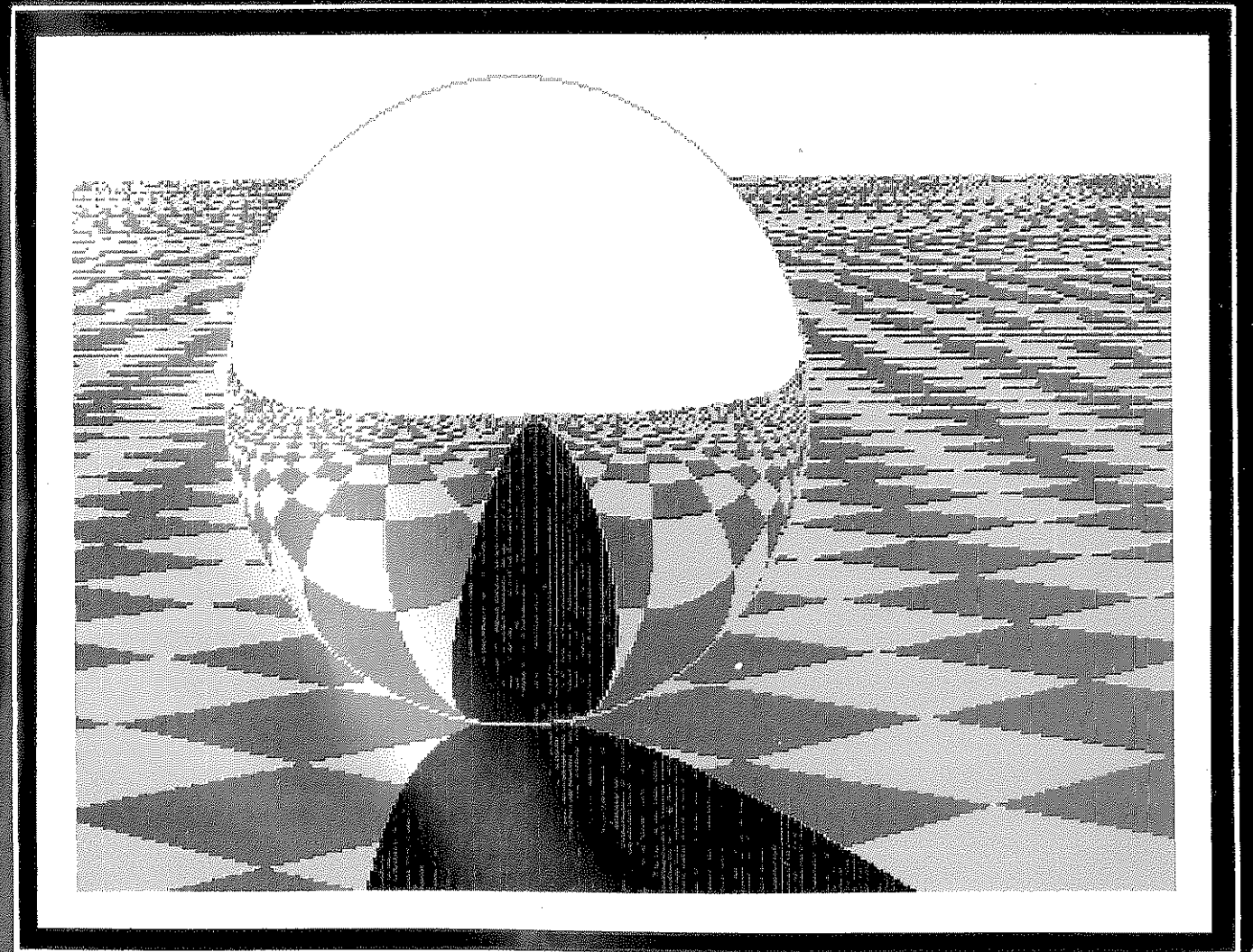


DAI
NAMIC

DAI
GLIC

35

8



tweemaandelijks tijdschrift juli-augustus 86

bimestriel juillet-août 86

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

International

COLOFON

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

DAInamic redactie :

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

Vormgeving : Ludo Van Mechelen.

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

Het abonnement loopt van januari tot december.

DAInamic verschijnt de pare maanden.
Bijdragen zijn steeds welkom.

CORRESPONDENTIE ADRESSEN.

Redactie en software bibliotheek

Wilfried Hermans	
Mottaart 20	Kredietbank Herselt
3170 Herselt	nr. 401-1009701-46
Tel. 014/54 59 74	BTW : 420.840.834

Lidgelden / Subscriptions

Bruno Van Rompaey	Generale
Bovenbosstraat 4	Bankmaatschappij
B-3044 Haasrode	Leuven
België	nr. 230-0045353-74
tel. : 016/46.10.85	

Voor Nederland :

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

Inzendingen : Games & Strategy

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

DAICLIC INFOS :

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

Conseil d'administration de IDC :

Président : Christian POELS, rue des Bas-Sarts 10,
B-4100 SERAING Tél. : 041/37.16.06
Secrétaire : Marc VANDEMEERSCH, av. Vert Bocage 17
B-1410 WATERLOO

Tél. : 02/354.13.63

Trésorier : Fabrice DULUINS, allée Tour Renard 4,
B-1400 NIVELLES Tél. : 067/21.82.10

Rédaction : Christian POELS
Soumissions logiciels : Marc VANDERMEERSCH
Inscriptions, vente logiciels : Fabrice DULUINS.
(mode de paiement : voir ci-dessous)

Cotisations :

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

Etranger : 1100 FB par mandat postal international
uniquement.

Services télématiques IDC :

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

Branches Régionales :

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

COPYRIGHT : Les articles publiés n'engagent que la
responsabilité de leur auteur. Toute reproduction, même
partielle, de ce magazine est interdite sans l'accord de
l'éditeur responsable.

INHOUD - SOMMAIRE

DAINAMIC 35 DAICLIC 8

1 INHOUD-SOMMAIRE	REDACTION
2 EDITO DAICLIC	
2 IDC SOFTWARE PC-DAI	D. BOITEAU
3 IDC SOFTWARE DISK-WIFE	X. DREZE
4 IDC BRUXELLES NEWS	DAIC
5 IDC PARIS NEWS DCA HARD	DCA
6 ASSEMBLEUR 8080	DCA
6 REGLAGES KEN-DOS	IDC BORDEAUX
7 MINITEL PART 3	IDC BORDEAUX
14 DAIDEV	DCA
19 CALCUL MATRICIEL	M. DECUYPER
24 DESSIN ANIME ASSISTE PAR ORD.	H. SAMAIN
27 L ENSEMBLE DE MANDELBROT	L. LAURENT
30 DAI QUI RIT CLAVIER AZERTY	A. MARIATTE
31 LOCK UNLOCK POUR VC 1541	DAINAMIC GERMANY
32 PETITES ANNONCES	IDC
33 FUNCTIETEKENAAR	F. DRUIJFF
37 X-BUS	H. RISON
39 TEKENPROGRAMMA	A. DE BRUIJN
41 PROGRAMMING CONTEST RESULTS	P. GOBERT
44 SOFTWARE PROMOTION	DAINAMIC
45 NEW DIRECTORY	K. VAN DE PERRE
53 PROGRAMMING IN MACHINE LANGUAGE	C. W. READ
59 RGB-MONITOR DIGITAAL	H. RISON
61 X-DOS	N. LOOIJJE
64 WALLPAPER	N. LOOIJJE

Edito 8

Toujours fidèles au poste, après un repos bien mérité, nous revoilà fin prêts (hum ...) pour affronter la rentrée qui s'annonce dès à présent bien chargée !

Avant tout, n'oubliez pas de réserver la journée du samedi 25 octobre 86 dans vos agendas ! Date du prochain Meeting que vous nous réclamiez depuis si longtemps.

Ce meeting s'annonce comme étant celui du changement ou de la cohabitation, pour employer un mot cher à nos amis français. Ce meeting ne sera plus consacré exclusivement à notre cher DAI puisque les clubs présenteront leurs nouvelles activités qu'ils développent autour d'autres micros. Tous les renseignements utiles concernant l'European Micro Meeting vous sont communiqués en page 13 (Dernière Minute).

Au sommaire de ce numéro, nous avons rassemblé de quoi contenter autant les fanas du fer à souder que les passionnés de software.

Une fois encore, nous tenons à remercier toutes les personnes qui directement ou indirectement nous apportent leur aide dans notre lourde tâche et notamment Werner Persoons, notre nouveau responsable software.

La Rédaction.

Le scoop de l'année !!!

Extraordinaire, formidable, incroyable... et utile : "IDC Software" va bientôt être en mesure de vous présenter un nouveau programme :

"PC - DAI"

Ce superbe programme, mis au point par M. Didier Boiteau, membre d'IDC BORDEAUX, vous permet (tenez-vous bien !) de lister le DIRECTORY au format MS-DOS/IBM PC à l'aide de votre DAI...

Jusque là, c'est pas mal me direz-vous, mais à quoi cela peut-il servir ? Evidemment, PC-DAI, n'est bien sûr pas limité à la lecture du DIRECTORY de n'importe quelle disquette provenant d'un IBM PC ou compatible. Il permet également de TRANSFERER les fichiers textes (ou autres fichiers, par exemple : programme BASIC...) de l'IBM vers le format DAI-KENDOS normal... et vice-versa. Stupéfiant n'est-ce-pas ??? Ce programme permettra donc à tout possesseur de KENDOS et d'IBM PC (je ne vise personne !) de transférer ses fichiers SANS DEVOIR PASSER par la RS-232 à l'aide d'un câble NUL-MODEM.

Plus de renseignements au sujet de ce programme dans un prochain numéro... mais dès à présent, apprenez-vous à faire un miracle : rendre le DAI (presque !) compatible IBM PC... ou du moins, en ce qui concerne ses fichiers... pas en ce qui concerne ses programmes en langage machine... évidemment !!!... mais pourquoi pas ses programmes en BASIC ?

Affaire à suivre donc !!!

Marc Vandermeersch

DISK-WIFE
Ne perdez plus votre temps...

Possesseurs de KEN-DOS, fini le temps perdu en de fastidieuses manipulations lorsque vous voulez mettre de l'ordre dans vos disquettes, fini le temps perdu en cherchant un programme qui se cache sur une de vos disquettes, sans que vous puissiez le retrouver, finis les disgracieux listings de directories !

En effet, le logiciel DISK-WIFE va vous permettre de faire cela en toute simplicité et en un temps record. Ce logiciel est divisé en deux parties, la première vous permettant de manipuler un DIRECTORY comme bon vous semble, et la deuxième gérant un fichier général de vos disquettes. Le tout étant contenu dans une EPROM de 8K se logeant sur votre carte KEN-DOS. Et accompagné d'un mode d'emploi détaillé (expliquant même comment reconstruire un directory).

1. Manipulation du directory

Le corps principal de cette partie de programme est constitué d'un mini-éditeur de directory qui permet d'effectuer, sur chaque programme du dir, n'importe laquelle des fonctions du KEN-DOS par le simple appui d'une touche. Vous vous promenez dans l'éditeur à l'aide des curseurs. Vous pouvez même définir, à l'aide de deux repères, des blocs de programme devant subir la même modification. Si vous voulez, par exemple, 'closer' tous les programmes de la disquette, il suffit de pointer le premier programme du directory ainsi que le dernier, puis d'appuyer sur C et DISK-WIFE fait le reste.

D'autre part, des commandes supplémentaires ont été ajoutées, tel que le rajout ou la suppression de l'adresse de départ d'un programme en langage machine, ou du code de protection d'un programme. Et cela toujours par simple appui sur une touche (en plus de l'adresse ou du code à ajouter).

Outre cet éditeur, DISK-WIFE permet encore de faire deux choses avec le directory : en faire une belle impression sur papier (sur 4 colonnes - une par bloc - ou sur une seule colonne, mais avec tous les détails nécessaires - même les codes d'accès) ou une copie de sécurité sur les pistes 1 ou 2, qui sont inutilisées par le système.

2. Fichier des programmes

Ce fichier permet de retrouver, en moins d'une seconde, le ou les noms des disquettes contenant un programme recherché. La recherche se fait, selon le nom (ou le début de celui-ci), dans un fichier pouvant contenir jusqu'à 640 directories et 16640 noms de programme (pour les 2 fois 800 K) !!!

Les noms de programmes sont insérés automatiquement par DISK-WIFE à partir des directories, et ne requièrent donc pas de fastidieuses manipulations. Il est possible de définir une grille de sélection, de manière à n'insérer que quelques types de programmes (basic, arrays, textes...).

De plus, le contenu du fichier peut être imprimé de trois manières différentes : soit en deux colonnes, une reprenant les noms des programmes et la deuxième indiquant les noms de leurs disquette d'origines, soit en n'imprimant que les noms des programmes, et ce sur 8 colonnes et en caractères condensés, ou encore en indiquant uniquement le nom des disquettes traitées.

DISK-WIFE par Xavier Dreze et IDC Software.
Prix EPROM comprise : 1500 Fb / 225 Ff

!! An English version (and explanation) is also available !! Bestaat ook in het Nederlands !!

Commentaires au sujet des nombres présentant la propriété indiquée.

On constate que les nombres de 4 chiffres dont les puissances se terminent par les nombres eux-mêmes sont 0625 et 9376, ceux de 5 chiffres 90625 et 09376, etc...

On peut ainsi écrire indéfiniment de nouveaux chiffres à gauche des nombres obtenus.

Le nombre "infini" final satisfait donc à l'équation $X \times X = X$ puisque, en calculant l'un après l'autre les chiffres du nombre X au carré, nous allons obtenir les mêmes chiffres que dans le nombre X ...

L'équation $X \times X = X$ a donc, en plus des solutions ordinaires connues $X = 0$ et $X = 1$, deux solutions infinies $X = \dots 109376$ et $X = \dots 90625$.

Avez-vous que vous en apprenez des choses en lisant la revue DAICLIC !

JEU-CONCOURS DAICLIC No 8

Comme vous êtes très forts, voici un nouveau problème que j'espère plus difficile !

Les nombres 12 et 42 possèdent une propriété intéressante: leur produit ne change pas lorsqu'on permute leurs chiffres. En effet: $12 \times 42 = 21 \times 24 = 504$.

On demande d'écrire un programme BASIC donnant les 14 égalités des produits de 2 nombres (différents) composés chacun de 2 chiffres (différents) possédant cette propriété.

Ce programme doit éliminer les solutions triviales du genre $12 \times 21 = 21 \times 12$, ainsi que les égalités doubles (si $12 \times 42 = 21 \times 24$, il est évident que $21 \times 24 = 12 \times 42$)

Vos réponses doivent être envoyées à Jacques MOENS Clos Fontaine des Ducs, 6 à 1310 La Hulpe (Belgique). Le programme Basic trouvant le plus rapidement la bonne réponse vaudra à son auteur de recevoir une cassette remplie de programmes au choix !

POST-SCRIPTUM

Je signale aux amateurs de ce genre de problèmes qui possèdent un modem que des casse-têtes divers sont proposés sur MICRONET 2. Bon amusement !

DAIC

ADDENDUM au JEU-CONCOURS du DAICLIC No 5

Une réponse nous est parvenue trop tard que pour être publiée dans le numéro précédent. Elle vaut cependant la peine d'être signalée, car elle donne la réponse en moins d'une seconde. Comme ce programme (en langage machine) est trop long que pour être publié ici, il sera envoyé gratuitement à tout lecteur intéressé qui en fera la demande. Il a été écrit par Paul GOBERT de Zoersel (Belgique), qui a tout de même reçu une cassette de programmes pour son effort.

D'autre part nous avons reçu simultanément les réponses aux Jeux 5 et 6 de Jean GUERARD de Meudon (France), (autant en emporte le poste...!) Comme son programme trouve la solution en 1,32 sec et est très court, nous vous le donnons ci-dessous:

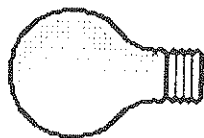
```
IMP FPT
1 D=1/2048:E=D*2:FOR M=9 TO 2 STEP -1:K=(1000*M-1)/(1000-M)
2 FOR JAC=102 TO 987/K:IF FRAC(K*TAC+D)>E THEN NEXT:END
3 PRINT TAC,K*TAC+D
```

REPONSE au JEU-CONCOURS du DAICLIC No 6

Les nombres de 3 chiffres demandés sont 376 et 625. Deux programmes BASIC trouvent la solution en moins d'une seconde: celui de Paul GOBERT et celui de Jean GUERARD (encore eux)!

```
Programme de Paul GOBERT (800 msec) :
IMP INT
1 POKE #1BE,#FF
10 FOR I=0 TO 9:IF I*1 MOD 10<>I GOTO 40:
FOR J=1 TO 1+90 STEP 10:IF J*J MOD 100<>J GOTO 30:
FOR K=J TO J+900 STEP 100:IF K*K MOD 1000<>K THEN NEXT K:
GO TO 30
15 IF K<100 GOTO 30
20 PRINT K,;NEXT K
40 NEXT J
50 PRINT (#FF-PEEK(#1BE))*20;" msec"

Programme de Jean GUERARD (440 msec) :
IMP INT
1 A=S:GOSUB 10:A=6:GOSUB 10:END
10 N=1 : K=10 : L=100
11 FOR X=1 TO 9:B=A+K*X:C=B*B:D=C-C/L:*L
12 IF D<>8 THEN NEXT:END
13 A=B: IF N=2 THEN PRINT A:RETURN
14 N=N+1 : K=K*10 : L=L*10 :GOTO 11
```



Jean GUERARD a également reçu une cassette de programmes au choix.

DCA HARD

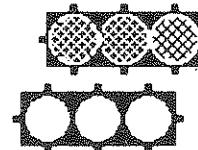
Certains d'entre vous nous ont demandé le prix des interfaces réalisées au club. Vous les trouverez ci-dessous. Ils vous paraîtront peut-être élevés, mais les prix que nous vous avons communiqués précédemment n'étaient qu'indicatifs et ils nous ont apparu, après consultation de divers détaillants sur Paris, que la réalité dépassait une fois de plus l'estimation. Aussi, si vous pouvez obtenir des prix avantageux sur tel ou tel composant, n'hésitez pas à nous en informer.

Les cartes vous sont proposées sous deux formes. La première, ou version kit (VK), comprend le circuit imprimé percé, les composants, et la notice de montage. La deuxième, ou version intégrale (VI), comprend la carte entièrement montée et testée. Pour certaines cartes nécessitant un minimum d'habileté dans la soudure ou de connaissances électroniques, nous vous conseillons fortement la version intégrale (dans ce cas, la carte est signalée par ***). A noter qu'aucune carte n'est livrée avec boîtier, certaines cartes pouvant en effet se regrouper ensemble. Enfin, deux autres paramètres définissent le prix: si vous êtes membres de DCA (prix club dans ce cas) et si vous habitez la France ou non.

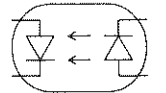
ANANUM: Faites l'acquisition de 8 données numériques et 8 données analogiques continues avec cette carte, début d'une série d'extensions du bus DCE. Remarque: ***

Prix club France: 460 FF (VK), 470 FF (VI)
 Prix club Europe: 475 FF (VK), 485 FF (VI)
 Prix France: 495 FF (VK), 505 FF (VI)
 Prix Europe: 510 FF (VK), 520 FF (VI)

DAISPOT: Vous reviez de vous faire un chenillard pro, ou de commander jusqu'à 8 canaux 220 V simultanés (jusqu'à 24 avec 3 cartes). C'est chose faite à présent avec cette carte entièrement isolée du secteur.



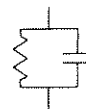
Prix club France: 350 FF (VK), 360 FF (VI)
 Prix club Europe: 365 FF (VK), 375 FF (VI)
 Prix France: 385 FF (VK), 395 FF (VI)
 Prix Europe: 400 FF (VK), 410 FF (VI)



DAIDEV: Dans la lignée d'ANANUM, voici une carte qui vous permettra de développer votre propre application compatible avec les ordres INP et OUT du Dai. Attention, la carte d'étude n'est pas fournie ainsi que le troisième connecteur, les cartes d'étude existant sous tellement de modèles...

Prix club France: 187 FF (VK), 197 FF (VI)
 Prix club Europe: 202 FF (VK), 212 FF (VI)
 Prix France: 206 FF (VK), 216 FF (VI)
 Prix Europe: 221 FF (VK), 231 FF (VI)

CABLE: Faire des cartes c'est bien, mais les relier au Dai c'est mieux! Voici un câble (1 m) pour ANANUM, DAIDEV et compagnie!



Prix club France: 88 FF (VI)
 Prix club Europe: 94 FF (VI)
 Prix France: 96 FF (VI)
 Prix Europe: 102 FF (VI)

DCA hard
 c/o Philippe Casier
 31ter rue de Paris
 92190 MEUDON
 FRANCE

Reglages KEN-DOS

ASSEMBLER EN 8080 DCA

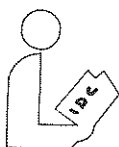
C'est le nom d'un ouvrage de 67 pages, en majeure partie en français (les appendices sont en anglais), qui initie les possesseurs d'un DAI à l'assembleur et au langage machine. Ce livre est consacré au DAI, il peut vous apporter les bases du langage machine tout en souplesse.

Vous pourriez être sceptique sur l'utilisation de ce livre si vous connaissez les ouvrages consacrés au 8080. Cependant, ce livre a l'avantage d'être écrit dans un langage simple. Les registres, pointeurs, instructions vous seront développés un à un avec un exemple simple et qui se rapporte à la logique du DAI.

Si vous en avez assez du basic et que vous voulez faire vos dents sur l'assembleur et sa rapidité, n'hésitez plus à commander cet ouvrage qui deviendra votre livre de chevet et un outil de travail, car sa simplicité, ses trucs et ses adresses en ROM lui sont des atouts indiscutables.

Pour tout renseignement ou commande écrire à :

DCA service documentation
c/o Pascal Hertzog
2, avenue Jean Moulin
94350 VILLIERS SUR MARNE
FRANCE



REGLAGES KEN-DOS IDC BORDEAUX



Ils sont très simples :

Avec le potentiomètre de 10 K Ohms, régler la tension à mesurer sur la pin 17 de l'IC 'C' entre 5,6 et 5,8 Volts.

Avec le potentiomètre de 100 K Ohms, régler la tension à mesurer sur la pin 2 de l'IC 'F' entre 1,5 et 1,6 Volts.

Avec le potentiomètre de 50 K Ohms, régler la fréquence à mesurer sur la pin 7 de l'IC 'F' à 2 Méga Hertz.

Si vous n'avez pas d'oscilloscope, vous pouvez essayer la méthode suivante :

Régler à mi-course le potar et taper l'instruction 'FORMAT'. Il doit afficher des 0 à cadence régulière (un environ toutes les secondes).

Sinon, bouger un peu le réglage et recommencer.

Après un bon FORMAT, taper ensuite VERIFY et le checksum des 80 pistes doit s'afficher à une cadence régulière, environ un toutes les secondes (sous réserve de posséder une disquette de bonne qualité).

Sinon, retoucher à peine votre réglage et recommencer le VERIFY.

Bien sur, se procurer dès que possible un oscillo...

Minitel part 3

DAI et MINITEL

3me

Par I.D.C. Bordeaux

le 14 / 04 / 86

Avant tout il est très important pour nous, vu l'ampleur de ces articles que vous nous envoyez un petit mot d'encouragement ou de protestation pour nous faire connaître :

- * votre intérêt dans cette lecture
- * votre aptitude à suivre nos élucubrations car nous sommes conscients du caractère parfois abscons de cette prose. (sur laquelle on peut aussi s'asseoir)

Passons à la pratique :

En effet à l'heure où est rédigé cet article notre émulateur MINITEL est fin prêt (c.f. la page d'information de I.D.C. Bordeaux de DAICLIC 7) aussi les différents labels mentionnés dans cet article correspondent à ceux réellement utilisés dans le source S.P.L. que nous tenons à votre disposition. La lecture de nos articles est quasi indispensable car le source n'a pu être commenté (il est tellement 'balaise' que les commentaires ne tiendraient pas dans le buffer S.P.L.)

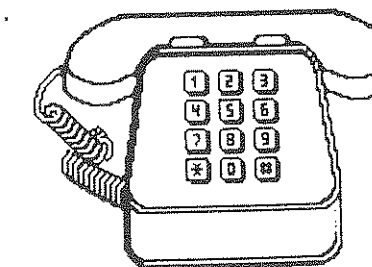
I / STRUCTURE DU PROGRAMME :

Il comprend 3 sous parties principales = 3 sous programmes différents fonctionnant simultanément grâce à la gestion de deux interruptions.

1/ *Communication* : grâce au détournement de l'interruption 4 (se reporter à la lecture de cet article qui paraîtra dès qu'il y aura suffisamment de place dans DAICLIC ...)

2/ *Clignoti - Clignota* : grâce au détournement de l'interruption 7, ce sous programme gère les différents types de clignotement pour les caractères et le curseur.

3/ *Affichage* : c'est la partie développée cette fois.



II / AFFICHAGE :

1/ *Ecran fictif - Ecran graphique* :

Ce sont deux zones de travail, deux portions de la mémoire vive du DAI, qui sont gérées séparément :

* L'écran graphique est l'écran réel, la portion de RAM du DAI correspondant au MODE 5, visible sur l'écran de votre moniteur, miroir du MINITEL.

* L'écran fictif est une portion de RAM de 3Kilo Octets, reflet codé de l'écran du MINITEL, stockant *in extenso* tous les caractères affichés ainsi que leurs attributs. (pour l'explication du calcul des 3 Ko. c.f. le premier article sur le MINITEL dans DAICLIC 4 p31 en bas ... = DAI et MINITEL 1)

a/ Rôle de l'écran fictif :

Il gère tout ce qui a été décrit dans notre 2me article (normalement DAICLIC 6 ! = DAI et MINITEL 2) propagation des attributs de groupe, espaces validants ... Ceci permet de s'affranchir de la notion d'attributs de groupe, afin de simplifier la traduction de cet écran fictif en écran graphique : chaque caractère contient en lui même la totalité de ses attributs, individuels et de groupe, ce qui implique une taille mémoire, non plus de 2 Kilo Octets comme dans le MINITEL, mais de 3 Ko.

b/ Codage utilisé pour créer cet écran fictif :

Nous savons que 3 Octets sont nécessaires pour coder tous les paramètres d'un caractère (c.f. DAI et MINITEL 1)

1er. OCTET :

Bit 0 à Bit 6 ; ils peuvent contenir 4 sortes de codage :

- pour un caractère normal VIDEOTEX : ils contiennent le code videotex de ce caractère (de #20H à #7FH).
- pour un caractère espace validant ils contiendront le code #18H.
- pour un 'trou' ce sera le code 00
- pour une case contiguë à un caractère double hauteur et/ou double largeur ça se complique un peu :

Un caractère double largeur s'étend sur deux cases contiguës sur l'écran graphique, chaque case de l'écran graphique correspond à trois octets mémoires dans le buffer 'écran fictif', donc 6 positions mémoires contiguës serviront pour un caractère double largeur, les trois premières contenant les caractéristiques attribuées à ce caractère et les trois suivantes contenant un code spécial (= #08H) juste pour préciser que cette place est occupée par un caractère double largeur, sans être autrement traduites sur l'écran graphique

Position : 1er Octet, 2me Octet, 3me Octet; 1er Oc. Bis, 2me Oc. Bis, 3me Oc. Bis.
Contenu : X X X #08 #08 #08
(X,X,X) : codent le caractère à afficher.

De même pour un caractère double hauteur ce sont les trois octets mémoire de l'écran fictif correspondant à la case de l'écran graphique, juste au dessus de la case contenant les codes du caractère à afficher, qui contiendront le code #08H (DUF!) :

(X,X,X) : codent le caractère à afficher

Position : Contenu :
1er Oc. Bis, 2me Oc. Bis, 3me Oc. Bis = #08, #08, #08 Case d'écran supérieure
1er Octet, 2me Octet, 3me Octet = X, X, X Case d'écran inférieure

Bit 7 : caractère clignotant (1) ou non (0)

2me OCTET :

Bit 0 à Bit 2 : Couleur du Caractère en code MINITEL

Bit 3 à Bit 4 : Jeu de caractères utilisé :
00H = G0
08H = G1
10H = G2
18H = Spécial :

On profite que reste libre la position 18H pour l'utiliser afin de préciser la présence soit d'un espace validant soit un trou soit un caractère 'contiguë' (redondance !)

Bit 5 : Double Largeur ou Non
Bit 6 : Double Hauteur ou Non
Bit 7 : Ne code Rien !



3me OCTET :

Bit 0 à Bit 3 : Couleur du Fond en code DAI
Bit 4 : Lignage ou Non
Bit 5 : Ne code Rien !
Bit 6 : Inversion Video ou Non
Bit 7 : Masquage ou Non

Cette méthode permet de séparer complètement les différents problèmes succités d'une part par le codage des différents attributs propres à un caractère (partie réglée par l'écran fictif que nous verrons en détail une prochaine fois) et d'autre part la gestion de l'écran graphique (que nous allons voir aujourd'hui) qui sera le simple reflet de l'écran fictif.

