI commands for writing and reading like LOAD, SAVE, LOADA, SAVEA and UTILITY W(rite) and R(ead) are valid for DCR and FDD. Only the command 'CHECK' does not work on the FDD.
All files must be saved to disk with a filename. When using filenames, max. 16 characters are allowed.
Existing files can be overwritten by using: "%:FILENAME".

The DCR commands are equivalent to the commands described in the MDCR manual, except that only the short versions are supported.
If a DCR is selected as back-up memory, all FDD commands are disabled (DEVICE MISMATCH error).



For the FDD, the following commands are available:

- FDD X : Selects disk drive 0-3 as back-up memory. DCR commands are disabled and will result in a DEVICE MISMATCH error.

- FORMAT : FORMAT"DISKETTENAME,ID". Formats a new diskette. Each diskette should have a name of max.16 characters and a unique identity code of 2 characters.

- SCRATCH: SCRATCH"FILENAME1,FILENAME2,.....". Removes indicated files from the diskette.

- RENAME : RENAME"NEWNAME=OLDNAME". Changes the name of a file on the diskette.

- COPY : COPY"NEWFILE=OLDFILE1,OLDFILE2,....". Merges several files together in a new file.

- INIT : INITializes the disk drive to the power-up state.

- VALID : The VALIDate command re-organizes the diskette to get the max. free space available.

- DIR : Reads the DIRectory from the diskette into the screen memory.
By means of the cursor keys, a BASIC or a UTILITY program can be loaded directly from the directory.
PRG-files will run automatically when loaded from the directory.
For USR-files, the UTILITY mode is entered.

- FVER : FVER"FILENAME". Verifies the recording of the indicated file by means of a checksum test.

Because all files are opened and closed automatically, no additional 'OPEN' and 'CLOSE' commands are required.

Several additional commands are available:

- BOOT : BOOT"OBJECTname,BASICname". Loads an object code program, followed by a BASIC program. This command is a replacement for the DBL bootstrap loader.

- UBL : UBL"FILENAME".
As 'BOOT',but for files consisting of a BASIC and a object code part, stored with the same filename with the following suffixes:
  - FILENAME/O object code part
  - FILENAME/B BASIC part

- CAS X : Selects an audio-cassette recorder as back-up memory. If no number is given, recorder 1 is selected.
DCR and FDD commands will cause a DEVICE MIS-

MATCH error.

- HSL X : Reads data from EPROM 0-3 in the High Speed data Loader.

- USR : Calls a USER defined object code subroutine via the I/O table address #02DA.

- LNON : Switches on the AUTO-LINENUMBER function. If in the direct mode the first input is a space then automatically a linenumber is generated.
- LNOFF : Switches off the AUTO-LINENUMBER function.

- <TAB> : Clear the screen.

- /C : The default cursor - flashing underline - is selected.

- /D : Output to the screen only.

- /E : Input from the edit buffer.

- /F : Sets Implicit Floating Point.

- /H : 'HELP'. Displays several important I/O directions.

- /I : Sets Implicit Integer.

- /M : Short command for MODE 0.

- /P : Output to parallel printer.

- /S : Output to screen and RS232 output.

- /T : Sets default text colours.

- /0 : Defaults various DAI pointers. It is a combination of: select mode 0, clear screen, set default text colours, set default cursor and select IMP INT.

--- : READ LINENUMBER: It is possible to read DATA statements on a given program line via:

```
10   A%=<linenumber>: CALLM #F33A,A%
20   READ ....
```

## ERROR MESSAGES:

For both the DCR and the FDD, the standard DAI loading error messages are used.
The FDD reports several error messages itself. They can be found in the VC1541 manual.
In addition, several DOS error messages will be displayed.

## USE OF DOS COMMANDS FROM BASIC:

All DOS commands are valid in direct mode only. To enable the use of these commands from BASIC, the commands must be placed in a 'REM'-statement, which follows a CALLM #F000 statement:

```
10 CALLM #F000: REM FDDO:DIR
20 A$="FILENAME": CALLM #F000: REM SCRATCH A$
```

## WHAT WILL IT COST:

The VC1541 disk drive can be bought in every computer shop. There is quite a difference in the price !!!

The interface card and the EPROM card together will cost 325 Dutch guilders. An interface cable (not required if you have already a flat cable for the DCR) will cost 50 Dutch guilders in addition, but can be also easily made yourself.

## HOW TO GET IT:

Send two completely filled out EUROCHEQUES (max. Dfl. 300.- per cheque) for the amount of:

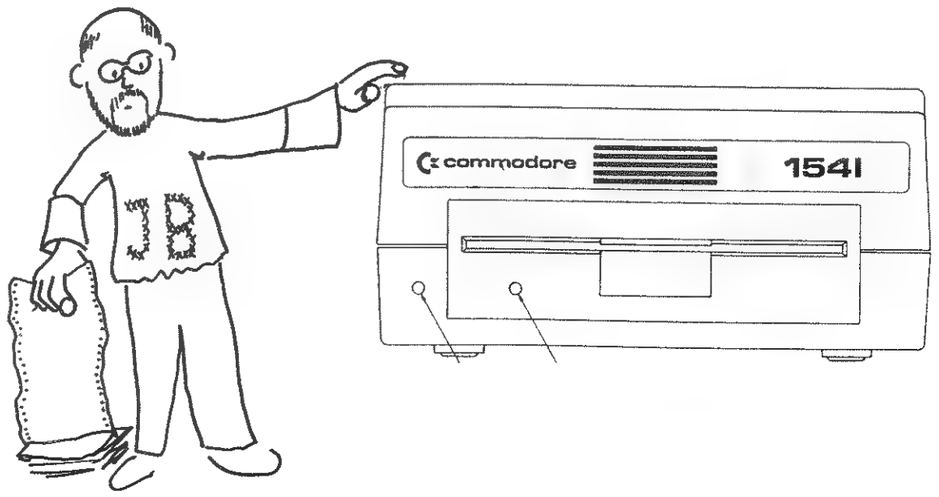
Dfl.325.- (interface + EPROM card)
or Dfl.375.- (idem + flat cable)
to:

Peter Wiegers
Corneliusstraat 5
6432 CZ Hoensbroek - The Netherlands

Delivery will take about 6-8 weeks.
For more information:

Jan Boerrigter
Fabritiusstraat 15
6174 RG Sweikhuizen - The Netherlands
tel. 04493-2093 (19-21 H)

The 'DAI-DOS 1541' is a co-production of: Jan Boerrigter, Hein Kop, Henk Rison and Peter Wiegers.



