Partie 5

Graphisme

5/0

Table des matières

5/1

Généralités

Les AMSTRAD CPC possèdent trois modes d'affichage qui seront utilisés en fonction du type de l'application.

MODE 0 : 25 lignes de 20 caractères, 16 couleurs, définition 160 × 200 pixels.

MODE 1 : 25 lignes de 40 caractères, 4 couleurs, définition 320 × 200 pixels.

MODE 2 : 25 lignes de 80 caractères, 2 couleurs, définition 640 × 200 pixels.

Aucune distinction n'est faite entre l'écran texte et l'écran graphique. Il existe même une instruction (TAG) qui permet d'afficher un texte alphanumérique à une position graphique (au pixel près) sur l'écran.

L'écran est divisé en 400 points verticaux et 640 points horizontaux respectivement repérés de 0 à 399 et de 0 à 639.

En MODE 0, comme la résolution est de 160 × 200, un point élémentaire occupe (640/160 =) 4 pixels horizontaux et (400/200 =) 2 pixels verticaux.

En MODE 1, comme la résolution est de 320 × 200, un point élémentaire occupe (640/320 =) 2 pixels horizontaux et (400/200 =) 2 pixels verticaux.

En MODE 2, comme la résolution est de 640 × 200, un point élémentaire occupe (640/640 =) 1 pixel horizontal et (400/200 =) 2 pixels verticaux.

A la mise sous tension, le MODE d'affichage est le MODE 1.

5/2

Tracé de points en BASIC

En BASIC, nous disposons de trois instructions pour accéder à un point élémentaire :

PLOT, PLOTR et GRAPHICS PEN.

PLOT <Abcisse>, <Ordonnée>[, <encre>[, <mode d'encre>]]

— <Abcisse> est compris entre 0 et 639,

— <Ordonnée> est compris entre 0 et 399.

Cet ordre permet d'afficher un point élémentaire aux coordonnées absolues spécifiées.

PLOTR <Déplacement en X>, <Déplacement en y>
[,<encre>[, <mode d'encre>]]

Cet ordre permet d'afficher un point élémentaire aux coordonnées relatives (par rapport au point courant) spécifiées.

Si X, Y est la position courante du curseur graphique, un point sera affiché en X+ <Déplacement en X>, Y+ <Déplacement en Y>.

GRAPHICS PEN [<encre>][, <Mode d'affichage du fond>]

Cet ordre n'existe pas sur CPC 464.

Si vous possédez un CPC 664 ou 6128, GRAPHICS PEN vous permettra de choisir une couleur de stylo (<encre>) pour dessiner, et la nature du fond de l'écran (<Mode d'affichage du fond> = 0 pour un fond opaque, et = 1 pour un fond transparent.)

5/3

Déplacement du curseur graphique en BASIC

En BASIC, nous disposons des instructions MOVE, MOVER, ORIGIN, XPOS et YPOS.

MOVE <abscisse>, <ordonnée>[, <encre>[<mode d'encre>]]

— <abscisse> est compris entre 0 et 639,

— <ordonnée> est compris entre 0 et 399.

Cet ordre permet de déplacer le curseur graphique aux coordonnées absolues spécifiées sans afficher le point élémentaire pointé.

Le paramètre <encre> permet, s'il est présent, de changer la couleur du stylo graphique.

Le paramètre <mode d'encre> permet, s'il est présent, de fixer le mode d'affichage graphique :

0 pour le mode normal,

1 pour le mode XOR,

2 pour le mode AND,

3 pour le mode OR.

MOVER <déplacement en X>, <déplacement en Y>
[,<encre>[<mode d'encre>]]

Cet ordre permet de déplacer le curseur graphique aux coordonnées relatives (par rapport au point courant) spécifiées sans afficher le point élémentaire pointé.

Le paramètre <encre> permet, s'il est présent, de changer la couleur du stylo graphique.

Le paramètre <mode d'encre> permet, s'il est présent, de fixer le mode d'affichage graphique :

0 pour le mode normal,

1 pour le mode XOR,

2 pour le mode AND,

3 pour le mode OR.

ORIGIN <abscisse>, <ordonnée>[, <gauche>, <droite>, <haut>, <bas>]

Cet ordre fixe les coordonnées du point qui sera pris comme origine de l'écran graphique. Par défaut, l'origine est en 0, 0.

Quand les paramètres <gauche>, <droite>, <haut>, <bas> sont précisés, ils définissent les dimensions de la fenêtre graphique.

XPOS indique l'abscisse courante du curseur graphique (entre 0 et 639).

YPOS indique l'ordonnée courante du curseur graphique (entre 0 et 399).

5/4

Tracé de droites en BASIC

En BASIC, nous disposons de quatre instructions relatives au tracé de droites : DRAW, DRAWR, GRAPHICS PEN et MASK.

DRAW <Abscisse>, <Ordonnee>[, <encre>[, <mode d'encre>]]

— <Abscisse> est compris entre 0 et 639,

— <Ordonnée> est compris entre 0 et 399.

Cet ordre permet de tracer un trait de la position courante du curseur graphique à la position absolue spécifiée.

DRAWR <Deplacement en X>, <Déplacement en Y> [, <encre>[, <mode d'encre>]]

Cet ordre permet de tracer un trait de la position courante du curseur graphique à la position relative (par rapport au point courant) spécifiée.

Si la position courante du curseur graphique est X,Y, le trait prendra fin en X+<Deplacement en X>, Y+<Deplacement en Y>.

GRAPHICS PEN [<encre>][, <Mode d'affichage du fond>]

Cet ordre n'existe pas sur CPC 464.

Si vous possédez un CPC 664 ou 6128, GRAPHICS PEN vous permettra de choisir une couleur de stylo (<encre>) pour dessiner, et la nature du fond de l'écran (<Mode d'affichage du fond> = 0 pour un fond opaque, et = 1 pour un fond transparent.)

MASK [<entier>][, <premier point tracé>]

— <entier> est compris entre 0 et 255,

— <premier point tracé> vaut 0 ou 1.

Cet ordre permet de définir le mode de tracé des lignes fabriquées à partir des ordres DRAW ou DRAWR.

Le paramètre <entier> est un nombre sur 8 bits où chaque bit représente un pixel. Si un bit est à 1, le pixel correspondant est allumé. Le pixel est éteint si le bit est à zéro.

Si le paramètre <premier point trace> vaut 0, le premier point n'apparaîtra pas. Le premier point apparaîtra si ce paramètre vaut 1.

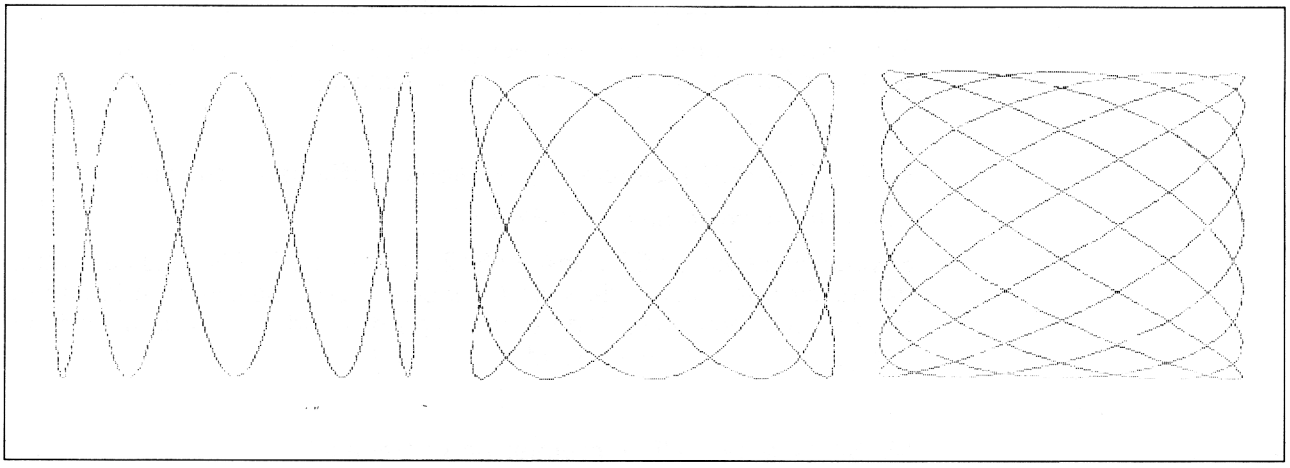
Pour illustrer

- le tracé de droites avec l'ordre DRAW,
- le positionnement du curseur avec l'ordre MOVE ou PLOT.

Nous vous proposons une série de programmes de tracé de courbes de type Y(t), X(t).

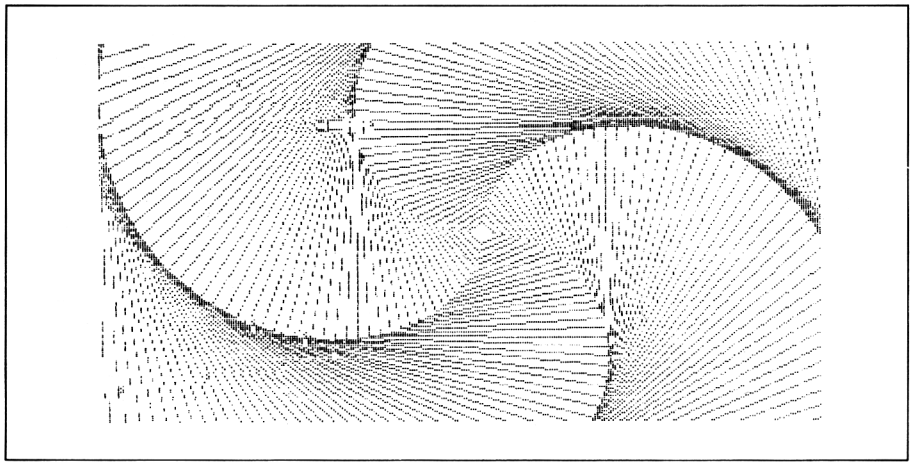
a) Courbes de Lissajous

```
10 CLS
20 FOR F=0.2 TO 4 STEP 0.2
30    PLOT 490,200
40    A=0
50    A=A+0.1
60    DRAW 300+190*COS(A*F), 200+190*SIN(A)
70    IF INKEY$ = "" THEN 50
80    CLS
90 NEXT
```



b) Rectangles tournants

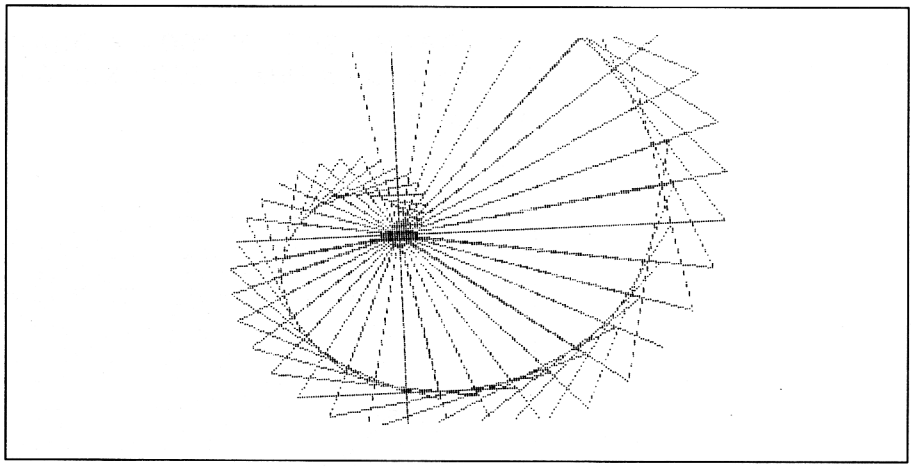
```
10 CLS:C=1
15 PLOT 290,200
20 FOR I=12 TO 600 STEP 8
30   A=1/200
40   X=I*SIN(A)
50   Y=I*COS(A)
70   DRAW 300-Y,200+X:DRAW 300-X,200-Y:DRAW 300+Y,200-X:DRAW 300+X,200+Y
80 NEXT
```

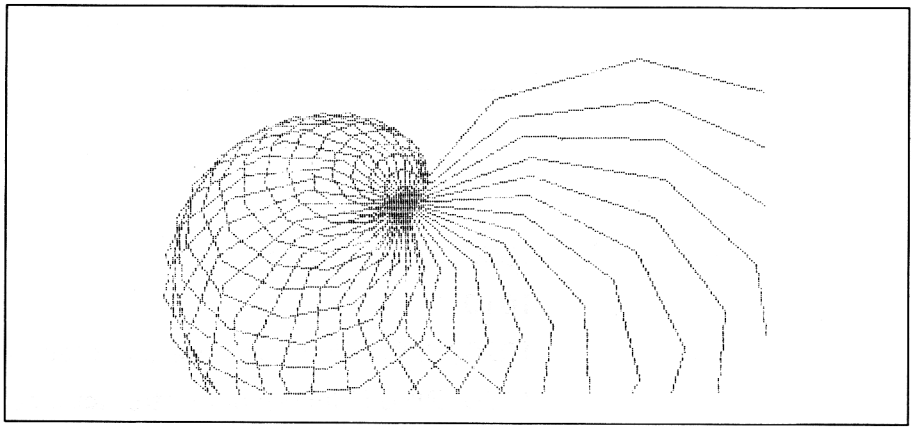


c) Spirales

```
1000 REM Spirales
1010 '
1020 CLS:INPUT''Nombre de cotes'';NC
1030 INPUT''Increment entre deux cotes'';IC
1040 INPUT''Increment d'angle'';IA
1050 DEG:CLS:C=1:U=0 'Cote initial et angle de depart
1060 MOVE 320,200
1070 'Boucle sans fin
1080 MOVE 320,200
1090 FOR N=1 TO NC
1100     AN=(360/NC)*(N-1)
```

```
1110     X=C*COS(U+AN)
1120     Y=C*SIN(U+AN)
1130     DRAWR X,Y
1140 NEXT N
1150 C=C+IC:U=U+IA
1160 GOTO 1070
```





5/5

Test de la couleur d'un point en BASIC

Deux commandes BASIC permettent de connaître la couleur d'un point de l'écran. Il s'agit de TEST et TESTR.