III/ ECRAN GRAPHIQUE :

Cette gestion est faite dans notre programme par la routine GNLEG i.e. général écran graphique.

1/ Localiser le caractère :

Il faut le localiser à la fois dans l'écran fictif et dans l'écran graphique. Par convention HL pointera la position dans l'E.graphique et OE la position dans l'E.fictif du caractère à afficher. Ces deux pointeurs sont dupliqués en RAM dans les variables POIGRA i.e. pointeur écran graphique et POIFIC i.e. pointeur écran fictif

Le calcul de ces adresses se fait par un sous programme à voir plus tard.

2/ Recherche de la matrice à POKER :

Une fois ces adresses respectives trouvées, il faut POKER sur l'écran graphique une matrice correspondant au caractère à afficher, cette matrice étant contenue dans une table différente pour chacun des trois jeu de caractères possibles.

Que longue et fastidieuse fut la frappe de cette table de matrices ! ciel, que de sacrifices dans ce monde d'ingratitude.

La routine générale de recherche de matrice quel que soit le jeu de caractères utilisé à pour label MATGNL i.e. matrice général.

Son utilisation :

A l'entrée : DE pointe l'écran fictif.

En sortie : BC contient l'adresse de la matrice à poker

Pour GO :

La table des matrices débute au label GO (si, si !!)

Comment trouver la bonne matrice dans GO, tel est le rôle de MATGO i.e. matrice GO utilisant la formule suivante :

Etant donné que la longueur d'une matrice est de 10 lignes (paramétrée cependant par MATY afin de varier facilement pour d'autres utilisations ...) on décide d'appeler le code Videotex à afficher : CODE on aboutit à :

ADRESSE = (CODE - #1F) * MATY + GO - 1

Pour G1 :
Il faut savoir que la correspondance CODE <=> matrice du semi graphique est la suivante :

Bit 0	Bit 1	
Bit 2	Bit 3	
Bit 4	Bit 6	Attention pas Bit 5 car ,

nous savons que les codes doivent être >= à #20H donc si le Bit 6 est à 0 alors il faut et il suffit que le Bit 5 (qui n'influe pas sur la forme du graphisme) soit mis à 1 pour que le code ait la valeur nécessaire. Et si Bit 6 = 1 alors le Bit 5 est indifférent.

Puis tout dépend si le caractère à afficher est ligné ou non :

* Pour un caractère **non ligné** le début de la table des matrices se trouve au label G1 (pas possible !)

* Pour un caractère **ligné** le début de la table des matrices = label G1P

Il suffit donc de tester si le caractère est ligné ou Non et on en déduit G1 ou G1P.

Si ce caractère à un code >= #60H (Bit 6 et Bit 5 à 1) alors on met le Bit 5 à 0 ce qui équivaut à faire -#20H.

Enfin la recherche de la matrice se calcule comme pour G0.

Pour G2 :

Le codage est interne au programme et n'a pas de rapport avec les codes MINITEL de G2. Exemple : le code de la livre anglaise est #21H, celui de la flèche curseur gauche est #22H, flèche curseur droite = #23H etc ...

Cette table contient aussi les caractères accentués et non plus le seul code de l'accent comme dans le MINITEL et aussi la matrice de l'erreur de transmission - point d'interrogation à l'envers.

Le début de la table commence au label !!! G2 !!!

3/ Couleur de cette matrice :

Dans l'écran fictif on appelle F la couleur codée comme couleur de fond (contenue dans le 3me Octet) (le cas des caractères clignotants sera traité dans un chapitre à part).

On appelle C la couleur codée comme couleur de caractère.

COLORE est la routine qui va déterminer la couleur de fond 'réelle' et la couleur de caractère 'réelle' (pour les pères hères qui croyaient que la couleur de fond était la couleur de fond!) et va les poker respectivement dans : COULFO i.e. couleur de fond et COULCA i.e. couleur de caractère.

Expliquons un peu :

Plusieurs tests sont faits successivement :

a/ Masquage :

S'il y a masquage du caractère alors le caractère est remplacé par un espace noir F=0 et C=0

b/ Inversion Video :

Uniquement pour les caractères des jeux G0 et G2, puisque les semi-graphiques ne sont pas atteints par une inversion video.

Dans ce cas F et C sont permutés, la couleur du fond devenant celle du caractère et inversement.

c/ Clignotement :

Ce test, portant sur la variable FCL sera développé plus tard.

Alors, à l'issue de ces trois tests, la valeur (qui n'attend pas le nombre des damnés hells) contenue dans F est celle de la couleur du FOND et dans C celle de la couleur du caractère.

*** CEPENDANT

Pour les caractères double hauteur, le haut du caractère peut avoir une couleur différente de celle du bas. Il faut donc refaire ces tests pour le haut du caractère et poker les résultats obtenus pour ce deuxième calcul dans COULFH i.e. couleur du fond haut et COULCH i.e. couleur du caractère haut.

4/ Pokage sur l'écran :

Puisque nous connaissons à présent :

- * la position
- * la matrice
- * les couleurs

(que ceux qui ont cru suivre le *distingo entre colora simplex et colora veritas* ne sortent pas de suite car ils vont encore avoir à affronter un terrible ennemi bien connu en *MODE 5 : la bavure*)

Cette bavure, décrite par Mr. BOUCHERON dans DAICLIC 4 p65 et dont l'explication va paraître dans un article séparé, nous oblige, afin de ne pas voir nos magnifiques matrices dégouliner lamentablement les unes sur les autres, à certaines précautions :

REGLE :

Rappelons qu'en *MODE 5* un point est codé dans un barreau de 8 points consécutifs, occupant 2 Octets en RAM video.

Le 1er. code la (COULEUR1 * #10) + COULEUR2 et le 2me code les points de COULEUR1 (= Bit à 1) et de COULEUR2 (= Bit à 0).

Pour éviter la bavure il faut et il suffit de commencer le 2me Octet (barreau de 8 points) par un 1 !.



DONC :

Il s'avère nécessaire, pour commencer un Octet par un 1 alors que l'Octet correspondant de notre matrice à Poker commence par un 0, de faire une 'inversion video' de notre ligne de matrice considérée, ainsi que d'inverser simultanément les couleurs de Fond et de Caractère.

(ce qui semble revenir au même à vos yeux Oh pères mortels !)

On fait de même pour chacune des 10 lignes de notre matrice à afficher. A la fin de ce saucissonnage la matrice paraît unie alors qu'elle n'a pas cessé d'être affichée avec ou sans inversion des couleurs selon la valeur du 1er Bit de la ligne à afficher.

Label où se trouve cette routine = POKCOG i.e. Pokage de la Couleur Genial.

Que ceux qui suivent encore nous envoient un télégramme de soutien pour nous assister dans notre rude tâche (et pour nous rassurer sur le taux d'audience de ces explications !).

POKCOG : utilisation

A l'entrée, dans la pile se trouvent BC et PSW dont A représente la ligne de 8 points de la matrice originale à poker, dont sera testé le bit de poids le plus fort, qui, s'il est égal à 0 déclenchera le processus d'inversion vidéo. 'B' contient la couleur réelle du Fond et 'C' la couleur réelle du Caractère.

Il faut considérer 4 sortes de pokage différentes pour cette matrice, selon la taille du caractère à afficher :

*Taille Standard : routine POKSTA

*Double Largeur : routine POKDL qui fait appel au sous programme REPADE i.e. répète l'Octet contenu dans 'A' en deux Octets

*Double Hauteur : routine POKDH qui pike 20 lignes graphiques. Pour les 10 lignes du bas du caractère, les couleurs seront celles des variables COULCA et COULFO et pour les 10 lignes du haut : COULCH et COULFH.

TESTHA (i.e. Test de Hauteur), pour cela, utilise 'H' comme pointeur de ligne de la matrice (varie de 1 à 10 D), puis met à #FFH le flag nommé FHAUT i.e. Flag Haut si la moitié du caractère à piker est atteinte ('H' = 5).

Alors POKCOL, sous programme général qui inclu POKCOG, sélectionne les bons couples de variables parmi les deux couples possibles : {COULCA, COULFO} et {COULCH, COULFH}; selon la valeur de FHAUT.

*Double Taille : combine les 2 routines Double Hauteur et Double Largeur.

POKGNL : rassemble ces sous routines pour effectuer la seule que réclame la taille du caractère à afficher.

A l'entrée : 'BC' contient l'adresse de la routine de la matrice à piker.
'DE' contient l'adresse du caractère dans l'écran fictif.
'HL' " " " " " l'écran graphique.

5/ Soulignement :

Pour G0 et G2, une fois les caractères pokés, il faut parfois les souligner (attribut de lignage), ce qui se résume à piker une ligne sous le caractère. Tous les tests (jeu de caractère, taille, lignage..) et toutes les opérations nécessaires à cette fonction sont réalisés par la routine SOULIG.

SOULIG :

A l'entrée : 'DE' contient l'adresse en écran fictif.
'HL' " " " " " écran graphique.

6/ Résumé :

Toutes ces opérations, séparées en sous programmes, sont appelées par la routine GNLEG i.e. Général Ecran Graphique.

GNLEG :

A l'entrée : 'DE' contient l'adresse en écran fictif.
'HL' " " " " " écran graphique.

Et un appel à GNLEG va donc dessiner le caractère voulu sur l'écran graphique.

IV/ CONCLUSION :

1/ Le compte de l'écran graphique est définitivement réglé, sauf pour le cas particulier des clignotements (caractères et curseur).

2/ Il reste à voir

a/La gestion de l'écran fictif i.e. la gestion des attributs.
cette seule partie prendra au moins 1 ou 2 articles ...

b/La réception des codes MINITEL et leur traduction :

*caractères visualisables

*caractères de contrôle (CODE >= #20H)

*attributs et protocoles

ceci aussi prendra une certaine place ...

AUSSI BON COURAGE !

Sébastien DUBOURG et Bruno DELANNAY de I.D.C.Bordeaux.

= EUROPEAN MICRO MEETING (EMIM) =

SAMEDI 25 OCTOBRE 1986

=====

=====

Le samedi 25 octobre 1986 (de 9h30 à 19 h00) à Nivelles (Belgique), aura lieu l'European Micro Meeting.

Tous les passionnés de micro-informatiques sont cordialement invités à cette manifestation exceptionnelle. Le DAI P.C. occupera toujours la place d'honneur. Les principaux clubs: IDC, IDC Bordeaux, IDC Bruxelles, IDC Charleroi, IDC Liège, IDC Paris et DAINamic seront représentés et présenteront leurs services spécifiques.

Le programme de cette journée sera fonction de chacune des associations représentées. Elles présenteront l'éventail de leurs activités, c'est ainsi que vous verrez des Amiga, Atari ST, MSX, PC-MS/DOS, ... cotoyer le DAI.

Un rendez-vous à ne manquer sous aucun prétexte.

COMMENT SE RENDRE AU MEETING ?

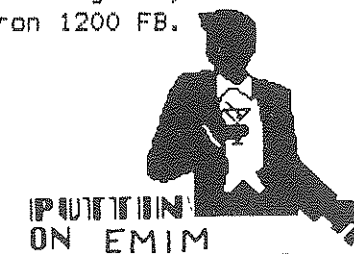
Autoroute PARIS-BRUXELLES, entre Charleroi et Bruxelles, sortie Nivelles Sud (sortie 19 dans la direction de Bruxelles, sortie 19 BIS dans la direction de Paris).

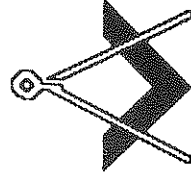
L'endroit : Motel Nivelles Sud situé à la sortie de l'autoroute, à 15 minutes de la gare. Madame pourra faire son shopping dans le centre commercial en face du Motel ou dans le centre commerçant de la ville (à 7 minutes), ou encore sur sur le marché (le matin, dans le centre à 5 minutes). Aucune excuse pour ne pas venir !

Un parking de 400 places est à votre disposition. Pour les étrangers, il est possible de loger sur place, le prix des chambres est d'environ 1200 FB.

ADRESSE :

EUROPEAN MICRO MEETING (EMIM)
MOTEL RESTAURANT "NIVELLES SUD"
Chaussée de Mons, 22
1400 Nivelles - BELGIQUE
Téléphone : (0)67 / 21.87.21





DAIDEV

La carte DAIDEV, comme son nom l'indique, donne la possibilité à tout daïste de développer sa propre application. Elle dispose de 3 fois 8 bits programmables bi-directionnels et accepte l'interruption extérieure du DAI. De plus elle est compatible avec la carte ANANUM déjà développée par le club.

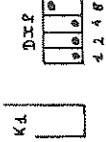
Deux parties la composent : A et B.

La partie A, la plus complexe, n'est autre qu'une carte à pastilles au pas de 2,54 de 10 cm x 16 cm (format Europe simple) sur laquelle est soudée un connecteur 64 broches femelle lui aussi faisant partie du standard Europe (référence C133-714A-64P très exactement !).

Pour la partie B, l'interface et décodeur d'adresse, il faut simplement de la patience et travailler avec soins. Utiliser un support pour le 8255. Le circuit imprimé étant un double faces à trous non métallisés, la référence des DIP à la masse ne doit pas être omise, pour cela utiliser un petit fil (ou queue de résistance) qui reliera les deux faces du circuit imprimé. Une fois le circuit réalisé et avant de placer les circuits intégrés, brancher le + d'une pile sur la broche 32 et le - sur la broche 29 du connecteur K1, placer ensuite le cavalier CA sur 5VDAI. Avec un voltmètre, contrôler la tension sur les pattes 7 (0v) et 14 (tension pile) du 8255. Si une tension incorrecte apparaît, elle vient d'un court-circuit probable : vérifiez alors votre carte. Si tout est ok vous pouvez maintenant placer les circuits intégrés (attention au sens !!!).

Pour relier le DAI à la carte (comme d'ailleurs à toutes les autres qui suivront) nous utilisons un câble plat. A une extrémité, nous avons un connecteur femelle 34 broches (même type que celui du DCA) à sertir auto-dénudant, et à l'autre extrémité, soudé fil à fil, un connecteur femelle 64 broches type Europe à wrapper. Pour souder, utilisez le plan du connecteur K1.

L'utilisation de la carte peut être faite sous basic avec les instructions INP et OUT. Pour l'exemple placer les interrupteurs DIP dans la position ci-dessous (on donne ainsi le numéro 8 à la carte).



Valeur sur la carte 0
adresse du port A #80
adresse du port B #81
adresse du port C #82
adresse logique de commande #83

Pour sortir des valeurs sur les ports A, B, C.
OUT #83,#80 place les ports en sortie.
OUT #80,#01 seul le bit 1 du port A est mis à 1
OUT #82,#FF le port C en entier est mis à 1
OUT #81,#0F les bits 1, 2, 3, 4 du port B sont à 1

Tout en exécutant le programme, contrôlez sur le connecteur K2 l'état des bits par rapport à la masse au moyen d'un voltmètre.

Pour entrer des valeurs sur la carte
OUT #83,#9B place les ports A, B, C en entrées
X%-INP(#80) copie le contenu du port A dans X

La méthode est identique pour les autres ports. Pour ceux d'entre vous qui aurais développé des cartes sur ce principe, n'hésitez pas à en faire profiter les autres par l'intermédiaire du club.

Enfin sachez que DAIDEV est disponible au club au prix de 260 FF en VK et 270 FF en VI + \$ ou \$\$\$ pour frais d'envoi et - 10 % si vous êtes membre du club. Vous trouverez aussi le câble de liaison (1m) pour 90 FF + \$ ou \$\$\$.

Pour D.C.A. R. Mayer & G. Dauchez

Nomenclature

Résistances
R1, à R6 : couche de carbone 220 kilo-ohms 1/4 de watt
R7 : couche de carbone 1 mégohms 1/4 de watt

Condensateur
C : céromique 20 nano farads 16 volts

Semi-conducteurs
T : NPN de faible puissance BCS49 ou équivalent
CI1 : CD4070
CI2 : CD4068
CI3 : 8255

Connectique
K1 : connecteur Europe 64 broches mâles
K2 : connecteur Europe 64 broches femelles à wrapper (à fin de pouvoir le couder)
K3 : connecteur Europe 64 broches mâles

Divers
2 supports 14 broches
1 support 40 broches
1 interdil 4 dips
1 cavalier + 3 picots (ou micro-inverseur pour circuit imprimé)
1 circuit imprimé double face 100 x 65
1 carte d'étude 100 x 160

Routine INP et OUT

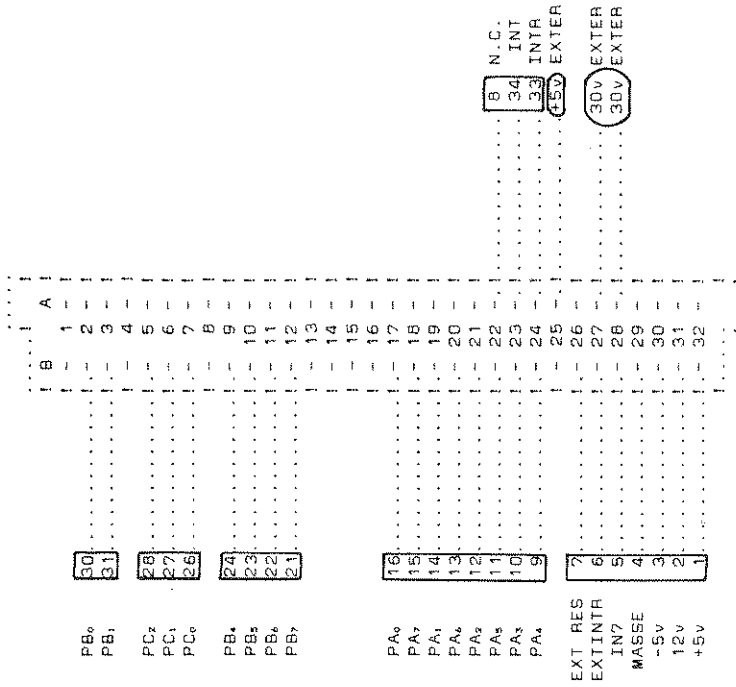
```
*****
*ECRITURE DU BUS DCE*
*****
écrit un octet à une adresse de la structure DCE
entrées : adresse en D
          donnée en E

D8C8 F5 RWOP PUSH PSW
D8C9 E5 PUSH H
D8CA 2103FE LXI H, :FE03 adresse du mot de controle
D8CB 3680 MVI M, :80 tous les ports en sortie
D8CC 28 DCX H port C adressé en HL
D8CD 36FE MVI M, :FE RAZ du signal BUS EXP
D8CE EB XCHG données en L, adresse en H
D8CF 2200FE SHLD : FE00 données en Pa, adres. en Pa
D8D0 EB XCHG adresse port C en HL
D8D1 34 INR M génère le signal BUS EXP
D8D2 36FD MVI M, :FD génère le signal d'écriture
D8D3 36FF MVI M, :FF RAZ du signal d'écriture
D8D4 35 DCR M RAZ du signal BUS EXP
D8D5 E1 POP H
D8D6 F1 POP PSW
D8D7 C9 RET

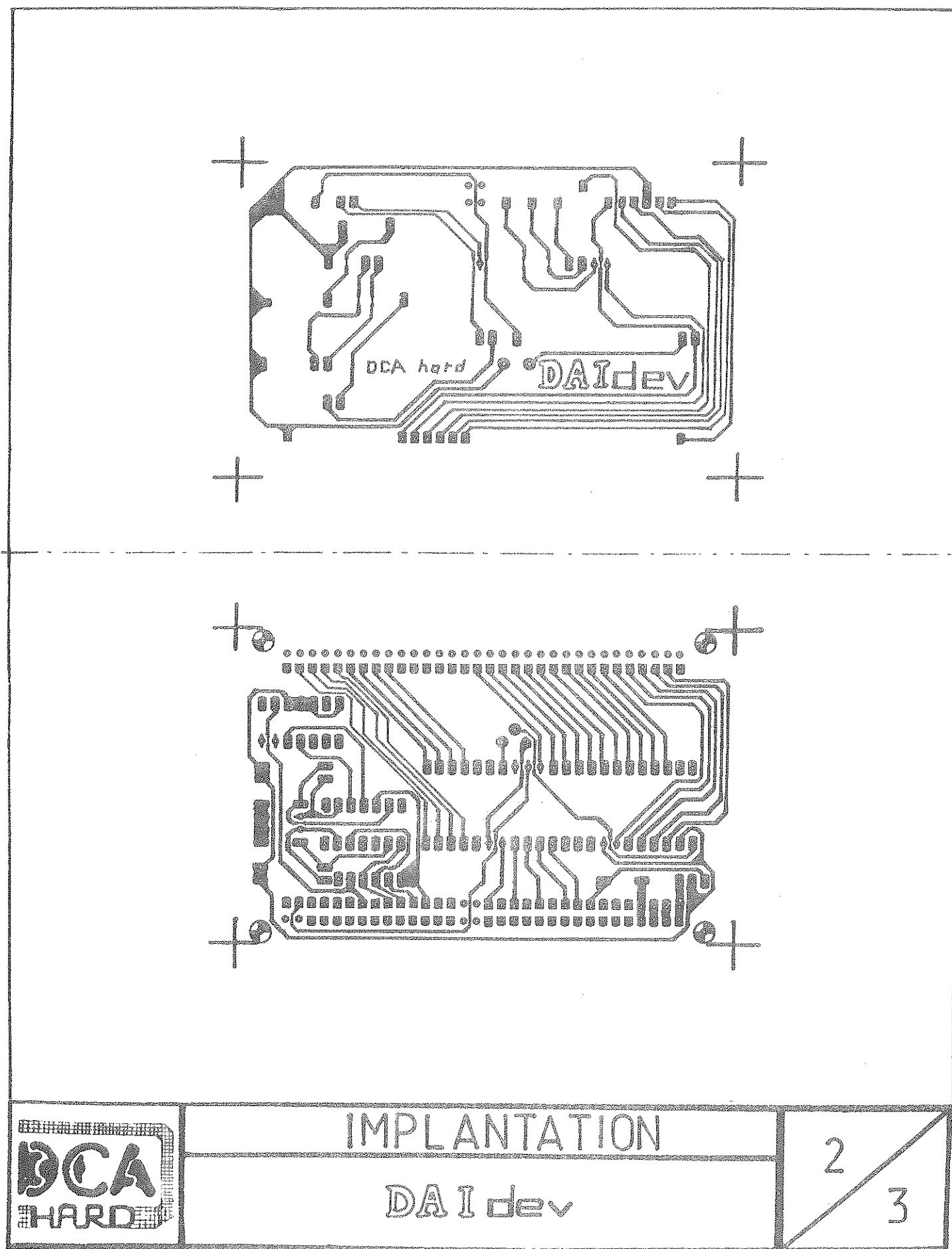
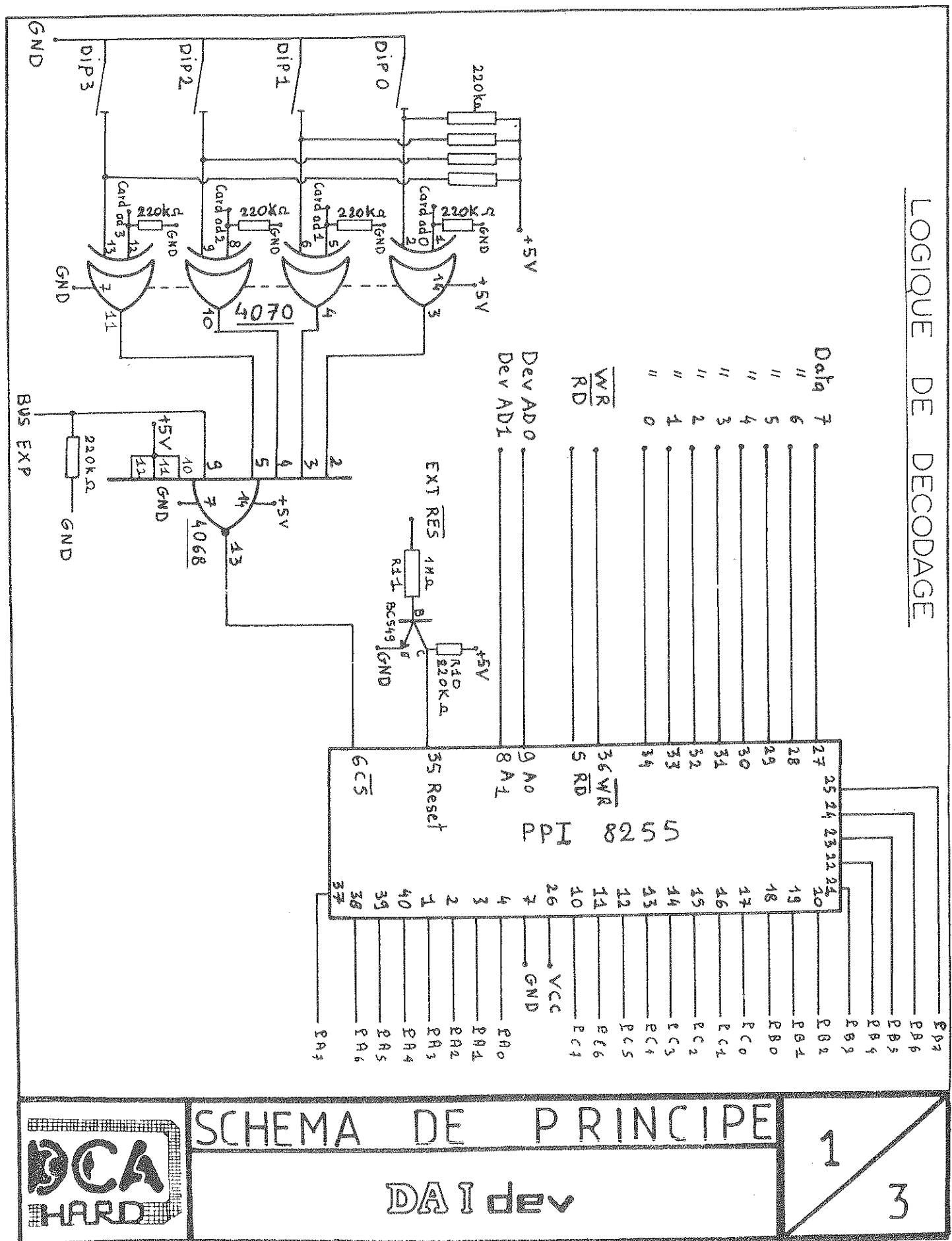
*****
*LECTURE DU BUS DCE*
*****
lit un octet à une adresse de la structure DCE
entrée : adresse en D
sortie : donnée en E

D8E0 F5 RWIP PUSH PSW
D8E1 E5 PUSH H
D8E2 2103FE LXI H, :FE03 adresse du mot de controle
D8E3 3690 MVI M, :90 Pa en entrée, le reste en S
D8E4 28 DCX H Pc adressé en HL
D8E5 36FE MVI M, :FE RAZ du signal BUS EXP
D8E6 7A MOV A,D adresse en A
D8E7 3201FE STA :FE01 met adresse en Pa BUS EXP
D8E8 34 INR M génère le signal de lecture
D8E9 36FB MVI M, :FB génère le signal de lecture
D8EA 3A00FE LDA :FE00 (échange de données effec.)
D8EB 5F MOV E,A données vers A
D8EC 36FF MVI M, :FF données dans E
D8ED 35 DCR M RAZ du signal de lecture
D8EE E1 POP H
D8EF F1 POP PSW
D8F0 C9 RET
D8F1 C9 RET
D8F2 C9 RET
D8F3 C9 RET
D8F4 C9 RET
D8F5 C9 RET
D8F6 C9 RET
D8F7 C9 RET
D8F8 E1 POP H
D8F9 F1 POP PSW
D8FA C9 RET
D8FB C9 RET
D8FC C9 RET
D8FD C9 RET
D8FE C9 RET
D8FF C9 RET
```

Brochage connecteur K1



Une partie seulement des bits du bus DCE est utilisé.
Correspondance signaux DCE-sigaux carte
PA0 à PA7 <=> Data 0 à Data 7
PB0 <=> Dev AD0
PB1 <=> Dev AD1
PB2 <=> Card ad0
PB3 <=> Card ad1
PB4 <=> Card ad2
PB5 <=> Card ad3
PC0 <=> BUS EXP
PC1 <=> WR
PC2 <=> RD



CALCUL MATRICIEL

Maurice DECUYPER, CHAUDFONTAINE (B)

Ce programme peut effectuer des travaux sur des systèmes linéaires (matrice):

Calcul de déterminant, Inversion de matrice, Résolution de systèmes

Il utilise pour cela l'algorithme de Banachiewicz qui est assez peu connu et qu'il serait fastidieux d'étudier ici. Retenons simplement que cette méthode passe par une matrice intermédiaire qui est la même quel que soit le but à atteindre. Le programme élabore cette matrice au cours d'une première phase (phase progressive). C'est au cours de cette première phase que les équations redondantes sont détectées et éliminées par exemple. La phase suivante diffère selon le travail demandé. La matrice intermédiaire peut être affichée mais elle ne présente un réel intérêt que pour ceux qui connaissent l'algorithme. Toutefois, on peut signaler que lors de la recherche du déterminant (non nul !) d'une matrice (carrée), il suffit de faire le produit des pivots (éléments dont le numéro de ligne = le numéro de colonne), la réponse obtenue ainsi ne sera cependant exacte qu'au signe près. Mais de toutes manières, la réponse finale donnée par le programme sera, elle, exacte (je l'espère...) y compris le signe. Lorsque le déterminant est nul, le système est incompatible... le programme n'élabore pas la matrice intermédiaire jusqu'au bout et on ne pourra dès lors pas l'afficher.