# VOEDINGSPERIKELEN

Werkend aan een programma werd het scherm ineens zwart. Niets bleek er nog te werken. Mijn DAI (rev 4) was volledig uitgevallen. Goede raad was duur. Ik heb de multimeter gepakt en al snel bleek wat er aan de hand was. De -5 volt spanning was geheel weggevallen. Na het losnemen van draadbrug J5 keerde de -5 volt spanning weer terug. De voeding was dus nog heel. Echter de rest was zo ongeveer kortgesloten: de schakelingen vormden samen een belasting van ongeveer 1,5 ohm, wat door de voeding als een kortsluiting wordt gezien, zodat de beveiliging in werking treedt. Aangezien het een zeer ingewikkeld apparaat is liet ik de reparatie liever aan deskundigen over. Toen kwamen de problemen. Middels de Fa vd Bend in Vlaardingen werd prijsopgave gevraagd bij de reparatieafdeling van een elektronicagigant in Den Haag. Mij werd n.l. van diverse kanten verzekerd dat de fa INDATA mijn DAI zeker zou herstellen, echter tegen forse uurtarieven en hoge transportkosten. Het was een vrij grote teleurstelling toen Den Haag liet weten d.m.v een schaderapport dat ook bij hen moest worden gerekend op minstens 4 uur werkplaatstarief tegen f80,-- per uur exclusief B.T.W. Daarbij kwam, dat de eigenlijke fout nog niet gevonden was. Wel kon men mij zeggen, dat de 8080 en de adresbuffers defect waren. Omdat het er op begon te lijken dat reparatie onbetaalbaar zou zijn heb ik uiteindelijk toen maar besloten zelf een poging tot reparatie te wagen. Zodoende heb ik machine weer opgehaald met een nieuwe 8080, 2x74LS241 (IC's 83 en 84), 1x7LS86 (IC 93), enkele stuks 2N3704, enkele tantaalelco's en diverse voetjes. Bij de fa vd Bend had men n.l. ondertussen ontdekt, dat T13 ook kapot was. Ik ben begonnen met de genoemde ic's gewoonweg te slopen. Want hoe kom je nu zo gauw aan een desoldeerstation? Iets lossolderen op een doorgemetalliseerde, dubbelzijdige print valt niet mee! Op de vrijgekomen plaatsen heb ik voetjes ingesoldeerd. Toen bleek na meting de kortsluiting nog steeds te bestaan. Blijkbaar zat de sluiting niet in de 8080. Er zat toen niets anders op dan de -5 volt leidingen een voor een door te snijden. Uiteindelijk werd de schuldige gevonden! Het was C44, die de opamps in IC 15 (in het geluidsgedeelte) ontkoppelt. Na vervanging van C44 en T13 werden de nieuwe ic's in de voetjes gestoken, de doorgesneden printbanen weer hersteld en toen de spanning erop. De - 5 volt was in orde, maarverder gebeurde weer niets! De 5501 bleek ook nog overleden te zijn. Ook die werd vervangen. Weer geen resultaat. Wel waren bleken de adres en datalijnen weer in orde. Na geruime tijd meten kwam ik bij de PALkaart. Daar werkten IC1 (LM 1889) en T2 (2N3704) niet, die eigenlijk niet verdacht werden aanvankelijk, alle signalen op connector 8 waren n.l. in orde. Toen die vervangen waren bleek mijn DAI tot mijn grote voldoening weer te werken. Mocht iemand ooit dezelfde problemen hebben, dat weet hij/zij nu wat er bij het wegvallen van de -5 volt spanning allemaal stuk gaat. Verder nog een verzuchting: Wat zou een reparatie makkelijk zijn als de print van fabriekswege geheel van voetjes zou zijn voorzien!

Tenslotte nog dit: in DAInamic nr 26 stond een tip over het "ontstoren van de resetschakelaar", zodat stoorpulsen geen kans meer zouden krijgen de DAI te resetten. Wanneer men een weerstand van 220 ohm van het middelste kontakt van de resetschakelaar naar de +5 volt soldeert, wordt T10 bij het inschakelen van de DAI nogal zwaar belast. Men bereikt hetzelfde, en dan zonder ekstra belasting als men van het middenkontakt naar 0 volt een keramische condensator van 56nf soldeert. Dit werkt prima!

Hartelijke groeten,

A.W.Verheijen
Loevestein 21
4254 EH Sleeuwijk
Nederland
tel: 01833-2392



# FWP tips

Christian LEQUESNE
"Les Bois"
72450 MONTFORT-LE-ROTROU

TIPS for FWP owners:

---

1 To add a long commentar in a basic program:
chose 60 characters/line in the default menu
24 lines/step
write,with FWP,your entire text.
Then use the function REPLACE to change lines to basic lines.
?"... line..."
Replace (everywhere) SHIFT CHAR DELETE 0D
by
"(quote) SHIFT CHAR DELETE 0D space space space ?(print) "(quote)
You will only have to write the basic line numbers before each ?(print)
Explanation :you have just surrounded each carriage return (#0D)
by one " (end of previous line) one PRINT one " (beginning of next line
Go to basic and POKE 309,2 to retrieve your text in a basic form.

---

2 To rename a variable (small program or part of a big program)
bring your basic program into FWP buffer 1
REPLACE old name by the new one.(all)
More interessing :to change the type of a variable.(impossible with SPU)
Replace X by X% (all) if X is a ! type for instance.
Then come back to basic. (& POKE 309,2)

---

3 To make a table with many lines and columns:
work in buffer 2 and prepare just a row with the tabulation
that you want.Like this:

```
!       !       !       !       !       !       !
!       !       !       !       !       !       !
!       !       !       !       !       !       !
-------------------------------------------------
```

Go back to buffer 1 and put as many markers 01 (shift char delete 01)
one upon the other,as you want rows ;then use as many times
the function GET.

---

4 By the same way ,you can print a short text many times in ONE print:
Place your text in buffer 2.
In buffer 1,put several markers 01 one upon the other
(the shortest the text is,the most markers you can put)
Use as many times the Get function (buffer 1 is very big ;)
Print the lot as a whole text'

---

Thank you to GER GRUITERS!

# DAInamic FRANCE

Cet utilitaire permet la réalisation de dessins 32 points x 32 points en Mode 6 et le stockage de ces dessins pour pouvoir les réutiliser dans vos jeux

Chargement de ce programme :

a) Taper la partie Basic et l'enregistrer sur cassette

b) Taper au moyen de la commande (S)ubstitute la partie langage machine et l'enregistrer: W3000 305F LM 3000 305F

c) Démarrer le programme avec RUN

Déscription générale

-Le dessin se fait dans un carré 32x32 en mode 2 (nettement plu reposant pour la vue !!) au moyen d'un point que l'on deplace a l'aide de PDL(0) et PDL(1)

-Appuyer sur le bouton du paddle fixe le point sur le dessin .

Commandes

| Appui sur | Donne |
|---|---|
| 1,2,3 ou 4 | Change la couleur du point avec le registre 1,2,3 ou 4 |
| S | Stocke le dessin (les octets le constituant) entre les adresses #5500 et #555F. |
| Z | Zoom :montre le dessin en mode 6 (transfert de #5500 a #555F sur l'écran mode 6.AVANT d'appuyer sur Z, appuyer sur S |
| R | Retour au mode 2 avec dessin (transfert de #5500 a #5600 sur l'ecran mode 2) |
| N | Nouveau dessin . Attention la commande N n'annule pas le dessin précedent . Si vous voulez vraiment l'effacer , faire N ET S . |

Lorsque le dessin est terminé , appuyer sur BREAK , taper UT , W5500 550F DESSIN 1 . Pour le recuperer , charger l'utilitaire et sa partie langage machine , charger le dessin , faire RUN puis R ou Z .

Les 4 couleurs utilisées peuvent en definir d'autres (voir l'article de Cedric sur les trames dans le DAInamic 25)

Le désassemblage de la routine ZOOM (#3022 #3040) montre que celle ci transfere arbitrairement les octets stockés de #5500 à #550F sur l'adresse #A000 (coin inferieur droit du carré) . Une toute petite modification à cette routine et il est trés facile de transferer ce qu'on veut OU on veut , mais la je vous laisse travailler !!!