TEST (<Abscisse>, <Ordonnée>)

— <Abscisse> est compris entre 0 et 639,

— <Ordonnée> est compris entre 0 et 399.

Déplace le curseur aux coordonnées absolues spécifiées, et renvoie la couleur du point élémentaire situé à cet endroit.

TESTR (<Déplacement en X>, <Déplacement en Y>)

Déplace le curseur aux coordonnées relatives (par rapport aux coordonnées courantes) spécifiées, et renvoie la couleur du point élémentaire situé à cet endroit.

5/6

Tracé de points et de droites en assembleur

L'équivalent des ordres BASIC MOVE, MOVER, PLOT, PLOTR, DRAW et DRAWR existe dans les ROM du FIRMWARE. Reportez-vous à la partie 4 chap. 2.7 pour avoir plus de détails à ce sujet.

Pour utiliser les programmes élémentaires situés dans le FIRMWARE (couramment appelés routines), il suffit de charger les registres demandés en entrée, et d'appeler le programme élémentaire.

Par exemple, pour appeler la routine « MOVE ABSOLUTE » située en #BBC0, il faudra faire :

En assembleur :

```
LD      DE, 100   ;Abscisse X = 100
LD      HL, 120   ;Ordonnee Y = 120
CALL    #BBC0     ;Appel de MOVE ABSOLUTE
```

En BASIC :

```
1000 FOR I=0 TO 9
1010  READ A:POKE &9000+I,A
1020 NEXT I
1030 DATA &11, 100, 0, &21, 120, 0, &CD, &C0, &BB, &C9
1040 CALL &9000
```

ou encore **PLOT 100, 120**

Les routines équivalentes aux PLOT, MOVE et DRAW BASIC sont les suivantes :

MOVE absolu :

CALL #BBC0

avec, en entrée, DE qui contient l'abscisse absolue (entre 0 et 639),
HL qui contient l'ordonnée absolue (entre 0 et 399).

MOVE relatif :

CALL #BBC3

avec, en entrée, DE qui contient le déplacement signé en abscisse,
HL qui contient le déplacement signé en ordonnée.

PLOT absolu :

CALL #BBEA

avec, en entrée, DE qui contient l'abscisse absolue (entre 0 et 639),
HL qui contient l'ordonnée absolue (entre 0 et 399).

PLOT relatif :

CALL #BBED

avec, en entrée, DE qui contient le déplacement signé en abscisse,
HL qui contient le déplacement signé en ordonnée.

LINE absolu :

CALL #BBF6

avec, en entrée, DE qui contient l'abscisse absolue du point final,
HL qui contient l'ordonnée absolue du point final.

LINE relatif :

CALL #BBF9

avec, en entrée, DE qui contient le déplacement signé en X du point final,
HL qui contient le déplacement signé en Y du point final.

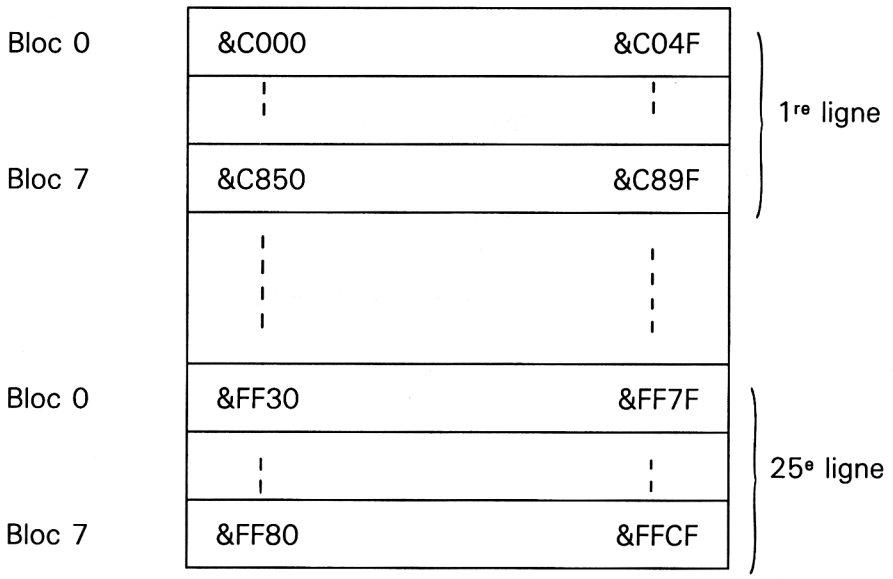
5/7

La mémoire d'écran

Une autre manière d'utiliser l'écran graphique est d'écrire dans la mémoire d'écran.

L'écran peut être considéré comme une mémoire RAM implantée entre les adresses &C000 et &FFFF, et donc, directement adressable par le micro-processeur. Quel que soit le mode d'affichage, la mémoire d'écran est divisée en huit blocs. Une bloc représente une ligne élémentaire (d'épaisseur un point élémentaire, et de largeur 80 octets).

Les blocs sont répartis comme suit :



Le bloc i (i compris entre 0 et 7) représente la ième ligne élémentaire de chacune des 25 lignes de l'écran, et fait donc 80 × 25 = 2000 octets. Pour des raisons de commodité de manipulation, un bloc a été défini sur 2^{12} octets (2048 octets), et les 48 octets supplémentaires de chaque ligne sont inutilisés.

Comme nous l'avons vu plus haut, la dimension en pixels d'un point élémentaire dépend du mode de résolution.

Dans le MODE 0, un octet comprend 2 points élémentaires de 4 pixels de large chacun.

Les octets sont codés comme suit :

Bit	0	1	2	3	4	5	6	7
Stylo	8	8	2	2	4	4	1	1
Point élémentaire	0	1	0	1	0	1	0	1

Ce tableau est à interpréter de la façon suivante :

— Si, par exemple, le bit 2 est à 1, le point élémentaire 0 sera allumé avec la couleur de stylo 2.

— De même, pour avoir le point 1 allumé avec la couleur 6, il faudra positionner à 1 les bits 3 et 5.

En MODE 1, un octet comprend 4 points élémentaires de 2 pixels de large chacun.

Les octets sont codés comme suit :

Bit	0	1	2	3	4	5	6	7
Stylo	2	2	2	2	1	1	1	1
Point élémentaire	0	1	2	3	0	1	2	3

Enfin, *en MODE 2*, un octet comprend 8 points élémentaires d'1 pixel de large chacun.

Les octets sont codés comme suit :

Bit	0	1	2	3	4	5	6	7
Stylo	1	1	1	1	1	1	1	1
Point élémentaire	0	1	2	3	4	5	6	7

En BASIC, on pourra afficher un ou plusieurs points sur l'écran par l'instruction **POKE <Adresse écran>, <Valeur sur 8 bits>.**

En assembleur, il faudra faire :

```
LD    A, <Valeur>
LD    (Adresse ecran),A
```

5/7.1

Effets vidéo sur CPC

Peut-être avez-vous remarqué dans certains jeux ou utilitaires, des effets vidéo agréables, au demeurant fort simples, mais qui produisent toujours leur effet de surprise.

Nous vous proposons dans ce chapitre de créer quelques instructions sous forme de RSX qui vous permettront d'effectuer des présentations intéressantes sur l'écran de votre micro-ordinateur préféré.

Ces nouvelles instructions, utilisent pour la plupart les registres du composant s'occupant de la gestion de l'écran des CPC, c'est-à-dire le 6845 de chez Motorola. Nous vous conseillons à cette occasion de relire les pages de la Partie 2 Chapitre 3.2 traitant de ce composant.

PROGRAMMATION DU 6845

Après avoir pris connaissance des registres du 6845, vous savez maintenant que pour accéder à l'un des registres de programmation de celui-ci (registres de contrôle R0 à R17), il vous faut donner le numéro de ce registre dans le registre d'adresse AR.

Le registre d'adresse AR se trouve à l'adresse numéro **&BC00**, le (ou les) registre(s) de contrôle se trouvant à l'adresse **&BD00**.

Ainsi en Basic, pour charger le registre de synchronisation horizontal avec la valeur **&4F**, il faudra entrer :

```
OUT &BC00,2     :   REM NUMERO DU REGISTRE R2
OUT &BD00,&4F   :   REM MODIFICATION
```

Cela donne, en assembleur :

```
LD      A,2             ;       NUMERO DU REGISTRE
LD      BC,0BC00H       ;       ADRESSE AR
OUT     (C),A           ;       ACCES R2
LD      A,04FH          ;       VALEUR
LD      BC,0BD00H       ;       ADRESSE REGISTRE CONTROL
OUT     (C),A           ;       MODIFICATION
```

Les nouvelles instructions Basic

I INIT6845

Comme nous allons modifier les caractéristiques des registres du 6845, nous avons, en premier lieu, pensé à créer une instruction permettant de reconfigurer l'état d'initialisation de ce composant.

Cette instruction vous permettra aussi, lorsque vous voudrez expérimenter quelques effets personnels, de restaurer l'écran qui sera peut-être perdu.

I SCROLLPRINT

Cette instruction dont la syntaxe est la suivante :

I SCROLLPRINT,positionx,positiony,@TEXTE$

permet d'afficher à la position **positionx,positiony** (comme **LOCATE positionx,positiony**), le message situé dans la variable alphanumérique TEXTE$.

Rien de bien original nous direz-vous ! Si ! Le message s'affichera caractère par caractère, avec un effet de scrolling arrière surprenant.

I ROULEH

Cette instruction fait disparaître le contenu de l'écran de votre CPC avec un effet de rouleau horizontal.

Il est possible de restaurer instantanément le contenu de celui-ci grâce à l'instruction I INIT6845.

Une utilisation intéressante consisterait à créer cet écran suite à l'utilisation de I ROULEH, c'est-à-dire en mode non visualisation, puis de le faire apparaître instantanément.

I DEROULEH

Cette instruction effectue le mouvement inverse sur l'écran du CPC et restitue le contenu caché de l'écran en le déroulant.

I TREMBLE

Cette instruction provoque le tremblement de l'écran dans le sens vertical.

L'arrêt de l'effet de cette instruction s'effectue par l'utilisation de I INIT6845.

I ONDULE

Cette instruction enivre votre écran en provoquant une légère ondulation du contenu de celui-ci.

Vous pourrez accentuer ou adoucir celui-ci en modifiant le réglage du bouton de synchronisation de votre moniteur.

Pour annuler cet effet, un seul remède : I INIT6845.

I ECRASE

Cette instruction provoquera un effet d'écrasement de toute l'image située sur l'écran. Effet qui pourra faire croire à un défaut sur l'ordinateur d'un de vos amis.

Là encore, on restitue l'état initial de l'écran à l'aide de I INIT6845.

I DESTROY

Pour aller plus loin dans la dégradation fictive de votre écran, voici une instruction qui modifie complètement la synchronisation de l'écran et qui en surprendra certainement plus d'un !

Le service après-vente vous conseille d'entrer, en mode aveugle, ou à tâtons, l'instruction I INIT6845.

Le programme Assembleur

L'écriture de ce programme créant de nouvelles instructions passe automatiquement par l'assembleur, mais, rassurons tout de suite les programmeurs férus du Basic, nous leur fournirons le chargeur adéquat.