On peut introduire n'importe quel système d'équations linéaires qu'il soit indéterminé, redondant (trop d'équations pour le nombre d'inconnues) ou incompatible, le programme est censé pouvoir le traiter. S'il détecte des équations redondantes, il les élimine en le signalant. Si le système obtenu après transformation est indéterminé, le programme donnera un système réduit en utilisant les dernières variables comme pseudo-constantes et toutes les autres seront donc exprimées en fonction de ces pseudo-constantes.

Le programme est prévu pour utiliser les drives **PRODATA** avec le **DOS 3.0**. On peut modifier la destination de la matrice (numéro du drive utilisé) et le nom du fichier pour la matrice à la ligne 230. La matrice est donc sauvegardée afin de permettre un autre calcul avec éventuellement des modifications des dimensions de la matrice et d'éléments particuliers de cette matrice.

Il est obligatoire de définir une précision de travail pour éviter les erreurs introduites par les arrondis. Tout élément introduit ou calculé (matrice intermédiaire) inférieur à la valeur définie sera simplement ramené à ZERO. En général, j'emploie 10 E-5 (on peut taper 0).

Le programme n'est "limité" que par les **DIM** et donc par la mémoire du **DAI**. On peut imaginer de travailler avec des matrices de 1000 x 1000 ! mais il faudrait un peu plus de place disponible. On s'arrêtera donc pour l'instant à des matrices de 50 x 50.

Cela représente déjà un travail considérable lors de l'introduction ! Si le 232^e élément que vous venez d'introduire est erroné: pis de panaque (!), notez la ligne et la colonne de celui-ci et vous pourrez le corriger à la fin.

Pour ceux qui ne possèdent pas les drives dont il est question, voici les commandes utilisées et leur signification.

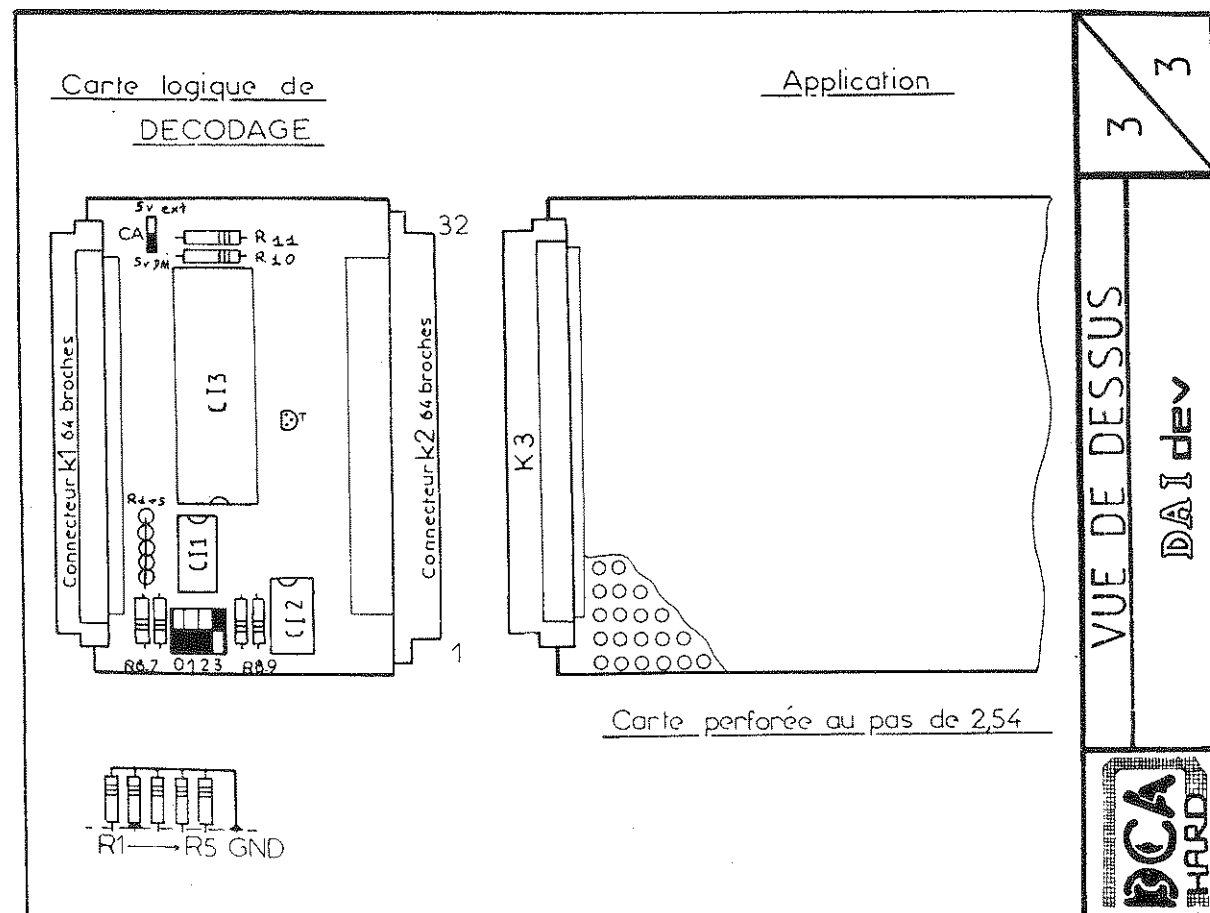
ASSIGN: Remplace tout dans un état standard (clavier, disquettes, écran) pour ce qui concerne les entrées-sorties uniquement ! (le **RS232** est déconnecté)

ASSIGN "OUTPUT TO RS232": Connecte le **RS232**.

Brochage connecteurs K₂ et K₃
(coté B)

- 1 : MASSE
- 2 : PA₁
- 3 : PA₂
- 4 : PA₃
- 5 : PA₄
- 6 : PA₅
- 7 : PA₆
- 8 : PA₇
- 9 : PA₈
- 10 : EXT RES
- 11 : +5v
- 12 : INTR
- 13 : EXT INTR
- 14 : IN7
- 15 : -5v
- 16 : PC₇
- 17 : PC₆
- 18 : PC₅
- 19 : PC₄
- 20 : PC₃
- 21 : PC₂
- 22 : PC₁
- 23 : PB₇
- 24 : PB₆
- 25 : PB₅
- 26 : PB₄
- 27 : PB₃
- 28 : PB₂
- 29 : PB₁
- 30 : PB₀
- 31 : PB₃
- 32 : +12v

BCA Hard-erratum: Sur le schéma de principe d'ANANUM PB₀ doit être relié à A₀ (broche 9 du 8255) et PB₁ à A₁ (broche B) et non pas le contraire comme c'était marqué. L'erreur n'existe cependant pas sur l'implémentation qui reste donc quand à elle parfaitement correcte. Nous pensons que vous avez déjà rectifié de vous-même mais vous prions de bien vouloir nous excuser.



DELETE: Détruit le fichier nommé sur la disquette.

DOS: Permet simplement de programmer des instructions du DOS.

NODOS: Revenir aux instructions BASIC.

Pour plus de facilité, regardez la syntaxe de ces opérations aux lignes 170-180-190 qui connectent le RS232 alors que la ligne 200 inhibe toute émission vers le RS232.

Il n'est pas possible de faire le tour des possibilités du programme ici et je ne fournis qu'un seul exemple qui rassemble un certain nombre des particularités très intéressantes pour les résolutions de systèmes.

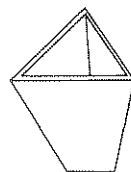
Le système initial est composé de 5 équations à 4 inconnues, il y a donc une équation de trop. Introduisons la matrice telle quelle. Le programme signale d'abord que l'équation numéro 3 est redondante (inutile) et la supprime. Puis, il détecte que l'équation numéro 4 est également redondante. Ici, il est bon de signaler que cette équation 4 est en fait la 5è dans le système initial. Cette équation est en effet devenue la 4è lors de la suppression de l'équation 3!

Le programme a donc éliminé 2 équations or il n'y en avait qu'une de trop, le système est donc à présent simplement indéterminé (une équation manque). L'inconnue X_4 (la dernière) est donc prise comme variable et les autres inconnues sont exprimées en fonction de X_4 , ce qui nous donne un système bien plus aisé à manipuler, vous en conviendrez.

EXEMPLE DE RESOLUTION DE SYSTEME (Précision : 1E-5)

Système à résoudre:

$$\begin{aligned} X_1 - 4X_2 - 7X_3 + X_4 - 23 &= 0 \\ X_2 - X_3 + 2X_4 + 10 &= 0 \\ 2X_1 - 9X_2 - 13X_3 &= -56 \\ -4X_1 - X_3 + 9X_4 + 22 &= 0 \\ -5X_1 + X_2 + 9X_3 + 2X_4 + 15 &= 0 \end{aligned}$$



Résultat: Equation numéro 3 redondante; je la supprime donc !
Equation numéro 4 redondante; je la supprime donc !
Ce système est indéterminé !!!

$$\begin{aligned} X_1 &= -5 + 2X_4 \\ X_2 &= 8 - X_4 \\ X_3 &= -2 + X_4 \\ X_4 &= X_4 \end{aligned}$$

```
10 REM VERSION: V6.6
30 REM CE PROGRAMME UTILISE L'ALGORITHME DE
40 REM BANACHIEWICZ.
50 REM IL PERMET LE CALCUL DE DETERMINANT (MAX:50X50)
60 REM AINSI QUE L'INVERSION D'UNE MATRICE CARREE
70 REM (MAX:25X25). IL PEUT AUSSI RESOUDRE UN
80 REM SYSTEME D'EQUATIONS LINEAIRES A PLUSIEURS
90 REM INCONNUES (MAX:50).
100 REM#####
110 REM Maurice DECUYPER janvier 1986
120 REM 11, rue A. DUMONT
130 REM 4601 CHAUDFONTAINE tel: 041/65.75.72

150 REM
160 REM
170 POKE #131,3:REM DOS
180 PRINT "ASSIGN","OUTPUT TO RS232"
190 POKE #131,PEEK(#1E):REM NODOS
200 POKE #131,1
210 CLEAR 4
220 CLEAR FRE:DIM ELEM(50.0,50.0),ELEMENT(50.0,50.0),COLAB(50.0),X$(5
0.0),INV(25.0,25.0)
230 FICHER$="MATRICE":DRIVE%=0:REM MODIFIEZ A VOTRE CONVENANCE CES P
ARAMETRES
240 NOM$=FICHER$+".FPT:"+MID$(STR$(DRIVE%),1,1)
250 POKE #131,1:PRINT CHR$(12):CURSOR 20,14:PRINT "BANACHIEWICZ"
260 PRINT TAB(20);"~~~~~ de Maurice DECUYPER"
270 PRINT :PRINT " Desirez-vous faire un calcul de: Determinant (D)"
280 PRINT TAB(34);"Resolution de systeme (R)"
290 PRINT TAB(34);"Inversion de matrice (I)"
300 CURSOR 25,CURY:POKE #75,63:SIGNE%=1:DTM%=1
310 A=GETC:IF A=0.0 THEN 310
320 IF A=ASC("D") THEN CALCUL%=0.0:COLON%=LIGNE%:GOTO 360
330 IF A=ASC("R") THEN CALCUL%=1.0:GOTO 360
340 IF A=ASC("I") THEN CALCUL%=2.0:COLON%=LIGNE%:GOTO 360
350 GOTO 310
360 PRINT CHR$(12):POKE #131,0:GOSUB 2380:CURSOR 0,14:ON CALCUL% GOTO
380,390
370 PRINT SPC(10);"DETERMINANT":PRINT SPC(10);"*****":GOTO 400
380 PRINT SPC(7);"RESOLUTION DE SYSTEME":PRINT SPC(7);"*****"
*****":GOTO 400
390 PRINT SPC(7);"INVERSION DE MATRICE":PRINT SPC(7);"*****"
*****":GOTO 400
400 POKE #75,95:GOSUB 2410:POKE #131,1:IF MEME%=1 THEN 660
410 IF MEME%=2.0 THEN 610
420 GOSUB 2410:POKE #131,1:INPUT "NOMBRE DE LIGNE: ";LIGNE%:PRINT :IF
LIGNE%<1.0 OR LIGNE%>50.0 THEN 420
430 IF CALCUL%=2 AND LIGNE%>25 THEN 420
440 IF CALCUL%=0 OR CALCUL%=2 THEN COLON%=LIGNE%:GOTO 460
450 INPUT "NOMBRE DE COLONNES: ";COLON%:PRINT :IF COLON%<1 OR COLON%
50 THEN 450
460 INPUT "PRECISION: ";PRECIS:IF PRECIS=0 THEN PRECIS=1E-5
470 ELEMENT(1.0,0.0)=PRECIS:PRINT CHR$(12)
480 ELEMENT(0.0,0.0)=LIGNE%+(COLON%/100.0)
490 CURSOR 18,15:PRINT "INTRODUCTION DE LA MATRICE"
500 PRINT TAB(18);"~~~~~";PREC=0.0
510 FOR LX=1 TO LIGNE%:FOR CX=1 TO COLON%
520 CURSOR 10,12:PRINT "ELEMENT PRECEDENT : ";PREC:SPC(10)
530 PRINT TAB(10);"ELEMENT L";LX;","C";CX;"=" :;INPUT ELEMENT(LX,CX):P
RINT
540 CURSOR 10,12:PRINT SPC(40):PRINT TAB(10);SPC(40)
550 PREC=ELEMENT(LX,CX):NEXT CX:NEXT LX
560 PRINT :PRINT " MERCI MERCI MERCI MERCI MERCI MERCI
MERCIS":PRINT :PRINT "EST-CE CORRECT (O/N)?"
```



```

570 A=BETC:IF A=0.0 THEN 570
580 IF A=ASC("N") THEN 605B 2280:GOTO 600
590 IF A<>ASC("O") THEN 570
600 SAVEA ELEMENT NOMS
610 PRINT CHR$(12):POKE #131,0:PRINT "PRECISION : ";PRECIS
620 PRINT SPC(20);"MATRICE DE DEPART"
630 PRINT SPC(20);"*****"
640 PRINT "Ligne, colonne: element"
650 FOR LX=1 TO LIGNEZ:FOR CX=1 TO COLONZ:PAUSE:PRINT "L";LX;"C";CX;
660 " ";ELEMENT(LX,CX):NEXT CX:PRINT "NEXT L";POKE #131,1
670 PRINT CHR$(12):CURSOR 20,14:PRINT "EN CALCUL":CURSOR 0,0:POKE #13
1,0:CT=0:IF CALCULZ=2 THEN 950
680 REM *** PHASE PROGRESSIVE *** CALCUL DES COLONNES ***
690 IF ELEMENT(1,0,1,0)=0.0 THEN COLA=1:GOTO 900
700 CX=CX+1:IF CX<=COLONZ THEN ELEMENT(LX,CX)=ELEMENT(1,0,CX):GOTO 690
710 LX=LX+1:IF LX<=COLONZ THEN CAL=0
720 CAL=CAL+ELEMENT(LX,CX):ELEMENT(LX,CX)=CAL:CAL=0
730 REM ***** CALCUL DES LIGNES *****
740 CX=CX+1:IF CX<=LIGNEZ THEN LIGNEZ=LIGNEZ+1
750 LX=LX+1:IF LX<=LIGNEZ THEN CAL=CAL+ELEMENT(LX,CX):ELEMENT(LX,LX):GOTO 7
50
760 ELEM(LX,CX)=ELEMENT(LX,CX)-CAL
770 IF CALCULZ=2.0 AND CX<=COLONZ/2.0 AND PIVOT=0.0 THEN 1170
780 CX=CX+1:IF ABS(ELEM(LX,CX-1,0))<=PRECIS THEN ELEM(LX,CX-1,0)=0.0:
GOTO 820
790 IF PIVOT<>0.0 THEN 850
800 IF CX-1,0=COLONZ AND CALCULZ=1,0 THEN 1000
810 COLAZ=LX:COLBZ=CX-1:GOTO 9999
820 IF LX=CX-1 THEN PIVOT=0:GOTO 800
830 IF PIVOT<>0.0 THEN 850
840 IF CX-1,0=COLONZ THEN 1110
850 IF CX=COLONZ THEN LIX=0:GOTO 750
860 CX=LX:LX=LX+1:IF LX<=LIGNEZ THEN LIX=0:CAL=0.0:GOTO 710
870 GOTO 1400
880 IF CX-1,0=COLONZ THEN 1100
890 GOTO 850
900 REM *** COLBZ=? ***
910 LX=LX+1
920 CX=CX+1:IF CX<=COLONZ THEN 1110
930 IF ELEMENT(1,0,CX)>0.0 THEN COLAZ=COLAZ+1:GOTO 1040
940 GOTO 920
950 LX=LX+1:IF LX<=COLONZ THEN COLONZ=COLONZ+2
960 ELEM(LX,CX)=1.0
970 LX=LX+1:IF LX<=LIGNEZ THEN 940
980 CX=0
990 GOTO 680
1000 POKE #131,1:PRINT CHR$(12):POKE #131,0:GOSUB 2380:CURSOR 5,14:PRI
NT "CE SYSTEME D'EQUATIONS EST INCOMPATIBLE":PRINT
1010 IF GETC<>32.0 THEN 1010
1020 GOTO 1200
1030 REM *** COLAZ**COLBZ ***
1040 IF CALCULZ=2.0 AND COLBZ<=COLONZ/2.0 THEN 1170
1050 IF CALCULZ<>0.0 THEN COLAZ(COLBZ)=COLBZ*COLAZ/100.0:GOTO 1070
1060 SIGNEZ=-SIGNEZ
1070 LX=0
1080 LX=LX+1:IF LX<=LIGNEZ THEN LX=COLAZ:LIX=0:CAL=0.0:PIVOT=1.0:GOTO 7
50
1090 A=ELEM(LX,COLAZ):ELEMENT(LX,COLAZ)=ELEMENT(LX,COLBZ):ELEMENT(L
X,COLBZ)=A:A=ELEM(LX,COLAZ):ELEMENT(LX,COLAZ)=A
1100 REM *** SUPPRIMER LA LIGNE LX ***
1110 IF CALCULZ=0.0 THEN 1190
1120 IF CALCULZ=2.0 THEN 1170
1130 PRINT "EQUATION NUMERO ";LX;" REDONDANTE; JE LA SUPPRIME D

```

```

ONC 1"
1140 LIGNEZ=LIGNEZ-1:IF LX<=LIGNEZ THEN 1710
1150 FOR LIX=LX TO LIGNEZ+1:FOR CIX=1 TO COLONZ:ELEMENT(LIX,CIX)=ELEME
NT(LIX,1,0,CIX):ELEMENT(LIX,CIX)=ELEMENT(LIX+1,0,CIX):NEXT CIX:NEXT LIX
1160 PIVOT=1,0:CAL=0.0:CAL=0.0:CAL=LX:GOTO 750
1170 GOSUB 2380:PRINT SPC(15);"CETTE MATRICE N'EST PAS INVERSIBLE,CAR S
ON DETERMINANT"
1180 PRINT SPC(12);"EST NUL !!!!!":GOTO 1200
1190 GOSUB 2380:PRINT SPC(15);"LE DETERMINANT EST NUL !":DTRZ=0
1200 REM ***** RECOMMENCER ? ***
1210 GOSUB 2410:POKE #131,1:PRINT CHR$(12):PRINT "PRECISION : ";PRECIS
1220 "vous venez de faire un autre calcul ?":PRINT
1230 PRINT TAB(25);"Sur la matrice ?"
1240 PRINT TAB(25);"Avec une nouvelle matrice ?"
1250 PRINT TAB(6);"SI VOUS NE DESIREZ PLUS UTILISER LE PROGRAMME [3]
1250 A=BETC:IF A=0.0 THEN 1250
1260 IF A=ASC("I") THEN GOSUB 1350:GOTO 1380
1270 IF A=ASC("2") THEN GOSUB 1350:GOTO 210
1280 IF A<>ASC("3") THEN 1250
1290 PRINT CHR$(12)
1300 POKE #131,REM DOS
1310 PRINT "DELETED",NOMS
1320 PRINT "ASSIGN"
1330 POKE #131,PEEK(#1E):REM NOMDS
1340 CLEAR 4:END
1350 REM ***** ROUTINE DE SEPARATION DES CALCULS IMPRIMES *****
1360 POKE #131,0:GOSUB 2380:PRINT "FOR X=1.0 TO 8.0:PRINT "-----";NEX
T :PRINT :PRINT :PRINT :GOSUB 2410:POKE #131,1
1370 RETURN
1380 CLEAR 4
1390 CLEAR PREDIM ELEM(50,0,50,0),ELEMENT(50,0,50,0),COLAB(50,0),X$(5
0,0),INV(25,0,25,0)
1400 LOADA ELEMENT NOMS
1410 LIGNEZ=INT(ELEMENT(0,0,0,0)):COLONZ=INT(IVAL(RIGHT$(STR$(ELEMENT(0
,0,0,0)),2)):PRECIS=ELEMENT(1,0,0,0)
1420 PRINT CHR$(12):PRINT "PRINT :PRINT "DESIREZ-VOUS LA MODIFIER ?"
1430 A=BETC:IF A=0.0 THEN 1430
1440 IF A=ASC("O") THEN 2440
1450 IF A<>ASC("N") THEN 1430
1460 MEMEZ=1:GOTO 250
1470 REM ***** AIGUILLAGE *****
1480 ON CALCULZ+1 GOTO 1520,1710,2100
1490 REM *** LIGNES: 1520 DETERMINANT ***
1500 REM *** 1710 RESOLUTION DU SYSTEME ***
1510 REM *** 2100 INVERSION DE LA MATRICE ***
1520 IF DTRZ=0.0 THEN CAL=0.0:GOTO 1580
1530 LX=LX+1:CAL=1,0
1540 CAL=CAL+ELEM(LX,CX)
1550 LX=LX+1:CAL=LX
1560 IF LX<=LIGNEZ THEN 1540
1570 CAL=CAL+SIGNEZ
1580 PRINT "PRINT TAB(15);"DETERMINANT=";CAL
1590 POKE #131,1:PRINT "Veuillez-vous des details (O/N) ?":CURSOR 0,0:PO
KE #131,0
1600 A=BETC:IF A=0.0 THEN 1600
1610 IF A=ASC("N") THEN 1200
1620 IF A<>ASC("O") THEN 1600
1630 REM *** GESTION D'IMPRESSANTE ***
1640 PRINT "PRINT SPC(20);"PHASE PROGRESSIVE":PRINT SPC(20);"*****"
1650 LX=LX+1:PRINT
1660 IF CALCULZ=0.0 AND CX=LX THEN PRINT CHR$(27);"E";POKE #131,1:CUR
SOR CURX-2,CURY:PRINT " ";CURSOR CURX-1,CURY:B=1,0:POKE #131,0
1670 PRINT "L";LX;"C";CX;" ";ELEMENT(LX,CX):IF B=1.0 THEN B=0.0:PRINT C
HR$(27);"F";CURSOR CURX-2,CURY:POKE #131,1:PRINT " ";CURSOR CURX-2,C

```

```

URY:POKE #131,0
1680 CX=CX+1:PAUSE:IF CX<=COLONZ THEN 1680
1690 LX=LX+1:IF LX<=LIGNEZ THEN PRINT "GOTO 1600
1700 PRINT "GOTO 1200
1710 REM *** RESOLUTION DU SYSTEME ***
1720 IF LIGNEZ<=COLONZ-1,0 THEN GOSUB 2380:PRINT "CE SYSTEME EST INDETE
RMINÉ !":GOSUB 2410:GOTO 1850
1730 IF LIGNEZ<=COLONZ-1 THEN 1000
1740 REM CALCUL DE LA PHASE REGRESSIVE
1750 FOR LX=LIGNEZ TO 1 STEP -1:CX=LX:CAL=ELEM(LX,COLONZ)
1760 FOR CIX=LIGNEZ TO CX STEP -1:CAL=CAL-ELEMENT(0,0,CIX):ELEMENT(LX,CIX
):NEXT CIX
1770 ELEMENT(0,0,CX)=CAL:ELEM(LX,CX):NEXT LX
1780 REM *** VERIFICATION DE L'ORDRE
1790 FOR CX=COLONZ-1 TO 1 STEP -1:IF INT(COLAB(CX))<>CX THEN NEXT:GOTO
1830
1800 REM *** ECHANGE
1810 CIX=INT(IVAL(RIGHT$(STR$(COLAB(CX)),2)))
1820 A=ELEMENT(0,0,CIX):ELEMENT(0,0,CX)=ELEMENT(0,0,CIX):ELEMENT(0,0,C
IX)=A:NEXT
1830 REM *** IMPRESSION ***
1840 PRINT "PRINT TAB(20);"RESULTATS":PRINT TAB(20);"*****"
1850 FOR CX=1 TO COLONZ-1:PAUSE:PRINT "X";CX;" ";ELEMENT(0,0,CX):NEXT
:GOTO 1590
1860 POKE #131,1:PRINT CHR$(12):VARZ=COLONZ-1-LIGNEZ
1870 FOR CX=COLONZ-1 TO COLONZ-VARZ-1 STEP -1:X$(CX)="X"+MID$(STR$(CX
),1,2):NEXT
1880 REM *** CALCUL DES T. I. ***
1890 FOR LX=LIGNEZ TO 1 STEP -1:CAL=ELEM(LX,COLONZ)
1900 FOR CX=LIGNEZ TO LX-1 STEP -1:CAL=CAL-ELEMENT(0,0,CX):ELEMENT(LX,CX
):NEXT CX
1910 ELEMENT(0,0,LX)=CAL:ELEM(LX,LX):X$(LX)=STR$(ELEMENT(0,0,LX)):NEXT
LX
1920 REM *** CALCUL DES COEFF. DES INCONNUES PRISES COMME VARIABLES**
1930 FOR LIX=LIGNEZ+1 TO COLONZ-1:FOR LIZ=LIGNEZ TO 1 STEP -1:CAL=ELEM
(LX,LIX)
1940 IF LIGNEZ=LX THEN 1960
1950 FOR CX=LIGNEZ TO LX+1 STEP -1:CAL=CAL-ELEM(LX,CX):ELEMENT(0,0,CX)
:NEXT CX
1960 ELEMENT(0,0,LX)=CAL:ELEM(LX,LX):IF ELEMENT(0,0,LX)=0.0 THEN 2010
1970 IF ELEMENT(0,0,LX)=(-1.0) THEN X$(LX)=X$(LX)+"-X"+MID$(STR$(LIX),
1,2):GOTO 2010
1980 IF ELEMENT(0,0,LX)=1.0 THEN X$(LX)=X$(LX)+"X"+MID$(STR$(LIX),1,2
):GOTO 2010
1990 IF ELEMENT(0,0,LX)>0.0 THEN X$(LX)=X$(LX)+""
2000 X$(LX)=X$(LX)+STR$(ELEMENT(0,0,LX))+ " X"+MID$(STR$(LIX),1,2)
2010 NEXT LX:NEXT LIX
2020 REM *** VERIFICATION DE L'ORDRE
2030 FOR CX=COLONZ-1 TO 1 STEP -1:IF INT(COLAB(CX))<>CX THEN NEXT:GOTO
2070
2080 REM *** ECHANGE
2090 CIX=INT(IVAL(RIGHT$(STR$(COLAB(CX)),2)))
2100 A=X$(CX):X$(CX)=X$(CIX):X$(CIX)=A:NEXT
2110 A=0.0:FOR COLINZ=LIGNEZ+1 TO COLONZ:A=A+1,0
2120 FOR CX=LIGNEZ TO 1 STEP -1:CAL=ELEM(CX,COLINZ):FOR CIX=LIGNEZ TO
CX STEP -1
2130 CAL=CAL-INV(A,CIX):ELEM(CX,CIX):NEXT CIX
2140 INV(A,CX)=CAL:ELEM(CX,CX):NEXT CX
2150 NEXT COLINZ

```

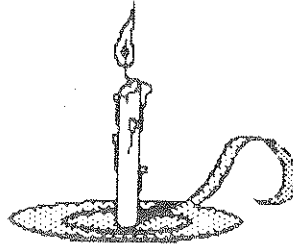
```

2160 REM *** VERIFICATION DE L'ORDRE DES COLONNES ***
2170 FOR CX=COLONZ/2 TO 1 STEP -1:IF INT(COLAB(CX))<>CX THEN NEXT:GOTO
2210
2180 REM *** ECHANGE DES 2 COLONNES ***
2190 CIX=INT(IVAL(RIGHT$(STR$(COLAB(CX)),2)))
2200 FOR LX=1 TO LIGNEZ:A=INV(LX,CX):INV(LX,CX)=INV(LX,CIX):INV(LX
):CIX=A:NEXT:GOTO 1590
2210 REM ***** TRANSPOSITION *****
2220 PRINT "PRINT TAB(17);"MATRICE INVERSE":PRINT TAB(17);"*****"
2230 PRINT "Ligne, colonne: element"
2240 PRINT "*****"
2250 FOR CX=1 TO LIGNEZ:FOR LX=1 TO LIGNEZ:PAUSE:PRINT "L";CX;" ";CIX;
";INV(LX,CX):NEXT LX
2260 PRINT "NEXT CX"
2270 GOTO 1590
2280 REM *** CORRECTION ***
2290 PRINT CHR$(12):PRINT "PRINT :PRINT
2300 PRINT "ENTREZ 0,0 POUR ARRETER.":PRINT
2310 PRINT "ELEMENT(S) A CORRIGER.":PRINT
2320 INPUT "LIGNE, COLONNE : ";LX,CX:PRINT
2330 IF LX=0.0 AND CX=0.0 THEN RETURN
2340 IF LX=0.0 OR CX=0.0 OR LX>LIGNEZ OR CX>COLONZ THEN PRINT "CET EMP
LACEMENT N'EXISTE PAS !":GOTO 2320
2350 PRINT "ANCIEN ELEMENT = ";ELEMENT(LX,CX)
2360 GOTO 2320
2370 REM *** PASSAGE EN MODE ELARGI
2380 REM *** CURSOR CURX-3,CURY:POKE #131,0
2390 PRINT CHR$(27);"M";CHR$(1):POKE #131,1:CURSOR CURX-3,CURY:PRINT
2400 RETURN
2410 REM *** PASSAGE EN MODE NORMAL
2420 PRINT CHR$(27);"0";POKE #131,1:CURSOR CURX-3,CURY:PRINT " ";
CURSOR CURX-3,CURY:POKE #131,0
2430 RETURN
2440 REM *** MEME MATRICE MAIS RETRAVAILLEE
2450 MEMEZ=2:PRINT "PRINT "NOMBRE DE LIGNE = ";LIGNEZ:PRINT "NOMBRE DE
COLONNES = ";COLONZ
2460 PRINT "DESIREZ-VOUS CHANGER LES DIMENSIONS ?"
2470 A=BETC:IF A=0.0 THEN 2470
2480 IF A=ASC("N") THEN 2520
2490 IF A<>ASC("O") THEN 2470
2500 INPUT "NOMBRE DE LIGNE = ";LIGNEZ:PRINT
2510 INPUT "NOMBRE DE COLONNES = ";COLONZ:PRINT
2520 PRINT "DESIREZ-VOUS MODIFIER DES ELEMENTS PARTICULIER ?"
2530 A=BETC:IF A=0.0 THEN GOSUB 2280:GOTO 2560
2540 IF A=ASC("O") THEN GOSUB 2280:GOTO 2560
2550 IF A<>ASC("N") THEN 2530
2560 PRINT "LA PRECISION ETAIT DE ";PRECIS;" . VOULEZ-VOUS LA CHANGER
?"
2570 A=BETC:IF A=0.0 THEN 2570
2580 IF A=ASC("N") THEN 2610
2590 IF A<>ASC("O") THEN 2570
2600 INPUT "NOUVELLE PRECISION : ";PRECIS:PRINT "IF PRECIS=0.0 THEN PR
ECIS=1E-5
2610 PRINT CHR$(12):PRINT "PRINT :PRINT "FAUT-IL SAUVER
LA MATRICE MODIFIEE ?"
2620 A=BETC:IF A=0.0 THEN 2620
2630 IF A=ASC("N") THEN 250:REM RETOUR AU PROGRAMME PRINCIPAL
2640 IF A<>ASC("O") THEN 2620
2650 ELEMENT(0,0,0)=LIGNEZ*(COLONZ/100.0):ELEMENT(1,0,0)=PRECIS
2660 GOTO 250
2680 END

```

Dessin animé assiste par ord.