Les 27 dessins de MEMODAI ont été réalisés avec cet utilitaire et toutes les parties graphiques de MEMODAI avec un utilitaire du meme type .

Amusez vous bien

Patrick Pedelaborde



-BASIC-

```
1     REM *********************************
2     REM *UTILITAIRE GRAPHIQUE 4 COULEURS*
3     REM *********************************
4     REM
5     REM IMPINT
6     REM
8     REM INITIALISATION
9     REM
10    COLORG 8 0 3 5:MODE 2:X=20:Y=10:C=0:Z=8
20    FILL 16,2 47,33 0:FILL 17,3 46,32 8:GOSUB 600
21    REM
22    REM BOUCLE PRINCIPALE
23    REM
25    REP=GETC
30    IF PDL(1)>250 THEN V=1:H=0:GOSUB 500
40    IF PDL(1)<5 THEN V=-1:H=0:GOSUB 500
50    IF PDL(0)<5 THEN V=0:H=-1:GOSUB 500
60    IF PDL(0)>250 THEN V=0:H=1:GOSUB 500
65    IF PEEK(#FD00) IAND #20<>0 THEN DOT X,Y C:Z=C:REM DESSIN DU POINT
70    IF REP=49 THEN C=20:GOSUB 600
80    IF REP=50 THEN C=21:GOSUB 600
90    IF REP=51 THEN C=22:GOSUB 600
100   IF REP=52 THEN C=23:GOSUB 600
120   IF REP=83 AND PEEK(#9D)=2 THEN CALLM #3000:SOUND 0 0 15 0 FREQ(444):WAIT TIME 30:SOUND OFF
 :REM STOCKAGE
130   IF REP=90 THEN MODE 6:CALLM #3022:REM ZOOM
140   IF REP=82 THEN MODE 2:CALLM #3041:GOSUB 600:REM RETOUR MODE 2
150   IF REP=78 THEN MODE 0:GOTO 10:REM ON RECOMMENCE !!
400   GOTO 25
497   REM
498   REM DEPLACEMENT DU POINT
499   REM
500   X1=X+H:Y1=Y+V:Z1=SCRN(X1,Y1)
510   DOT X1,Y1 C:DOT X,Y Z:Z=Z1:X=X1:Y=Y1:WAIT TIME 10:RETURN
597   REM
598   REM INDICATION COULEUR POINT
599   REM
600   FILL 52,2 60,10 0:FILL 53,3 59,9 C:RETURN
```

- LANGAGE MACHINE -

```
>D3000 305F
3000 C5 21 00 55 11 10 80 0E 20 06 08 1A 77 13 23 05
3010 C2 0B 30 C5 06 10 13 0E C2 16 30 C1 0D C2 09 30
3020 C1 C9 C5 11 00 55 21 0C A0 0E 20 06 08 1A 77 13
3030 23 05 C2 2D 30 C5 01 52 00 09 C1 0D C2 2B 30 C1
3040 C9 C5 11 00 55 21 10 BA 0E 20 06 08 1A 77 17 23
3050 05 C2 4C 30 C5 01 10 00 09 C1 0D C2 4A 30 C1 C9
```

# JEU DE CARACTERES

Certains lecteurs ont remarqué des differences entre une partie des caractères présenté dans le N0 25 et ceux qu'ils ont l'habitude de voir sur leurs écrans. Ceci est du au fait que la largeur des lignes en mode texte (Octets impairs des mots de commande égal à 7A) est réglée par le DAI à 10 points.
Cette valeur 10 à été choisie pour permettre d'une part de voir les caractéres alphanumériques en entier (Y compris le jambage des minuscules). D'autre part cela permet d'avoir un bon espacement entre chaque ligne (2 Points min.), ce qui accroit le confort et la lisibilité.
Par contre le dessin des lettres et sigles dans la MEM (Memoire morte) du générateur de caractéres est lui fixé à 16 points en hauteur.
Conclusion pour chaque sigle affiché il y a 6 points qui sont invisibles !. Les lettres et chiffres n'utilisent pas les possibilitées additionelles offertes par ces six rangées de points. Par contre les semi-graphiques sont pour la plupart programmés sur toute la hauteur de leur matrice. Si vous désirez vous en servir en entier il vous sera donc nécessaire de regler les mots de commandes de lignes à la largeur maximale soit 15 (7F dans le mots de commande).

Exemple :

Pour passer tout l'écran en mode 'Super-haut'
(On aura 16 lignes de texte au lieu de 24)

```
  1   REM ** MODE SUPER HAUT 30/01/85 C.D. **
 10   PRINT CHR$(12)
 20   FOR I%=#B815 TO #BFEF STEP 134
 30   POKE I%,#7F
 40   NEXT
 50   REM ** EXEMPLE DE SEMI-GRAPHIQUES **
 60   CURSOR 10,20:PRINT CHR$(6);CHR$(7)
 70   CURSOR 10,21:PRINT CHR$(14);CHR$(15)
 80   CURSOR 20,15:PRINT CHR$(17);CHR$(17)
```

Pour remettre l'écran dans son état initial il suffit de taper PRINT CHR$(12)
Pour Passer en mode 'Super-haut' la I eme ligne uniquement, taper :

```
POKE #BFEF-134*(24-I),#7F
```

Pour la remettre bien taper #7A au lieu de #7F

C.Dufour



# EXTENSIONS DAIminiCALC

Les lignes de BASIC qui suivent peuvent etre ajouté directement à DAI mini-CALC pour éviter le IMP INT avant le RUN . De plus vous aurez la possibilité de representer sur un graphe cartésien la ligne du tableau qui est pointée par le curseur.Il suffira pour cela d'utiliser la commande /G (Dessin en Mode 4).

(IMPINT)

```
    7   FOR I=#275 TO #29E:POKE I,16:NEXT

 1000   ... IF XX=0 AND XZ=0 THEN DD=6
 6150   IF PK=71 THEN GOSUB 26000:GOSUB 4500:GOTO 6200
 9100   GOSUB 9550:IF IMIN<>0 THEN GOSUB 55000
 9550   IMIN=0:FOR KK=1 TO 30:IF A(KK,JJ)<>0 THEN
         IMIN=IMIN+1
 9560   NEXT :RETURN
26000   MAX=A(YZ,1):MODE 4:COLORT 0 3 5 10
26010   FOR I=1 TO 14:IF A(YZ,I)>MAX THEN MAX=A(YZ,I)
26020   NEXT :EY=YMAX/MAX:EX=XMAX/13:AY=A(YZ,1)*EY:
         AX=0
26030   FOR I=1 TO 14:NY=A(YZ,I)*EY:NX=(I-1)*EX:
         DRAW AX,AY NX,NY 23
26040   DRAW NX,0 NX,NY 21:AX=NX:AY=NY:NEXT
26050   IF GETC=0 THEN 26050:MODE 0:RETURN
```

Olivier Pattiniez

---

* Les cassettes DCR certifiées sont toujours aussi difficile à se procurer à des prix interessants en France. Si vous connaissez un bon fournisseur, envoyez-nous son adresse !

* Programmeurs en langage machine & assembleur, des 'calculatrices' capable de convertir les nombres en decimal-hexadecimal-octal sont disponibles pour moins de 300 Frs. Ecrire à DAInamic France pour plus de précisions.