Le listing assembleur est le suivant :

```
                        ;************************
                        ;*                      *
                        ;*  INSTRUCTIONS RSXs   *
                        ;*     PROPOSANT DES    *
                        ;*    EFFETS   VIDEO    *
                        ;*                      *
                        ;************************
                        ;
                        ;
                        ;*** TABLE DES EQUIVALENCES ***
                        ;
                        ;INSERTION DES RSX
                        ;       KL-LOG-TEXT
                        KLOGTX:     EQU  0BCD1H
                        ;
                        ;REGISTRE D'ADRESSE 6845
                        ;       ADRESS REGISTER
                        AREG:       EQU  0BC00H
                        ;
                        ;REGISTRES DE CONTROLE 6845
                        ;       CONTROL REGISTER
                        CREG:       EQU  0BD00H
                        ;
                        ;VECTEUR POUR MODE ECRAN
                        ;       SCR-GET-MODE
                        SGMODE:     EQU  0BC11H
                        ;
                        ;POSITIONNEMENT CURSEUR
                        ;       TXT-SET-CURSOR
                        TSCURS:     EQU  0BB75H
                        ;
                        ;AFFICHAGE CARACTERE
                        ;       TXT-OUTPUT
                        TXTOUT:     EQU  0BB5AH
                        ;
                        ;
                        ;************************
                        ;
                        ;
                        ;************************
                        ;*   EMPLACEMENT  DU    *
                        ;*       PROGRAMME      *
                        ;************************
                        ;
                                    ORG  0A000H
                        ;
                                    LOAD 0A000H
                        ;
                        ;
                        ;**********************
                        ;* INSERTION DES RSXs *
                        ;**********************
                        ;
                        DEBUT:      EQU  $
A000 210FA0                         LD   HL,KERNEL          ;ARESERVER POUR RSX
A003 0113A0                         LD   BC,VECTEU          ;TABLE INSTRUCTIONS

A006 3EC9                           LD   A,0C9H             ;POUR EVITER UN
A008 3200A0                         LD   (DEBUT),A          ;NOUVEL APPEL

```

```
 61 A00B CDD1BC              CALL KLOGTX             ;INITIALISE RSXs
 62
 63 A00E C9                  RET                     ;FIN
 64                ;
 65                ;
 66                ;*** RESERVE POOUR KERNEL ***
 67                ;
 68                KERNEL:     EQU  $
 69 A00F 00000000            DEFB 00,00,00,00
 70                ;
 71                ;*** TABLE INSTRUCTIONS ***
 72                ;
 73                VECTEU:     EQU  $
 74 A013 2DA0                DEFW TABLE
 75                ;
 76                ;*** TABLE DES SAUTS ***
 77                ;
 78 A015 C368A0              JP   IN6845
 79 A018 C390A0              JP   ROULE
 80 A01B C3ABA0              JP   DEROUL
 81 A01E C3C7A0              JP   TREMBL
 82 A021 C3D6A0              JP   ECRASE
 83 A024 C3E5A0              JP   SCRTXT
 84 A027 C384A1              JP   DESTRO
 85 A02A C393A1              JP   ONDULE
 86                ;
 87                ;
 88                ;*** SYNTAXE INSTRUCTIONS ***
 89                ;
 90                TABLE:      EQU  $
 91                ;
 92 A02D 494E4954            DEFB "INIT684"
 92 A031 363834
 93 A034 B5                  DEFB "5"+080H
 94                ;
 95 A035 524F554C            DEFB "ROULE"
 95 A039 45
 96 A03A C8                  DEFB "H"+080H
 97                ;
 98 A03B 4445524F            DEFB "DEROULE"
 98 A03F 554C45
 99 A042 C8                  DEFB "H"+080H
100                ;
101 A043 5452454D            DEFB "TREMBL"
101 A047 424C
102 A049 C5                  DEFB "E"+080H
103                ;
104 A04A 45435241            DEFB "ECRAS"
104 A04E 53
105 A04F C5                  DEFB "E"+080H
106                ;
107 A050 5343524F            DEFB "SCROLL"
107 A054 4C4C
108 A056 5052494E            DEFB "PRIN"
109 A05A D4                  DEFB "T"+080H
110                ;
111 A05B 44455354            DEFB "DESTRO"
111 A05F 524F
112 A061 D9                  DEFB "Y"+080H
113                ;
```

```
114 A062 4F4E4455             DEFB "ONDUL"
114 A066 4C
115 A067 C5                   DEFB "E"+080H
116                 ;
117                 ;
118                 ;**********************
119                 ;*                      *
120                 ;*   EXECUTION  DES     *
121                 ;*    DIFFERENTES       *
122                 ;*    RSXs  VIDEO       *
123                 ;*                      *
124                 ;**********************
125                 ;
126                 ;
127                 ;*** INITIALISATION 6845 ***
128                 ;
129                 IN6845:     EQU  $
130 A068 2190A0               LD   HL,FININI          ;POINTE FIN TABLE
131                 ;D'INITIALISATION DU 6845
132                 ;POUR UNE FREQUENCE DE 50 Hz
133 A06B 0100BC               LD   BC,AREG            ;REGISTRE D'ADRESSE
134 A06E 0E0F                 LD   C,0FH              ;REGISTRE 15
135 A070 2B                   DEC  HL
136                 ;
137                 I68452:     EQU  $
138 A071 ED49                 OUT  (C),C              ;POINTE REGISTRE CONTROL
139 A073 04                   INC  B                  ;REGISTRE DE CONTROLE
140 A074 7E                   LD   A,(HL)             ;VALEUR DE LA TABLE
141 A075 ED79                 OUT  (C),A              ;DANS LE REGISTRE CONTROL
142 A077 05                   DEC  B                  ;REGISTRE D'ADRESSE
143 A078 0D                   DEC  C                  ;SUIVANT OU PRECEDENT
144 A079 2B                   DEC  HL
145                 ;LA TABLE
146 A07A 79                   LD   A,C                ;REG CONTROL DANS A
147 A07B FEFF                 CP   0FFH               ;SI TOUS Y SONT PASSES
148 A07D 20F2                 JR   NZ,I68452          ;SINON ENCORE
149 A07F C9                   RET                     ;SI OUI FIN
150                 ;
151                 I68451:     EQU  $                  ;TABLE 50 Hz
152 A080 3F                   DEFB 03FH               ;H-TOTAL
153 A081 28                   DEFB 028H               ;H-DISPLAY
154 A082 2E                   DEFB 02EH               ;H-SYNC
155 A083 8E                   DEFB 08EH               ;SYNC-WIDTH
156 A084 26                   DEFB 026H               ;V-TOTAL
157 A085 00                   DEFB 00H                ;V-ADJUST
158 A086 19                   DEFB 019H               ;V-DISPLAY
159 A087 1E                   DEFB 01EH               ;V-SYNC
160 A088 00                   DEFB 00H                ;INTERLACE
161 A089 07                   DEFB 07H                ;SCAN LINE
162 A08A 00                   DEFB 00H                ;CURSOR START
163 A08B 00                   DEFB 00H                ;CURSOR END
164 A08C 3000                 DEFB 030H,00H           ;START RAM
165 A08E C000                 DEFB 0C0H,00H           ;CORSOR POS.
166                 FININI:     EQU  $
167                 ;
168                 ;
169                 ;*** EFFACEMENT ECRAN PAR ***
170                 ;***     MODE ROULEAU       ***
171                 ;***      HORIZONTAL        ***
172                 ;
```

```
                ROULE:      EQU  $
A090 3E27                   LD   A,027H          ;40 CARACTERES DEFINIS
                ;
                ROUL3:      EQU  $
A092 0100BC                 LD   BC,AREG         ;POINT ADRESS REGISTER
A095 0E01                   LD   C,01            ;REGISTRE 1
A097 ED49                   OUT  (C),C
A099 04                     INC  B               ;POINTE CONTROL REGISTER
A09A ED79                   OUT  (C),A           ;EFFET REUSSI
A09C 47                     LD   B,A             ;SAUVE A
                ;
A09D 210020                 LD   HL,02000H       ;TEMPO
                ROUL4:      EQU  $
A0A0 2B                     DEC  HL
A0A1 7C                     LD   A,H
A0A2 B5                     OR   L
A0A3 20FB                   JR   NZ,ROUL4
A0A5 78                     LD   A,B             ;RECUPERE A
A0A6 3D                     DEC  A               ;1 CARACTERE DE MOINS
A0A7 04                     INC  B
A0A8 10E8                   DJNZ ROUL3           ;AU SUIVANT
                ;
A0AA C9                     RET                  ;FIN
                ;
                ;
                ;*** REMISE EN FORME HORIZONTALE *
                ;***      DE  L'ECRAN VIDEO      *
                ;
                DEROUL:     EQU  $
A0AB 3E00                   LD   A,00H           ;ON COMMENCE A ZERO
                DEROU3:     EQU  $
A0AD 0100BC                 LD   BC,AREG
A0B0 0E01                   LD   C,01H           ;REGISTRE 1
A0B2 ED49                   OUT  (C),C           ;EN ACCES
A0B4 04                     INC  B               ;CONTROL REGISTER
A0B5 ED79                   OUT  (C),A           ;EFFET VIDEO
A0B7 F5                     PUSH AF              ;SAUVE AF
                ;
A0B8 210020                 LD   HL,02000H       ;TEMPO
                DEROU4:     EQU  $
A0BB 2B                     DEC  HL
A0BC 7C                     LD   A,H
A0BD B5                     OR   L
A0BE 20FB                   JR   NZ,DEROU4
                ;
A0C0 F1                     POP  AF              ;RECUPERE AF
A0C1 3C                     INC  A               ;UN CARACTERE DE PLUS
A0C2 FE29                   CP   029H            ;PLUS DE 40?
A0C4 20E7                   JR   NZ,DEROU3       ;NON, RECOMMENCER
                ;
A0C6 C9                     RET                  ;OUI,FIN
                ;
                ;
                ;*** EFFET DE TREMBLEMENT ***
                ;
                TREMBL:     EQU  $
A0C7 0100BC                 LD   BC,AREG         ;ADRESS REGISTER
A0CA 3E08                   LD   A,08H           ;REGISTRE 8
A0CC ED79                   OUT  (C),A           ;EN ACCES
A0CE 0100BD                 LD   BC,CREG         ;CONTROL REGISTER
```

```
233 A0D1 3E01                LD   A,01H             ;MODE ENTRELACE
234 A0D3 ED79                OUT  (C),A             ;EFFET
235                ;
236 A0D5 C9                  RET                    ;FIN
237                ;
238                ;
239                ;*** EFFET ECRASEMENT ***
240                ;
241                ECRASE:   EQU  $
242 A0D6 0100BC              LD   BC,AREG           ;ADRESS REGISTER
243 A0D9 3E08                LD   A,08H             ;REGISTRE 8
244 A0DB ED79                OUT  (C),A             ;EN ACCES
245 A0DD 0100BD              LD   BC,CREG           ;CONTROL REGISTER
246 A0E0 3E03                LD   A,03H             ;MODE ENTRELACE VIDEO
247 A0E2 ED79                OUT  (C),A             ;EFFET
248                ;
249 A0E4 C9                  RET                    ;FIN
250                ;
251                ;
252                ;*** SROLLING HORIZONTAL ***
253                ;***  POUR L'AFFICHAGE   ***
254                ;***       D'UN TEXTE    ***
255                ;
256                SCRTXT:   EQU  $
257 A0E5 FE03                CP   03H               ;3 PARAMETRES?
258 A0E7 205A                JR   NZ,ERREU1         ;NON ERREUR
259                ;
260 A0E9 DD7E04              LD   A,(IX+4)          ;RECUPERE X
261 A0EC FE50                CP   050H              ;PLUS QUE 80?
262 A0EE D268A1              JP   NC,ERREU2         ;OUI ERREUR
263 A0F1 DD7E02              LD   A,(IX+02H)        ;RECUPERE Y
264 A0F4 FE19                CP   019H              ;PLUS QUE 25?
265 A0F6 D268A1              JP   NC,ERREU2         ;OUI ERREUR
266                ;
267 A0F9 DD6601              LD   H,(IX+01H)        ;ADRESSE HAUTE
268 A0FC DD6E00              LD   L,(IX+00H)        ;ADRESSE BASSE
269 A0FF 7E                  LD   A,(HL)            ;NB DE LETTRES
270 A100 3240A1              LD   (NBLET),A         ;A RANGER
271 A103 23                  INC  HL                ;SUIVANT
272 A104 7E                  LD   A,(HL)            ;ADRESSE BASSE
273 A105 3242A1              LD   (OCTET0),A        ;CHAINE A RANGER
274 A108 23                  INC  HL                ;SUIVANT
275 A109 7E                  LD   A,(HL)            ;ADRESSE HAUTE
276 A10A 3241A1              LD   (OCTET1),A        ;CHAINE A RANGER
277                ;
278 A10D 0601                LD   B,01H             ;PREMIER CARACTERE
279                ;
280                SCRTX2:   EQU  $
281 A10F C5                  PUSH BC                ;SAUVER
282 A110 DD7E04              LD   A,(IX+04H)        ;REPREND X
283 A113 C1                  POP  BC                ;RECUPERE BC
284 A114 C5                  PUSH BC                ;RESAUVE
285 A115 90                  SUB  B                 ;X-I
286 A116 E5                  PUSH HL                ;SAUVE HL
287 A117 2140A1              LD   HL,NBLET          ;ADRESSE
288 A11A 86                  ADD  A,(HL)            ;X-I+LEN(CHAINE$)
289 A11B E1                  POP  HL                ;RECUPERE HL
290 A11C 67                  LD   H,A               ;POSITIONX DANS A
291 A11D DD6E02              LD   L,(IX+02H)        ;PREND Y
292 A120 CD75BB              CALL TSCURS            ;POSITION CURSEUR
```

```
293                     ;
294 A123 C1             POP   BC                ;REPREND BC
295 A124 C5             PUSH  BC                ;RESAUVER
296 A125 3A41A1         LD    A,(OCTET1)        ;ADRESSE HAUTE
297 A128 67             LD    H,A               ;CHAINE DANS H
298 A129 3A42A1         LD    A,(OCTET0)        ;ADRESSE BASSE
299 A12C 6F             LD    L,A               ;DE CHAINE DANS L
300                     ;
301                     SCRTX1:   EQU   $
302 A12D 7E             LD    A,(HL)            ;LETTE EN POSITION HL
303 A12E E5             PUSH  HL                ;A SAUVER
304 A12F CD5ABB         CALL  TXTOUT            ;AFFICHER CARACTERE
305                     ;
306 A132 E1             POP   HL                ;RECUPERE HL
307 A133 23             INC   HL                ;SUIVANT
308 A134 10F7           DJNZ  SCRTX1            ;ENCORE UN CARACTERE?
309                     ;
310 A136 C1             POP   BC                ;REPREND BC
311 A137 04             INC   B                 ;DECALE POSITION
312 A138 3A40A1         LD    A,(NBLET)         ;RECUPERE LONGUEUR
313 A13B 3C             INC   A                 ;PLUS UNE
314 A13C B8             CP    B                 ;MESSAGE ENTIER?
315 A13D 20D0           JR    NZ,SCRTX2         ;NON RECOMMENCER
316                     ;
317 A13F C9             RET                     ;FIN
318                     ;
319                     NBLET:    EQU   $
320 A140 00             DEFB  00H               ;RESERVE POUR LONGUEUR
321                     ;
322                     OCTET1:   EQU   $
323 A141 00             DEFB  00H
324                     OCTET0:   EQU   $
325 A142 00             DEFB  00H
326                     ;
327                     ;
328                     ERREU1:   EQU   $
329 A143 214FA1         LD    HL,TEXTE1         ;1ER MESSAGE
330                     ;
331                     ERR11:    EQU   $
332 A146 7E             LD    A,(HL)            ;AFFICHAGE
333 A147 B7             OR    A                 ;DU
334 A148 C8             RET   Z                 ;MESSAGE
335 A149 CD5ABB         CALL  TXTOUT            ;POINTE
336 A14C 23             INC   HL                ;PAR
337 A14D 18F7           JR    ERR11             ;REGISTRE HL
338                     ;
339                     TEXTE1:   EQU   $
340 A14F 0A0D           DEFB  0AH,0DH
341 A151 494C2046       DEFB  "IL FAUT "
341 A155 41555420
342 A159 33205041               DEFB  "3 PARAMETRES"
342 A15D 52414D45
342 A161 54524553
343 A165 0A0D                   DEFB  0AH,0DH
344 A167 00                     DEFB  00H
345                     ;
346                     ERREU2:   EQU   $
347 A168 216DA1         LD    HL,TEXTE2         ;2EE MESSAGE
348 A16B 18D9           JR    ERR11             ;A AFFICHER
349                     ;
```

```
350                     TEXTE2:     EQU  $
351 A16D 0A0D                       DEFB 0AH,0DH
352 A16F 45525245                   DEFB "ERREUR DE "
352 A173 55522044
352 A177 4520
353 A179 504F5349                   DEFB "POSITION"
353 A17D 54494F4E
354 A181 0A0D                       DEFB 0AH,0DH
355 A183 00                         DEFB 00H
356                     ;
357                     ;
358                     ;*** EFFET DE DESTRUCTION ***
359                     ;***      SUR L'ECRAN      ***
360                     ;***        DU CPC         ***
361                     ;
362                     DESTRO:     EQU  $
363 A184 0100BC                     LD   BC,AREG          ;EDRESS REGISTER
364 A187 0E09                       LD   C,09H            ;MAX RASTER ADRESS
365 A189 ED49                       OUT  (C),C
366 A18B 0100BD                     LD   BC,CREG
367 A18E 3E00                       LD   A,00H            ;VALEUR 0
368 A190 ED79                       OUT  (C),A
369                     ;
370 A192 C9                         RET                   ;FIN
371                     ;
372                     ;
373                     ;*** EFFET D'ONDULATION ***
374                     ;
375                     ONDULE:     EQU  $
376 A199 0100BC                     LD   BC,AREG          ;ADRESS REGISTER
377 A196 3E03                       LD   A,03H
378 A198 ED79                       OUT  (C),A
379 A19A 0100BD                     LD   BC,CREG          ;CONTROL REGISTER
380 A19D 3E01                       LD   A,01H
381 A19F ED79                       OUT  (C),A
382                     ;
383 A1A1 C9                         RET                   ;FIN
384                     ;
385                     ;
386                                 END
```