DESSIN ANIME ASSISTE PAR ORDINATEUR H. SAMAIN pour DCA



Aborder l'animation sur ordinateur, peut se faire de plusieurs facon suivant la méthode utilisée pour générer les images.

La méthode la plus simple est de rentrer les images en les dessinant sur l'écran. C'est cette méthode qui servira de modèle pour un premier programme de type expérimental.

On pourra ensuite utiliser des sous-programmes de traitement de l'image : translation, rotation... en trois dimensions...

Une fois que ces sous-programmes sont assimilés, on pourra alors envisager de programmer la suite de ces traitements de l'image et accéder ainsi à l'automatisation de la création de notre animation.

Le deuxième point à aborder est l'utilisation de l'image créée. LA, deux ou trois méthodes :

* L'image est affichée sur l'écran et reste ainsi pendant que l'ordinateur calcule l'autre image qui prendra alors la suite.

Avantage : pas besoin de stocker chaque image en mémoire. On peut alors accéder à la haute résolution puisque la mémoire n'est partagée qu'en deux pages : page écran plus page de l'image en cours de calcul.

Inconvénient : on est confronté au problème de la lenteur. Si on calcule l'image, les calculs ne devront pas être compliqués et de toute manière, ils devront être écrits en langage machine, ce qui ne pose pas de problème pour la translation mais demande un certain effort pour la rotation en 3D ou l'effet d'éclatement... De plus cette méthode est inconcevable si l'on rentre les images en les dessinant.

* Chaque image est stockée d'une manière ou d'une autre en mémoire centrale. Un très court programme en langage machine les fait défiler une fois les images entrées en mémoire.

Avantage : Pas de problème de temps. Les calculs peuvent être longs et en basic. On peut rentrer les images en les dessinant.

Inconvénient : Le Dai n'est pas le CRAY-2 et sa mémoire ne fait que 48 Ko. Pas question donc d'animer une pleine page haute résolution.

* L'ordinateur se fait aider par un système spécialisé dans l'animation d'image : le magnétoscope et peut être un jour le vidéodisque (on peut toujours rêver non ?!...). Alors là, à nous les images haute résolution animées, d'autant plus qu'on pourra ajouter au DAI une unité de stockage de masse type disquette pour stocker les coordonnées de l'image en cours car un objet en volume demande une sacrée mémoire pour être digitalisé. Seul problème : raccorder le DAI à un magnétoscope (sinon la télé au magnétoscope). Un joli problème posé aux mordus de l'interface et j'espère qu'ils relèveront le défi (si ce n'est déjà fait) !

Le programme du jour

Il traite l'animation en utilisant les méthodes de rentrée successive des diverses images puis d'affichage séquentiel de celles-ci.

Lignes 30 à 120 : on dessine à l'écran (MODE 2A) en noir sur blanc. On déplace le curseur par les flèches. Une fois ce curseur à l'endroit désiré, un appui sur SPACE laissera un point sur l'écran immortalisant ainsi un élément unitaire du dessin. Le passage du curseur sur un point noirci gommara ce dernier. Le confort minimum est assuré : ainsi peut-on tracer une droite en pointant le point de départ par "(" et le point d'arrivée par ")". D'autres fonctions élémentaires peuvent être ajoutées facilement. En appuyant sur RETURN, on passe à la suite des événements...

Ligne 1015 : permet de laisser à l'écran l'ancienne page, ce qui sera bien utile pour dessiner l'image suivante. Si on choisit l'option noire, on utilisera l'ancienne sans devoir la redessiner. Exemple : un décor. Si on choisit l'option bleue, l'ancien dessin restera visible mais ne sera pas pris en compte lors de la digitalisation suivante.

Lignes 1020 à 1120 : scrutation de l'écran. Tout octet contenant au moins un point noir est stocké à l'emplacement mémoire correspondant à la page sous la forme : adresse du point (2 octets) - pixel (1 octet).

Lignes 1200 à 1208 : permet de continuer ou de rattraper une page qui ne convient pas.

Lignes 1210 à 1213 : animation. Le nombre d'images maximum à la seconde pourra être de 17.

Ligne 1215 : défilement automatique ou programmation de la suite des images.

Lignes 1218 à 1300 : défilement automatique.

POKE *2FF, no de page : informe le programme en langage machine du numéro de la page. MODE 2A : efface l'écran. CALLM *3000 : appelle le programme en langage machine affichant les points stockés auparavant. L'appui sur une touche permet à chaque instant de revenir sur n'importe quel point du travail.

Lignes 1310 à 1410 : programmation des images. La seule différence avec le défilement automatique est que l'on donne la suite des pages à afficher : no de page, RETURN, no de page, RETURN, ... , no de page, RETURN, 255, RETURN.

Lignes 1500 à 1510 : programme en langage machine (38 octets) localisé en *3000.

Il ne me reste plus qu'à vous souhaiter de belles animations. Si, vous aussi, vous avez travaillé dans ce domaine, n'hésitez pas à nous contacter !

Enfin, pour les paresseux du clavier le programme est disponible à la softthèque.

```
1 REM *****
2 REM * Dessin Anime Assiste par Ordinateur *
3 REM * programme conçu par H. Samain *
4 REM * membre de D.C.A. *
5 REM *****
6 CLEAR 15000
10 MODE 2A:COLORG 15 12 0 3:C=15.0:PRINT CHR$(12)
20 FOR I=*3000 TO *3025:READ J:POKE I,J:NEXT I
30 I=GETC:IF I=0.0 THEN 30
35 DOT X,Y C
37 C=15.0
40 IF I=16.0 THEN Y=Y+1.0:GOTO 100
50 IF I=17.0 THEN Y=Y-1.0:GOTO 100
60 IF I=18.0 THEN X=X-1.0:GOTO 100
70 IF I=19.0 THEN X=X+1.0:GOTO 100
80 IF I=32.0 THEN CURSOR 40,2:PRINT X,Y:C=0.0:GOTO 100
85 IF I=13.0 THEN 1000
86 IF I=40 THEN D1=X:D2=Y:DOT X,Y 3:C=3.0:GOTO 30
```

```

87 IF I=41.0 THEN DRAW D1,D2 X,Y 0:C=0.0:GOTO 30
90 GOTO 30
100 DOT X,Y 0
120 GOTO 30
1000 N=N+1.0
1005 Z=*400+(N-1.0)*256.0
1010 PRINT "C'ETAIT LA PAGE N";N
1015 INPUT "VOULEZ VOUS GARDER CETTE PAGE EN (B)LEU OU EN (N)OIR ";NOUB$
1020 FOR W=*BB23 TO *BB13 STEP -2.0
1030 FOR I=0.0 TO 40.0:W1=W+24.0*I
1040 S=PEEK(W1):IF S=0 THEN 1100
1050 IF NOUB$="N" THEN 1060
1055 POKE W1.0:POKE W1-1.5
1060 Z=Z+1.0:POKE Z,INT(W1/256.0)
1070 Z=Z+1.0:POKE Z,INT(256.0*(W1/256.0-INT(W1/256.0)))
1080 Z=Z+1.0:POKE Z,S
1090 M=M+1.0
1100 NEXT I
1110 NEXT W
1120 POKE *400+(N-1)*256.M
1200 PRINT :INPUT "UNE AUTRE PAGE";R$:G=0.0
1205 PRINT :PRINT :PRINT :PRINT :IF R$<>"OUI" THEN 1210
1206 INPUT "VOULEZ VOUS LA PAGE (S)UIVANTE OU UNE (A)UTRE ";SOUA$
1207 IF SOUA$="S" THEN M=0.0:GOTO 30
1208 PRINT :PRINT :PRINT :INPUT "NUMERO DE LA PAGE ";N:N=N-1.0:M=0.0:GOTO 30
1210 PRINT :PRINT :PRINT :PRINT "ANIMATION":PRINT
1212 INPUT " COMBIEN D'IMAGES PAR SECONDES ";R
1213 PRINT :PRINT :PRINT :PRINT :R1=ABS(50.0/R-3.0)
1215 INPUT " VOULEZ VOUS PROGRAMMER LA SUITE DES IMAGES ";R$
1216 PRINT :PRINT :PRINT
1217 IF R$="OUI" THEN 1310
1218 PRINT "DEFILEMENT AUTOMATIQUE ":INPUT "DEPUIS LA PAGE ";TO
1219 PRINT :INPUT "A LA PAGE ";T1:PRINT
1220 FOR T=TO TO T1
1225 POKE *2FF,T
1230 MODE 2A:CALLM *3000
1240 WAIT TIME R1
1245 IF GETC<>0 THEN G=9.0
1250 NEXT T
1260 IF G=9.0 THEN 1200
1300 GOTO 1220
1310 PRINT :PRINT :PRINT :PRINT :PRINT " Programmation ":PRINT
1315 S%=*3080
1316 L=0.0
1320 INPUT P%:POKE S%,P%
1322 IF P%>255 THEN L=L-1.0:GOTO 1330
1323 L=L+1.0
1325 POKE *2FF,P%:MODE 2A:CALLM *3000
1327 S%=S%+1:GOTO 1320
1330 FOR T=0.0 TO L
1360 P%=PEEK(*3080+T)
1370 POKE *2FF,P%
1380 MODE 2A:CALLM *3000
1390 WAIT TIME R1
1395 IF GETC<>0 THEN G=9.0
1400 NEXT T:IF G=9.0 THEN 1200
1410 GOTO 1330
1500 DATA 245,197,213,229,1,0,4,58,255,2,61,202,18,48,4,195,10,48,10,103,3,10,0
1510 DATA 95,3,10,18,37,194,20,48,225,209,193,241,201

```



L'ensemble de Mandelbrot



NO BOZOS

* EXPLORATION DE L'ENSEMBLE DE MANDELBROT *

L'ensemble de Mandelbrot est considéré par certains comme l'objet le plus complexe jamais défini par les mathématiciens.

Il appartient à la catégorie des fractals, mais, contrairement à ceux-ci, qui donnent en général des images géométriques et répétitives, donc sans grand intérêt, l'ensemble de Mandelbrot et ses environ immédiats donnent, autour d'un thème commun, des variations infinies, au point qu'on peut faire du véritable "tourisme informatique" en explorant ses frontières.

En choisissant judicieusement les couleurs de représentation, on peut obtenir des effets esthétiques remarquables, et des images tellement étranges qu'on les croit difficilement générées par des opérations mathématiques simples.

Les seules limites imposées à l'exploration sont données par la précision de l'ordinateur qui empêche de pousser le grossissement trop loin, et le temps de calcul (en BASIC entre 6 et 24 heures pour une image).

L'ensemble de Mandelbrot et ses environs intéressants sont localisés dans le plan complexe, dans le carré allant de $X=-2$ à $X=+0,5$ et de $Y=-1,25$ à $Y=+1,25$.

Le principe de la représentation consiste à décomposer le carré à explorer (qui peut n'être qu'une petite partie du carré précédent) en points (par exemple 200 x 200) et à tester chaque point en effectuant la série d'itérations suivante :

soient $C=X_0+iY_0$, le nombre complexe représentant le point et Z , le nombre complexe variable dont la valeur initiale est 0 et qui vaut après chaque itération :

$$(nouvelle\ valeur\ de\ Z) = (valeur\ precedente\ de\ Z)^2 + C$$

- le test se fait après chaque itération sur le module M de Z :

- quand M dépasse 2, on arrête l'itération et on donne au point (converti en coordonnées d'écran) une couleur fonction du nombre d'itérations, puis on recommence pour le point suivant.

Si M ne dépasse jamais 2 quel que soit le nombre d'itérations (pratiquement, on se limite à 100), le point appartient à l'ensemble de Mandelbrot et est représenté en noir.

Cette méthode de représentation, due au mathématicien J. Hubbard, donne des images beaucoup plus riches que la simple représentation de l'ensemble seul.

J'ai écrit ces programmes après avoir lu l'article de A. Dewdney sur l'ensemble de Mandelbrot paru en octobre 1985 dans la revue "Pour la Science" (et dont je conseille la lecture).

Pour raccourcir le temps de calcul (environ quatre fois), j'ai utilisé deux artifices, valables seulement quand la zone à explorer est symétrique par rapport à Ox :

- dessin simultané des deux demi-images.

- "accélérateur dans le noir", qui permet, avec certaines restrictions, d'éviter de devoir effectuer 100 itérations sur chaque point noir de l'intérieur de l'ensemble. Cet accélérateur doit être déconnecté dans les parties concaves suivant Ox , sous peine de les combler (voir deuxième image).

Dans tous les cas, il y aurait moyen de gagner beaucoup de temps en réécrivant la partie itérative du programme en assembleur.

L. Laurent B-1180 Bruxelles

```
100 REM ENSEMBLE DE MANDELBROT CAS SYM /OX V1.2
110 REM ***** L LAURENT 1986 *****
120 REM L'ENSEMBLE EST REPRESENTE EN NOIR
130 REM L'EXTERIEUR DE L'ENSEMBLE UTILISE 3 COULEURS
140 REM SUIVANT LE NOMBRE D'ITERATIONS POUR DEPASSER 2
150 REM COMME ON NE DISPOSE QUE DE 4 COULEURS ELLES
160 REM SONT UTILISEES 2 FOIS (LE NOIR UNE SEULE FOIS)
170 REM LE MODE 6A ET 100 ITERATIONS SONT NECESSAIRES
180 REM POUR UNE BONNE DEFINITION
190 REM LE PROGRAMME EST ALORS TRES LENT (6 A 8 HEURES)
200 REM ON PEUT MODIFIER LES COULEURS PAR APRES
210 REM AVEC UN COLOGR (PAS DE COMMANDE DE MODE !)
220 REM ET SAUVER L'IMAGE PAR LA PROCEDURE DU MANUEL
230 REM L'ACCELERATEUR PERMET D'ALLER PLUS VITE EN
240 REM REMPLISSANT L'INTERIEUR DE LA ZONE NOIRE
250 REM A PARTIR DE 20 OU 5 POINTS DE SA LIMITE (SUI-
260 REM VANT MODE SANS EFFECTUER LES ITERATIONS. NE PAS
270 REM L'UTILISER DANS LES PARTIES CONCAVES /OX
280 REM POUR L'ACTIVER. PRESSER 'A' ET 'REPEAT'
300 REM POUR LE DESACTIVER. PRESSER 'N' ET 'REPEAT'
310 REM POUR CHOISIR LES NOMBRES D'ITERATIONS LIMITES
320 REM CORRESPONDANT AUX TROIS COULEURS REPRESENTANT
330 REM L'EXTERIEUR DE L'ENSEMBLE. UTILISER LE
340 REM PROGRAMME DE TEST QUI DONNE LE NOMBRE D'ITERA-
350 REM TIONS POUR UN CERTAIN NOMBRE DE POINTS
360 REM OU ESSAYER EN MODE 2A
370 PRINT CHR$(12);POKE #131,1:COLORT 13 0 0 0
380 PRINT "ENSEMBLE DE MANDELBROT (CAS SYM /OX) V1.2 * L.LAURENT 1986 *"
390 COLORG 15 0 14 3 322 -1
400 PRINT "MODE 6A. SINON MODIFIER LIGNE 410"
410 MODE 6A:YM%-200:REM MODE 2A:YM%-50 /MODE 4A:YM%-100
420 INPUT "NOMBRE D'ITERATIONS":N0%
430 PRINT "REMI NOMBRE D'ITER LIMITE POUR CHAQUE COULEUR"
440 INPUT "L1":L1%:REM S1 N0%:L1:COUL.ROUGE
450 PRINT
460 INPUT "L2":L2%:REM " L2:COUL.BLANCHE
470 PRINT
480 INPUT "L3":L3%:REM " L3:COUL.JAUNE
490 PRINT
500 INPUT "L4":L4%:REM " L4:COUL.ROUGE
510 PRINT
520 INPUT "L5":L5%:REM " L5:COUL.BLANCHE
530 PRINT
540 INPUT "L6":L6%:REM " L6:COUL.JAUNE
550 PRINT
560 INPUT "L7":L7%:REM " L7:COUL.ROUGE
570 PRINT
580 INPUT "L8":L8%:REM " L8:COUL.BLANCHE
```

```
590 PRINT
600 INPUT "XMINI (min -2.05)":A
610 PRINT "IF (-A<2.0 OR A<0.5) THEN 600
620 INPUT "XMAXI (>XMINI,max 0.5)":B
630 PRINT "DEC%:YM%/10
640 IF (B-A<0 OR B=A OR B<0.5) THEN 620
650 C=(B-A)/2.0
660 D=C
670 FOR X%-0 TO YM%
680 X1%:X2%:DEC%:REM DECALAGE DE L'IMAGE
690 K%-0
700 X0-A:X1%*(B-A)/YM%
710 IF X0<0.25 THEN K2%=-1:COLORT 13 0 0 0:REM SUPPRESS ACCEL.
720 FOR Y%-0 TO YM%/2.0
730 ACC%:GETC:ACC%-GETC:REM ACCELERATEUR
740 IF ACC%<65.0 THEN K2%=-5:COLORT 11 0 0 0
750 IF ACC%>78.0 THEN K2%=-1:COLORT 13 0 0 0
760 K1%:0.5*YM%/K2%
770 YP%-YM%-Y%
780 Y0-C:Y%*(D-C)/YM%
790 XZ-0:YZ-0:N%-0
800 IF N%>N0% THEN 1010
810 X1Z-XZ
820 XZ-XZ*XZ-YZ*YZ-X0
830 YZ-2.0*X1Z*YZ-Y0
840 M-XZ*YZ*YZ
850 IF M<4.0 THEN 900
860 REM
870 REM
880 N%-N%+1
890 GOTO 800
900 IF N%>1.8% THEN DOT X1%:Y%:15:DOT X1%:Y%:15:GOTO 990
910 IF N%>1.7% THEN DOT X1%:Y%:14:DOT X1%:Y%:14:GOTO 990
920 IF N%>1.6% THEN DOT X1%:Y%:13:DOT X1%:Y%:13:GOTO 990
930 IF N%>1.5% THEN DOT X1%:Y%:12:DOT X1%:Y%:12:GOTO 990
940 IF N%>1.4% THEN DOT X1%:Y%:11:DOT X1%:Y%:11:GOTO 990
950 IF N%>1.3% THEN DOT X1%:Y%:10:DOT X1%:Y%:10:GOTO 990
960 IF N%>1.2% THEN DOT X1%:Y%:9:DOT X1%:Y%:9:GOTO 990
970 IF N%>1.1% THEN DOT X1%:Y%:8:DOT X1%:Y%:8:GOTO 990
980 GOTO 1030
990 K%-0:PRINT X0:-Y0:N%
1000 GOTO 1030
1010 DOT X1%:Y%:0:DOT X1%:Y%:0:K%-K%-1:0:PRINT X0:-Y0:N%
1020 IF K%>K1% THEN DRAW X1%:Y%:K1%:YM%/2:0:DRAW X1%:Y%:K1%:YM%/2
0:GOTO 1040:REM ACCELERATION DANS LE NOIR
1030 NEXT Y%
1040 NEXT X%
1050 C1-YM%/10:0:C2-YM%:0:C3-1.1*YM%
1060 DRAW C1:0:C3:0
1070 DRAW C3:0:C2:0
1080 DRAW C3:C2:C1:0
1090 DRAW C1:C2:C1:0
1100 PRINT "ENS.MANDELBROT XMINI-'A':XMAXI-'B'
1110 PRINT
" L1%:"/L2%:"/L3%:"/L4%:"/L5%:"/L6%:"/L7%:"
"/L8%
```

```
260 INPUT "L5":L5%:REM " L5:BLANCHE
270 PRINT
280 INPUT "L6":L6%:REM " L6:JAUNE
290 PRINT
300 INPUT "L7":L7%:REM " L7:ROUGE
310 PRINT
320 INPUT "L8":L8%:REM " L8:BLANCHE
330 PRINT
340 INPUT "XMINI (min -2.05)":A
350 PRINT "IF (-A<2.0 OR A<0.5) THEN 340
360 INPUT "XMAXI (>XMINI,max 0.5)":B
370 PRINT
380 IF (B-A<0 OR B=A OR B<0.5) THEN 360
390 INPUT "YMINI (compris entre -1.25 et 1.25)":C
400 REM
410 REM
420 D=C-B-A
430 DEC%-YM%/10
440 REM
450 FOR X%-0 TO YM%
460 X1%:X2%:DEC%:REM DECALAGE DE L'IMAGE
470 REM
480 X0-A:X1%*(B-A)/YM%
490 REM
500 FOR Y%-0 TO YM%
510 REM
520 Y0-C:Y%*(D-C)/YM%
530 XZ-0:YZ-0:N%-0
540 IF N%>N0% THEN 750
550 X1Z-XZ
560 XZ-XZ*XZ-YZ*YZ-X0
570 YZ-2.0*X1Z*YZ-Y0
580 M-XZ*YZ*YZ
590 IF M<4.0 THEN 640
600 REM
610 REM
620 N%-N%+1
630 GOTO 540
640 IF N%>1.8% THEN DOT X1%:Y%:15:GOTO 730
650 IF N%>1.7% THEN DOT X1%:Y%:14:GOTO 730
660 IF N%>1.6% THEN DOT X1%:Y%:13:GOTO 730
670 IF N%>1.5% THEN DOT X1%:Y%:12:GOTO 730
680 IF N%>1.4% THEN DOT X1%:Y%:11:GOTO 730
690 IF N%>1.3% THEN DOT X1%:Y%:10:GOTO 730
700 IF N%>1.2% THEN DOT X1%:Y%:9:GOTO 730
710 IF N%>1.1% THEN DOT X1%:Y%:8:GOTO 730
720 GOTO 760
730 PRINT X0:Y0:N%
740 GOTO 760
750 DOT X1%:Y%:0:PRINT X0:Y0:N%
760 NEXT Y%
770 NEXT X%
780 C1-YM%/10:0:C2-YM%:0:C3-1.1*YM%
790 DRAW C1:0:C3:0
800 DRAW C3:0:C2:0
810 DRAW C3:C2:C1:0
820 DRAW C1:C2:C1:0
830 PRINT "ENS.MANDELBROT XMINI-'A':YMINI-'C':XMAXI-'B'
" L1%:"/L2%:"/L3%:"/L4%:"/L5%:"/L6%:"/L7%:"
"/L8%
```

```
RECHERCHE 2.1.
100 REM MANDELBROT SYM. TEST V1.1 * L.LAURENT *
110 PRINT "ENSEMBLE DE MANDELBROT (SYM/OX) TEST POUR CHOIX COUL."
120 PRINT CHR$(12)
130 REM MODE 2A:YM%-50 /MODE 4A:YM%-100 /MODE 6A:YM%-200
140 YM%-200
150 INPUT "NOMBRE D'ITERATIONS":N0%
160 PRINT " "
170 INPUT "XMINI (min -2,max 0.5)":A
180 PRINT " "
190 IF (-A<2.0 OR A<0.5) THEN 170
200 INPUT "XMAXI (>XMINI,max 0.5)":B
210 PRINT
220 IF (B-A<0 OR B=A OR B<0.5) THEN 200
230 PRINT
240 C=(B-A)/2.0
250 D=C
260 FOR X%-0 TO YM%:STEP 20
270 X0-A:X1%*(B-A)/YM%
280 FOR Y%-0 TO YM%:2.0:STEP 5
290 YP%-YM%-Y%
300 Y0-C:Y%*(D-C)/YM%
310 XZ-0:YZ-0:N%-0
320 IF N%>N0% THEN 400
330 X1Z-XZ
340 XZ-XZ*XZ-YZ*YZ-X0
350 YZ-2.0*X1Z*YZ-Y0
360 M-XZ*YZ*YZ
370 IF M<4.0 THEN 400
380 N%-N%+1
390 GOTO 320
400 CURSOR X%/5:Y%/5:PRINT N%.
410 NEXT Y%
420 NEXT X%
430 CURSOR 0 0

RECHERCHE 3.1.
100 PRINT CHR$(12);POKE #131,1
110 PRINT "ENSEMBLE DE MANDELBROT (CAS GEN.) V1.2 * L.LAURENT *"
120 COLORG 15 0 14 3
130 PRINT "MODE 6A. SINON MODIF LIGNE 140"
140 MODE 6A:YM%-200:REM MODE 2A:YM%-50 /MODE 4A:YM%-100
150 REM
160 INPUT "NOMBRE D'ITERATIONS N0%":N0%
170 PRINT "REMI NOMBRE D'ITER LIMITE POUR CHAQUE COULEUR.L1:L2:L3:N0%"
180 INPUT "L1":L1%:REM S1 N0%:L1:COUL.ROUGE
190 PRINT
200 INPUT "L2":L2%:REM " L2:BLANCHE
210 PRINT
220 INPUT "L3":L3%:REM " L3:JAUNE
230 PRINT
240 INPUT "L4":L4%:REM " L4:ROUGE
250 PRINT
```


DAI qui rit : clavier azerty

DAI QUI RIT (EXCLUSIVITE DAICLIC !):

Clavier AZERTY en vaut deux !