---

# FACSIMILE on DAI

The facsimile is a primarily commercial technique for transmission of written documents, weather maps, satellite pictures and press photographs. When the first successfully tested facsimile equipments appeared in great numbers on the user market, this technique also entered the field of amateur radio. However, a facsimile station can also be realized via pure electronic methods. A microcomputer naturally lends itself to do that. In order to show the associated advantages and limitations, I wrote an article on the subject at the begining of 1983 in the periodical "RTTY". With the kind permission of the "RTTY" editor's office, a great deal of its text as well as all its figures have been borrowed from the periodical in question.

The facsimile device consists of the following component parts:

1)A DAI computer 48KB and a Memocom Digital Cassette Recorder(MDCR)
2)A FAX-timer
3)An FM-to-BCD converter
4)A program

These individual points are described hereunder:

1)The DAI Personal Computer.

With its high colour-graphic resolution of 255*335 pixels and 4 grey shades, the DAI PC has , at least on that point, taken the leadership since a long time, so that it seemed well-suited for facsimile. Only the 4 shades of grey represent a limitation. It can only be hoped that the PC will in the future be equipped on a larger scale with graphic memory. A MDCR is used as program and graphic memory. Due to its high transfer rate of 6000 bauds, it is very good for loading and storing pictures.

2)The FAX-timer.

In the facsimile technique, one works in a quartz-synchronous fashion, which means that the recording and the broadcast has to take place with a precision of 10 to the power minus 5 per rotation. Figure 1 shows by means of a weather map that was received directly from Meteosat 2 (geostationary meteorological satellite operated by the European Space Agency), the recording format of a facsimile picture. Beginning below on the right, a starting signal of 300 Hz is first broadcast for a duration of 3 seconds. Thereafter follows the phasing signal. It consists of a 12.5 ms long, black picture data signal and of a 237,5 ms long, white picture data signal The next picture lines begin with a start of line signal. The signal is designed in order to produce an automatic amplifier regulation. Above, at the end of the picture, a stop signal can be seen. It lasts for 5 seconds and has a frequency of 450 Hz. The FAX-timer is also used for the synchronization of the broadcast and of the reception. It delivers, with quartz precision, the start signal for each line. The duration of the recording for one line is adjusted in the program by means of a waiting loop.

The circuitry consists of a quartz oscillator which vibrates on 256 KHz, and of two dividers. A J-K flipflop ensures a sensing ratio of 1:1.The principle of the programmable dividers originates from the periodical "UKW-Berichte",number 4/79, where it is described in full details. The FAX-timer by hardware has the following advantages on the software solution:



(a)The timing can be lengthened by means of two keys, in order to bring the picture in phase (i.e.,to center the picture on the screen)

(b)Owing to the 1:1 sensing ratio, one can use the signal for testing purposes.By means of a waiting loop in the program the duration of a line is adjusted in such a way that one period of the rotation number in force exactly fills the screen.

(c)The timer can also be replaced by other signal sources. In that way, the DAI and a telecopier can be linked in phase with each other, in order to transfer pictures from the copier to the DAI, and vice versa.

3)The FM-to-BCD decoder

The circuitry of the FM-to-BCD decoder originates from Volker Wraase.It was published in the"CQ-dl" and is therein very well described. The LF signal is transformed into a rectangular signal by the operational amplifier IC1. The following differential element generates on the falling side sharp impulses which trigger an oscillator and a monoflop (IC5). The oscillating frequency is about 16 times the black frequency. The oscillator and the monoflop are adjusted by means of the potentiometers VR1 and VR2, in such a way that the following binary counter IC6 counts up to 16 during the black frequency (1500Hz) and stops counting during the white frequency (2000Hz). These values are stored by IC7 until the next period. A diode display is connected to the exit of IC7. It consists of a 4 to 16 decoder which steers 16 lightdiodes(for 16 shades of grey). Only that way can one judge if the tuning is good and if VR1 and VR2 are correctly adjusted. By means of switch S2, the synchronizing signal can be incorporated in the picture for testing purposes.

4)The program.

In its original version, the program originates from the QUASAR software house in Haltern, but it has been modified and improved in relation with several important points:

(a) Broadcasting

In the beginning I generated the broadcasting signal directly with the sound exit of the DAI. In the meantime however, I have used a AM and FM modulator which is steered by 2 bits of the DCE bus. In that way I can send pictures to the telecopier or broadcast them by radio.

(b) Receiving.

The resolution of 225*335 pixels and 4 shades of grey is of course not normally sufficient for facsimile. I have therefore tried to generate pictures by sofware, with a greater number of shades of grey. In this working mode, the program puts always 2 points together, so that 9 shades of grey are possible. This naturally halves the resolution. On the diagram (figure 2) it can be seen that a strong error happens in the middle area. A subsidiary program now eliminates numbers 8, 9, 10,11 and fills the remaining ones with 15. In this way one obtains a grey distribution which is usable.

(c) Generating a test picture.

(d) Inserting a call sign in the graphic data to be broadcast.

(e) Picture loading from and storing on MDCR.

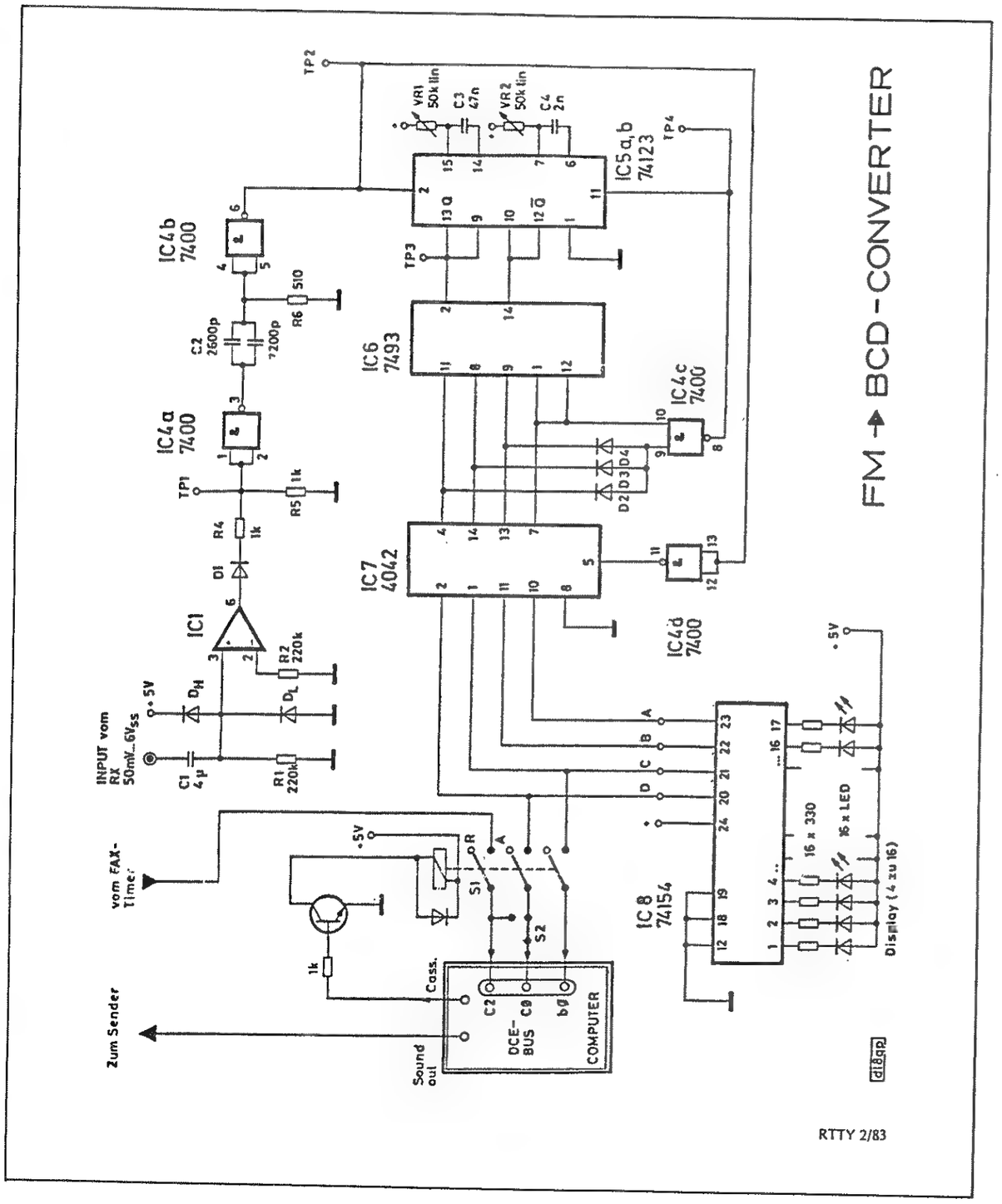
(f) Indicating the scanning speed (rotation number)