Lignes 9 à 34 : définitions des équivalences utilisées

Lignes 54 à 115 : définition des instructions

Lignes 129 à 166 : instruction I INIT6845

Lignes 173 à 195 : instruction I ROULEH

Lignes 201 à 223 : instruction I DEROULEH

Lignes 229 à 236 : instruction I TREMBLE

Lignes 242 à 249 : instruction I ECRASE

Lignes 257 à 355 : instruction I SCROLLPRINT

Lignes 363 à 370 : instruction I DESTROY

Lignes 376 à 383 : instruction I ONDULE

Le chargeur Basic

```
10 REM ****************************
20 REM **                            **
30 REM ** GENERATION DES CODES      **
40 REM **  MACHINES POUR CREER       **
50 REM **    DES RSXs D'EFFETS       **
60 REM **          VIDEO             **
70 REM **                            **
80 REM ****************************
90 REM
100 REM *** RESERVATION MEMOIRE ***
110 REM
120 MEMORY &9FFF
130 OPENOUT"MACHIN."
140 MEMORY HIMEM-1
150 CLOSEOUT
160 REM
170 controle = 0
180 REM
190 REM *** MISE EN MEMOIRE ***
200 REM
210 FOR A = &A000 TO &A1A1
220 READ B$:POKE A,VAL("&"+B$)
230 controle = controle + VAL("&"+B$)
240 NEXT A
250 REM
260 IF controle = 77713 THEN GOTO 380
270 MODE 2
280 PRINT CHR$(7)
290 PRINT "Vous avez fait une erreur"
300 PRINT "en entrant les DATAs ou le no
mbre de controle 77713"
310 PRINT "Verifiez les svp"
320 REM
330 PRINT:PRINT
340 STOP
350 REM
360 REM *** sauvegarde ***
370 REM
380 MODE 2
390 PRINT "SAUVEGARDE DU PROGRAMME BINAI
RE CREE"
400 PRINT:PRINT
410 SAVE "EFFETVID.BIN",B,&A000,&1A2
420 PRINT:PRINT
430 PRINT "Le programme est sauvegarde s
ous le nom"
440 PRINT
450 PRINT "      EFFETSON.BIN"
460 PRINT:PRINT
```

```
470 PRINT "Pour utiliser le fichier bina
ire ainsi cree,"
480 PRINT "inserez dans votre programme
les lignes suivantes:
490 PRINT "      MEMORY &9FFF"
500 PRINT "      OPENOUT " + CHR$(34) + "
BIDON.BID"+CHR$(34)
510 PRINT "      MEMORY HIMEM - 1"
520 PRINT "      CLOSEOUT"
530 PRINT "      LOAD " + CHR$(34) + "EFF
ETVID.BIN" + CHR$(34) + ",&A000"
540 PRINT "      CALL &A000"
550 PRINT "      ... suite du programme
..."
560 PRINT:PRINT
570 END
580 REM
590 REM *** CODES HEXADECIMAUX ***
600 REM
610 REM *** BLOC 1 ***
620 REM
630 DATA 21,0F,A0,01,13,A0,3E,C9
640 DATA 32,00,A0,CD,D1,BC,C9,00
650 DATA 00,00,00,2D,A0,C3,68,A0
660 DATA C3,90,A0,C3,AB,A0,C3,C7
670 DATA A0,C3,D6,A0,C3,E5,A0,C3
680 DATA 84,A1,C3,93,A1,49,4E,49
690 DATA 54,36,38,34,B5,52,4F,55
700 DATA 4C,45,C8,44,45,52,4F,55
710 REM
720 REM *** BLOC 2 ***
730 REM
740 DATA 4C,45,C8,54,52,45,4D,42
750 DATA 4C,C5,45,43,52,41,53,C5
760 DATA 53,43,52,4F,4C,4C,50,52
770 DATA 49,4E,D4,44,45,53,54,52
780 DATA 4F,D9,4F,4E,44,55,4C,C5
790 DATA 21,90,A0,01,00,BC,0E,0F
800 DATA 2B,ED,49,04,7E,ED,79,05
810 DATA 0D,2B,79,FE,FF,20,F2,C9
820 REM
830 REM *** BLOC 3 ***
840 REM
850 DATA 3F,28,2E,8E,26,00,19,1E
860 DATA 00,07,00,00,30,00,C0,00
870 DATA 3E,27,01,00,BC,0E,01,ED
880 DATA 49,04,ED,79,47,21,00,20
890 DATA 2B,7C,B5,20,FB,78,3D,04
900 DATA 10,E8,C9,3E,00,01,00,BC
910 DATA 0E,01,ED,49,04,ED,79,F5
```

```
920 DATA 21,00,20,2B,7C,B5,20,FB
930 REM
940 REM *** BLOC 4 ***
950 REM
960 DATA F1,3C,FE,29,20,E7,C9,01
970 DATA 00,BC,3E,08,ED,79,01,00
980 DATA BD,3E,01,ED,79,C9,01,00
990 DATA BC,3E,08,ED,79,01,00,BD
1000 DATA 3E,03,ED,79,C9,FE,03,20
1010 DATA 5A,DD,7E,04,FE,50,D2,68
1020 DATA A1,DD,7E,02,FE,19,D2,68
1030 DATA A1,DD,66,01,DD,6E,00,7E
1040 REM
1050 REM *** BLOC 5 ***
1060 REM
1070 DATA 32,40,A1,23,7E,32,42,A1
1080 DATA 23,7E,32,41,A1,06,01,C5
1090 DATA DD,7E,04,C1,C5,90,E5,21
1100 DATA 40,A1,86,E1,67,DD,6E,02
1110 DATA CD,75,BB,C1,C5,3A,41,A1
1120 DATA 67,3A,42,A1,6F,7E,E5,CD
1130 DATA 5A,BB,E1,23,10,F7,C1,04
1140 DATA 3A,40,A1,3C,B8,20,D0,C9
1150 REM
1160 REM *** BLOC 6 ***
1170 REM
1180 DATA 00,00,00,21,4F,A1,7E,B7
1190 DATA C8,CD,5A,BB,23,18,F7,0A
1200 DATA 0D,49,4C,20,46,41,55,54
1210 DATA 20,33,20,50,41,52,41,4D
1220 DATA 45,54,52,45,53,0A,0D,00
1230 DATA 21,6D,A1,18,D9,0A,0D,45
1240 DATA 52,52,45,55,52,20,44,45
1250 DATA 20,50,4F,53,49,54,49,4F
1260 REM
1270 REM *** BLOC 7 ***
1280 REM
1290 DATA 4E,0A,0D,00,01,00,BC,0E
1300 DATA 09,ED,49,01,00,BD,3E,00
1310 DATA ED,79,C9,01,00,BC,3E,03
1320 DATA ED,79,01,00,BD,3E,01,ED
1330 DATA 79,C9,00,00,00,00,00,00
1340 REM
1350 REM ***** FIN DE DATA ******
1360 REM
1370 END
```

Après sauvegarde et vérification, vous lancerez ce programme et suivrez les instructions affichées à l'écran.

Pour utilisation, vous entrerez dans votre programme les lignes suivantes :

```
10      MEMORY &9FFF
20      OPENOUT "TRUC."
30      MEMORY HIMEM - 1
40      CLOSEOUT
50      LOAD "EFFETVID.BIN",&A000
60      CALL &A000
```

PROGRAMME DE DÉMONSTRATION

Nous vous conseillons, pour une première utilisation, d'entrer le programme de démonstration suivant, qui permet de s'assurer du bon fonctionnement du programme, surtout si vous avez utilisé le chargeur Basic.

```
10 REM ****************************
20 REM **                          **
30 REM **      PROGRAMME DE        **
40 REM **    DEMONSTRATION DES     **
50 REM **      EFFET VIDEO         **
60 REM **                          **
70 REM ****************************
80 REM
90 CALL &BB4E
100 MEMORY &9FFF
110 OPENOUT "BIDON.BID"
120 MEMORY HIMEM - 1
130 CLOSEOUT
140 LOAD "EFFETVID.BIN",&A000
150 CALL &A000
160 MODE 2
170 REM
180 |INIT6845: REM  - De preference, tou
jours initialiser le 6845 avant les effe
t videos -
190 REM
200 PHRASE1$ = "Voici une demonstration
d'effet videos"
210 PHRASE2$ = "que vous pourrez reutili
ser pour"
220 PHRASE3$ = "personnaliser vos progra
mmes"
230 PHRASE4$ = "FRAPPEZ SUR UNE TOUCHE P
OUR LA DEMO SUIVANTE"
```

```
240 REM
250 |SCROLLPRINT,10,10,@PHRASE1$
260 |SCROLLPRINT,13,12,@PHRASE2$
270 |SCROLLPRINT,15,14,@PHRASE3$
280 PRINT CHR$(24)
290 |SCROLLPRINT,7,20,@PHRASE4$
300 PRINT CHR$(24)
310 REM
320 CALL &BB06
330 REM
340 |SCROLLPRINT,7,20,"ALLEZ ON  ENROULE
 POUR LA DEMONSTRATION SUIVANTE"
350 FOR I = 1 TO 1300
360 NEXT I
370 REM
380 |ROULEH
390 REM
400 MODE 1
410 FOR I = 1 TO 12
420 PRINT"VOICI UN ECRAN BIEN REMPLI POU
R ROULER"
430 PRINT
440 NEXT I
450 REM
460 CALL &BB06
470 REM
480 REM
490 |DEROULEH
500 CALL &BB06
510 REM
520 MODE 1
530 PRINT:PRINT:PRINT
540 PRINT"Je suis un ordinateur tres emo
tif"
550 PRINT
560 PRINT"aussi, si vous appuyez mainten
ant"
570 PRINT
580 PRINT"sur une touche, j'aurai peur"
590 REM
600 CALL &BB06
610 REM
620 |TREMBLE
630 REM
640 CALL &BB06
650 REM
660 |INIT6845
670 REM
680 MODE 1
690 PRINT:PRINT:PRINT
```

```
700 PRINT"Une petite ondulation vous pla
irait"
710 PRINT:PRINT:PRINT
720 PRINT"APPUYEZ SUR UNE TOUCHE POUR VO
IR"
730 REM
740 CALL &BB06
750 REM
760 |ONDULE
770 REM
780 CALL &BB06
790 REM
800 |INIT6845
810 REM
820 MODE 0
830 PRINT:PRINT:PRINT
840 |SCROLLPRINT,5,5,"DESTROY !"
850 CALL &BB06
860 |DESTROY
870 CALL &BB06
880 |INIT6845
890 MODE 0
900 PRINT:PRINT:PRINT
910 PRINT "TEXTE ECRASE"
920 |ECRASE
930 CALL &BB06
940 |INIT6845
```

Après lancement, vous verrez apparaître tous les effets décrits plus haut.