(Alain MARIATTE)

Il est très facile de redéfinir les touches du clavier du DAI. Il suffit de savoir comment s'opère la gestion des codes ASCII. Lorsqu'une touche est frappée, le système transforme les numéros de ligne et colonne (matrice du clavier) en une valeur qui sert d'offset dans la table des codes ASCII. Cette table réside en ROM 3, à l'adresse *E8C5, et son début est pointé par le contenu de *2A7 et *2A8. Nous pouvons donc écrire une nouvelle table en RAM, en assignant à chaque touche le code désiré. Mettons en *2A7,*2A8 la nouvelle adresse de la table, et c'est tout ! C'est aussi un moyen de placer des caractères semi-graphiques dans des lignes de REM. Pour améliorer la présentation des listings, il suffit d'assigner dans la table un de ces caractères à une touche, et le tour est joué.

Profitons de la simplicité de ce programme pour voir quelques trucs bien utiles :

- Le code ASCII du curseur se trouve en *75.

- L'autostart d'un programme machine en fin de chargement est réalisé par un saut donné en *2D4, où se trouve l'adresse de la routine qui stoppe le moteur du magnétophone. Remplaçons cette adresse par celle de notre programme, et le lancement devient ainsi automatique, dès la fin du chargement.

- Protection du programme machine : notre routine est placée en début de RAM utilisateur, et il faut ajuster le pointeur *29B à l'adresse qui suit la fin de notre routine, éventuellement ajuster la taille de l'espace réservé aux chaînes et tableaux à sa valeur par défaut (*100 en *29D,*29E), puis ajuster les autres pointeurs Basic en appelant la routine *DEB8. Le retour au Basic se fait par un saut à *C7A0.

Ne pas oublier de sauver le code machine à partir de *2D4, avec ce pointeur modifié (saut au programme utilisateur). Plus besoin ainsi d'avoir à se rappeler l'adresse de lancement du programme chargé. Et c'est bien agréable !

```

0000 ;*****
0000 ;*** CLAVIER AZERTY ***
0000 ;*** pour le DAI ***
0000 ;*** (c) Alain MARIATTE ***
0000 ;*** SOURCE SPL ***
0000 ;*****
0000 ; Mettre *EC en *2D5, *02 en *2D6 et sauve le
0000 ;programme machine (V 2D4 395 NOW). Ainsi, il est
0000 ;autostart' au chargement et se positionne en clavier
0000 ;AZERTY. Les pointeurs basic sont ajustés, et le
0000 ;programme est protégé. Pour repasser en :
0000 ; - QERTY = CALL#318 (curseur habituel)
0000 ; - AZERTY = CALL#30A (curseur tilde)
0000 ;*****
0000 0=02A7 KBTPT EQU 2A7H ;PTR TO TABLE ASC.CODES
0000 0=E8C5 KEYTU EQU 0E8C5H ;DAI ASCII TABLE
0000 0=029B HEAP EQU 29BH ;START HEAP
0000 0=029D HSIZE EQU 29DH ;SIZE OF HEAP
0000 0=DEB8 RNEW EQU 0DEB8H ;RUN NEW
0000 0=C7A0 BASIC EQU 0C7A0H ;JUMP TO BASIC
0000 0=075 CURIN EQU 75H ;CURSOR INFORMATION
0000 0=02D4 RCL0SE EQU 2D4H ;STOP CASS. MOTORS
0000 0=D445 ST0PH EQU 0D445H ;STOP MOTORS ROUTINE
0000
0000 02EC CD45D4 INIT ORG 2ECH ;STOP MOTORS ROUTINE
02EF 2145D4 LXI B ST0PH ;RESTORE VECTOR
02F2 22D502 SHLD RCL0SE+1H
02F5 219703 LXI H END+1H ;END OF HELP
02F8 229B02 SHLD HEAP ;
02FB 210001 LXI H 100H ;DEFAULT SIZE
02FE 229D02 SHLD HSIZE ;
0301 CDB8DE CALL RNEW ;ADJUST POINTERS
0304 CDA003 CALL AZERTY
0307 C3A0C7 JIP BASIC ;INTO BASIC MONITOR
030A E5 AZERTY PUSH H
030B 212603 LXI H TABLE
030E 22A702 SHLD KBTPT
0311 217500 LXI H CURIN
0314 367E MVI H ;NEV TABLE IN USE
0316 E1 POP H
0317 C9 RET
0318 E5 QERTY PUSH H
0319 21C5E8 LXI H KEYTU
031C 22A702 SHLD KBTPT
031F 217500 LXI H CURIN
0322 365F MVI H ;CURRENT CURSOR
0324 E1 POP H
0325 C9 RET
0326
0326 ;--- TABLE REMPLACANT CELLE EN ROM3 (E8C5H-E934H) ---
0326 303132 TABLE DB '0123456789:;,-.!'
0336 0D DB 0DH
0337 514243 DB 'BCDEFGHIJKLMNOP'
0347 415253 DB 'ARSTUVWXYZ[\^_`
0354 000810 DB 0H,0H,10H,11H,12H,13H
035F 212223 DB 9H,90H,0H,0H,5FH
0365 27 DB '12345678'
0366 28292A DB '()*+<=>?'
036E 0D DB 0DH
036F 716263 DB 'abcdefghijklnop'
037F 617273 DB 'rstuvwxyz[\^_`
038C 000814 DB 0H,0H,14H,15H,16H
0391 170980 DB 17H,9H,00H,0H,0H
0396 END END

```

Lock/unlock pour VC 1541

LOCK et UNLOCK pour le floppy Commodore

Article original : DAInamic Allemagne

Traduction : Frederic BACQUET



Chaque possesseur du VC-1541 connaît la signification de l'asterisque (*) devant le filetype qui signale un fichier ouvert. Le signe 'plus petit que' (<) est par contre moins courant. Comme l'asterisque, on ne peut pas l'effacer avec l'ordre SCRATCH. Cette impossibilité est malheureusement passée sous silence dans le manuel du floppy. Le but de cet article est de chercher pourquoi le DOS ne fournit pas d'instructions comme 'LOCK' et 'UNLOCK' pour permettre et interdire l'effacement de fichiers.

Le bit 6 de l'octet filetype est responsable de cette protection contre l'effacement. S'il est à 1, SCRATCH ne fonctionne pas pour ce fichier.

Le programme listé ci-dessous donne la possibilité d'envoyer dans la RAM du floppy des programmes écrits en assembleur 6502. On le démarre avec les ordres UC et US. Ceci présente contrairement à l'instruction ME (Memory Execute) la possibilité de transmettre des paramètres (dans le cas présent, un ou plusieurs noms de fichiers).

Un fichier est protégé avec COM"UC:nom-de-fichier" et déprotégé avec UD. La syntaxe est la même que celle SCRATCH, les jokers et nom de fichiers multiples peuvent aussi être utilisés. Avec l'asterisque, on peut par exemple protéger tous les fichiers. Le programme en LM est enregistré sur la disquette sous le type USR. Cette méthode présente par rapport au stockage dans un secteur et appel avec BE (Block Execute) l'avantage de ne pas perdre le programme lors d'un VALIDATE.

Malheureusement, les fichiers user limitent l'efficacité du LOCKage des fichiers, et ils ne sont pas mentionnés dans le manuel. De plus, ils devraient être utilisés pour l'autodémarrage après un RESET ou une mise sous tension, mais cette possibilité pourtant prévue dans le DOS ne fonctionne pas.

Tout d'abord au sujet de l'organisation d'un fichier user, les deux premiers octets représentent l'adresse de départ du programme : l'octet de poids faible puis celui de poids fort. Le groupe de deux octets suivant annonce le nombre d'octets qui suivent. Après les données, le checksum. Après le checksum peuvent suivre d'autres programmes avec leur adresse de départ, etc... La longueur est alors annoncée avec seulement un octet, les longs programmes devant donc être décomposés en plusieurs parties.

Un fichier user doit être envoyé dans la RAM du floppy avec un '&' (ex. COM"&UN/LOCK"). le '&' en première position sera lu lors du traitement du nom. Après chargement, le programme sera automatiquement exécuté. Dès la fermeture avec RTS, le bloc précédent sera exécuté, etc... jusqu'à ce que le premier soit atteint, ceci pouvant être désagréable dans le cas d'un long programme (voir plus haut). Il faut ou bien que le pointeur de pile soit adapté au dernier bloc, ou bien que le premier octet d'un bloc programme soit un RTS. Dans ce cas, on peut empêcher l'autostart du fichier entier.

En plus des messages d'erreur habituels, 2 nouveaux sont ajoutés lorsqu'on utilise des fichiers user :

- 50 RECORD NOT PRESENT (il y a une erreur de checksum).

- 51 OVERFLOW IN RECORD (le dernier bloc programme ne contient pas assez de données [l'octet de longueur est incorrect]).

Le fichier user est créé avec le programme Basic suivant.

Petites annonces

```
10 OPEN 1 8 2 "&UN/LOCK,U,W"
20 READ ADDR IF ADDR *FFFF THEN CLOSE 1 END
30 CHKSUM=0.BYTE=ADDR IAND *FF GOSUB 70
40 BYTE=ADDR SHR 8 GOSUB 70
50 READ BYTE GOSUB 70 FOR I=1 TO BYTE
60 READ BYTE GOSUB 70.NEXT.PUT# 1.CHKSUM.GOTO 20
70 CHKSUM=(CHKSUM+BYTE) MOD *FF.PUT# 1.BYTE.RETURN
100 DATA #610,102,*A0,*05,*B9,*70,*06,*99,*00,*05,*88,*10,*F7,*60
110 DATA #A9,*29,*8D,*56,*06,*A9,*BF,*8D,*57,*06,*4C,*33,*06,*A9,*09
120 DATA #8D,*56,*06,*A9,*40,*8D,*57,*06,*A9,*0A,*8D,*2A,*02,*20,*EE,*C1,*20
    *98,*C3
130 DATA #20,*20,*C3,*20,*CA,*C3,*A9,*00,*85,*86,*20,*9D,*C4,*30,*15,*20,*B7
    *DD,*90,*0B
140 DATA #A0,*00,*B1,*94,*EA,*EA,*20,*B9,*C8,*E6,*86,*20,*8B,*C4,*10,*EB,*20
    *10,*06
150 DATA #A5,*86,*85,*80,*68,*68,*A9,*00,*4C,*78,*C8,*4C,*29,*06,*4C,*1C,*06
    99999
```



PETITES ANNONCES

Vends DAI en panne mais reparable : 10000 Fb. KENDOS 2 x 800 K et nombreux programmes : 50000 Fb (les 2 pour 55000 Fb). Filip D'Haene, bld P. H. Spaak 6, B-7900 Leuze-en-Hainaut.

Vends KENDOS 2 x 800 K plus une quarantaine de disquettes (soit une des plus grandes programmatheques existant sur le DAI)+ en EPROM sur la carte KENDOS: COM FWP,CP/M,DISK-WIFE. Prix : 45000 Fb. Marc Vandermeersch (IDC asbl).

Vends Memocom avec cables et EPROM, poignees de jeu, processeur arithmetique, cables divers, cassettes DCR et AUDIO pleines de programmes, plusieurs kilos de documentation. PRIX tres BAS. Contactez Remi Arnaud au 55398582, ou ecrivez Chemin des Pepinieres, F-87340 La Jonchere Saint Maurice.

Vends interface VC1541 + TOS + 2 eeproms (DCR & 1541) pour 2500 Fb. Victor Nijs, rue Marchand 5/7, B-4530 Hermalle.

Vends synthetiseur vocal (decrit dans DAICLIC 7) completement monte, alimentation exterieure (pas de parasites), le tout dans un tres beau boitier. Prix : 4000 Fb soit le prix des composants. Vends Souris cablee pour le DAI : 2000 Fb. Werner Persoons, rue de la Cordialite 14, B-1080 Bruxelles.

Vends Memocom MDCR avec cables et EPROM (TOS) ou echange contre Commodore 64. Prix sacrifie. K7 DCR. Philippe Verhaege, rue de la Ture 8, B-7500 Tournai.

Vends SOURIS cablee pour le DAI + pgm de DAO : 2500 Fb. F. Duluins (IDC).

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

Functietekenaar

In dit artikel wilde ik een door mij geschreven programma bij U introduceren.

Dit programma heet 'Functie Tekenaar' en uit die naam kan al worden opgemaakt dat het een programma is dat de gebruiker in staat stelt er functies mee te laten tekenen.

In de afleveringen van de laatste DAIamics heb ik nogal uitvoerig over allerlei functies gesproken, die in de DAI standaard zijn opgenomen.

Maar er zijn natuurlijk ontelbaar veel meer functies. Die functies, die op te bouwen zijn met de standaard-functies zijn met dit programma heel eenvoudig te onderzoeken.

Het lijkt mij daarom zinnig dit programma te bespreken.

Functie Tekenaar

Het programma zal binnenkort bij DAIamic uitgebracht worden tesamen met een aantal statistiek programma's en het lineaire regressie programma dat al eerder in DAIamic werd aangekondigd.

Inleiding

Het programma FT (functietekenaar) is een krachtig programma dat, alhoewel voornamelijk in basic geschreven de gebruiker in staat stelt allerlei functies eenvoudig te onderzoeken.

Het programma is gebruikersvriendelijk opgezet. Het maakt gebruik van een aantal machinetaal routines die al tijdens de intro op hun plaats gezet worden. Het gebruik van deze routines maakt het echter wel aan te raden na gebruik van FT de DAI even te resetten alvorens iets anders te gaan doen.

Invoer

De te onderzoeken functies worden eenvoudig tijdens het programma ingetikt. Het programma geeft dan ruime edit mogelijkheden. De functies moeten worden opgegeven in de volgende vorm :

f: X - 'een of andere expressie met eventueel de variabele X'

Voor diegenen die nu op school zitten of daar niet te lang geleden op zaten de meest gebruikelijke vorm

om functies in op te geven.

Voor degenen die meer vertrouwd zijn met de vorm $f(x) = \text{'expressie'}$ en dit ook hier in dit programma graag zouden zien geldt dat dit eenvoudig zelf in basic is te wijzigen.

Het zal duidelijk zijn dat de expressie in een voor de DAI begrijpelijke vorm moet zijn.

De aldus ingevoerde functie wordt in de vorm :

100 Y='de ingevoerde expressie'

opgenomen in de listing van het basic-programma.

De ingevoerde expressie wordt als string behandeld tot hij de gewenste vorm van een basicregel heeft. Deze regel wordt vervolgens in het basic-programma opgenomen.

Hiervoor gaat mijn dank uit naar de wonderdokter Nico Looije.

Fout bij invoer

Is de ingevoerde expresie niet voor de DAI begrijpelijk grijpt een aangepaste versie van de errortrapping in en kunt u de invoer corrigeren.

Hiervoor is een soort editbuffer gecreerd zodat de editing simpel is.

Deze zelfde editbuffer kan ook als dat gewenst wordt, gebruikt worden bij de eerste keer invoeren.

In absolute noodgevallen kan voor zeer uitgebreide functievoorschriften de aanwezige ruimte op beeld ontoereikend zijn. In dat geval kan de functie altijd nog als regel 100 opgenomen worden. Zou zelfs dat niet voldoende zijn, dan kunnen alle regelnummers tussen 90 en 110 de Y stap voor stap opbouwen.

Snijpunten

Er kunnen twee functies tegelijk in het geheugen van de DAI opgeslagen worden.

Met deze mogelijkheid kunnen simpel snijpunten van twee functies in het onderzochte interval bepaald worden.

Meer dan twee is - na een tamelijk eenvoudige aanpassing - wel mogelijk maar lijkt mij niet zo nodig bij een normaal gebruik.

Intro

Het programma start met het introductieplaatje en wacht dan op

een toetsaanslag. De eerste keer wordt tijdens dit wachten een tweetal machinetaal programma's weggePOKEd.

Bij een latere run na een onderbreking zal dit niet meer geschieden.

Onderbreking

Loopt het programma om een of andere reden vast of wenst u het te onderbreken kunt U er meestal met de [BREAK]-toets weer uitkomen.

We kunnen dan herstarten met 'CONT' of 'RUN'. (geen gePOKE ditmaal)

Met 'RUN1000' om te starten met hoofdmenu in default en 'RUN1020' om te starten met hoofdmenu met behoud van de ingestelde waarden.

Het is natuurlijk mogelijk na 'RUN' een keus te bieden uit de hiervoor beschreven mogelijkheden maar aangezien het hier exclusieve toepassingen betreft heb ik daar niet voor gekozen.

De op de voorgrond staande functie blijft in alle gevallen behouden. Hij staat immers in regel 100.

De andere functie, die in een string staat zal bij de 'RUN' gereset worden.

Hoofdmenu

In het hoofdmenu krijgen we de volgende tekst in beeld :

F U N C T I E - T E K E N A A R

F.H. Druijff

Ga met de cursor naar de gewenste positie en tik daar dan de nieuwe waarde in. Nieuwe functie met 'E' of 'F' of 'G'.

Geen verandering bij onmogelijke waarde. Eindig met [RETURN]

En verder een hele lijst met keuze en instelmogelijkheden die ik de straks de revue laat passeren.

Om een nieuwe functie in te voeren kunnen we uit meerdere mogelijkheden kiezen.

'F', 'G' en 'E'

De plaats van de cursor is niet van belang. Hierop is jammer genoeg een kleine uitzondering. Als we bezig zijn met het invoeren van een van de in te stellen waarden zou daar

eventueel een van die letters in kunnen voorkomen. Zij worden dan als invoer voor die waarde beschouwd. Ik geef ter verduidelijking een voorbeeld :

U wilt de maat op de x-as gelijk aan het getal e hebben. Bij de maat voor x tikt U hiervoor EXP(1) in zodat de DAI nu deze waarde bepaalt.

Het zou dan erg lastig zijn als het programma op de 'E' van 'EXP' zou reageren om de functie te veranderen.

'F' De functie die aanwezig was wordt verwijderd uit beeld. (Op regel 100 staat hij dan nog wel !) En dan kunnen we een nieuwe functie gaan intikken.

'G' De functie die er stond wordt op de plaats van de tweede functie gezet en die komt nu in beeld. Deze functie kan dan aangepast / veranderd worden.

'E' De functie blijft in beeld staan en op een EDITbuffer-achtige manier kunt U de functie wijzigen.

De cursor springt als er geen fout gemaakt werd naar de positie terug die hij innam voor de functieverandering.

Keuzes

We krijgen in het hoofdmenu een keuze aangeboden tussen de volgende alternatieven :

assen / rooster

In het eerste geval krijgen we alleen de assen (als die tenminste in beeld staan) en de maatverdeling. Bij de keuze 'rooster' krijgen we ruitjes.

punten / lijnen

Bij de keuze 'punten' worden alleen de uitgerekende punten neergezet (als die op het beeld passen) en bij de keuze 'lijnen' worden de opvolgende punten verbonden. In combinatie met de tekennauwkeurigheid kan de teken-snelheid hiermee enorm worden opgevoerd maar we interpoleren en daar kunnen fouten door ontstaan. Vandaar mijn keuze voor default 'punten'.

geluidssignaal aan / uit

Komt bij de berekening voor een punt een fout te voorschijn (delen door nul of de wortel uit een negatief

getal bij voorbeeld) wordt bij die x-waarde zowel boven als onder in beeld een punt gezet in een afwijkende kleur. Wij worden nogeens extra op deze bijzonderheid atent gemaakt door een geluidssignaal.

Instellingen

tekennauwkeurigheid = 10.0

Dit betekent dat we om de tien horizontale BEELDpunten een punt gaan bepalen. Aangezien het programma in een gemodificeerde MODE 6A werkt hebben waarden boven de 100 weinig en waarden boven de duizend geen zin.

Hoe groter de waarde bij de tekennauwkeurigheid is hoe sneller de functie getekend zal zijn.

Waarden onder de een kunnen wel degelijk zinvol zijn om de 'slingers' van een functie goed te volgen maar minder dan 1/256ste is onzinnig. Het kost alleen maar extra tijd.

maat voor beide assen = 1.0
vermenigvuldigingsfactor
voor maat = 1.0

In feite worden alleen de maat op de x-as en de maat op de y-as in het programma gebruikt, maar voor gebruikersgemak is gekozen voor het ook opgeven van de maat voor beide assen.

Een hier opgegeven maat staat na tekenen ook niet meer op deze plaats maar is verwerkt in de maten van de afzonderlijke assen.

Voor de vermenigvuldigingsfactor heb ik bewust een wiskundig onjuiste aanpak gekozen. Als iemand zegt 'maak de maat eens tien maal zo groot' bedoelt hij meestal 'vergroot de grafiek met een factor tien' of beter gezegd 'zoom eens in' want het beeldscherm wordt niet groter. Om het door hem gewenste effect te verkrijgen moet de maat door een factor tien gedeeld worden. We tellen per hokje met 1/10de in plaats van 1 op. De maat wordt dus kleiner. Toch bleek in de praktijk dat ik en ook enige anderen zich regelmatig vergisten en verkleinden als ze wilden vergroten en omgekeerd. Daarom toch vermenigvuldigingsfactor als het deelfactor zou moeten zijn. Zowel naam als effect zijn simpel te veranderen voor gebruikers die wel wiskundig correct wensen te werken.

De maten voor alleen x-as en y-as zijn ook apart in te stellen, evenals

de vermenigvuldigingsfactoren daarvoor.

functie-translatie in de x-richting en y-richting

Met deze waarden kunnen we de gehele grafiek verschuiven. Ook hier is gekozen voor de intuïtieve notatie.

Wil ik de hele grafiek vijf hokjes naar boven verschuiven geef ik (bij maat 1 tenminste) een translatie van 5 op in de y-richting.

Hebben we geen maat 1 meer kan ik ook gewoon vijf hokjes opgeven door bij de maatgebonden translatie 5 bij de y-richting in te geven. Net zoals de vermenigvuldigingsfactoren worden deze maatgebonden translaties verrekend in de 'echte' translatie.

Onmogelijke waarden

Bij elk van de vorige waarden zijn bepaalde waarden uitgesloten.

Een maat van 0 is onzinnig.

Tikt iemand dan toch zoiets in zal deze onzin worden vervangen door de vorige waarde. Heeft iemand dus per ongeluk een waarde verminkt kan hij altijd door 0 op te geven de oorspronkelijke waarde weer terug krijgen.

Domein

Tot slot kan het domein van de functie beperkt worden om te lange berekeningstijden te voorkomen.

Deze optie is echter minder krachtig dan men zich zou wensen.

Alle gewenste beeld x'en worden omgewerkt naar 'echte' x-waarden en die worden dan gecontroleerd of ze in het domein zitten of niet. Als ze er niet in zitten wordt de bijbehorende y niet uitgerekend. Alleen hier zit er tijdswinst. Om de domeingrenzen om te rekenen naar beeldgrenzen is lastig bij monotone continue functies en onmogelijk (voor de DAI) bij functies die dat niet zijn.

Cursor

De uitgangspositie van de cursor is tussen de twee blokken in, net iets boven het midden van het beeld.

Zit de cursor daar niet maar elders druk dan op de spatiebalk, waarna de cursor naar de uitgangspositie zal gaan. Het verschijnsel komt door het 'LIST'en van de regel met de functie terwijl een toets

(al of niet bewust) werd ingedrukt.

Voor de cursor worden meerdere symbolen gebruikt. Het 'normale' teken voor de cursor is CHR\$(29). Tijdens invoer van nieuwe parameters wordt CHR\$(95) gebruikt en tijdens invoer een nieuwe functie CHR\$(11).

Bij de foutmeldingen worden weer andere cursortekens gebruikt.

Bewegen over beeld

De cursor kan simpel over het beeld heen bewogen worden in verticale richting door middel van de cursortoetsen voor op en neer.

Bevindt de cursor zich in het bovenste blok (op de drempel wordt hij wit) kan men met de cursortoetsen voor links en rechts de instelling veranderen.

Na een eventuele foutmelding komt de cursor automatisch weer op de vorige plaats terug.

Het invoeren van een nieuwe parameter doet men simpel door hem in te tikken op de juiste plaats.

Return

Met het drukken op de [RETURN]-toets verdwijnt het menu en gaat de DAI de grafiek van de opgegeven functie (trachten te) tekenen. De cursor zal op dat moment CHR\$(29) moeten zijn anders wordt de [RETURN] gezien als einde invoer.

Dit is dus een van de redenen voor de verschillende cursors.

Tekenen

De DAI maakt de grafiek van de opgegeven functie rekening houdend met de gewenste parameters.

Tijdens het tekenen staat ook de tekst 'Ik teken nu !!!' in beeld.

Dit; omdat het bij een verkeerde parameterkeuze heel erg lang kan duren voordat er iets in beeld verandert. Is de DAI klaar verandert de tekst 'Ik teken nu !!!' in een mini-keuzemenu : 'D/E/F/G/M/T/cursor'.

Submenu 1

'E', 'F' en 'G' hebben hier dezelfde betekenis als in het hoofdmenu. De functie wordt echter in splitmode opgegeven en de grafiek die we net

maakten verdwijnt dus niet.

'D' en 'M' laten ons beide naar het hoofdmenu terugkeren. Met 'D' worden echter alle parameters weer op hun default waarde gezet.

Met 'T' komt een klein kruisje in beeld en verandert het submenu in het volgende submenu : 'M/O/Y/X/Y/cursor'

Submenu 2

We bewegen met de cursor het kruisje in stappen van tien beeldpunten over het scherm. Met shift erbij bewegen we in stappen van een beeldpunt.

'M' - ga terug naar het hoofdmenu.
'O' - Plaats kruisje in de oorsprong.

In de andere gevallen gaan we de functie opnieuw tekenen. Het vorige beeld raken we hierbij kwijt.

De plaats van de grafische cursor (kruisje) wordt nu het beeldmidden.

Tevens wordt een vergrotingsfactor tien toegepast. We krijgen zo een soort loupe, die we op de functie leggen. Met 'V' worden beide maten aangepast; met 'X' alleen de x-maat en vanzelf met 'Y' alleen de y-maat.

In het eerste submenu (na tekenen) konden we ook de cursorknoppen gaan gebruiken. Doen we dit krijgen we vanzelf submenu 3 : 'cursor/T/G/M'.

Submenu 3

Met de cursortoetsen kunnen we lijnen over het scherm laten bewegen.

Deze lijnen komen van de beeldranden af en gaan naar het midden toe.

We kunnen zo een rechthoek afbakken, die als we de returntoets hebben ingedrukt, wordt opgeblazen tot een beeldvullend geheel.

Hebben we een van de lijnen per ongeluk iets te ver gezet kunnen we door de shift gelijktijdig met de cursor te bedienen de richting omdraaien.

Een vervelende bijkomstigheid van deze oplossing was dat de maten op x-as en y-as niet gelijk bleven. Door echter 'G' te geven wordt de vergroting minder sterk maar in ieder geval zodanig dat de maten gelijk blijven. Ook kan het vervelend zijn als er een translatie van de oor-

X-bus-card

" X " BUS CARD

De DAI is inwendig uitgerust met een connector waarop adres lijnen, data lijnen en diverse controle signalen uitkomen.

Deze connector wordt de "X" bus connector genoemd en is zeer geschikt voor uitbreidingsmogelijkheden. Een van deze mogelijkheden is een Eprom kaart waarbij de vrije adres ruimte van #F000 tot #F7FF gebruikt wordt voor een besturings programma b.v. voor DCR of High Speed Loader.

Door middel van z.g. bank switching kan men hier ook grotere bestanden in onder brengen.

Wij zullen echter alleen het schema voor de besturing van een Eprom van het type 2716, welke 2Kbyte kan bevatten, behandelen.

Beschrijving schema.

Het schema is zeer eenvoudig. Het bestaat uit twee IC's n.l de Eprom 2716 welke aangesloten wordt op de adres lijnen A0 t/m A10, de data lijnen D0 t/m D7 en \overline{OE} .

De selectie van de Eprom vindt plaats d.m.v een TTL bouwsteen van het type 74LS151.

Als de signalen A12 t/m A15 "H" zijn (#F0) en A11 is "L" gaat de uitgang (het signaal W) op pin 6 laag. Dit is het \overline{CE} signaal voor de Eprom en maakt deze actief (Mits \overline{OE} actief is).

De adres counter kan nu het adres bereik van 0 tot 7FF doorlopen. Dit is tot alle bits van A0 t/m A10 "H" zijn. Gedurende deze tijd kan men de 8 data bits via D0 t/m D7 uitlezen.

Wordt nu echter A11 "H", adres #FB00, dan schakelt de uitgang van de 74LS151 om op "H" omdat de ingangs signalen niet meer aan de juiste voorwaarden voldoen en maakt de Eprom inactive.