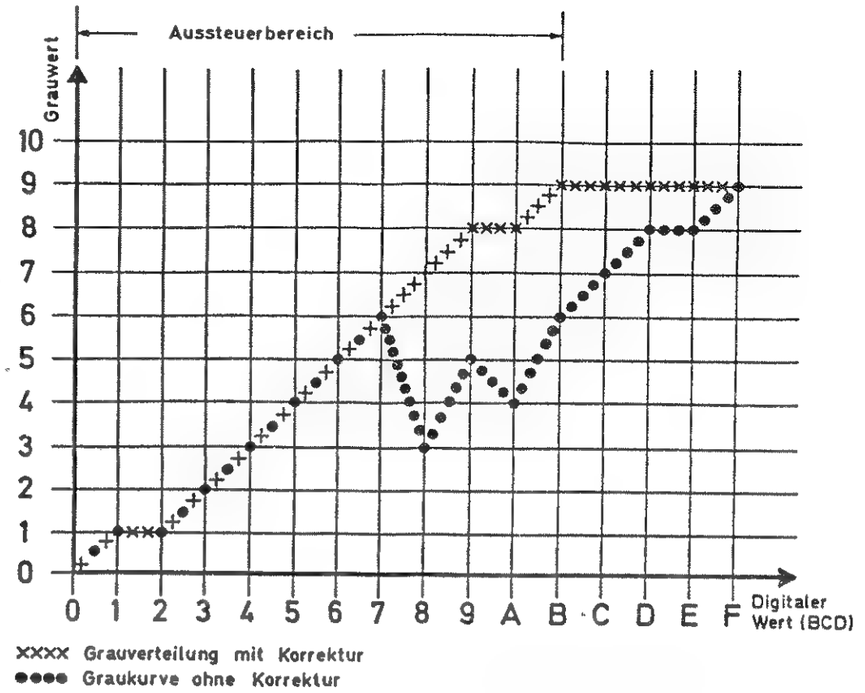
(g) Obtaining the picture on the printer.

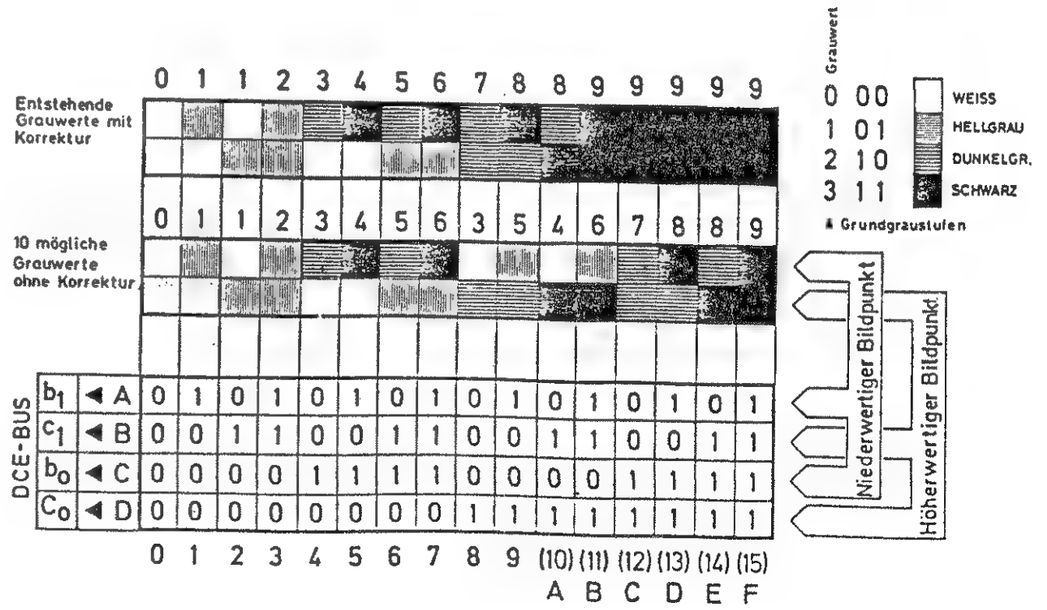
I hope that, being stimulated by the present article, numerous DAI users will get busy with the problem of picture processing, as this subject will gain more and more significance in the future.

I remain at your disposal for further information.

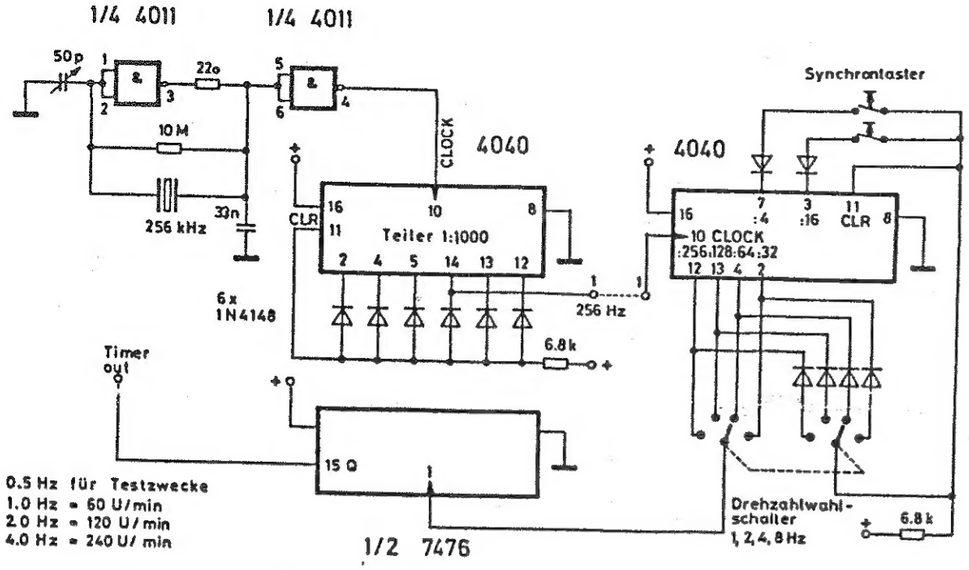
Willi Sicking, Jägerweg,31,4423 Gescher 02542/611.