5/8

Caractères graphiques et signes spéciaux

Ils occupent les positions ASCII 123 à 255 et sont affichables au moyen de leur code ASCII.

Par exemple, le code de valeur ASCII 254 sera affichable en BASIC par PRINT CHR$(254).

En assembleur, on utilisera la routine du FIRMWARE GRA WR CHAR située en #BBFC de la manière suivante :

En assembleur :

```
LD      A, 254      ;Code du caractere a afficher
CALL    #BBFC       ;Affichage
```

En BASIC :

```
1000 FOR I=0 TO 5
1010 READ A:POKE &9000+I,A
1020 NEXT I
1030 DATA &3E, 254, &CD, &FC, &BB, &C9
1040 CALL &9000
```

Reportez-vous à l'annexe 3 Partie 11 où les caractères graphiques AMSTRAD sont donnés.

5/9

Sprites : définition de caractères par l'utilisateur

Des caractères de la dimension d'un caractère normal (8 × 8 pixels) sont définissables par l'utilisateur, en BASIC ou en ASSEMBLEUR.

1) En BASIC

... Les ordres SYMBOL et SYMBOL AFTER permettent d'y arriver.

SYMBOL AFTER [<entier>]

où <entier> est compris entre 0 et 255.

Cet ordre fixe le nombre de caractères redéfinissables à 255-<entier>. Par défaut, si aucun argument n'est précisé, seize caractères redéfinissables seront permis. Ils seront accessibles par **CHR$(240)** à **CHR$(255)**.

Remarque :

Pour des raisons internes, la commande MEMORY interdit l'utilisation de SYMBOL AFTER (l'erreur « improper argument » sera générée si vous tentez d'utiliser SYMBOL AFTER après MEMORY).

SYMBOL <Numéro du caractère>, <8 Données sur 8 bits>

Le numéro du caractère sera compris entre 0 et 255 – n, où n est l'argument de la commande SYMBOL AFTER.

Le caractère redéfini de numéro i sera accessible par

CHR$(255 – n + i)

où n est l'argument de la commande SYMBOL AFTER.

A titre d'exemple, voici un petit programme écrit en BASIC qui vous permettra de définir vos propres caractères :

```
1000 REM Redefinition de caracteres
1010 CLS:DIM T(8, 8)
1020 FOR I=1 TO 8
1030   PRINT « ..... »
1040 NEXT I
1050 X=1 : Y=1
1060 LOCATE X,Y
1070 A$=INKEY$:IF A$="" " THEN 1070
1080 CA=ASC(A$)
1090 IF A$="P" " THEN T(Y, X)=1:PRINT "*"
1100 IF A$="V" " THEN T(Y, X)=0:PRINT"."
1110 IF CA=243 AND X<8 THEN X=X+1
1120 IF CA=242 AND X>1 THEN X=X-1
1130 IF CA=240 AND Y>1 THEN Y=Y-1
1140 IF CA=241 AND Y<8 THEN Y=Y+1
1150 IF CA<> 13 THEN 1060 'Boucle de saisie
1160 LOCATE 1,15 : PRINT "Donnees de la commande SYMBOL :
":PRINT
1170 FOR I=1 TO 8
1180   CA=0
1190   FOR J=1 TO 8
1200     CA=CA+(2^(8-J))*T(I, J)
1210   NEXT J
1220   PRINT CA;
1230 NEXT I
```

Les lignes 1010 à 1040 affichent la grille de redéfinition du caractère.

Les lignes 1050 à 1150 permettent de saisir le caractère par les commandes suivantes :

— les touches-flèches permettent de se déplacer dans la grille,

— la touche P permet d'allumer un pixel,

— la touche V permet d'éteindre un pixel.

Les lignes 1160 à 1230 affichent les données correspondant au caractère saisi qu'il faudra entrer dans la commande SYMBOL.

2) En ASSEMBLEUR

... Vous pouvez redéfinir des caractères grâce aux routines du FIRMWARE **TXT SET MATRIX** et **TXT SET M TABLE.**

TXT SET M TABLE est l'équivalent de SYMBOL AFTER.

Le registre pair DE doit contenir le premier caractère de la table (entre 0 et 255).

Le registre pair HL doit contenir l'adresse de départ de la table de redéfinition des caractères.

L'appel à TXT SET M TABLE se fait de la manière suivante :

```
LD    DE,<Numero du caractere>
LD    HL,<Adresse de la table>
CALL  #BBAB
```

TXT SET MATRIX est l'équivalent de SYMBOL.

Le registre A doit contenir le numéro du caractère à redéfinir. Le registre pair HL doit contenir l'adresse de la matrice où se trouvent les octets de redéfinition.

L'appel à TXT SET MATRIX se fait de la manière suivante :

```
LD    A,<Numero du caractere>
LD    HL,<Adresse de la matrice de redefinition>
CALL  #BBA8
```

L'adresse entrée dans HL doit pointer sur l'adresse mémoire du premier octet de redéfinition du caractère entré dans le registre A.

5/9.1

Définition de caractères multiples

Grâce à ce programme, vous pourrez définir des caractères graphiques dans une grille de 6 × 10 caractères, soit 48 × 80 pixels.

Ce programme utilise les concepts manipulés par le précédent, et, en particulier les mêmes commandes.

```
1000 '=======================================
1010 ' Redefinition de caracteres multiples
1020 '=======================================
1030 '
1040 '----------------
1050 'Initialisation
1060 '----------------
1070 MODE 1
1080 DIM T(48,80)
1090 X=1:Y=1 'Position de depart
1100 '-----------------------------
1110 'Gestion du curseur graphique
1120 '-----------------------------
1130 LOCATE INT((X-1)/2)+1,INT((Y-1)/2)+1
1140 L$=INKEY$:IF L$="" THEN 1140
1150 L=ASC(L$)
1160 IF UPPER$(L$)="P" THEN T(Y,X)=1:CP=1:GOSUB 2000:PRINT CHR$(CA) 'Plein
1170 IF L$="V" THEN T(Y,X)=0:CP=0:GOSUB 2000:PRINT CHR$(CA) 'Vide
1180 IF L=243 AND X<80 THEN X=X+1
1190 IF L=242 AND X>1 THEN X=X-1
1200 IF L=241 AND Y<48 THEN Y=Y+1
1210 IF L=240 AND Y>1 THEN Y=Y-1
1220 IF L<>243 AND L<>242 AND L<>241 AND L<>240 AND L<>80 AND L<>86 AND L<>112 A
ND L<>118 AND L<>13 THEN PRINT CHR$(7) 'Action refus
ee
1230 IF L<>13 THEN 1130 'Boucle principale
1240 '-----------------------
1250 'Affichage des resultats
1260 '-----------------------
1270 CLS
1280 PRINT"1) Sur ecran"
1290 PRINT"2) Sur imprimante"
1300 PRINT:INPUT "Votre choix";R
1310 IF R<>1 AND R<>2 THEN 1300
1320 IF R=1 THEN C=1 ELSE C=8
1330 MODE 2
```

```
1340 FOR I=1 TO 6
1350   FOR J=1 TO 10
1360     PRINT#C:PRINT#C,"Ligne"I"Colonne"J":";
1370     FOR K=1 TO 8
1380       A=0
1390       FOR L=1 TO 8
1400         A=A+(2^(8-L))*T(K+(I-1)*8,L+(J-1)*8)
1410       NEXT L
1420       PRINT#C,A;
1430     NEXT K
1440   NEXT J
1450 NEXT I
1460 '
2000 '=========================
2010 'Zone des sous-programmes
2020 '=========================
2030 '
2040 '-----------------------------------------
2050 'Carre plein ou Carre vide (CP=1 OU CP=0)
2060 '-----------------------------------------
2070 S=0
2080 IF INT(X/2)<>X/2 AND INT (Y/2)<>Y/2 THEN GOSUB 2140
2090 IF INT(X/2)=X/2 AND INT (Y/2)<>Y/2 THEN GOSUB 2220
2100 IF INT(X/2)<>X/2 AND INT (Y/2)=Y/2 THEN GOSUB 2300
2110 IF INT(X/2)=X/2 AND INT (Y/2)=Y/2 THEN GOSUB 2380
2120 CA=128+S 'Caractere a afficher
2130 RETURN
2140 '-----------------------------------------
2150 'Caractere en haut et a gauche
2160 '-----------------------------------------
2170 IF CP=1 THEN S=S+1
2180 IF T(Y,X+1)=1 THEN S=S+2
2190 IF T(Y+1,X)=1 THEN S=S+4
2200 IF T(Y+1,X+1)=1 THEN S=S+8
2210 RETURN
2220 '-----------------------------------------
2230 'Caractere en haut et a droite
2240 '-----------------------------------------
2250 IF CP=1 THEN S=S+2
2260 IF T(Y,X-1)=1 THEN S=S+1
2270 IF T(Y+1,X-1)=1 THEN S=S+4
2280 IF T(Y+1,X)=1 THEN S=S+8
2290 RETURN
2300 '-----------------------------------------
2310 'Caractere en bas et a gauche
2320 '-----------------------------------------
2330 IF CP=1 THEN S=S+4
2340 IF T(Y-1,X)=1 THEN S=S+1
2350 IF T(Y-1,X+1)=1 THEN S=S+2
2360 IF T(Y,X+1)=1 THEN S=S+8
2370 RETURN
2380 '-----------------------------------------
2390 'Caractere en bas et a droite
2400 '-----------------------------------------
2410 IF CP=1 THEN S=S+8
2420 IF T(Y-1,X-1)=1 THEN S=S+1
2430 IF T(Y-1,X)=1 THEN S=S+2
2440 IF T(Y,X-1)=1 THEN S=S+4
2450 RETURN
```

Lignes 1000 à 1090 : Initialisation

Lignes 1100 à 1230 : Gestion du curseur graphique
(affichage ou effacement d'un pixel sur l'écran)

Lignes 1240 à 1460 : Affichage des données à programmer par SYMBOL sur écran ou imprimante

Lignes 2000 à 2450 : Zone des sous-programmes.
Calcul du motif à afficher en fonction de la position du point courant et du contexte précédent.

5/9.2

Définition et animation de sprites

Si vous êtes amateur de jeux d'arcades, vous savez très certainement ce que sont les sprites. Pour les néophites, les sprites (ou lutins) sont des objets graphiques qui se déplacent sur l'écran sans l'effacer. Dans ce paragraphe, nous allons :

– étudier un générateur de sprites en **MODE 1** ;
– étudier une RSX (¦SPRITE) qui permettra aux programmeurs Basic de déplacer très simplement jusqu'à 8 sprites en même temps sur un écran en **MODE 1** ;
– utiliser la RSX ¦ SPRITE pour effectuer une démonstration du déplacement d'un sprite en **MODE 1**.

UN PEU DE THÉORIE

Comme vous le savez (voir Partie 5 Chapitre 7), l'écran peut être considéré comme un bloc mémoire dans lequel chaque octet représente deux, quatre ou huit points élémentaires (MODES 0, 1 et 2). En ce qui nous concerne (MODE 1), chaque octet représente quatre points élémentaires ainsi codés :

Bit	7	6	5	4	3	2	1	0
Stylo	2	2	2	2	1	1	1	1
Point	0	1	2	3	0	1	2	3

Interprétation du tableau :

– chaque point élémentaire peut avoir quatre couleurs ;
– lorsque tous les bits correspondant à un point élémentaire valent 0, le point a la couleur du fond de l'écran **(PEN 0)** ;

– pour allumer le point élémentaire le plus à gauche avec la couleur 1, il faut stocker la valeur **&H08** (bit 3 à 1) dans la mémoire concernée ;
– pour allumer le point le plus à gauche avec la couleur 2, il faut stocker la valeur **&H80** (bit 7 à 1) dans la mémoire concernée ;
– pour allumer le point le plus à gauche avec la couleur 3, il faut stocker la valeur **&H88** (bits 7 et 3 à 1) dans la mémoire concernée ;
– le codage des 3 autres points élémentaires suit la même démarche logique :

Point 1 :		Point 2 :		Point 3 :	
Couleur	**Valeur**	**Couleur**	**Valeur**	**Couleur**	**Valeur**
0	&H00	0	&H00	0	&H00
1	&H04	1	&H02	1	&H01
2	&H40	2	&H20	2	&H10
3	&H44	3	&H22	3	&H11

La RSX que nous allons étudier permet de manipuler des sprites de 8 points élémentaires sur 8, ce qui représente 16 octets : 2 octets pour représenter une ligne de 8 points élémentaires, et 8 lignes élémentaires.