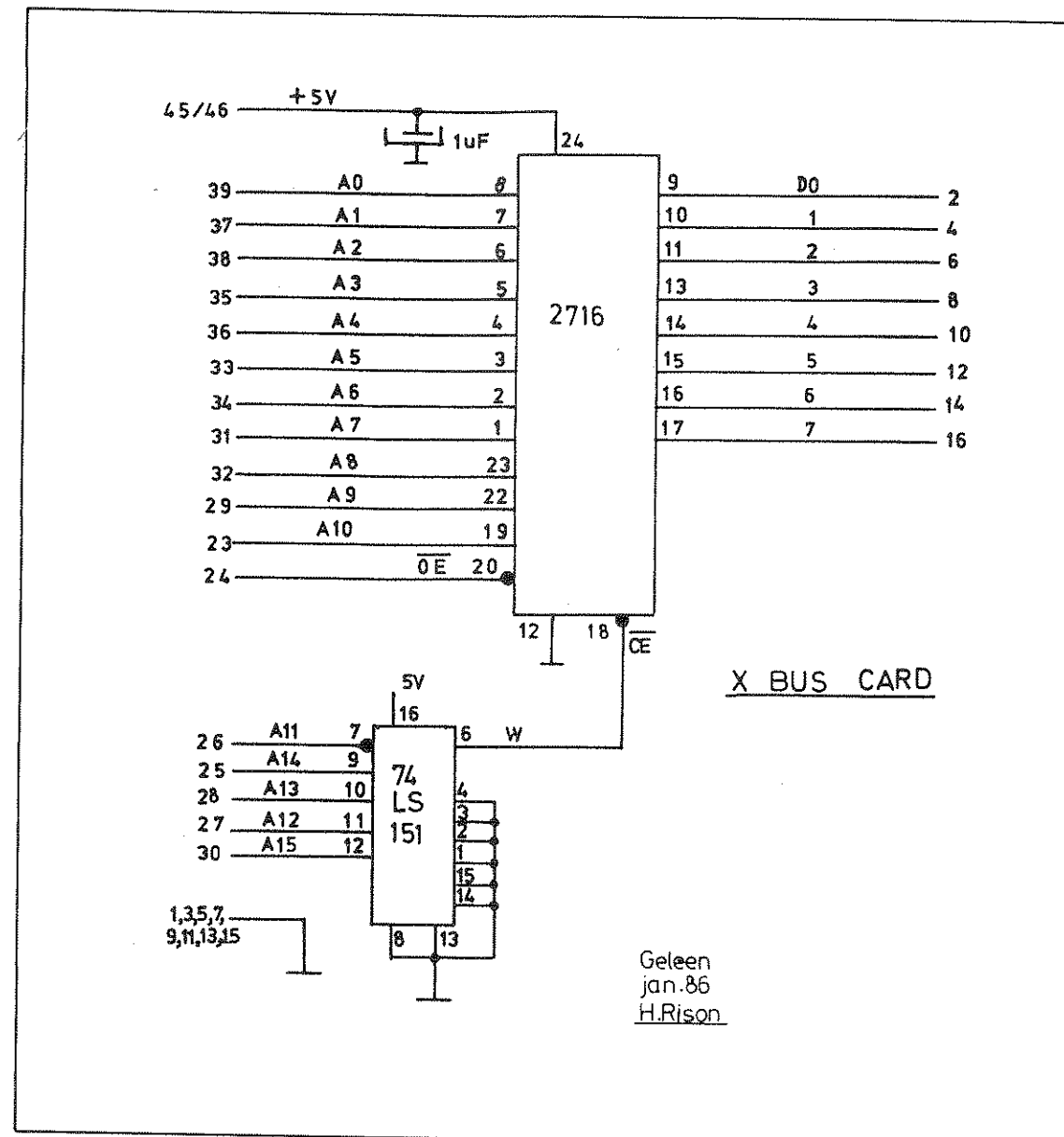
De constructie van deze kaart is niet moeilijk. De connector is meestal in een goede elektronika zaak verkrijgbaar evenals een stukje gaatjesboard twee IC voeten en een condensator.

Als U klaar bent met de bouw meet dan eerst alle signalen van de connector naar de IC voeten door en zorg ervoor dat er nergens sluiting is.

Een voortijdig einde van Uw DAI is het laatste wat ik U zou wensen.

Geleen, jan 1986
H. Rison

Tekenprogramma



Cont from p. 35

sprong plaatsvindt, maar met de keuze 'T' voorkomt men dat.

Ik weet dat het natuurlijk onlogisch is om met 'T' juist NIET transleren aan te geven maar dit onthoudt toch wel het eenvoudigst.

De volgende DAInamic een volledige aankondiging van de hele band met wat plaatjes ter illustratie.

Frank H. Druiff

Hieronder vindt u de listing van een gebruikersvriendelijk tekenprogramma. Nadat u 4 kleuren naar wens hebt opgegeven, tekent u de contouren als volgt: u stuurt met de cursortoetsen een knipperend blokje (bij SHIFT + cursor loopt het blokje met 10 stappen tegelijk). Drukt u op de spatiebalk, dan zal op de plaats van het blokje een puntje blijven staan. U stuurt het blokje verder, en als u dan weer op de spatiebalk drukt dan ontstaat er een lijn tussen de vorige en de huidige positie van het blokje. Als u klaar bent met de contouren, dan kunt u gesloten vlakken inkleuren. Dit gaat met basic-paint, dus niet zo snel. Bent u in het bezit van de machinetaal versie, dan kunt u het programma aanpassen. Andere mogelijkheden zijn: per regel de kleuren veranderen, binnen een rechthoekig vlak de contour uitwissen, de tekening ondertitelen in vergrote letters, en van mode 6A naar 6 gaan. Het gaat te ver om daar de details van uit te leggen, men probeer deze mogelijkheden zelf uit.

Summary

This is a user-friendly drawing program. You can set the contours of the drawing by using the cursorkeys (+ SHIFT: 10 steps at a time). After you finished this, there are several possibilities: PAINT, change the colours below a certain line, change to mode 6. If you want to make a photograph of the drawing, you can have it subtitled. Type it in and try it !

Aad de Bruijn
Heerenlaan 14
NL-3218 VL Heenvliet

```

10 GOTO 500:REM *** 100-180 MOVE DOT ***
100 C=SCRN(X,Y):DOT X,Y 21:WAIT TIME 2:H=GETC:DOT X,Y C:IF H=0 GOTO 100
110 IF H<16 GOTO 180:IF H>23 GOTO 180:V=H/20*9+1
120 ON H MOD 4 GOTO 140,150,160
130 Y=Y+V:IF Y<YMAX GOTO 170:Y=YMAX:GOTO 170
140 Y=Y-V:IF Y>0 GOTO 170:Y=0:GOTO 170
150 X=X-V:IF X>0 GOTO 170:X=0:GOTO 170
160 X=X+V:IF X<XMAX GOTO 170:X=XMAX
170 CURSOR 3,3:PRINT X;" ":CURSOR 3,2:PRINT Y;" "
180 RETURN:REM *** 200-260 PAINT ***
200 IF SCRN(X,Y+1)<>CK THEN Y=Y+1:GOTO 230:REM BOVEN
210 IF SCRN(X+1,Y)<>CK THEN X=X+1:GOTO 200:REM RECHTS
220 IF SCRN(X,Y-1)<>CK THEN Y=Y-1:GOTO 210:REM BENEDEN
230 IF SCRN(X-1,Y)<>CK THEN X=X-1:GOTO 220:REM LINKS
240 K=X-1:IF X=BX THEN IF Y=BY THEN IF F=1 GOTO 260:F=1
250 K=K+1:IF SCRN(K,Y)<>CK GOTO 250:DRAW X,Y K-1,Y VK:GOTO 200
260 RETURN:REM *** 300-330 DRAW LINE ***
300 GOSUB 100:IF H=13 GOTO 400:IF H<>32 AND H<>8 AND H<>9 GOTO 300
310 LK=CK:IF H=8 OR H=9 THEN LK=AK:F=9-H
320 IF F=0 THEN DOT X,Y LK:F=1:P=X:Q=Y
330 DRAW X,Y P,Q LK:P=X:Q=Y:GOTO 300
400 REM *** PREPARE PAINT ***
410 IF X<12 GOTO 950:X=12:Y=0:PRINT CHR$(12);
420 PRINT "PAINT in colour";VK1;" or";VK2;" ";;POKE #75,#FF:INPUT VK
430 POKE #75,32:IF VK<>VK1 AND VK<>VK2 GOTO 410:PRINT CHR$(12);
440 PRINT "x = 12 Move blinking dot into field to PAINT,"
450 PRINT "y = 0 then press the spacebar."
460 GOSUB 100:IF H<>32 GOTO 460:X=X+1
470 X=X-1:IF SCRN(X,Y)<>CK GOTO 470:X=X+1:BX=X:BY=Y:F=0:GOSUB 240:GOTO 1100

```

```

500 REM *** INITIALISATION ***
510 COLORT 8 0 0 15:MODE 0:PRINT CHR$(12);:POKE #75,32
520 FOR REG=0 TO 15:POKE #BFEE-REG*#86,192+REG:NEXT:POKE #B78E,#C8
530 FOR REG=3 TO 20:POKE #BFEC-2*REG,#FF:NEXT
540 FOR REG=0 TO 15:PRINT TAB(5-LEN(STR$(REG)));REG:NEXT
550 PRINT :PRINT "give the numbers of:":PRINT
560 PRINT "backgroundcolour":PRINT "contourcolour"
570 PRINT "first PAINT-colour ":PRINT "second PAINT-colour"
580 POKE #75,#FF:A$=" "+CHR$(137)+" "
590 CURSOR 20,4:INPUT AK:CURSOR 20,3:INPUT CK
600 CURSOR 20,2:INPUT VK1:CURSOR 20,1:INPUT VK2
610 COLORG AK CK VK1 VK2:MODE 6A
620 FILL 3,0 5,YMAX CK:FILL 6,0 8,YMAX VK1:FILL 9,0 11,YMAX VK2
630 GOSUB 1000:GOTO 300
700 REM *** PHOTOGRAPHY ***
710 FILL 3,0 11,YMAX AK
720 PRINT CHR$(12);"subtitle ..... 1":PRINT "MODE 6 ..... 2"
730 PRINT :PRINT "(after taking the picture press the spacebar)";
740 H=GETC:IF H=49 GOTO 760:IF H=50 GOTO 750:GOTO 740
750 MODE 6:CALLM #D6DA:GOTO 790
760 PRINT CHR$(12):INPUT "subtitle (max. 36 char.) ";T$:IF LEN(T$)>36 GOTO 760
770 PRINT CHR$(12);:G=#6A:POKE #7557,G:POKE #74D1,G:POKE #744B,G:POKE #73C5,G
780 COLORT 8 0 8 0:CURSOR 18-LEN(T$)/2,3:PRINT T$:CALLM #D6DA:COLORT 8 0 0 15
790 FILL 3,0 5,YMAX CK:FILL 6,0 8,YMAX VK1:FILL 9,0 11,YMAX VK2:GOTO 1100
800 REM *** CHANGE COLOUR ***
805 PRINT CHR$(12);"change colour of:":PRINT
810 PRINT "contour ... 1":PRINT "field ..... 2";
815 H=GETC:IF H=49 GOTO 820:IF H=50 GOTO 900:GOTO 815
820 X=12:Y=0:PRINT CHR$(12);
825 PRINT "x = 12      move the blinking dot to the bottom left corner of"
830 PRINT "y = 0      the rectangular area where the contour colour must"
835 PRINT "              be changed, then press RETURN";
840 GOSUB 100:IF H<>13 GOTO 840:XL=X:YB=Y
845 CURSOR 39,3:PRINT "top right corner of ";
850 GOSUB 100:IF H<>13 GOTO 850:XR=X:YT=Y
860 PRINT CHR$(12);"change into colour number";AK;",";VK1;" or";VK2;" ";
870 POKE #75,#FF:INPUT PK:IF PK<>AK AND PK<>VK1 AND PK<>VK2 GOTO 860
880 POKE #75,32:FOR X=XL TO XR:FOR Y=YB TO YT:IF SCRN(X,Y)=CK THEN DOT X,Y PK
890 NEXT:GOTO 1100
900 X=0:Y=0:PRINT CHR$(12);
910 PRINT "x = 0      move the blinking dot into the column (x < 12 !)"
920 PRINT "y = 0      of which the colour must be changed, and to the"
930 PRINT "              highest line of the change, then press RETURN";
940 GOSUB 100:IF H<>13 GOTO 940:IF X>11 GOTO 940
950 PRINT CHR$(12);:INPUT "change into colour number ";PK
960 POKE #75CO+Y*90,#CO+(X/3*#10)+PK:GOTO 1100
1000 REM *** MENU 1 ***
1010 PRINT CHR$(12);:POKE #75,32:X=12:Y=0
1020 PRINT "x = 12      draw line";A$;"spacebar      RETURN while x < 12:"
1030 PRINT "y = 0      erase line";A$;"char delete    colour of this field"
1040 PRINT "              reset & dot";A$;"tab & spacebar  can be changed, from"
1050 PRINT "              PAINT";A$;"(x > 11) RETURN    y to bottom screen";
1060 RETURN
1100 REM *** MENU 2 ***
1110 X=12:Y=0:PRINT CHR$(12);"PAINT ..... 1"
1120 PRINT "change colour ..... 2"
1130 PRINT "draw ..... 3"
1140 PRINT "photograph screen ..... 4";
1150 H=GETC:IF H<49 OR H>52 GOTO 1150
1160 ON H-48 GOTO 400,800,1170,700
1170 F=0:GOSUB 1000:GOTO 300

```

Programming contest: results

Hello programmer,

first of all we wish to thank all people who sent us a program to solve the problem we stated in DAInamic 32 - DAIClic 5, for the work they did searching and especially commenting the programs. All programs were very well written and commented, and it is a pity that there is always only one winner. Most of the programs were faster than we had made as examples, and some were a lot faster than we even dreamed of. Try to understand the next paragraphs, or type the following program in your computer, and you will be astonished by the speed this program can solve the problem. For a detailed description of the problem, look in DAInamic 32 (january-february 1986).

We will not describe all programs we received, since this would fill a whole editon on itself, and would interest few people. The one we are going to describe is the one who did the job the fastest of all. It is written in BASIC, so everyone should be able to make it work on his computer. If you want to read the detailed description, do not panic if you don't understand some, too mathematical, explanations. We didn't understand them too in the beginning, but if you look at the figures, you will understand.

The program uses the WARNSDORF rule to find the solution : see : RECREATIONS MATHEMATIQUES by M. KRAITCHIK, BRUXELLES, Imprim. STEVENS freres, 1930 ; this is :

- At each step, choose the square that communicates with the fewest possibles issues with the unoccupied part of the matrix .

- If at a step several squares verify this condition, one is free to choose one of these squares.

On figure 1 for example there are 3 squares possible from square 10, noted A, B, and C. The number of squares you can reach from A is 3, from B it is 6 and from C it is 5 : the rule tells you to pick location A to become square 11.

Once the program has found a solution, it uses the EULER transformation to search new solutions. This is explained in figure 2 :

Consider the circuit going from 1 to N (and N is 100 in this case), and X is one of the squares that communicates with N (100). We can then substitute the circuit 1 .. N by 1..X followed by N..(X+1) in opposite direction, which will cover the same squares as before. We also get a new location for N (= 100) and can keep going on this way.

There are also many possibilities of changing data, for example we can make a chess-like " horse move " by changing the value of var. NMOVE in line 50 and DATA in lines 90 and 100, or change the dimension of the matrix (change DMX and DMY in line 50).

That was all about the program. The following are the winners for each language class:

BASIC : Herbertz Alain : 46 sec 480/1000

PASCAL: Marc Expeels

D-BASIC: John Mitchell

mach. lang.: no volunteers

All other candidates should have got a little surprise too ;
if not, please write a note. If you wish to receive other contesting programs
than the one we described, please write a note too on the following address:

Paul Gobert
De Bergen 49
B-2241 ZOERSEL

We hope you and your DAI got a lot of fun making the
programs, and if you have other interesting computer problems, you can also
send them to that address. Maybe it is worth another contest.

Paul Gobert, Luc Maes.

PAGE 01 -- SOLUTION CONTEST

```

5 GOSUB 600

10 REM *****
20 REM * Initialisations and Limitations *
30 REM *****
40 CLEAR 1000
50 NMOVE=8:DMX=10:DMY=10
60 DIM DX(NMOVE),DY(NMOVE),A(DMX,DMY)
70 FOR I=1 TO NMOVE:READ DX(I):NEXT I
80 FOR I=1 TO NMOVE:READ DY(I):NEXT I
90 DATA -3,3,0,0,-2,-2,2,2
100 DATA 0,0,-3,3,-2,2,-2,2
110 NP=1:XP=1:YP=1:A(XP,YP)=NP
120 CL=LOGT(DMX*DMY)*1.00001+2:PF$=LEFT$("#####",CL)

130 REM *****
140 REM * Search of a valid square *
150 REM *****
160 NDIR=0:MIN=NMOVE+1
170 FOR I=1 TO NMOVE
180 1 XN=XP+DX(I):YN=YP+DY(I)
190 1 IF (XN<1 OR XN>DMX OR YN<1 OR YN>DMY) GOTO 330
200 1 IF (A(XN,YN)>0) GOTO 330

210 1 REM *****
220 1 REM * Choose the square with the fewest *
230 1 REM * moving possibilities *
240 1 REM *****
250 1 NUM=0:XT=XN:YT=YN
260 1 FOR J=1 TO NMOVE
270 2 XN=XT+DX(J):YN=YT+DY(J)
280 2 IF (XN<1 OR XN>DMX OR YN<1 OR YN>DMY) GOTO 310
290 2 IF (A(XN,YN)>0) GOTO 310
300 2 NUM=NUM+1:IF (NUM>=MIN) GOTO 330
310 1 NEXT J
320 1 MIN=NUM:NDIR=I
330 NEXT I

340 REM *****
350 REM * No valid square left or save the one found *
360 REM *****
370 IF (NDIR=0) GOTO 430
380 NP=NP+1:XP=XP+DX(NDIR):YP=YP+DY(NDIR):A(XP,YP)=NP
390 GOTO 160

```

```

400 REM *****
410 REM * Solution print *
420 REM *****
430 POKE #38,#C9:GOSUB 700:FOR J=1 TO DMY:PRINT :FOR I=1 TO DMX:PRINT
TAB(I*4);A(I,J);:NEXT I:NEXT J:POKE #38,0

440 REM *****
450 REM * Search for other solutions *
460 REM *****
470 NDIR=RND(NMOVE)+1
480 XN=XP+DX(NDIR):YN=YP+DY(NDIR)
490 IF (XN<1 OR XN>DMX OR YN<1 OR YN>DMY) GOTO 470
500 IF ((A(XP,YP)-A(XN,YN))=1) GOTO 470
510 CP=A(XN,YN)
520 FOR I=1 TO DMX

```

PAGE 02 -- SOLUTION CONTEST

```

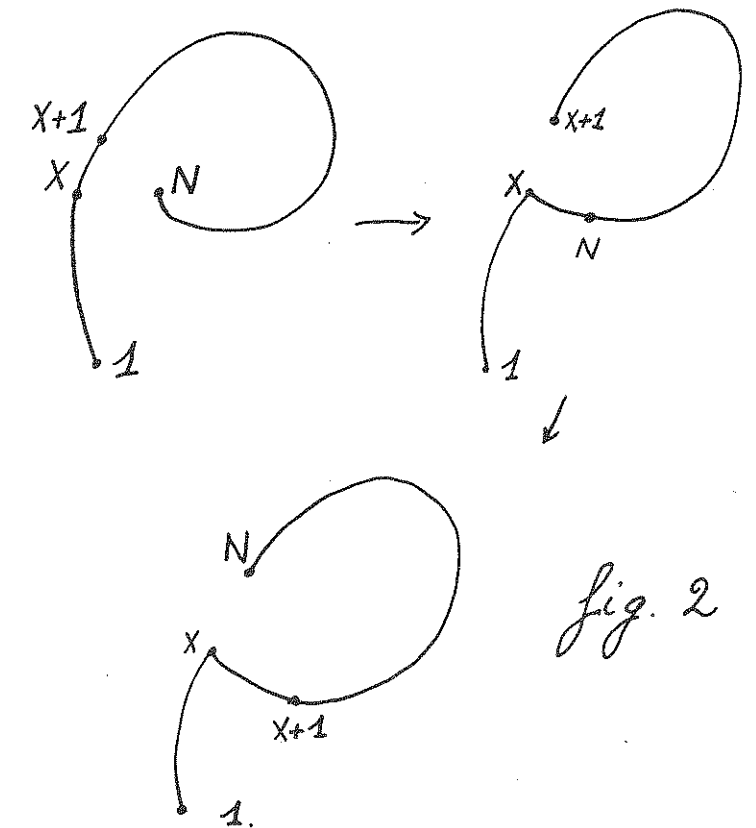
530 1 FOR J=1 TO DMY
540 2 IF (A(I,J)<>(CP+1)) GOTO 560
550 2 XP=I:YP=J
560 2 IF (A(I,J)>CP) THEN A(I,J)=NP+1+CP-A(I,J)
570 1 NEXT J
580 NEXT I
590 PRINT :GOTO 430

600 REM ***** TIMER SUBROUTINES *****
610 REM THESE WERE NOT IN THE ORIGINAL PROGRAM
620 POKE #38,#C9:POKE #1BE,#FF:POKE #1BF,#FF:POKE #38,0:RETURN
630 TIMER=(#FFFF-(PEEK(#1BE)+(PEEK(#1BF) SHL 8)))*20:CURSOR 0,15:PRINT "
Tijd: ";TIMER;" msec. ":RETURN

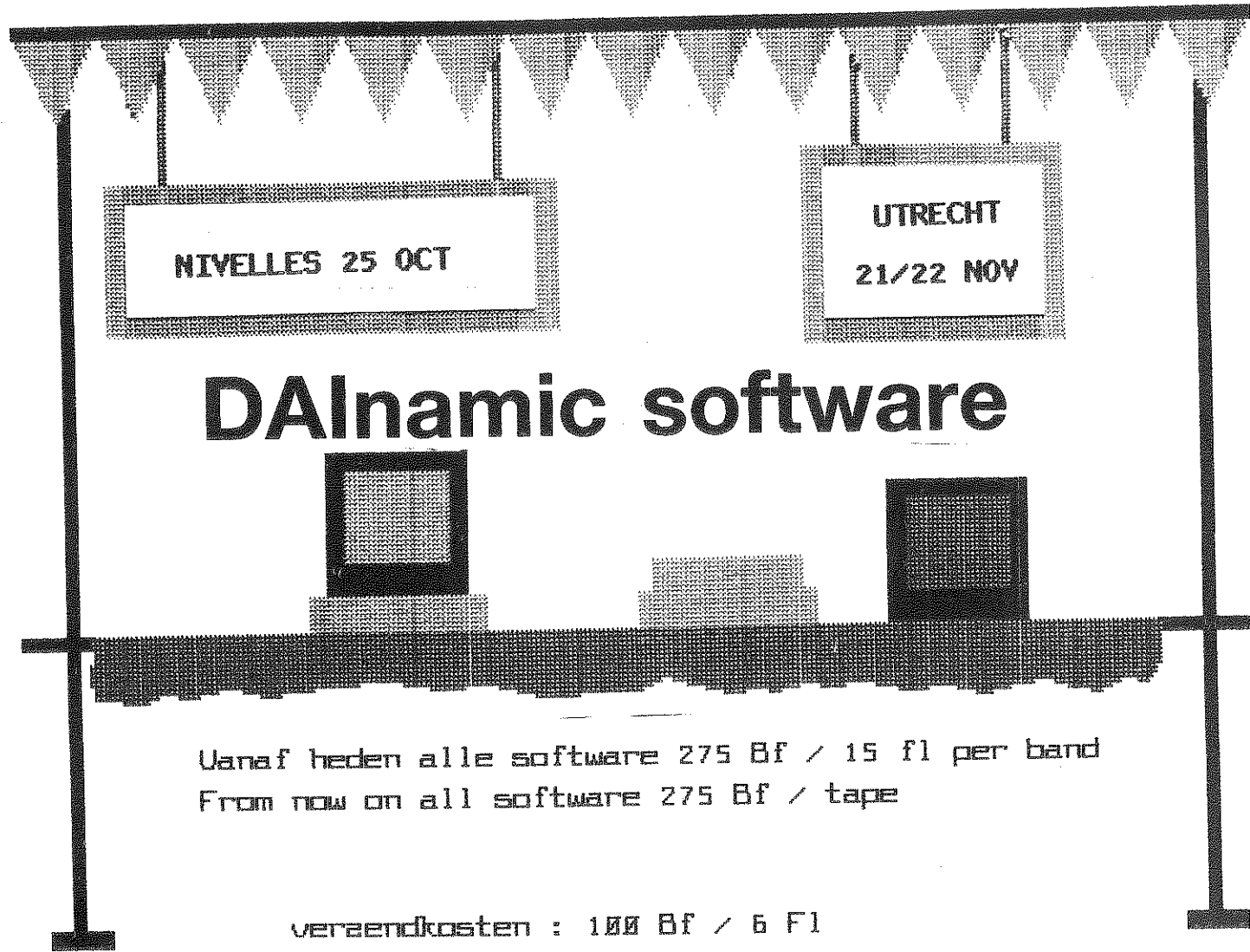
```

1		2		3			
							a1
					b2	4	
				b3			b1
				c1			(A)
			c2	b4	(B)	5	↑
					8	↖	↑
		c3		b5	(C)	←	10
				7	b6	6	
			c4				c5

fig. 1



GROTE SOFTWARE-PROMOTIE !!! SUPER SOFTWARE-PROMOTION !!



DAInamic software

Vanaf heden alle software 275 Bf / 15 fl per band
From now on all software 275 Bf / tape

verzendkosten : 100 Bf / 6 Fl

extra for DCR : 300 Bf / 16.50 Fl / tape

Op 25 oct & 21/22 nov geen verzendkosten op de beurs !

On 25 oct & 21/22 nov no mailingcosts if ordered on the meeting !!

Vooruitbetaling voor alle bestellingen , levertijd 2 weken

All orders should be paid in advance, please allow 2 weeks for delivery

SUPERFONT = 2 AUDIO / 4 DCR !