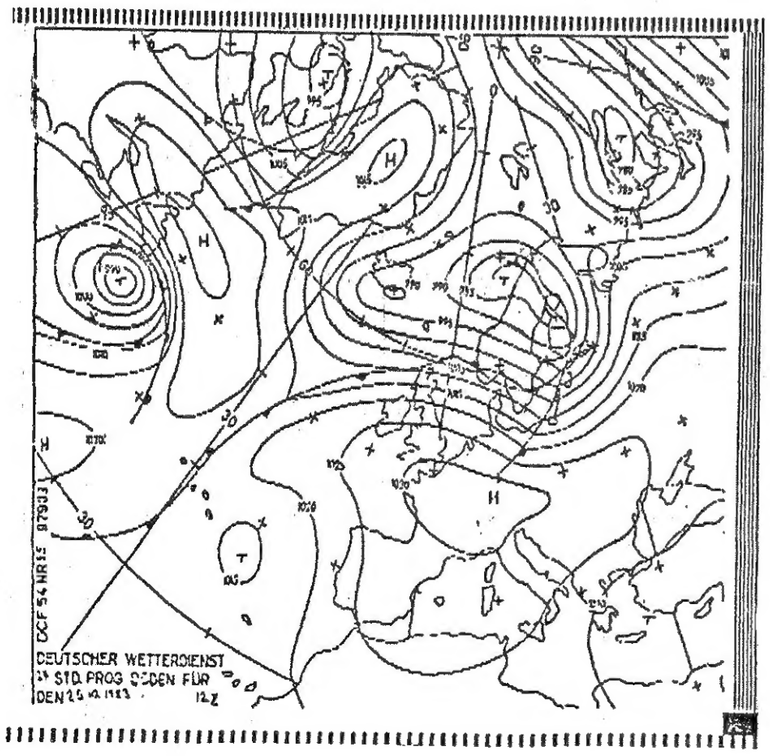




Die 0 gekennzeichneten Nullen werden zu 1 gesetzt.




FAX-TIMER




see also front page of Newsletter 25.



## program identification

| | | |
|---|---|---|
| title | : | COULEURS MULTIPLES |
| author | : | C.DUFOUR |
| purpose | : | gives many color-scales |
| comment | : | |

```
2     REM PGM DE COULEURS MULTIPLES
4     MODE 0:PRINT CHR$(12)
10    CURSOR 0,0:COLORG 0 1 1 1
12    CURSOR 19,14
14    INPUT "ENTREZ:HAUTEUR DES LIGNES[0-15]";C%
20    FOR I%=#BFEF TO #BF00 STEP -8
30    POKE I%,#30+C%:POKE I%-2,0:POKE I%-3,0
32    POKE I%-4,#30+C%:POKE I%-6,#AA:POKE I%-7,0
40    NEXT
50    FOR I%=#BFEE TO #BF00 STEP -4
60    READ A%:IF A%=#FF THEN 300
70    POKE I%,A%
80    NEXT
100   DATA #80,#81,#A1,#8C,#AC,#8F,#AF
101   DATA #8B,#AB,#82,#A2,#84,#A4,#83,#A3,#8A,#AA,#8E
102   DATA #AE,#8F,#AF,#8D,#AD,#85,#A5
103   DATA #87,#A7,#80,#A0,#88,#A8,#8F
104   DATA #AF,#80,#A0,#80,#A0,#80,#A0
105   DATA #8B,#AB,#84,#A4,#81,#A1,#85
106   DATA #A5,#8E,#AE,#8A,#AA,#83,#A3
107   DATA #80,#A0,#90,#FF
```

---

```
1     REM M.DIERCKX SINGING IN THE MORNING ...
2     ENVELOPE 0 16
5     Z=RND(40)+10
10    FOR X=100 TO 1000 STEP Z:SOUND 1 0 15 0 X:NEXT
20    GOTO 5
```