Les quatre couleurs possibles pourront être modifiées à volonté à l'aide d'instructions Basic INK (voir Partie 4).

LE GÉNÉRATEUR DE SPRITES

Avant de parler de la RSX !SPRITE, nous allons étudier un générateur de sprites qui vous permettra de créer vos propres sprites de 8 points élémentaires sur 8, et comportant jusqu'à quatre couleurs.

Comment utiliser le programme

Saisissez et exécuter le programme Basic suivant :

```
1000 '=======================
1010 ' Definition de sprites
1020 '=======================
1030 '
1040 '- - - - - - - - - - - - - - - - - -
1050 ' Tableaux de donnees du programme
1060 '- - - - - - - - - - - - - - - - - -
1070 '
1080 DIM t(8,8) 'Tableau du sprite
1090 DIM t2(4,4) 'Valeur des couleurs
1100 t2(1,1)=0:t2(1,2)=0:t2(1,3)=0:t2(1,4)=0
1110 t2(2,1)=8:t2(2,2)=4:t2(2,3)=2:t2(2,4)=1
1120 t2(3,1)=128:t2(3,2)=64:t2(3,3)=32:t2(3,4)=16
1130 t2(4,1)=136:t2(4,2)=68:t2(4,3)=34:t2(4,4)=17
1140 '
1150 '- - - - - - - - - -
1160 ' Trace de la grille
1170 '- - - - - - - - - -
1180 '
1190 MODE 1
1200 GRAPHICS PEN 1
1210 FOR i=1 TO 9
1220   MOVE i*20,100
1230   DRAW i*20,260
1240   MOVE 20,80+i*20
1250   DRAW 180,80+i*20
1260 NEXT i
1270 '
1280 '- - - - - - - - - - - -
1290 ' Ecran de presentation
1300 '- - - - - - - - - - - -
1310 '
1320 LOCATE 20,1
1330 PRINT"Couleurs possibles"
1340 FOR i=0 TO 3
1350   LOCATE 23,i+5
1360   PAPER i
1370   PRINT" "
1380 NEXT i
1390 PAPER 0
1400 LOCATE 15,12
1410 PRINT "Donnees correspondantes"
1420 LOCATE 15,15
1430 PRINT"00 00 00 00 00 00 00 00"
1440 LOCATE 15,16
1450 PRINT"00 00 00 00 00 00 00 00"
1460 '
1470 LOCATE 1,22
1480 PRINT"   Deplact = Fleches, Couleur = Copy"
1490 PRINT"   Pixel   = Enter,   Fin      = Q"
```

```
1500 '
1510 '- - - - - - - - - - - - - - - - - -
1520 ' Boucle principale de definition
1530 '- - - - - - - - - - - - - - - - - -
1540 '
1550 PAPER 0
1560 LOCATE 22,5
1570 PRINT">"
1580 cc=1 'Couleur courante=1
1590 l=1:c=1:sl=1:sc=1:GOSUB 1710
1600 a$=INKEY$:IF a$="" THEN 1600
1610 a=ASC(a$)
1620 sc=c:sl=l 'Sauvegarde ligne et colonne
1630 IF a=243 THEN c=c+1:GOSUB 1710 'Vers la droite
1640 IF a=242 THEN c=c-1:GOSUB 1710 'Vers la gauche
1650 IF a=240 THEN l=l-1:GOSUB 1710 'Vers le haut
1660 IF a=241 THEN l=l+1:GOSUB 1710 'Vers le bas
1670 IF a=224 THEN GOSUB 1950 'Changement de couleur
1680 IF a=13 THEN GOSUB 2080 'Affichage d'un pixel
1690 IF UPPER$(a$)<>"Q" THEN 1600
1700 END
1710 '
1720 '- - - - - - - - - - - - - - - - - -
1730 ' Affichage de la case courante
1740 '- - - - - - - - - - - - - - - - - -
1750 '
1760 IF l=9 THEN l=8
1770 IF l=0 THEN l=1
1780 IF c=9 THEN c=8
1790 IF c=0 THEN c=1
1800 MASK 255
1810 MOVE 20+(sc-1)*20,260-(sl-1)*20
1820 DRAW 40+(sc-1)*20,260-(sl-1)*20
1830 DRAW 40+(sc-1)*20,240-(sl-1)*20
1840 DRAW 20+(sc-1)*20,240-(sl-1)*20
1850 DRAW 20+(sc-1)*20,260-(sl-1)*20
1860 MASK 85
1870 MOVE 20+(c-1)*20,260-(l-1)*20
1880 DRAW 40+(c-1)*20,260-(l-1)*20
1890 DRAW 40+(c-1)*20,240-(l-1)*20
1900 DRAW 20+(c-1)*20,240-(l-1)*20
1910 DRAW 20+(c-1)*20,260-(l-1)*20
1920 MASK 255
1930 '
1940 RETURN
1950 '
1960 '- - - - - - - - - - - - - - -
1970 ' Changement de couleur
1980 '- - - - - - - - - - - - - - -
1990 '
```

```
2000 sc=cc 'Sauvegarde couleur courante
2010 cc=cc+1
2020 IF cc=5 THEN cc=1
2030 LOCATE 22,sc+4
2040 PRINT " "
2050 LOCATE 22,cc+4
2060 PRINT ">"
2070 RETURN

2080 '
2090 '- - - - - - - - - - - -
2100 ' Affichage d'un pixel
2110 '- - - - - - - - - - - -
2120 '
2130 GRAPHICS PEN cc-1
2140 MOVE 20+(c-1)*20,260-(l-1)*20
2150 DRAW 40+(c-1)*20,260-(l-1)*20
2160 DRAW 40+(c-1)*20,240-(l-1)*20
2170 DRAW 20+(c-1)*20,240-(l-1)*20
2180 DRAW 20+(c-1)*20,260-(l-1)*20
2190 MOVE 25+(c-1)*20,255-(l-1)*20
2200 FILL cc-1
2210 GRAPHICS PEN 1
2220 MASK 85
2230 MOVE 20+(c-1)*20,260-(l-1)*20
2240 DRAW 40+(c-1)*20,260-(l-1)*20
2250 DRAW 40+(c-1)*20,240-(l-1)*20
2260 DRAW 20+(c-1)*20,240-(l-1)*20
2270 DRAW 20+(c-1)*20,260-(l-1)*20
2280 MOVE 25+(c-1)*20,255-(l-1)*20
2290 MASK 255
2300 t(l,c)=cc-1
2310 '
2320 IF c<5 THEN s=1 ELSE s=2
2330 octet=(l-1)*2+s
2340 IF octet <9 THEN LOCATE 12+octet*3,15 ELSE LOCATE octet
*3-12,16
2350 tot=0
2360 IF c<5 THEN FOR z=1 TO 4:tot=tot+t2(t(l,z)+1,z):NEXT z

2370 IF c>4 THEN FOR z=5 TO 8:tot=tot+t2(t(l,z)+1,z-4):NEXT
z
2380 PRINT HEX$(tot,2)
2390 RETURN
```

Une grille de 8 points sur 8 apparaît sur l'écran. C'est dans cette grille que vous définirez vos sprites.

La partie haute de l'écran laisse apparaître les quatres couleurs disponibles. Une flèche vers la droite (>) indique quelle est la couleur courante, c'est-à-dire la couleur de tracé. Pour changer la couleur courante, appuyez autant de fois que nécessaire sur la touche <Copy>.

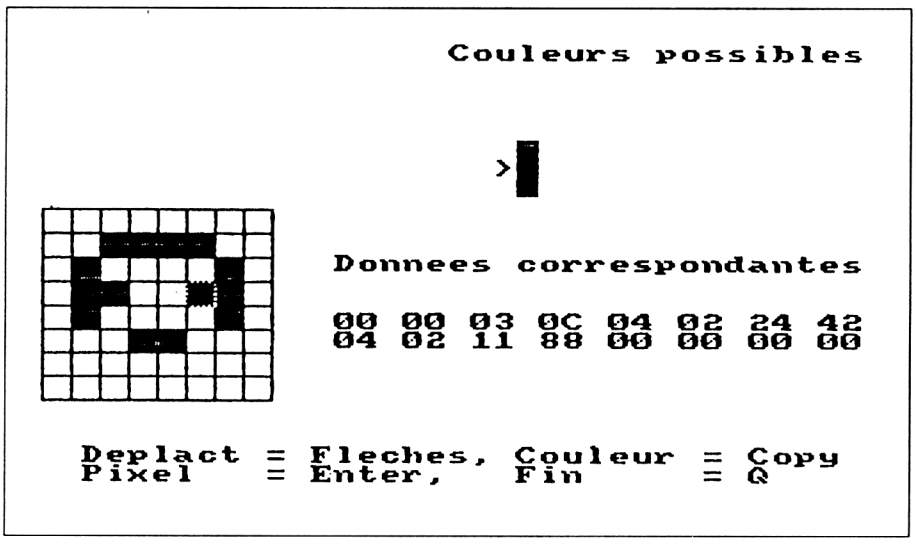
La partie centrale de l'écran affiche les données hexadécimales correspondant au sprite en cours de création. Lorsque vous avez défini un sprite, notez ces données sur un bout de papier. Elles seront par la suite intégrées au programme d'animation...

Enfin, la partie basse de l'écran résume les diverses actions possibles.

Pour définir un sprite, respectez les étapes suivantes :

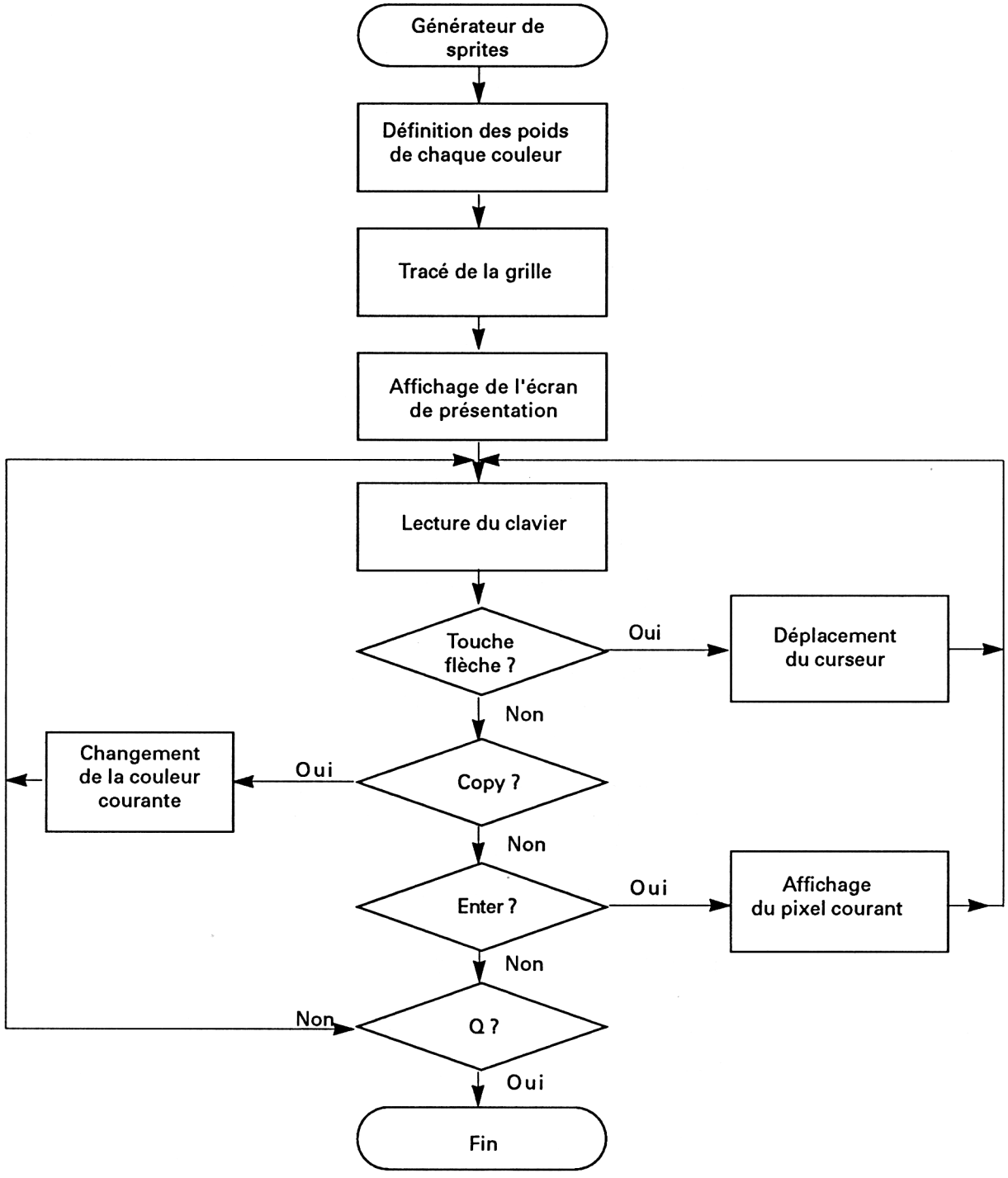
– positionnez le « curseur » sur la case de votre choix (le curseur est symbolisé par une case discontinue) à l'aide des touches flèches ;
– choisissez éventuellement la couleur de la case courante à l'aide de la touche <Copy> du clavier ;
– appuyez sur la touche <Enter> du clavier pour colorier une case courante ;
– recommencez les opérations qui viennent d'être décrites jusqu'à ce que le sprite soit entièrement défini ;
– notez alors les données hexadécimales correspondantes.