NEW SOFTWARE :

FUNCTIETEKENAAR + LINEAIRE REGRESSIE + STATISTIEK

```

SPL V1.1 PAGE 1
*****
; This routine creates a directory of a DCR-cassette. It
; gives information about the state and the length of a
; normal DAI - file on tape and gives the possibility to
; LOAD (and evt. to RUN) a BASIC or a MLP - programme,
; without leaving the directory.
; The first part is used to initiate the DAI pointers.
; After some screen organisations the cassette name will
; be printed together with the number of files and the
; number of the totally used bytes on tape. If there are
; names in the filename buffer, then the first name will
; be printed (press spacebar to print next name).
; If there are no names in the Filename buffer then "OUT
; OF DATA" is printed.
; To load a file : press "L". The routine skips to the
; right file and reads the filename. If it is the sear-
; ched name then :
; 1 : if a BASIC-file : programme load
; if a MLP-file : programme load
; if an other file : no loading, but
; 2 : jump to BASIC-monitor.
; To load and to run a file : press SHIFT + "L".
; The routine skips to the right file and reads the
; file name. If it is the searched name then
; if a BASIC-file : programme load & run
; if a MLP-file : programme load & start
; executing at first byte from the file.
; So take care : if it is a file with an-
; other startadres, a JMP stadr must be
; placed at the beginning of the file.
; if an other file: jump to the BASIC-monitor.
; If no loading was expected ( DNA or SPL sources , data
; files... ) then press "S". The routine skips "n" files
; and jump to BASIC.
; When a cassette must be updated, a name is asked.
; After put in the cassette name, the routine will read
; all files and check their data as follows :
; 1. Read filename ( = first block on tape )
; check + store it,
; 2. Read the different parts of a file
; (if an MLP file : second block = adres,
; third block = data)
; (if a BASIC file second block = text,
; third = symboltable)
; (if another -normal- file :
; second block = length,
; third block = data)
; check these parts,
; 3. and places a "+" (OK) or "-" (BAD)
; sign after the programme name.
; Only if reading a "USER"-name or a "2 END" - name the
; cassette is rewinding to the beginning of the tape to
*****
TITL 'NEW DIRECTORY DCR/1 KVDP 05/08/86 21.30'
*****
; save the directory as "USER". After saving an over-
; view of the new directory is given.
; Structure of a name in the filename-buffer:
; 1th byte : highbyte length of file
; 2th byte : lowbyte length of file
; 3th byte : length filename
; 4th byte : file-type
; 5th byte + n : text filename
; n + 1th byte : space-byte (#20)
; n + 2th byte : status byte ("+" or "-")
; n + 3th byte : CR-byte (#0D)
; n + 4th byte : 1th byte next name
; Length of the filenamebuffer : 1167 bytes. If this
; space is not enough, use DIRECTORY DCR/2.
;
; Koen Van de Perre
; Gagelweg 48
; B-2170 GOOREIND
; 03/6632008
*****
; Length Cassette-name
; BASIC-command 'RUN'
*****
; Initialisation
;
; Save number of skip files
; Temporary save area for filename
; if loading a file was request
; Temporary save area to cal-
; culate the length of the
; different blocks of a file
; Calculate length of directory
;
; Initialisation
;
; Shift-Locked key
;
; Used by saving Directory
;
*****
40H EQU
5BH EQU
28H EQU
67H EQU
6AH EQU
8CH EQU
0D5H EQU
100H EQU
100H EQU
104H EQU
113H EQU
117H EQU
118H EQU
122H EQU
126H EQU
127H EQU
180H EQU
13EH EQU
EBUF EQU
LB1k1 EQU
LB1k1+4H EQU
LB1k2 EQU
LB1k2+4H EQU
LB1k3 EQU
LB1k3+4H EQU
LUSER EQU
LIMPTAB EQU
LPHBGN EQU
LPHSIZE EQU
LSHLK EQU
LWOPEN EQU
LWBLK EQU
LWLOS EQU
POROM EQU
SPSAV EQU
LcasNm EQU
BASRUN EQU
TOPSCR EQU
MACC EQU
HIZE EQU
SVSBOT EQU
LOPVAR EQU
STKGOS EQU
RDJPF EQU
RUNFL EQU
ERSFL EQU
CONFL EQU
STACK EQU
FILES EQU
EBUF EQU
LB1k1 EQU
LB1k2 EQU
LB1k3 EQU
LUSER EQU
IMPTAB EQU
PHBGN EQU
PHSIZE EQU
SHLK EQU
WOPEN EQU
WBLK EQU
WLOS EQU
SPSAV EQU
LcasNm EQU
BASRUN EQU
TOPSCR EQU
MACC EQU
HIZE EQU
SVSBOT EQU
LOPVAR EQU
STKGOS EQU
RDJPF EQU
RUNFL EQU
ERSFL EQU
CONFL EQU
STACK EQU
FILES EQU
EBUF EQU
LB1k1 EQU
LB1k2 EQU
LB1k3 EQU
LUSER EQU
IMPTAB EQU
PHBGN EQU
PHSIZE EQU
SHLK EQU
WOPEN EQU
WBLK EQU
WLOS EQU

```

```

SPL V1.1 PAGE 2 NEW DIRECTORY DCR/1 KVDP 05/08/86 21.30
*****
; save the directory as "USER". After saving an over-
; view of the new directory is given.
; Structure of a name in the filename-buffer:
; 1th byte : highbyte length of file
; 2th byte : lowbyte length of file
; 3th byte : length filename
; 4th byte : file-type
; 5th byte + n : text filename
; n + 1th byte : space-byte (#20)
; n + 2th byte : status byte ("+" or "-")
; n + 3th byte : CR-byte (#0D)
; n + 4th byte : 1th byte next name
; Length of the filenamebuffer : 1167 bytes. If this
; space is not enough, use DIRECTORY DCR/2.
;
; Koen Van de Perre
; Gagelweg 48
; B-2170 GOOREIND
; 03/6632008
*****
; Length Cassette-name
; BASIC-command 'RUN'
*****
; Initialisation
;
; Save number of skip files
; Temporary save area for filename
; if loading a file was request
; Temporary save area to cal-
; culate the length of the
; different blocks of a file
; Calculate length of directory
;
; Initialisation
;
; Shift-Locked key
;
; Used by saving Directory
;
*****
40H EQU
5BH EQU
28H EQU
67H EQU
6AH EQU
8CH EQU
0D5H EQU
100H EQU
100H EQU
104H EQU
113H EQU
117H EQU
118H EQU
122H EQU
126H EQU
127H EQU
180H EQU
13EH EQU
EBUF EQU
LB1k1+4H EQU
LB1k2+4H EQU
LB1k3+4H EQU
LUSER EQU
LIMPTAB EQU
LPHBGN EQU
LPHSIZE EQU
LWOPEN EQU
LWBLK EQU
LWLOS EQU
POROM EQU
SPSAV EQU
LcasNm EQU
BASRUN EQU
TOPSCR EQU
MACC EQU
HIZE EQU
SVSBOT EQU
LOPVAR EQU
STKGOS EQU
RDJPF EQU
RUNFL EQU
ERSFL EQU
CONFL EQU
STACK EQU
FILES EQU
EBUF EQU
LB1k1 EQU
LB1k2 EQU
LB1k3 EQU
LUSER EQU
IMPTAB EQU
PHBGN EQU
PHSIZE EQU
SHLK EQU
WOPEN EQU
WBLK EQU
WLOS EQU

```



```

0000 @=02CE ROPEN EQU 2CEH
0000 @=02D1 RBLCK EQU 2D1H
0000 @=02D4 RCLDS EQU 2D4H
0000
0000 @=02EC HBN EQU 2ECH
0000 @=AA55 ORIG1 EQU 0AA55H
0000 @=0BFA LORG EQU 0BF69H
0000 @=B34F BufEnd EQU 0B34FH
0000
0000 @=B577 LINE3 EQU 0B577H
0000 @=B5FD LINE4 EQU 0B5FDH
0000 @=BDD7 LINE19 EQU 0BDD7H
0000 @=BF69 LINE22 EQU 0BF69H
0000 @=BFEF LINE23 EQU 0BFEFH
0000
0000 @=C7A0 RINIT EQU 0C7A0H
0000 @=C7D3 M6G1N$ EQU 0C7D3H
0000 @=C892 RUMBAS EQU 0C892H
0000 @=CEE4 SELB0 EQU 0CEE4H
0000 @=D101 SHINIT EQU 0D101H
0000 @=D121 CMPD$H EQU 0D121H
0000
0000 @=D2AD ERROR EQU 0D2ADH
0000 @=D387 RBUEX EQU 0D387H
0000
0000 @=D560 KBINIT EQU 0D560H
0000 @=D7EC BYTCUR EQU 0D7ECH
0000 @=D988 KBEI EQU 0D988H
0000 @=D9DB CLKEI EQU 0D9DBH
0000 @=DAFF PMSGR EQU 0DAFFH
0000
0000 @=DB32 PSTR EQU 0DB32H
0000
0000 @=DB53 PINT EQU 0DB53H
0000
0000 @=DC33 OTDAT$ EQU 0DC33H
0000 @=DD1F INLINX EQU 0DD1FH
0000 @=DD60 DJTC EQU 0DD60H
0000 @=DE39 DADM EQU 0DE39H
0000
0000 @=DE7C FILL EQU 0DE7CH
0000 @=DE8B RNEW EQU 0DE8BH
0000
0000 @=E8C5H KEYTU EQU 0E8C5H
0000 @=EA04 L3E159 EQU 0EA04H
0000 @=ECC1 Z2Res EQU 0ECC1H
0000 @=ED99 602 EQU 0ED99H
0000
0000 @=F012 REMIND EQU 0F012H
0000 @=F015 SKIP EQU 0F015H
0000 @=F29C CPMN2 EQU 0F29CH
0000 @=F2D3 XTL4 EQU 0F2D3H
0000 @=F31F DRCLDS EQU 0F31FH
0000 @=F348 DOPEN EQU 0F348H
0000 @=F3D5 DSTART EQU 0F3D5H
0000 @=F41D RDAAR EQU 0F41DH
0000 @=F43D ROPAMB EQU 0F43DH
0000 @=F458 R0H4R EQU 0F458H
0000 @=F900 STTOP EQU 0F900H

```

```

0000 @=FD06 PORO EQU 0FD06H
0000
0000
0000
AA55 21EC02 INIT EQU 21EC02H
AA58 229802 PHBGN EQU 229802H
AA5B 210001 SHLD LXI H
AA5E 229D02 SHLD LXI H
AA61 CD88DE CALL PHSIZE
AA64
AA64 21C5E8 LXI H KEYTU
AA67
AA67 CD40D5 CALL
AA6A CD01D1 CALL KBINIT
AA6D CD88D9 CALL SHINIT
AA70 CDDBD9 CALL KBEI
AA73
AA73 3E10 MVI A 10H
AA75 117502 LXI D IMPTAB
AA78 219002 LXI H IMPTAB+1BH
AA7B CD7CDE CALL FILL
AA7E
AA7E FB EI
AA7F
AA7F CDE4CE CALL SELB0
AA82 00 NOP
AA83 00 NOP
AA84 2100F9 LXI H STTOP
AA87 222701 SHLD STACK
AA8A F9 SPHL
AA8B
AA8B XRA A
AA8C 3F22601 STA CONFL
AA8F 67 MOV M,A
AA90 6F MOV L,A
AA91 220001 SHLD SYS60T
AA94 220401 SHLD LDPVAR
AA97 221301 SHLD STK6DS
AA9A 322201 STA ERSFL
AA9D
AA9D 211701 LXI H RDIPF
AAA0 77 MOV M,A
AAA1 3D DCR A
AAA2 23 INX H
AAA3 77 MOV M,A
AAAA
AAAA ***** Start from BASIC or UTILITY *****
AAAA4
AAAA4 BEGIN MVI A OFFH
AAAA6 RST 5
AAAA7 18 DB
AAAA8
AAAA8 21F7AD LXI H TXTCOL
AAAAE EF RST 5
AAAAF 6H
AAAD CDFFDA CALL PMSGR
AAAB FBAD DW Title$
AA2 3E6A MVI A 6AH

```

```

AAB4 32EFBF STA LINE23
AAB7 3269BF STA LINE22
AABA 210003 EQU 300H
AABD EF EQU 9H
AABE 09 EQU 9H
AABF 3EDF EQU 0DFH
AAC1 3276B5 STA LINE3-1H
AAC4 CDEFDA CALL PMSGR
AAC7 23AE DW MENU$
AAC9
AAC9 21D7BD LXI H LINE19
AACD 228A00 SHLD TOPSCR
AACF 21FD85 LXI H LINE4
AAD2 228C00 SHLD ENDSOR
AAD5
AAD5 3E0C StrtPr MVI A 0CH
AAD7 EF RST 5
AAD8 03 DB 3H
AAD9
AAD9 Print cassettename + filenames
AAD9
AAD9 LXI H 1100H
AAD9 RST 5
AAD9 DB 9H
AADE 218DAE LXI H CasNm$
AAE1 CD32DB CALL PSTR
AAE4 2186AE LXI H AanNm$
AAE7 E7 RST 4
AAE8 0C DB 0CH
AAE9 CD53DB CALL PINT
AAEC CDEFDA CALL PMSGR
AAEF 0CAE DW Files$
AAF1 218AAE LXI H Bytes
AAF4 E7 RST 4
AAF5 0C DB 0CH
AAF6 CD53DB CALL PINT
AAF9 CDEFDA CALL PMSGR
AAFC 15AE DW Bytes$
AAFE
AAFE 21D500 LXI H MACC
AB01 E7 RST 4
AB02 69 DB 69H
AB03 2189AE LXI H AanNm$+3H
AB06 7E MOV A,M
AB0A B7 STA PNames
AB0B C218AB JNZ PNames
AB0E E5 EQU EMPTY
AB0F CDFFDA PUSH H
AB12 33DC DW OTDAT$

```

```

AB14 E1 POP H
AB15 C3SFAB JMP SELECT
AB18
AB18 21COAE PNames LXI H BgnNm$
AB1B 0E00 MVI C 0H
AB1D
AB1D 23 PrName INX H
AB1E 23 INX H
AB1F 79 MOV A,C
AB20 32B001 STA FILES
AB23 3C INR A
AB24 32D800 STA MACC+3H
AB27 FE0A CPI OAH
AB29 DCBEAD CC PSpace
AB2C 7E MOV A,M
AB2D B7 ORA A
AB2E CAD5AA StrtPr
AB31 E5 PUSH H
AB32 CD53DB CALL PINT
AB33 CD8EAD CALL PSpace
AB38 E1 POP H
AB39 E5 PUSH H
AB3A
AB3A 2B DCX H
AB3B 7E MOV A,M
AB3C 32D800 STA MACC+3H
AB3F 2B DCX H
AB40 7E MOV A,M
AB41 32D700 STA MACC+2H
AB44 CD53DB CALL PINT
AB47 3E09 MVI A 9H
AB49 CD60DD CALL OUTC
AB4C 21D500 LXI H MACC
AB4F E7 RST 4
AB50 69 DB 69H
AB51 EF RST 5
AB52 0C DB 0CH
AB53 2E0B MVI L 0BH
AB55 EF RST 5
AB56 09 DB 9H
AB57
AB57 E1 POP H
AB58 228EAE SHLD NmPtr
AB5B CD32DB CALL PSTR
AB5E 0C INR C
AB5F AF SELECT
AB60 32C302 STA XRA A
AB63 CF RST 1
AB64 15 DB 15H
AB65 D439AC JC BASIC
AB68 FE20 CPI ,

```



```

ACD1 E5 PUSH H RdFile
ACD2 210000 LXI H OH
ACD5 E5 PUSH H
ACD6 010000 LXI B OH
ACD9 AF XRA A
ACDA CDCAD CALL
ACDD 214801 LXI H LB1k3+2H
ACE0 22E2AD SHLD SLB1ok+1H
ACE3 E1 POP H
ACE4 CCCAD CZ
ACE7 C2F3AC JNZ
ACEA E1 OK
ACEB CDC4AD CALL
ACEE 3E2B MVI A
ACFO C3F9AC JMP
ACF3 AC'3 BAD1
ACF4 CDC4AD CALL
ACF7 3E2D MVI A
ACF9 77 FileRD MOV M,A
ACFA CDECD7 CALL
ACFD 23 INX H
ACFE 3E0B MVI A
AD00 77 MOV M,A
AD01 CD1FF3 CALL
AD04 EF RST 5
AD05 03 DB
AD06 21D500 ChkEnd LXI H
AD09 E7 RST 4
AD0A 69 DB
AD0B AD0B POP H
AD0C 34 INR M
AD0D 117CAE LXI D USER$
AD10 CD21D1 CALL CMPD$H
AD13 C21EAD JNZ ChkNkt
AD16 AF UserRd XRA A
AD17 77 MOV M,A
AD18 CDADAD CALL REM1
AD1B C36AAD JMP WRDIR
AD1E 1176AE ChkNkt LXI D END$
AD21 CD21D1 CALL CMPD$H
AD24 F5 PUSH PSW
AD25 ;
AD25 ; *** Update name-length ***
AD25 ; This routine is used every time a file is read,
AD25 ; except the USER-file.
AD25 ;
AD25 ;
AD25 ; UpLngt PUSH H
AD26 7E MOV A,M
AD27 3D DCR A
AD28 324001 STA LB1k1+2H
AD2B 3E03 MVI A 3H
AD2D ;

```

```

AD2D ; Save endadres name
AD2D ; Check without name
AD2E 86 ADD M
AD2F 77 MOV M,A
AD2F ;
AD2F ; Check any filetype, no print
AD2F ;
AD2F ; Set for 3e blok
AD2F ; in ChkBlk-routine
AD2F ;
AD2F 214001 LXI H LB1k1+2H
AD32 CDE7AD CALL ADDByt
AD35 214401 LXI H LB1k2+2H
AD38 CDE7AD CALL ADDByt
AD3B 214801 LXI H LB1k3+2H
AD3E CDE7AD CALL ADDByt
AD41 ;
AD41 2ABEAE LHLD NMPtr
AD44 3AD700 LDA MACC+2H
AD47 77 MOV M,A
AD48 23 INX H
AD49 3AD800 LDA MACC+3H
AD4C 77 MOV M,A
AD4D ;
AD4D 21BAAE LXI H Bytes
AD50 E7 RST 4
AD51 4E DB
AD52 E7 RST 4
AD53 OF DB
AD54 ;
AD54 ; Set Name-pointer to next Filename-adres
AD54 ;
AD54 ;
AD54 ; Get namelength-adres
AD55 CD39DE UpdPtr CALL DADM
AD58 3600 MVI M OH
AD5A 22BEAE SHLD NMPtr
AD5D E5 PUSH H
AD5E 21B9AE UpAant LXI H AantNms+3H
AD61 34 INR M
AD62 E1 POP H
AD63 F1 POP PSW
AD64 C286AC JNZ RdChck
AD67 ;
AD67 CDB5AD CALL REMALL
AD6A ;
AD6A 9=0009 USER$ SET 9H
AD6A ;
AD6A ;
AD6A ; 1 (Recognize-byte) +
AD6A ; 1 (Filetype) +
AD6A ; 2 (Length filename) +
AD6A ; 4 (Filename) +
AD6A ; 1 (Checksum byte)
AD6A ;
AD6A 9=0007 USER2 SET 7H
AD6A ;
AD6A ;
AD6A ; 1 (Recognize-byte) +
AD6A ; 2 (Length blok2 (=adres)) +
AD6A ; 1 (Checksum byte) +
AD6A ; 2 (adres) +
AD6A ; 1 (Checksum byte)
AD6A ;
AD6A 9=0BFA USER3 SET BufEnd-ORIG1
AD6A ; Length Directory +

```

```

AD6A 9=0BFF USER3 SET USER3+5H
AD6A ;
AD6A 9=090F TotUsr SET USER$+USER2+USER3
AD6A ;
AD6A 21D500 WRDIR LXI H MACC
AD66 E7 RST 4
AD6E 69 DB
AD6F ;
AD6F 210809 LXI H TotUsr-4H
AD72 224C01 SHLD LUSER+2H
AD75 214E01 LXI H LUSER+2H
AD78 CDE7AD CALL ADDByt
AD7B ;
AD7B 21BAAE LXI H Bytes
AD7E E7 RST 4
AD7F 4E DB
AD80 4E RST 4
AD81 OF DB
AD82 ;
AD82 ; ***** Save directory *****
AD82 ;
AD82 ;
AD82 21594A LXI H INIT
AD85 114FB3 LXI D EndNms$
AD88 7B MOV A,E
AD89 95 SUB L
AD8A 5F MOV E,A
AD8B 7A MOV A,D
AD8C 9C SBB H
AD8D 57 MOV D,A
AD8E D5 PUSH D
AD8F ;
AD8F ; Write header
AD8F ;
AD8F 3E31 MVI A 31H
AD91 2182AE LXI H USER$
AD94 CDC502 CALL WOPEN
AD97 ;
AD97 ; Write beginadres
AD97 ;
AD97 2187AE LXI H Iniptr
AD9A 110200 LXI D 2H
AD9D CDC802 CALL WBLK
ADAA ;
ADAA ; Write data
ADAA ;
ADAA ;
ADAA D1 POP D
ADAA 2155AA LXI H INIT
ADAA CDC802 CALL WBLK
ADA7 ;
ADA7 CDCB02 CALL WCLDS
ADA4 C3A4AA JMP BEGIN
ADAD ;
ADAD ; ***** Subroutines *****
ADAD ;
ADAD ;
ADAD ; *** Rewind file(s) ***
ADAD ;
ADAD ;
ADAD 3E01 REM1 MVI A 1H
ADAF 32B001 STA FILES
ADB2 C3BAAD JMP REW

```

```

AD85 3E3F REWALL MVI A OFFH
AD87 32B001 STA FILES
AD8A CD12F0 REW CALL REWIND
AD8D C9 RET
AD8E ;
AD8E ; *** Print a space ***
AD8E ;
AD8E 3E20 PSpace MVI A ' '
ADCO CD40DD CALL QUITC
ADC3 C9 RET
ADC4 ;
ADC4 ; *** Set a space on NmPtr + update NmPtr ***
ADC4 3E20 PSpace MVI A 20H
ADC6 77 MOV M,A
ADC7 CDECD7 CALL BYTCUR
ADCA 23 INX H
ADCB C9 RET
ADCC ;
ADCC ; *** First part of Check a Blok
ADCC ; (routine starts on adres OF287H)
ADCC ;
ADCC ; ChkBlk
ADCC C5 PUSH B
ADCC F5 PUSH PSW
ADCE CD3DF4 CALL RDPAMB
AD01 F1 POP PSW
AD02 C1 POP B
AD03 CDD3F2 CALL XTHL4
AD06 00 NOP
AD07 00 NOP
AD08 00 NOP
AD09 C5 PUSH B
ADDA E5 PUSH H
ADDB 47 MOV B,A
ADDC D5 PUSH D
ADDD E5 PUSH H
ADDE CD1DF4 CALL RDRADR
ADE1 220000 SLBlk SHLD OH
ADE4 ;
ADE4 C39CF2 JMP CMPN2
ADE7 ;
ADE7 ; *** Add a number of bytes ( H,L = pointer) in MACC
ADE7 ; Change 2 bytes
ADE7 5E ADDByt MOV E,M
ADE8 23 INX H
ADE9 56 MOV D,M
ADEA 73 MOV M,E
ADEB 2B DCX H
ADEC 72 DCX M,D
ADEE 2B DCX H
ADEF E7 RST 4
ADF0 4E DB
ADF1 ;
ADF1 2189AE LXI H Plus5
ADF4 ;
ADF4 E7 RST 4
ADF5 4E DB
ADF6 C9 RET
ADF7 ;

```

DAInamic software

As announced on p.44, you can complete your DAI software library now at the bargain price of 275 Bf/ tape.

This offer concerns all the titles we released since the start of our activities. So dive into your old DAInamic issues and order now the titles you always wanted

The NEW DIRECTORY, published in this issue, is also available on cassette/DCR. DCR and audio versions are included on both media!

This is how the SUPER PROMOTION works:

- all titles should be ordered from DAInamic software Mottaart 20, 3170 Hertselt, or on one of the meetings.
- the price for one tape is 275 Bf /15 Fl.
- extra for DCR : 300 Bf/16.50 Fl
- mailing costs 100 Bf/6 Fl
- there is no mailing cost if you order on one of the meetings.
- all orders should be paid in advance.
- The prices in guiden apply for orders from J. Van Dunne Rotterdam.
- please allow 2 weeks for delivery.

```
SPL V1.1 PAGE 15 NEW DIRECTORY DCR/1 KUDP 05/08/86 21.30
***** Data buffer *****
ADF7 080008 TXTCOL DB BH,0H,BH,0CH
ADF7 0C Title$ DB 0CH
ADFB 444952 'DIRECTORY DCR/1'
ADFC 00 DB
AEOB 00 ' Files -'
AE14 00 ' Bytes used'
AE15 204279 Bytes$ DB 0DH,0DH,0H
AE20 0D0D00 DB BEH,56H
AE23 BE56 MENU$ DB '...
AE25 ZEZE DB BCH,0D9H
AE27 BCD9 ' NEXT'
AE29 204649 ' FILE L..'
AE33 BD1B DB BDH,1BH
AE35 20206C '...
AE3A BD1B 'LOAD'
AE3C 3A 'RUN'
AE3D BBF2 'S..'
AE3F 202053 DB 0B7H,95H
AE44 B795 ' U..UPDATE'
AE46 202055 DB 0H
AE51 00
AE52
AE52 Data buffer Updating part
AE52 OC DB
AE52 OD DB
AE53 InpNm$ DB
AE54 4E414D 'NAME + DATE (max 35 characters) :'
AE75 00 DB 0H
AE76 05 END$ DB 5H
AE77 322045 '2 END' DB 5H
AE7C 05 USER1$ DB 'USER'
AE7D 315553 DB 4H
AE82 04 USER$ DB 'USER'
AE83 555345 DB INIT
AE87 55AA IniPtr DB 0H,0H,0H,5H
AE89 000000 Plus5 DB
AE8D CasNm$ DB 5H
AE9D 05 DB ;length-byte
AE9E 1B DB ;Escape
AE9F 44 DB 'D' ; tabulation
AE90 0B DB DB ; on liith position
AE91 00 DB DB ; end tab.
AE92
AE92 *** From here all data is cleared
AE92 if updating Directory
AE92
AE92 0D DB
AE93 LCasNm-SH DS
AE93 DB
AE96 000000 ArgNm$ DB 0H,0H,0H,0H
AE96 000000 Bytes DB 0H,0H,0H,0H
AE9E 0000 NmiPtr DB 0H,0H
AECO ORIG2 DB $+OFH$/OFF0H ;Set next address
AECO DB ; end with 0
AECO 000000 BgnNm$ DB 0H,0H,0H
AECO DB DS BufEnd-$ ;Space for filenames
AEC3 EndNm$ END
B34F
```

Programming in machine language

PROGRAMMING THE DAI IN MACHINE AND ASSEMBLY LANGUAGES
Part 2 - SOME HELP FROM DAI's ROM

In Part 1 there was a little machine language programme to put three letters on the screen. The screen location was specified so as to keep that first example programme short and simple; we did not cocern ourselves with the whereabouts of the cursor. Now we will! Positioning the cursor during normal running is part of the DAI's 'housekeeping' duty. The section of memory (RAM) devoted to housekeeping' is from address #0000 to #02EB, plus a block from #F800 to #FFFF. Addresses #0072 and #0073 hold a pointer to the current location of the cursor; that is, the data held in #0072-#0073 is the screen RAM address of the cursor at any instant. By finding that address and loading it with a character, the character will appear on the screen at the cursor position. The instruction for loading an address, or data, held by a pointer is LHLD plus the address of the pointer. Here it would be LHLD 0072 (Hex code 2A 72 00). LHLD means Load the HL registers Directly with the value stored at the address quoted. Thus with the cursor at character position 1 on the 6th line down, the screen address is #BD49, and the operation of machine code 2A 72 00 would result in BD being loaded into register H and 49 into register L. There is no similar instruction for loading the other register-pairs with 2-byte values, via pointers. Here is a trial programme. The ORG (origin) 300 in the first line is not translated into machine code; it is an assembler directive, indicating that the object code is to be loaded into memory at #300 onwards:

Programme 1.

Label	Source	Object code	Comments
	ORG #300		Programme starts at #300
	NOP	00	No operation
	MVI A 30	3E 30	Load A with #30 (ASCII '0')
	LHLD 0072	2A 72 00	Load HL with the address that is stored at 0072-0073.
	MOV M,A	77	Move #30 from A to the screen location (address is in HL)
WAIT	JMP WAIT	C3 07 03	Wait in an endless loop.

The last instruction, labelled WAIT, is just a jump to itself, similar to a BASIC line 100 GOTO 100. Pressing the BREAK key will give escape from the loop but it also switches the DAI out of Utility and back to the BASIC mode. If the above section of code is run in Utility with G300 an '0' will appear on the screen at the cursor location but it will almost immediately be wiped out again by the cursor. To overcome that the cursor must be moved on to the next character position. That address can be obtained by decrementing the HL pair twice, stepping over the colour byte on to the next character position. That new address has to be stored in the cursor pointers #0072, #0073. The instruction SHLD 0072 (Store HL Direct) will perform this task. It stores the byte held in register L at address #0072 and the byte in H at address #0073. The Hex op code for SHLD is 22. The revised example is:

Programme 2.

Label	Source	Object Code	
	ORG #300		
	NOP	00	
	MVI A 30	3E 30	Character "0" into A
	LHLD 0072	2A 72 00	Cursor address into HL

```

MOV M,A      77      Print "0"
DCX H        2B      Step HL pair over the colour
DCX H        2B      byte to next character position
SHLD 0072    22 72 00 Store this new address at 0072/73
WAIT JMP WAIT C3 0C 03 Endless loop

```

This moves the cursor on satisfactorily but only if it is not at the end of the line; in that situation the HL pair would have to be decremented 14 times to step over Line Control Byte, Colour Control Byte and the unused character positions at the start and end of each screen line. There must also be some way of determining if the line end has been reached. A subroutine for cursor handling can therefore be quite long. Obviously there is already one in the DAI which is automatically called whenever anything is typed to the screen. There is also a routine which takes a character from register A and puts it on the screen, just as we have done in our machine code programming examples, and it calls in the cursor handling subroutine. It is in ROM starting at address #DD60 and ending at #DD6F. As it is a subroutine we call it with the instruction CALL #DD60 (CD 60 DD). When it has completed its job it reaches #DD6F where there is a RET (C9) instruction to return control to our programme. The next programme example uses that routine to put 10 digits on the screen, but as it also introduces another new instruction, a 'conditional call' some comment on that must come first.

Until now we have used straightforward jumps, calls or returns to branch to another part of our programmes; the JMP, CALL and RET instructions used were 'unconditional'. There are many occasions when branching depends upon some condition being met; for those occasions conditional jumps, conditional calls or conditional returns are used. There are 24 of them listed in the 8080 Instruction Set. Here are the most commonly used ones in mnemonic form, with the hex op codes in brackets:

Jumps	Calls	Returns	Conditions for branching
JZ (CA)	CZ (CC)	RZ (C8)	When the Zero flag is set
JNZ (C2)	CNZ (C4)	RNZ (C0)	When the Zero flag is Not set
JC (DA)	CC (DC)	RC (D8)	When the Carry flag is set (denoting register overflow caused by a carry or a borrow).
JNC (D2)	CNC (D4)	RNC (D0)	When the Carry flag is Not set
JM (FA)	CM (FC)	RM (FB)	When the Sign flag is set (denoting Minus)

The flags referred to are in the Flag register. In effect each flag is a single bit register; if the bit is 1 the flag is said to be 'set' and if the bit is 0 the flag is 'not set' (restored). In general, arithmetic and logical instructions set or restore flags in accordance with the results of their operations, whereas moving and loading instructions do not.

Programme 3.

Label	Source Code	Object Code	Comment
	ORG #300		Start at #300
	NOP	00	
	MVI B 0A	06 0A	Counter. Load B with 10 (#A)
	MVI A 30	3E 30	Load A with ASCII 0
PRINT	CALL #DD60	CD 60 DD	Call 'Output-a-character' routine
	INR A	3C	A+1=ASCII code of next digit.
	DCR B	05	Subtract 1 from counter
	JNZ PRINT	C2 05 03	If counter <> 0 print next digit
	HLT	76	If counter is 0 then stop.