# DAI draws DAI

```
1       REM ================================================================
2       REM COPYRIGHT BY Paolucci Roberto - Via del Crocifisso 4 - Ancona.
3       REM ================================================================
10      CLEAR 15000:DIM STACK(255.0,2.0):COLORT 15 0 0 0:COLORG 15 1 1 10:MODE
        5:COL!=0.0
20      READ A!,B!,C!,D!:DRAW A!,B! C!,D! COL!
35      A!=C!:B!=D!
40      READ C!,D!
50      IF C!=400.0 THEN READ A!,B!:GOTO 40
60      IF C!=999.0 THEN 80
70      DRAW A!,B! C!,D! COL!:GOTO 35
80      FILL 140,129 199,136 0:FILL 200,128 202,132 0:FILL 120,24 206,30 0:FILL
        107,32 230,38 0
90      FILL 102,40 236,46 0:FILL 97,48 230,54 0:FILL 105,56 236,62 0:FILL 242,
        32 244,34 0
120     M!=131.0/15.0:FOR I!=1.0 TO 14.0:N!=(M!*I!)+105.0:DRAW N!,56 N!,62 15:
        NEXT
130     M!=133.0/15.0:FOR I!=1.0 TO 14.0:N!=(M!*I!)+103.0:N1!=(M!*I!)+99.0:DRAW
        N!,40 N!,46 15
140     DRAW N1!,48 N1!,54 15:NEXT:DRAW N1!,48 N1!,54 0
150     M!=123.0/14.0:FOR I!=1.0 TO 12.0:N!=(M!*I!)+117.0:DRAW N!,32 N!,38 15:
        NEXT I!:DRAW N!,32 N!,38 0
160     FOR I!=146.0 TO 196.0 STEP 3.0:DRAW I!,134 I!,136 15:NEXT I!
175     GOSUB 40200
180     X!=110.0:Y!=100.0:C!=5.0:VFLAG!=3.0:F!=1.4:A$="DAI":GOSUB 40240
185     X!=140.0:Y!=106.0:VFLAG!=3.0:F!=0.5:A$="PERSONAL":GOSUB 40240
190     X!=140.0:Y!=100.0:VFLAG!=3.0:F!=0.5:A$="COMPUTER":GOSUB 40240
200     X!=81.0:Y!=136.0:C!=6.0:C1!=0.0:GOSUB 10000:X!=258.0:Y!=136.0:GOSUB
        10000
210     X!=83.0:Y!=136.0:C!=7.0:C1!=0.0:GOSUB 10000:X!=254.0:Y!=138.0:GOSUB
        10000
230     X!=130.0:Y!=162.0:C!=5.0:GOSUB 10000
240     FOR I!=130.0 TO 210.0 STEP 5.0:DRAW I!,166 I!,172 15:NEXT I!
250     FILL 124,176 216,236 0:FILL 124,178 215,234 15
280     FOR Z!=0.0 TO 2.0*PI STEP 5E-2:A!=SIN(Z!):B!=COS(Z!)
290   1  DRAW 60,168 60+14*A!,168+14*B! 0:DRAW 276,168 276+14*A!,168+14*B! 0
300   1  DRAW 74,204 74+5*A!,204+5*B! 0:DRAW 290,204 290+5*A!,204+5*B! 0
310     DRAW 54,214 54+10*A!,214+10*B! 0:DRAW 272,214 272+10*A!,214+10*B! 0:
        NEXT Z!
360     FOR Z!=0.0 TO 2.0*PI STEP 5E-2:A!=SIN(Z!):B!=COS(Z!)
370   1  DOT 60+3*A!,168+3*B! 15:DOT 276+3*A!,168+3*B! 15
380     DOT 54+3*A!,214+3*B! 15:DOT 272+3*A!,214+3*B! 15:NEXT Z!
500     FILL 242,32 244,34 5:A!=91.0/16.0:B!=56.0/16.0:COL!=0.0
510     FOR X!=124.0 TO 215.0-A! STEP A!:FILL X!,178 X!+A!,234 COL!:COL!=COL!+
        1.0:GOSUB 600:NEXT
520     FOR X!=215.0-A! TO 124.0 STEP -A!:FILL X!,178 X!+A!,234 COL!:COL!=
        COL!-1.0:GOSUB 600:NEXT
530     FOR Y!=178.0 TO 234.0-B! STEP B!:FILL 124,Y! 215,Y!+B! COL!:COL!=COL!+
        1.0:GOSUB 600:NEXT
540     FOR Y!=234.0-B! TO 178.0 STEP -B!:FILL 124,Y! 215,Y!+B! COL!:COL!=
        COL!-1.0:GOSUB 600:NEXT
550     GOTO 510:SOUND OFF
600     SOUND 0 0 15 0 FREQ(X!*Y!/10.0):WAIT TIME 5:SOUND 2 0 15 0
        FREQ(Y!*X!/10.0):RETURN
1100    DATA 60,20,66,0,276,0,282,20,240,120,226,136,200,136,204,128
1120    DATA 138,128,140,136,114,136,100,120,60,20,68,20,70,12,270,12
1140    DATA 272,20,234,120,106,120,68,20,104,86,236,86,272,20,282,20
1160    DATA 400,400,82,22,100,64,240,64,256,22,82,22,400,400,82,236,82,134,40,
        140,40,230,82,236
1180    DATA 96,230,96,146,82,134,400,400,256,236,300,232,300,140,256,134
```





```
1200    DATA 256,236,240,230,240,144,256,134,400,400,92,64,248,64,400,400,94,68,
        246,68
1220    DATA 400,400,120,170,120,240,220,240,220,170, 210,160,130,160,120,170
1240    DATA 400,400,202,128,200,136,999,999

10000   REM C COLORE PER RIEMPIRE C1 COLORE DEL BORDO.
10020   GOSUB 10150:GOSUB 10180:GOSUB 10330
10030   RETURN

10100   REM PUSH
10110   STACK(P!,1.0)=X!:STACK(P!,2.0)=Y!:P!=P!+1.0:RETURN

10120   REM POP
10130   P!=P!-1.0:X!=STACK(P!,1.0):Y!=STACK(P!,2.0):RETURN

10140   REM CERCA IL BORDO DESTRO.
10150   FOR ED!=X! TO XMAX-1.0
10160 1  IF SCRN(ED!,Y!)<>C1! AND SCRN(ED!,Y!)<>C! THEN NEXT:RETURN
10170 1  ED!=ED!-1.0:RETURN

10180 1  REM CERCA IL BORDO SINISTRO.
10190 1  FOR ES!=X! TO 1.0 STEP -1.0
10200 2   IF SCRN(ES!,Y!)<>C1! AND SCRN(ES!,Y!)<>C! THEN NEXT:RETURN
10210 2   ES!=ES!+1.0:RETURN

10220 2   REM SEGUE IL BORDO   VERSO DESTRA.
10230 2   CC!=0.0:FEND!=0.0:FOR ES!=X! TO XMAX-2.0
10240 3    IF SCRN(ES!,Y!)<>C1! AND SCRN(ES!,Y!)<>C! THEN RETURN
10250 3    CC!=CC!+1.0:IF CC!>=DIFF! THEN FEND!=1.0:RETURN
10260 2   NEXT ES!:RETURN

10270 2   REM SEGUE IL BORDO VERSO SINISTRA.
10280 2   CC!=0.0:FEND!=0.0:FOR ED!=X! TO 2.0 STEP -1.0
10290 3    IF SCRN(ED!,Y!)<>C1! AND SCRN(ED!,Y!)<>C! THEN RETURN
10300 3    CC!=CC!+1.0:IF CC!>=DIFF! THEN FEND!=1.0:RETURN
10310 2   NEXT ED!:RETURN

10320 2   REM MASTER
10330 2   IF DY!=0.0 THEN DY!=-1.0
10345 2   DY1!=-DY!:GOSUB 10110:Y!=Y!+DY!
10350 2   GOSUB 10420:GOSUB 10480
10360 2   IF FEND!=1.0 THEN 10390
10370 2   DRAW ES!,Y! ED!,Y! C!:DIFF!=ED!-ES!:Y!=Y!+DY!
10380 2   X!=ES!:GOSUB 10420:X!=ED!:GOSUB 10480:GOTO 10360
10390 2   IF DY!=DY1! THEN DY!=0.0:FEND!=0.0:RETURN
10400 2   FEND!=0.0:DY!=-DY!:GOSUB 10130:GOTO 10350

10410 2   REM ESTREMO SINISTRO.
10420 2   IF SCRN(X!,Y!)=C1! THEN GOSUB 10230:RETURN
10435 2   IF SCRN(X!,Y!)=C! THEN FEND!=1.0:RETURN
10440 2   GOSUB 10190:RETURN

10460 2   REM ESTREMO DESTRO.
10480 2   IF SCRN(X!,Y!)=C1! THEN GOSUB 10280:RETURN
10490 2   IF SCRN(X!,Y!)=C! THEN FEND!=1.0:RETURN
10500 2   GOSUB 10150:RETURN

40000 2   REM LETTERE.
40200 2   DIM CAR$(90.0)
40210 2   FOR Z!=65.0 TO 90.0:READ A$
```

```
40220 3    IF A$="STOP" THEN RETURN
40230 2   READ CAR$(Z!):NEXT:RETURN
40240 2   X1!=X!:Y1!=Y!:IF F!=0.0 THEN F!=1.0
40250 2   FOR M!=0.0 TO LEN(A$)-1.0
40260 3    T$=MID$(A$,M!,1)
40270 3    GR$=CAR$(ASC(T$))
40280 3    FOR N!=0.0 TO LEN(GR$)-1.0 STEP 4.0
40290 4     IF VFLAG!=1.0 GOTO 40430
40300 4     IF MID$(GR$,N!,1)="/" THEN X!=X!+(8.0*F!):GOTO 40400
40310 4     ZZ!=VAL(MID$(GR$,N!,1)):YY!=VAL(MID$(GR$,N!+1,1))
40320 4     JC5=X!+ZZ!*F!:JC6=Y!+YY!*F!
40330 4     JC7=X!+VAL(MID$(GR$,N!+2,1))*F!:JC8=Y!+VAL(MID$(GR$,N!+3,1))*F!
40340 4     DRAW JC5,JC6 JC7,JC8 C!
40350 4     IF F!<1.5 THEN GOTO 40390
40360 4     JC9=X!+1+VAL(MID$(GR$,N!+2,1))*F!
40370 4     JC10=Y!+1+VAL(MID$(GR$,N!+3,1))*F!
40380 4     DRAW X!+1+ZZ!*F!,Y!+1+YY!*F! JC9,JC10 C!
40390 3    NEXT N!
40400 3    IF X!+8*F!>=XMAX THEN X!=X1!:Y!=Y!-10.0*F!
40410 2   NEXT M!
40420 2   RETURN
40430 2   IF MID$(GR$,N!,1)="/" THEN Y!=Y!-9.0*F!:GOTO 40480
40440 2   JC1=X!+VAL(MID$(GR$,N!+1,1))*F!:JC2=Y!-VAL(MID$(GR$,N!,1))*F!
40450 2   JC3=X!+VAL(MID$(GR$,N!+3,1))*F!:JC4=Y!-VAL(MID$(GR$,N!+2,1))*F!
40460 2   DRAW JC1,JC2 JC3,JC4 C!
40470 1  NEXT N!
40480 1  IF Y!-9*F!<0 THEN Y!=Y1!:X!=X!-9.0*F!
40490   NEXT M!
40500   RETURN
40650   DATA A,11155155135315373755/,B,11171747144414152535556/,C
40660   DATA 12162747475621414152/,D,1117114152561747/
40670   DATA E,1117115114441757/,F,111714441757/,G,121627572151515335343/
40680   DATA H,111714545157/
40690   DATA I,214131372747/,J,122121415257/,K,111713572451/,L,11171151/
40700   DATA M,11171735353435575751/,N,111751571652/,O,1216274756522141/,P
40710   DATA 1117144417475556/
40720   DATA Q,121627475653213133351/,R,11171747565514442451/,S
40730   DATA 12212141525324441516274747 56/,T,17573137/
40740   DATA U,111721415157/,V,1317535713313153/,W,11175157113333513334/
40750   DATA X,11121716515257561256 1652/
40760   DATA Y,161756571634345631 34/,Z,175712561151/,STOP
```