Exemple de sprite :



Le programme en détail

La logique du programme obéit à l'ordinogramme suivant :



La ligne 1080 définit le tableau dans lequel seront stockées les données (allumé/éteint) des points du sprite :

DIM t(8,8) 'Tableau du sprite

La ligne 1090 définit le tableau qui contient le poids de chaque point élémentaire en fonction de sa couleur :

DIM t2(4,4) 'Valeur des couleurs

Chaque octet contient la définition de quatre points élémentaires successifs. Les quatres points sont ainsi codés :

	Points			
Couleurs	1	2	3	4
0	0	0	0	0
1	8	4	2	1
2	128	64	32	16
3	136	68	34	17

Les données de ce tableau s'interprètent ainsi :

– lorsque le point élémentaire 1 est allumé avec la couleur 0 (couleur de fond), l'octet de définition est inchangé ;
– lorsque le point élémentaire 1 est allumé avec la couleur 1, l'octet de définition doit être incrémenté de 8 ;
– lorsque le point élémentaire 1 est allumé avec la couleur 2, l'octet de définition doit être incrémenté de 128 ;
– lorsque le point élémentaire 1 est allumé avec la couleur 3, l'octet de définition doit être incrémenté de 136 ;
– de ce qui précède, vous pouvez déduire vous même le codage des points élémentaires 2 à 4.

Les données de ce tableau sont stockées dans t2 entre les lignes 1100 et 1130 :

```
1100 t2(1,1)=0:t2(1,2)=0:t2(1,3)=0:t2(1,4)=0
...
1130 t2(4,1)=136:t2(4,2)=68:t2(4,3)=34:t2(4,4)=17
```

Le tracé de la grille est effectué entre les lignes 1190 et 1260 :

```
1190 MODE 1
1200 GRAPHICS PEN 1
1210 FOR i=1 TO 9
```

```
1220    MOVE i*20,100
1230    DRAW i*20,260
1240    MOVE 20,80+i*20
1250    DRAW 180,80+i*20
1260 NEXT i
```

La ligne 1200 spécifie que la grille doit être tracée avec la couleur PEN 1.

Les diverses données qui accompagnent la grille sont affichées sur l'écran entre les lignes 1320 et 1480 :

– les quatre couleurs possibles sont affichées en haut et à gauche de l'écran à l'aide d'une boucle FOR NEXT dans laquelle la couleur du papier *(paper)* est modifiée :

```
1320 LOCATE 20,1
1330 PRINT "Couleurs possibles"
1340 FOR i=0 TO 3
1350    LOCATE 23,i+5
1360    PAPER i
1370    PRINT " "
1380 NEXT i
```

– les données hexadécimales correspondant au sprite sont initialisées à 0 et affichées à l'aide d'instructions PRINT :

```
1400 LOCATE 15,12
1410 PRINT  "Données correspondantes"
1420 LOCATE 15,15
1430 PRINT  "00 00 00 00 00 00 00 00"
1440 LOCATE 15,16
1450 PRINT  "00 00 00 00 00 00 00 00"
```

– enfin, le rappel des commandes est affiché en bas de l'écran à l'aide d'instructions PRINT :

```
1470 LOCATE 1,22
1480 PRINT"    Deplact = Fleches, Couleur = Copy"
1490 PRINT"    Pixel   = Enter,   Fin     = Q"
```

Le programme se poursuit par la boucle principale de définition dans laquelle les diverses touches pressées sur le clavier sont interprétées.

Après diverses initialisations, le sous-programme situé en ligne 1710 est appelé. Ce sous-programme affiche la case courante dans la grille :

```
1590 1=1:c=1:sl=1:sc=1:GOSUB 1710
```

La case courante affichée, le programme se met en attente d'une action au clavier :

```
1600 a$=INKEY$:IF a$="" THEN 1600
```

Lorsqu'une touche est pressée, son code ASCII est stocké dans la variable a :

```
1610 a=ASC(a$)
```

Les ligne et colonne courantes sont sauvegardées dans les variables sc et sl :

```
1620 sc=c:sl= l 'Sauvegarde ligne et colonne
```

Lorsque la touche pressée est une touche flèche, la variable l ou c est modifiée en conséquence :

```
1630 IF a=243 THEN c=c+1:GOSUB 1710 'Vers la droite
1640 IF a=242 THEN c=c-1:GOSUB 1710 'Vers la gauche
1650 IF a=240 THEN 1=1-1:GOSUB 1710 'Vers le haut
1660 IF a=241 THEN 1=1+1:GOSUB 1710 'Vers le bas
```

Lorsque la touche <Copy> est pressée, le sous-programme de changement de couleur est activé :

```
1670 IF a=224 THEN GOSUB 1950 'Changement de la couleur
```

Lorsque la touche <Enter> est pressée, le sous-programme d'affichage d'un pixel est activé :

```
1680 IF a=13 THEN GOSUB 2080 'Affichage d'un pixel
```

Enfin, lorsque la touche "Q" est pressée, le programme prend fin. Dans tous les autres cas, la touche est ignorée et le programme se met en attente d'une autre action au clavier :

```
1690 IF UPPER$(a$)<>"Q" THEN 1600
```

Le listing se termine par les trois sous-programmes dont nous venons de parler :

Les lignes 1710 à 1940 affichent la case courante dans la grille.
Les paramètres nécessaires en entrée sont les nouvelles ligne (l) et colonne (c) de la case courante.

Si ces données sont incorrectes (par exemple suite à l'appui sur la touche flèche vers la droite alors que le curseur se trouve dans la dernière colonne de la grille), elles sont révisées :

```
1760 IF l=9 THEN l=8
1770 IF l=0 THEN l=1
1780 IF c=9 THEN c=8
1790 IF c=0 THEN c=1
```

La case courante est repérée dans la grille par une ligne discontinue. L'ancienne case courante (de coordonnées sc,sl) est affichée avec une ligne continue :

```
1800 MASK 255
1810 MOVE 20+(sc-1)*20,260-(sl-1)*20
1820 DRAW 40+(sc-1)*20,260-(sl-1)*20
1830 DRAW 40+(sc-1)*20,240-(sl-1)*20
1840 DRAW 20+(sc-1)*20,240-(sl-1)*20
1850 DRAW 20+(sc-1)*20,260-(sl-1)*20
```

et la nouvelle case courante est affichée avec une ligne discontinue :

```
1800 MASK 85
1810 MOVE 20+(c-1)*20,260-(l-1)*20
1820 DRAW 40+(c-1)*20,260-(l-1)*20
1830 DRAW 40+(c-1)*20,240-(l-1)*20
1840 DRAW 20+(c-1)*20,240-(l-1)*20
1850 DRAW 20+(c-1)*20,260-(l-1)*20
```

Les lignes 1950 à 2070 permettent de changer la couleur courante.

La couleur courante est sauvegardée puis incrémentée :

```
2000 sc=cc 'Sauvegarde couleur courante
2010 cc=cc+1
```

Lorsque la nouvelle couleur est égale à 5 (le nombre de couleurs disponibles étant égal à 4), la variable **cc** est initialisée à 1 pour pointer sur la première couleur disponible :

```
2020 IF cc=5 THEN cc=1
```

Le signe **">"** pointant sur l'ancienne couleur courante est effacé et le nouveau signe est affiché :

```
2030 LOCATE 22,sc+4
2040 PRINT " "
2050 LOCATE 22,cc+4
2060 PRINT ">"
```

Les lignes 2080 à 2390

– affichent un point élémentaire dans la grille lorsque la touche <Enter> est pressée,
– modifient et affichent les valeurs hexadécimales correspondantes.

Le remplissage de la case sélectionnée se fait à l'aide de l'instruction FILL :

```
2130 GRAPHICS PEN cc-1
```

```
2140 MOVE 20+(sc-1)*20,260(sl-1)*20
...
2180 DRAW 20+(sc-1)*20,260-(sl-1)*20
2190 FILL cc-1
2200 GRAPHICS PEN 1
2220 MASK 85
2230 MOVE 20+(sc-1)*20,260-(sl-1)*20
...
2280 DRAW 20+(sc-1)*20,260-(sl-1)*20
2290 MASK 255
```

La couleur de la case affichée est mémorisée dans le tableau t :

```
2300 t(1,c)=cc-1
```

Le curseur est positionnné sur la donnée hexadécimale à modifier :

```
2320 IF c<5 THEN s=1 ELSE s=2
2330 octet=(l-1)*2+s
2340 IF octet <9 THEN LOCATE 12+octet*3,15 ELSE
LOCATE octet*3-12,16
```

La nouvelle valeur hexadécimale est calculée à partir des tableaux t et t2 :

```
2350 tot=0
2360 IF c<5 THEN FOR z=1 TO 4:tot=tot+t2(t(l,z)+1,z):NEXT z
2370 IF c>4 THEN FOR z=5 TO 8:tot=tot+t2(y(l,z)+1,z-4):NEXT z
```

et affichée sur deux digits :

```
2380 PRINT HEX$(tot,2)
```

LA RSX D'AFFICHAGE ¦SPRITE

Comment afficher un objet sur l'écran et le déplacer sans effacer le décor ? Eh bien tout simplement en mémorisant le décor qui se trouve sous le sprite et en le réaffichant lors du prochain déplacement. Ce raisonnement simpliste est la base de la RSX ¦SPRITE.

Lors du premier affichage du sprite sur l'écran, aucun décor n'a encore été mémorisé. Il est donc inutile d'afficher la zone de mémoire correspondant au décor. Par contre, il faut sauvegarder le décor qui se trouve sous le sprite avant de l'afficher. Lors des futurs déplacements de sprite, il faudra :

- restituer le décor sauvegardé ;
- mémoriser le nouveau décor ;
- afficher le sprite à la nouvelle position.