Explanation: The accumulator A is loaded with the ASCII code for digit 0 which is put on screen by PRINT. The value in A is then incremented to make 31, the hex ASCII code for digit 1. Register B is used to count 10 digits printed. It is initially loaded with #A (10) and decremented after each print by the DCR B instruction. The JNZ (Jump when Not Zero) instruction checks the Zero flag; if the flag is not set the programme loops back to PRINT and the next digit is printed. The loop-back is repeated until the DCR B finally reduces the count in register B to 0 and consequently sets the Zero flag. At that stage the condition for the jump is no longer met and the programme goes to the next instruction HLT. Both INR (Increment a Register) and DCR (Decrement a Register) instructions set the Sign and Zero flags, if appropriate, but do not affect the Carry flag. Where one flag-setting instruction follows another as in this programme, the second overwrites any flags set by the first. Thus the conditional instruction is only checking flags which result from the DCR B operation, not from the INR A.

With the programme entered into RAM starting at #300 the Utility Display command D300 30F should produce the following:-

```
0300 00 06 0A 3E 30 CD 60 DD 3C 05 C2 05 03 76 00 00
```

Initialize with Z3 and run it with G300. The HLT instruction stops the microprocessor when 10 digits have been printed; use the RESET button to start the micro again.

There is another way of checking when the last digit is reached without using a register as a counter. The value in register A can be 'compared' to see if it has reached 3A, that is, beyond 39 the hex ASCII code for 9. A suitable instruction is CPI (Compare Immediate data) which compares the immediate data with the contents of the accumulator and sets the flags as if the immediate data had been subtracted from the accumulator. CPI does not change the value in the accumulator as would an actual subtraction (SUI). Here is the changed programme:

Programme 4.

```

ORG #300
NOP
MVI A 30      3E 30
PRINT CALL #DD60 CD 60 DD
INR A        3C
CPI 3A      FE 3A      Is data in A=#3A ?
JNZ PRINT   C2 03 03   If A<>3A print next digit.
HLT        76         Stop when A=3A

```

The setting of the Zero flag determines whether the programme loops back or continues. But CPI sets the Carry flag too so our JNZ PRINT could be replaced by JC (Jump if Carry) PRINT (DA 03 03) with no obvious change in results. Remember that as far as the flags are concerned, CPI 3A subtracts #3A from the value in register A. Therefore if A contains less than #3A there will be a 'borrow' and a borrow sets the Carry flag. There will obviously be no borrow and hence no Carry if A equals or is greater than #3A. JNZ would jump if the value in A is either less or greater than #3A.

The DD60 routine can also be used in a machine code programme to clear the screen by calling it to print #0C thus:-

```

MVI A 0C      3E 0C      This is the same as the BASIC
CALL #DD60    CD 60 DD    command PRINT CHR$(12)

```

It is now time to introduce a couple more ROM routines. The one already used (DD60) to print a character from the A register is usually known by the assembler label OUTC (Output a Character). Most of the often used ones have easy to remember labels which can be found in "The DAI pc Firmware Manual" if you have one. Without that manual it would be worthwhile building up your own list or annotating Colin Hards' ROM Index which appeared in DAInamic 18. Many ROM calls can be found in assembly language programmes published in DAInamic. A useful routine is GETC (Get a Character from the keyboard) which is at #D6BE. It checks for input and returns with either:

- (a) the ASCII value of the key pressed in register A, or
- (b) the Carry flag set if the BREAK key is pressed, or
- (c) the Zero flag set if no key has been pressed.

The routine does not wait for a key to be pressed; the programme must do that by recalling D6BE until the Zero flag changes from the set to unset condition, eg:-

Programme 5.

```

ORG #300
NOP
KEYBD  CALL #D6BE    CD BE D6    Get character from keyboard
RC      RC          DB          Return when BREAK key pressed
JZ KEYBD  CA 01 03    Loop back if no key pressed, else
CALL #DD60 CD 60 DD    print character in reg A on screen
JMP KEYBD C3 01 03    Go back for next character

```

The programme monitors the keyboard and displays characters typed on the screen. The only exit from the programme is by means of the BREAK key which sets the Carry flag, allowing the instruction RC (return on carry) to be effected. Return will be to the main programme from whence this was called. In this example return is to the Utility mode because the programme is called with the Utility G300 command. Instead of RC we could use JC xxxx, so that control of the programme is directed to a more obvious address; for example, JC END would be a conditional jump to a routine labelled END that displays an "End of programme" message.

There is a ROM routine which will print a message. It is called PMSG and can be found at address #DAD4. The message has to be already stored, in ASCII code, at a known address. That address is loaded into the HL register pair before calling PMSG. The routine runs through the message a byte at a time, calling the OUTC routine to print each character it finds. The last byte of the message must be 00 (not ASCII 0); when PMSG finds that, it stops printing and returns control to the main programme. You can see the routine with the Utility Display facility by typing DDAD4 DAES. Try dis-assembling the machine code displayed (converting the hex bytes back into assembly code). One can learn quite a lot about machine language programming by seeing how the experts wrote the routines hidden away in the DAI's ROM! You will find one instruction in the displayed routine that we have not dealt with yet: it is B7 at address DAD7. Hex code B7 is ORA A which is a Logical operation to "OR" the value in register A with itself. That looks futile because anything ORed with itself does not change. However OR operations do flag their results with Sign and Zero (but not Carry) and that is why this particular instruction is there. ORA A will set the Zero flag only if A is empty (contains 00). That is how the routine looks for the 00 denoting the end of the message.

All the various OR instructions operate on the accumulator, and leave the result there. They are:

```

ORI data      OR the Immediate data with A
ORA register  OR the quoted register with A
ORA M         OR the Memory (addressed by HL) with A

```

Exclusive-OR instructions are similar and flag their results with Sign and Zero flags. Their mnemonics are XRI data, XRA register and XRA Memory. XRA A is frequently used; it "exclusive ORs" the value in register A with itself, thus zeroing the register. Some assemblers use an alternative mnemonic ZAR (Zero A Register) instead of XRA A but the hex op code for both is AF.

The next programme demonstrates how the three ROM routines can be used to put a programme menu on the screen and determine which option the user chooses. Note that the menu message includes the ASCII characters #0C and #0D to clear the screen and to move the cursor down a line.

Now is the time to load your assembler so that you can write this programme in assembly language. If you are using the DNA assembler give the Buffer Dimension as 5 (for 5Kbyte); if using SPL answer the Start query with 4000. The following programme is written with the SPL assembler. The object code is placed by the ORG instruction at #6000 onwards, an area of memory that is clear of both DNA and SPL assemblers. The source instructions TITL, EQU (Equates), DB (data byte) and END are all assembler directives. The semicolon is the same as a BASIC REM. The addresses of the ROM routines being used are given to the assembler by the EQU commands.

Users of DNA will need to make the following changes when typing-in the source code:-

- (a) Use an asterisk instead of the semicolon in the Label column.
- (b) Omit semicolons at start of Comments column.
- (c) Insert a colon before Hex addresses and values.
- (d) Omit the H after Hex addresses and values.
- (e) Replace DB + data by DATA + data.
- (f) Replace DB + string in quotes by ASC + string in quotes.

Programme 6.

Addr	Object code	Label	Instrn	Operand	Source code	Comment
0000		TITLE	TITL	'FILE WRITER/WORD PROCESSOR'		
0000						
0000						
0000						
0000	@=DD60	OUTC	EQU	0DD60H		;Output a character
0000	@=D6BE	GETC	EQU	0D6BEH		;Get character from keyboard
0000	@=DAD4	PMSG	EQU	0DAD4H		;Print a message
0000						
0000			ORG	6000H		;Programme starts at #6000
6000	212660	MENU	LXI H	MENU\$;Load HL with address of menu
6003	CDD4DA		CALL	PMSG		;Display menu on screen.
6006	CDBED6	KEYBD1	CALL	GETC		;Get key pressed
6009	DA0060		JC	MENU		;If BREAK key pressed
600C	CA0660		JZ	KEYBD1		;No key pressed, so loop back
600F	FE31		CPI	'1'		;Key 1 pressed ?
6011	CAE560		JZ	LOAD1		;Yes, go to Load routine
6014	FE32		CPI	'2'		;2 pressed ?
6016	CAE660		JZ	SAVE1		;Yes, go to Save routine
6019	FE33		CPI	'3'		;3 pressed ?
601B	CAE760		JZ	EDIT1		;Yes, go to Edit routine
601E						; etc, etc.....
601E	FE55		CPI	'U'		;U pressed ?
6020	CA74EA		JZ	0EA74H		;Then switch to Utility
6023	C30660		JMP	KEYBD1		;If any other key pressed
6026						

```

6026 0C0D0D MENU$ DB 0CH,0DH,0DH ;#C clears screen,
6029 ; #D is Carriage Return
6029 202020 DB ' M E N U '
603F 0D0D DB 0DH,0DH
6041 20204B DB ' Key 1 TO LOAD A FILE FROM CASSETTE'
6067 0D DB 0DH
6068 20204B DB ' Key 2 TO RECORD CURRENT FILE ON
CASSETTE'

6094 0D DB 0DH
6095 20204B DB ' Key 3 TO DISPLAY OR EDIT THE FILE'
60BA 0D0D DB 0DH,0DH
60BC 206574 DB ' etc...'
60C3 0D0D DB 0DH,0DH
60C5 20204B DB ' KEY U TO SWITCH TO UTILITY'
60E3 0D00 DB 0DH,0H ;Note last byte=00
60E5 ;
The following lines are temporary. Until the various subroutines have
been written, jumps to the LOAD, SAVE and EDIT labels will find NOP
instructions; control will step over them to the LXI H instruction that
displays the 'Not Available' message.
60E5 ;
60E5 00 LOAD1 NOP
60E6 00 SAVE1 NOP
60E7 00 EDIT1 NOP
60E8 21F960 LXI H NAVMsg ;Load HL with addr of message,
60EB CDD4DA CALL PMSG ;Print message
60EE CDBED6 KEYBD2 CALL GETC ;Get character from keyboard
60F1 FE20 CPI ' ' ;Is it a space ?
60F3 C2EE60 JNZ KEYBD2 ; No, loop back to look again
60F6 C30060 JMP MENU ; Yes, go back to menu
60F9 0C0D0D NAVMsg DB 0CH,0DH,0DH
60FC 205072 DB ' Programme not available.'
6115 205072 DB ' Press space-bar to return to menu...'
613A 00 DB 0H ;00 signifies end of message
613B ;
613B END END

```

You may notice that the object code resulting from DB with string messages looks rather odd. That is because the assembler only lists the first three data items although all data in the DB statement will be assembled into memory, as you can see by checking the instruction addresses; eg: the difference between 60FC and 6115 is the same as the number of characters (including spaces) of the message ' Programme not available.'

When your source programme is in the assembler check it for errors with the assembler's error check command, ? with SPL or #E. with DNA. That should find any syntax errors but it will not find things like wrong addresses which so often result in a self-destruct routine. Therefore save your source code on cassette before assembling it. We will extended it in subsequent articles in this series. When assembled the programme can be run in Utility by Z3 (Return) G6000 (Return). The machine code programme can be saved as a type 1 file with the Utility Write command. It is customary when saving type 1 files to add the address of the programme after the file name as a personal reminder of its location in memory. eg: W6000 613A ..FILE WRITER (#6000-#613A)

RGB-monitor digitaal

RGB MONITOR (DIGITAAL)

De vorige keer heb ik de mogelijkheid beschreven om zelf een analoge monitor op de DAI aan te sluiten. Deze maal zullen we bekijken hoe dit mogelijk is met een digitale monitor.

Een digitale monitor heeft normaal vieringangssignalen voor kleur generatie n.l. Rood, Groen, Blauw en een Intensity signaal. Met dit laatste kan men de intensiteit van een kleur beïnvloeden. Dit signaal kunnen we voor de DAI niet gebruiken.

We hebben zijn dus beperkt tot maar 8 kleuren, zwart meegerekend.

Buiten deze voor de kleuren noodzakelijke signalen hebben we nog een horizontaal en een vertikaal synchronisatie signaal nodig. Deze kunnen positieve of negatief zijn. In verschillende typen monitors is dit d.m.v. een schakelaar in te stellen.

De aansluiting aan de monitor is meestal een 9 polige Canon connector waarvan helaas de aansluitingen niet altijd hetzelfde zijn.

De aansluitingen op het schema zijn die van een door ons gebruikte monitor.

Omdat pin 9 hierin niet werd gebruikt hebben we deze genomen om het audio signaal naar de monitor te brengen.

Beschrijving schema.

Als we het schema vergelijken met die van de analoge kaart dan valt ons op dat we gebruik maken van dezelfde Prom waarvan de inhoud gelijk gebleven is. De signalen hiervan gaan naar een 74LS374.

Inplaats van alle uitgangen te gebruiken nemen we er nu maar drie.

Pin 6 voor het "R" (rood) signaal, pin 15 voor het "G" (groen) signaal en pin 19 voor het "B" (blauw) signaal. Een extra buffer is niet noodzakelijk daar de ingang van de monitor TTL niveau is.

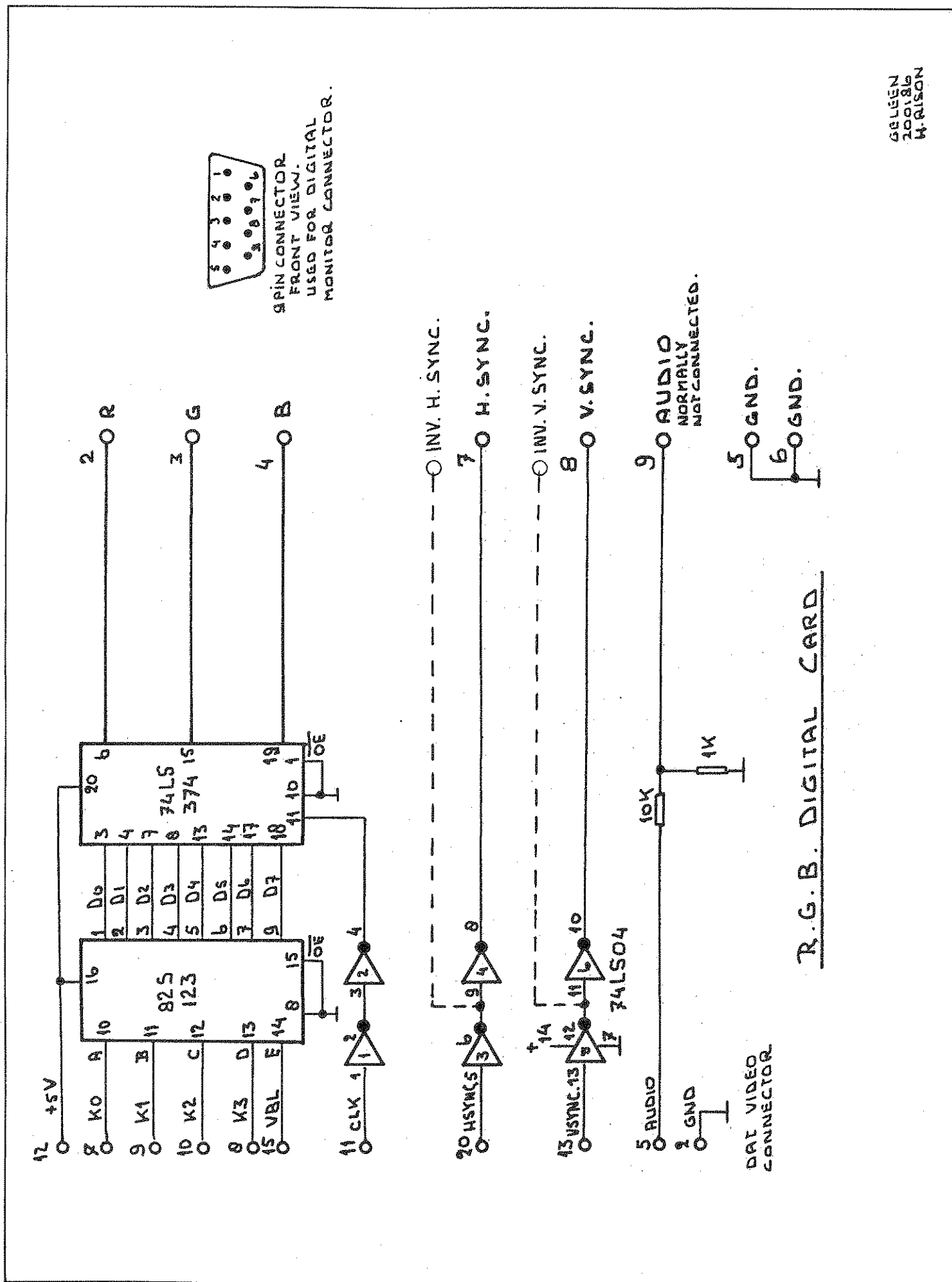
De beide synchronisatie signalen worden verkregen door ze twee keer te inverteren en naar de monitor te sturen. Moet het een inverted sync. zijn dan direkt na de eerste inverter afnemen.

De twee overgebleven inverters gebruiken we voor de Clock pulse naar de 74LS374.

De signalen zijn allen terug te vinden op de connector van het main board naar de video print.

Pin 2 = Gnd.	Pin 10 = K2
Pin 3 = -5V	Pin 11 = Clk.
Pin 5 = Audio	Pin 12 = +5V
Pin 7 = K0	Pin 13 = V sync.
Pin 8 = K3	Pin 15 = VLB
Pin 9 = K1	Pin 20 = H sync.

Gelezen, febr. 1986
H. Rison



```

XX      XX  DDDDDDDD  000000  SSSSSSS
XX      XX  DDDDDDDD  00000000  SSSSSSSSS
  XXXX    DD    DD 00    00  SS    SS
    XX    DD    DD 00    00  SSSSSSSSS
  XXXX    DD    DD 00    00    SS
XX      XX  DD    DD 00    00  SS    SS
XX      XX  DDDDDDDD  00000000  SSSSSSSSS
XX      XX  DDDDDDDD  000000  SSSSSSS
  
```

***** XDOS a disk operating for KENDOS *****

Why a new operating system for KENDOS hardware ?
 When I bought my KENDOS drives, the only advantage was the increased speed of loading and saving. The operating system was very minimal and had several points which lead to a decreased compatibility with existing hardware and software.

The KENDOS operating system changed system addresses, aborted programs (I came to like LOADING ERRORS, these could be trapped). The spaghetti like syntax with &,%,+/,",0123 and drivenumbers which could be placed anywhere. Drive 1 2 and 3 were stepchildren and working from these drives was possible but every KENDOS instruction had to be changed. The choice of filetypes became very limited and confusing. Assembler programs written as 'DBS' could not be changed back to 'SRC' unless you knew the BIOS routines. Before calling BIOS routines tables and pointers had to be set up. Programs were read and written in one block which lead to problems with programs with two long blocks. A hardcopy of the directory was a complete mess. The 1.5k memory used by KENDOS is better than DAIDOS but problems often occurred when overwriting this buffer. Disk, audiocassette and MDCR could not be used at the same time without extra commands. The other advantages of a disk operating system namely random and sequential access were not so random respectively uhhh?

In XDOS these disadvantages have been corrected/overcome. The speed of the system has not been decreased due to the fact that the diskformat has not been changed.

The main goals for the development of XDOS have been:

- user-friendliness
- increased software compatibility
- easy interfacing disk routines
- true random and sequential access possibilities.

DIFFERENCES BETWEEN KENDOS V3.X & XDOS V1.0X

A list of the most important changes and enhancements.

- 1 XDOS has its own 8 KByte RAMspace
 - 1.5 K extra user RAM (no DAI RAM used)
 - increased compatibility (no relocation of programs)
 - no more problems with pictures
 - no buffer adjustment

- 2 A new command interpreter/processor with:
 - easier syntax with abbreviations
 - hexadecimal and decimal numbers
 - direct and indirect addressing
 - upper and lowercase commands
 - easy errortrapping
 - 3 BDOS supports MDCR and AUDIO storage
 - no special switching (DCR/CAS) is needed
 - enhanced MDCR commands
 - copying of files requires no special program
 - 4 enhanced filenames
 - 15 instead of 14 characters
 - less illegal characters
 - provisions for grouping of filenames wildcard characters
 - 5 all filetypes remain unchanged
 - the standard filetypes are used (0 1 2 etc.)
 - filetypes may be defined by the user
 - 6 default filemode
 - filemodes select the type and number of background memory
 - once a filemode has been selected it stays selected and if not explicitly overruled all commands act on this filemode.
 - 7 generic commands
 - because grouping files is possible by means of filemode filename and filetype commands work on selected groups of files instead of one or all files.
 - conversational commands for extra security and ease.
 - 8 XDOS fully supports the DAI two blockstructure
 - blocks are loaded and saved separately
 - each block may be 64Kbyte.
 - 9 XDOS fully supports DAI loading saving protocol
 - LOADING direct displays the filename
 - SAVEing direct asks for confirmation
 - standard errorhandling in any program
 - suppression of errors during LOADING and SAVEing is possible
 - return to caller is always possible.
 - extended saving protocol. The Zeroflag is set on succesful completion of WOPEN, WBLOCK & WCLOSE.
 - extended loading protocol. The Carryflag is now also set on succesful completion of ROPEN .
 - 10 XDOS supports true RANDOM files
 - randomfiles may be as big as the space on a diskette
 - recordlength may be from 1 to 1024 bytes.
 - each random file may have 65535 records.
 - 11 XDOS supports sequential files
 - sequential files may be as big as the space on a diskette.
 - all characters are legal no file markers
 - 12 INPUT/OUTPUT redirection.
 - easy handling of input from keyboard, files and memory.
 - easy handling of output to screen, files and memory.
 - 13 programmable functionkeys
 - keys or ASCII values from the input source can be redefined to give a string of ASCII characters which may of course be commands.
 - 14 diskettes can be read and written manually
 - blocks of 1 byte to 64Kbyte can be read and written manually anywhere on disk
 - 15 BDOS calls
 - all information is documented
 - all data is passed in the registers
 - all commands return to caller with carry set on succesfull completion if unsuccessfull the A register contains the returncode
 - 16 extensive status display
 - display of system flags, date, PFkeys, commands, diskette-status, memory allocation and opened files
 - 17 jobs
 - Ascii command files to configure/set up program environment etc.
 - 18 XBASIC
 - the most important XDOS commands and random and sequential access are available in runtimecode. all programs are upward compatible with DAI BASIC.
 - all XDOScommands can be issued from XBASIC.
 - commands for reading writing sequential files
 - BASIC programs can be saved and loaded as ASCII files
 - commands for random records with numeric and alphanumeric fields, string formatting commands
 - XBASIC control structures support structured programming. WHILE,UNTIL,FOREVER, repetitive and selective control stuctures.
 - modulair programming with subprograms to be called as procedures with VALUE and ADDRESS parameters and LOCAL variables.
 - programs are automatically precompiled for speed and structural errors are immediatly detected.
 - XBASIC supports a structured listing with indentation for control structures (also in EDITmode).
- XDOS/XBASIC comes as a package including:
1. XDOS operating system (24 Kbyte) in 2 27128 eproms. The system adds over 50 XDOS commands. XBASIC language (8 Kbyte) with over 40 new commands. A modification must be made to set 2 sockets for 27128 eproms (one scratch and two jumpers).
 2. 8 Kbyte static RAM used for buffering. A modification must be made to enable writing into the RAM (a scratch and a wire).

3. A diskette with manual and conversion utility for old diskettes and some demo programs.
4. Old KENDOS system may be ordered optionally in 1 16K eeprom fit for the third socket (Dfl 30,-- extra)

The price for the complete package is Dfl 150,-- and can be ordered from

N.P. Looije
 Paludanushof 22
 3151 CM Hoek van Holland
 Nederland.

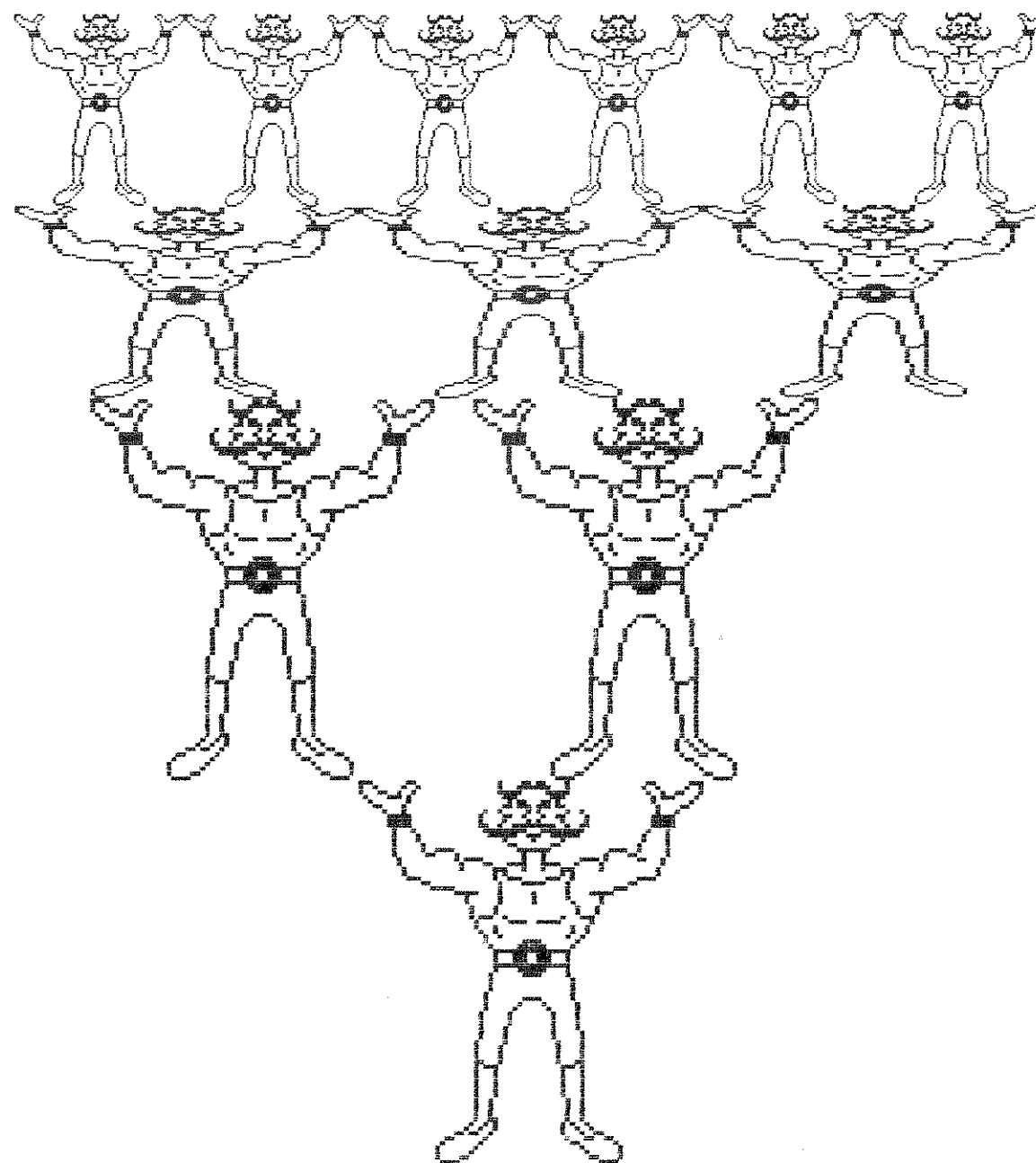
The price is to be paid in advance by means of an international postal money order. Please indicate 'XDOS' on your order. Dutch members can transfer this amount on bank account 32.32.77.314 of N.P. Looije. Allow 28 days for delivery.

```

10 GOTO 100
: REM Wallpaper N.P.LOOIJE 4/84
20 MAX=RND(10)
: FOR A=0 TO MAX
:   A(A)=RND(256)
: NEXT
30 FOR A=#BFFF TO #B350 STEP -MAX-1
40   FOR B=0 TO MAX
:   POKE A-B,A(B)
:   NEXT
50   IF GETC<>0 THEN 70
:   NEXT
60 IF GETC=0 THEN 60
70 MODE 0
: PRINT CHR$(12)
: CURSOR 0,0
: GOTO 20
100 MODE 0
: PRINT CHR$(12)
110 CURSOR 21,15
: PRINT "Wallpaper-program"
120 PRINT TAB(21);"-----"
130 PRINT TAB(10);"You don't have to be a first-class"
140 PRINT TAB(10);"machinelanguage-programmer to create"
150 PRINT TAB(10);"wallpaper patterns by inventing a"
160 PRINT TAB(10);"good crash routine."
170 PRINT TAB(10);"Use the keyboard to start and restart"
180 DIM A(10)
: GOTO 60

```

SUPERFONT



SUPERFONT ... SOURCE OF IMAGINATION!

SUPERFONT .. SOURCE OF IMAGINATION!

SUPERFONT .. SOURCE OF IMAGINATION!

SUPERFONT .. SOURCE OF IMAGINATION!

SUPERFONT .. SOURCE OF IMAGINATION!

SUPERFONT .. SOURCE OF IMAGINATION

SUPERFONT .. SOURCE OF IMAGINATION