# MOVING TEXT

```
2       REM Moving text / Thijs Berkx / dec. '83
10      CLEAR 200

11      REM
12      REM You can shorten this program by removing all
13      REM lines with linenumbers that are not multiples
14      REM of ten.
15      REM
16      REM input number of lines and text
20      GOSUB 100

21      REM
22      REM Calculate start and end of screen (=source)
23      REM Transfer start and end of source to mlp.
30      GOSUB 200

31      REM
32      REM The text is shifted as many times as there are
33      REM characters. Each time a new screen is build and
34      REM moved to a separate buffer.
35      REM SCR%=screennumber; NRCH%=number of characters
36      REM build screen  gosub 300
37      REM move screen to buffer  gosub 400
38      REM shift characters in text  gosub 500
39      REM
40      FOR SCR=1 TO NRCH
50    1  GOSUB 300:GOSUB 400:GOSUB 500
60      NEXT SCR

61      REM
62      REM Now a buffer is source and the screen is
63      REM destination. Transfer destination to mlp.
70      GOSUB 600

71      REM
72      REM By moving the buffers one after another to
73      REM the screen a moving text is created  gosub 700
74      REM Keep text moving until space-bar is hit.
75      REM
80      G=GETC:IF G<>32 THEN GOSUB 700:GOTO 80
90      END

93      REM
94      REM input number of lines and text
95      REM
96      REM NRLN=number of lines
97      REM NRCH=number of characters
98      REM MXNRCH=maximum number of characters
99      REM
100     MODE 0:PRINT CHR$(12):CURSOR 10,13
110     PRINT "Number of moving lines (1-24) ";
120     INPUT NRLN:PRINT :IF NRLN<1.0 OR NRLN>24.0 THEN GOTO 100
130     MXNRCH=FRE/NRLN/#86
140     IF MXNRCH>60 THEN MXNRCH=60
150     PRINT TAB(10);"Type text of maximum";MXNRCH;" characters."
160     PRINT :PRINT " ----:----:----:----:----:----:----:----:----:----:
        ----"
170     INPUT TEXT$:NRCH=LEN(TEXT$)
180     IF NRCH<1 OR NRCH>MXNRCH THEN GOTO 100
```

```
PAGE 02  --  MOVING BLOCK

190     RETURN

191     REM
192     REM Calculate start and end of screen
193     REM Transfer source to mlp
194     REM
195     REM STSCR=start of screen; EDSCR=end of screen
196     REM STSRC=start of source; EDSC=end of source
197     REM NRLN=number of lines
198     REM
200     STSCR=#BFF0-(12+NRLN/2)*#86
210     STSRC=STSCR
220     POKE #305,STSRC IAND 255
230     POKE #306,STSRC SHR 8
240     EDSCR=STSCR+NRLN*#86
250     EDSRC=EDSCR
260     POKE #308,EDSRC IAND 255
270     POKE #309,EDSRC SHR 8
280     RETURN

291     REM
292     REM build screen
293     REM
294     REM NRLN=number of lines
295     REM NRCH=number of characters
296     REM
300     PRINT CHR$(12):CURSOR 0,11+NRLN/2+NRLN MOD 2
310     FOR I=1 TO NRLN
320   1  FOR J=1 TO 60/NRCH:PRINT TEXT$;:NEXT J
330   1  PRINT
340     NEXT I
350     RETURN

391     REM
392     REM move screen to buffer
393     REM
394     REM STBF=start buffer (destination)
395     REM NRLN=number of lines
396     REM SCR=screennumber
397     REM mlp "move block" starts at #300
400     STBF=#B350-SCR*NRLN*#86
410     POKE #30B,STBF IAND 255
420     POKE #30C,STBF SHR 8
430     CALLM #300
440     RETURN

491     REM
492     REM shift characters in text
493     REM
500     TEXT$=RIGHT$(TEXT$,1)+LEFT$(TEXT$,NRCH-1)
510     RETURN

591     REM
592     REM Transfer destination to mlp.
593     REM
594     REM STSCR=start of screen
595     REM
600     POKE #30B,STSCR IAND 255
610     POKE #30C,STSCR SHR 8
620     RETURN
```

```
PAGE 03  --  MOVING BLOCK

691     REM
692     REM move buffers in sequence to screen
693     REM
694     REM NRBF=number of buffers
695     REM STSRCBF=start of source buffer
696     REM EDSRCBF=end of source buffer
697     REM mlp "move block" at #300
698     REM NRCH%= number of characters
699     REM

700     FOR BFNR=1 TO NRCH
710   1  STSRCBF=#B350-BFNR*NRLN*#86
720   1  POKE #305,STSRCBF IAND 255
730   1  POKE #306,STSRCBF SHR 8
740   1  EDSRCBF=STSRCBF+NRLN*#86
750   1  POKE #308,EDSRCBF IAND 255
760   1  POKE #309,EDSRCBF SHR 8
780   1  CALLM #300
790     NEXT BFNR
800     RETURN
```

```
PC UTILITY V3.3
>D300 315

0300 F5 C5 D5 E5 11 CC 64 21 08 6A 01 0A B7 CD 4F DE
0310 E1 D1 C1 F1 C9 FF
```