Comment utiliser la RSX

La RSX est bien entendu écrite en Assembleur. En voici le listing :

```
1                               ORG  9000H
2                               LOAD 9000H
3                       ;------------------------------
4                       ; RSX SPRITE
5                       ; Format : |SPRITE,N,X,Y
6                       ; Entree : N=Numero du sprite
7                       ;          X=Abscisse du sprite
8                       ;          Y=Ordonnee du sprite
9                       ; Sortie : Affichage du sprite
10                      ;------------------------------
11                      ;
12                      ;
13                      ;--------------------------------
14                      ; Declaration des constantes
15                      ; et des variables du programme
16                      ;--------------------------------
17                      ;
18                      LOGEXT:     EQU  0BCD1H         ;KL LOG EXT
19                      DOTPOS:     EQU  0BC1DH         ;SCR DOT POS
20                      NEXTLINE:   EQU  0BC26H         ;SCR NEXT LINE
21                      SAUVDEC:    EQU  8100H          ;@ sauv decor
22                      PREMAFF:    EQU  8200H          ;Indic 1er aff
23                      NUMERO:     DS   1              ;Numero du sprite
24                      ABS:        DS   2              ;Abscisse du sprite
25                      ORD:        DS   2              ;Ordonne du sprite
26                      ADR:        DS   2              ;Adresse ecran
27                      ESPRIT:     DS   2              ;Adresse sprite
28                      ANCXY:      DS   32             ;Adresse anc coord
29                      BUF:        DS   4              ;Zone RAM pour LOG EXT
30 902D 3290            PTRTAB:     DW   TABLE          ;Pointeur TABLE
31 902F C34390                      JP   SPRITE         ;Affichage du sprite
```

```
32 9032 53505249 TABLE:      DB    "SPRIT"
32 9036 54
33 9037 C5                   DB    "E"+80H
34 9038 00                   DB    0                 ;Fin de table
35              ;
36              ;---------------------------------
37              ; Definition de la RSX
38              ;---------------------------------
39              ;
40              DEFRSX:      EQU   $                 ;Point d'entree
41 9039 012D90               LD    BC,PTRTAB         ;Ptr table definition
42 903C 212990               LD    HL,BUF            ;Buffer pour LOG EXT
43 903F CDD1BC               CALL  LOGEXT            ;Definition de la RSX
44 9042 C9                   RET
45              ;
46              ;---------------------------------
47              ; Traitement de SPRITE
48              ;---------------------------------
49              ;
50              SPRITE:      EQU   $                 ;Point d'entree
51 9043 DD6601               LD    H,(IX+1)
52 9046 DD6E00               LD    L,(IX+0)
53 9049 220390               LD    (ORD),HL
54 904C DD6603               LD    H,(IX+3)
55 904F DD6E02               LD    L,(IX+2)
56 9052 220190               LD    (ABS),HL          ;Sauv abscisse
57 9055 DD7E04               LD    A,(IX+4)
58 9058 320090               LD    (NUMERO),A        ;Sauv No Sprite
59              ;
```

```
60                ;- - - - - - - - - - -
61                ; Test 1er affichage
62                ;- - - - - - - - - - -
63                ;
64 905B 210082              LD   HL,PREMAFF
65 905E 3A0090              LD   A,(NUMERO)
66 9061 3D                  DEC  A
67 9062 1600                LD   D,0
68 9064 5F                  LD   E,A
69 9065 19                  ADD  HL,DE
70 9066 7E                  LD   A,(HL)          ;Indic 1er aff
71 9067 B7                  OR   A
72 9068 CAC190              JP   Z,SAUVEDE       ;Sauvegarde decor
73                ;
74                ;- - - - - - - - - - - -
75                ; Affichage ancien decor
76                ;- - - - - - - - - - - -
77                ;
78                ; Calcul adresse ecran
79                ;
80 906B 210990              LD   HL,ANCXY
81 906E 3A0090              LD   A,(NUMERO)
82 9071 3D                  DEC  A
83 9072 87                  ADD  A,A
84 9073 87                  ADD  A,A
85 9074 1600                LD   D,0
86 9076 5F                  LD   E,A
87 9077 19                  ADD  HL,DE           ;Ordonnee
88 9078 5E                  LD   E,(HL)
89 9079 23                  INC  HL
90 907A 56                  LD   D,(HL)
```

```
 91 907B 23                 INC   HL
 92 907C 4E                 LD    C,(HL)          ;Abscisse
 93 907D 23                 INC   HL
 94 907E 46                 LD    B,(HL)
 95 907F 60                 LD    H,B
 96 9080 69                 LD    L,C
 97 9081 CD1DBC             CALL  DOTPOS
 98 9084 220590             LD    (ADR),HL        ;@ ecran
 99              ;
100              ; Calcul adresse decor
101              ;
102 9087 210081             LD    HL,SAUVDEC      ;@ 1er decor
103 908A 111000             LD    DE,16
104 908D 3A0090             LD    A,(NUMERO)
105              BIS:       EQU   $
106 9090 3D                 DEC   A
107 9091 2803               JR    Z,FININC        ;Fin incrementation
108 9093 19                 ADD   HL,DE
109 9094 18FA               JR    BIS             ;Boucle incrementation
110              FININC:    EQU   $
111 9096 220790             LD    (ESPRIT),HL     ;@ decor
112              ;
113              ; Affichage
114              ;
115 9099 ED5B0790           LD    DE,(ESPRIT)
116 909D 2A0590             LD    HL,(ADR)
117 90A0 3E08               LD    A,8
118 90A2 47                 LD    B,A
119              BISDEC1:
LD A,(D
```

```
120 90A3 1A                 LD    A,(DE)
121 90A4 77                 LD    (HL),A
122 90A5 CD26BC             CALL  NEXTLINE          ;SCR NEXT LINE
123 90A8 13                 INC   DE
124 90A9 13                 INC   DE
125 90AA 10F7               DJNZ  BISDEC1
126
127 90AC ED5B0790           LD    DE,(ESPRIT)
128 90B0 13                 INC   DE
129 90B1 2A0590             LD    HL,(ADR)
130 90B4 23                 INC   HL
131 90B5 3E08               LD    A,8
132 90B7 47                 LD    B,A
133             BISDEC2:
LD A,(D
134 90B8 1A                 LD    A,(DE)
135 90B9 77                 LD    (HL),A
136 90BA CD26BC             CALL  NEXTLINE          ;SCR NEXT LINE
137 90BD 13                 INC   DE
138 90BE 13                 INC   DE
139 90BF 10F7               DJNZ  BISDEC2
140             ;
141             ;- - - - - - - - - - -
142             ; Sauvegarde du decor
143             ;- - - - - - - - - - -
144             ;
145             SAUVEDE:    EQU   $
146             ;
147             ; Sauvegarde des coordonnees
148             ;
149 90C1 210990             LD    HL,ANCXY
150  90C4  340090            ID    A, (NUMERO)
```

```
151 90C7 3D                 DEC   A
152 90C8 87                 ADD   A,A
153 90C9 87                 ADD   A,A
154 90CA 1600               LD    D,0
155 90CC 5F                 LD    E,A
156 90CD 19                 ADD   HL,DE
157 90CE ED5B0190           LD    DE,(ABS)
158 90D2 73                 LD    (HL),E
159 90D3 23                 INC   HL
160 90D4 72                 LD    (HL),D
161 90D5 ED5B0390           LD    DE,(ORD)
162 90D9 23                 INC   HL
163 90DA 73                 LD    (HL),E
164 90DB 23                 INC   HL
165 90DC 72                 LD    (HL),D
166             ;
167             ; Calcul de l'adresse ecran
168             ;
169 90DD ED5B0190           LD    DE,(ABS)
170 90E1 2A0390             LD    HL,(ORD)
171 90E4 CD1DBC             CALL  DOTPOS          ;SCR DOT POSITION
172 90E7 220590             LD    (ADR),HL        ;Sauvegarde adresse
173             ;
174             ; Sauvegarde
175             ;
176 90EA 210081             LD    HL,SAUVDEC      ;@ 1er decor
177 90ED 111000             LD    DE,16
178 90F0 3A0090             LD    A,(NUMERO)
179             BIS2:       EQU   $
```

```
 180 90F3 3D                    DEC   A
 181 90F4 2803                  JR    Z,FININC2          ;Fin incrementation
 182 90F6 19                    ADD   HL,DE
 183 90F7 18FA                  JR    BIS2               ;Boucle incrementatior
 184              FININC2:      EQU   $
 185 90F9 220790                LD    (ESPRIT),HL        ;@ decor
 186              ;
 187 90FC ED5B0790              LD    DE,(ESPRIT)
 188 9100 2A0590                LD    HL,(ADR)
 189 9103 3E08                  LD    A,8
 190 9105 47                    LD    B,A
 191              BISSAV1:
LD A,(H
 192 9106 7E                    LD    A,(HL)
 193 9107 12                    LD    (DE),A
 194 9108 CD26BC                CALL  NEXTLINE           ;SCR NEXT LINE
 195 910B 13                    INC   DE
 196 910C 13                    INC   DE
 197 910D 10F7                  DJNZ  BISSAV1
 198              ;
 199 910F ED5B0790              LD    DE,(ESPRIT)
 200 9113 13                    INC   DE
 201 9114 2A0590                LD    HL,(ADR)
 202 9117 23                    INC   HL
 203 9118 3E08                  LD    A,8
 204 911A 47                    LD    B,A
 205              BISSAV2:
LD A,(H
 206 911B 7E                    LD    A,(HL)
 207 911C 12                    LD    (DE),A
 208 911D CD26BC                CALL  NEXTLINE           ;SCR NEXT LINE
 209 9120 13                    INC   DE
```

```
212                 ;
213                 ;- - - - - - - - - - -
214                 ; Affichage du sprite
215                 ;- - - - - - - - - - -
216                 ;
217 9124 210080             LD    HL,8000H          ;@ 1er sprite
218 9127 111000             LD    DE,16
219 912A 3A0090             LD    A,(NUMERO)
220                 BIS3:    EQU   $
221 912D 3D                 DEC   A
222 912E 2803               JR    Z,FININC3         ;Fin incrementation
223 9130 19                 ADD   HL,DE
224 9131 18FA               JR    BIS3              ;Boucle incrementati
225                 FININC3: EQU   $
226 9133 220790             LD    (ESPRIT),HL
227                 ;
228                 ; Affichage
229                 ;
230 9136 ED5B0790           LD    DE,(ESPRIT)
231 913A 2A0590             LD    HL,(ADR)
232 913D 3E08               LD    A,8
233 913F 47                 LD    B,A
234                 BISCOL1:
LD A,(D
235 9140 1A                 LD    A,(DE)
236 9141 77                 LD    (HL),A
237 9142 CD26BC             CALL  NEXTLINE          ;SCR NEXT LINE
238 9145 13                 INC   DE
239 9146 13                 INC   DE
```

```
240 9147 10F7                   DJNZ BISCOL1
241                   ;
242 9149 ED5B0790               LD   DE,(ESPRIT)
243 914D 13                     INC  DE
244 914E 2A0590                 LD   HL,(ADR)
245 9151 23                     INC  HL
246 9152 3E08                   LD   A,8
247 9154 47                     LD   B,A
248                   BISCOL2:
LD A,(D
249 9155 1A                     LD   A,(DE)
250 9156 77                     LD   (HL),A
251 9157 CD26BC                 CALL NEXTLINE          ;SCR NEXT LINE
252 915A 13                     INC  DE
253 915B 13                     INC  DE
254 915C 10F7                   DJNZ BISCOL2
255                   ;
256                   ;
257                   FIN:      EQU  $                ;Fin du programme
258 915E C9                     RET
259                   ;
260                   ;---------------------------------
261                   ; Zone des sous-programmes
262                   ;---------------------------------
263                   ;
264                             END
```

```
ABS         9001 ADR        9005 ANCXY      9009 BUF        9029
BIS         9090 BISDEC1    90A3 BISDEC2    90B8 BIS2       90F3
BISSAV1     9106 BISSAV2    911B BIS3       912D BISCOL1    9140
BISCOL2     9155 DOTPOS     BC1D DEFRSX     9039 ESPRIT     9007
FININC      9096 FININC2    90F9 FININC3    9133 FIN        915E
LOGEXT      BCD1 NEXTLINE   BC26 NUMERO     9000 ORD        9003
PREMAFF     8200 PTRTAB     902D SAUVDEC    8100 SPRITE     9043
SAUVEDE     90C1 TABLE      9032
```

La version Assembleur est intéressante pour comprendre le fonctionnement des sprites, mais ce sont surtout les données hexadécimales correspondantes qui seront utilisées en programmation Basic.

Pour utiliser cette RSX , procédez comme suit :

– Définissez la RSX en exécutant le programme situé en **&H9039** :

```
CALL &H9039
```

– Avant d'afficher pour la première fois le sprite **N** (où N est compris entre 1 et 8) sur l'écran, initialisez la mémoire **&H8200+N-1** à zéro par un **POKE**. Par exemple pour le sprite 1 :

```
POKE &H8200,0
```

puis affichez le sprite en spécifiant les coordonnées d'affichage. Par exemple pour afficher le sprite 1 aux coordonnées **x=100, y=50** :

```
|SPRITE,1,100,50
```

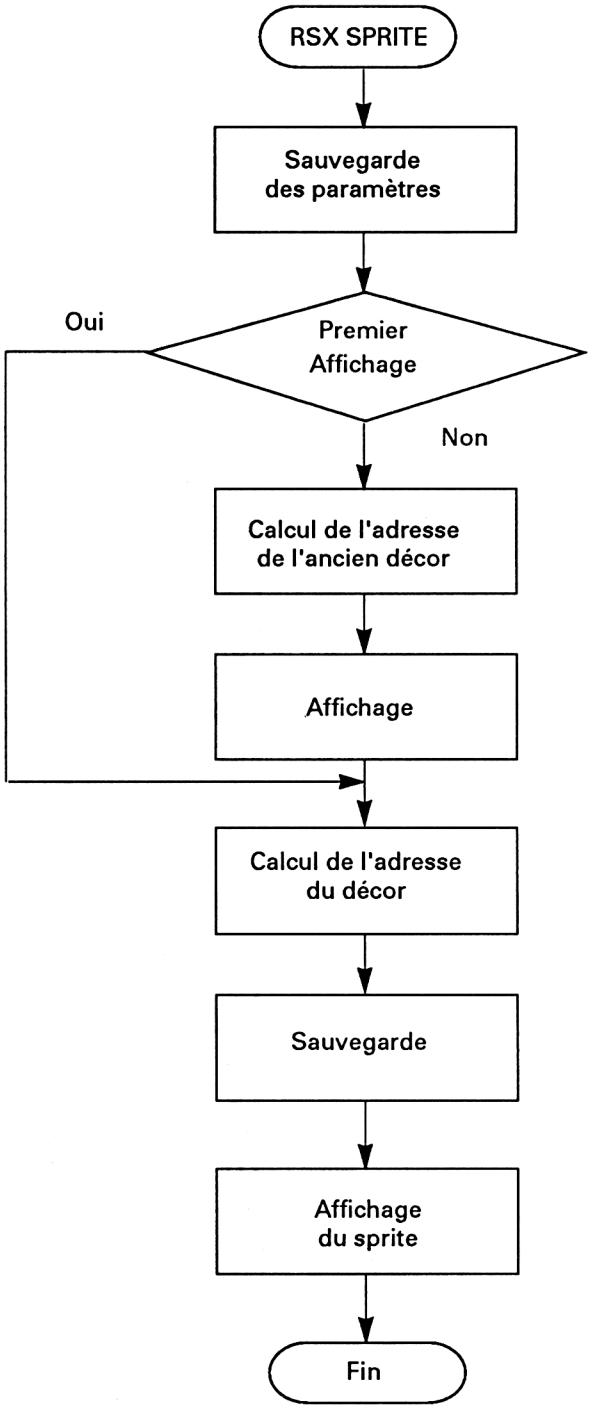
– Lorsque le sprite **N** (où N est compris entre 1 et 8) a été affiché une fois, initialisez la mémoire **&H8200+N-1** à un par un **POKE**. Par exemple pour le sprite 1 :

```
POKE &H8200,0
```

puis affichez le sprite comme dans l'exemple ci-dessus.

Le programme en détail

La logique de la RSX apparaît dans l'ordinogramme suivant :



La RSX est définie à l'aide du petit programme situé entre les lignes 40 et 44 :

```
DEFRSX:   EQU     $             ;Point d'entrée
          LD      BC,PTRTAB     ;Ptr table définition
          LD      HL,BUF        ;Buffer pour LOG EXT
          CALL    LOGEXT        ;Définition de la RSX
```

Lorsque le mot **SPRITE** sera reconnu par le Basic, le programme situé à l'étiquette **SPRITE** sera exécuté. En effet, le tableau **PTRTAB** pointe sur l'étiquette **SPRITE** et contient le mot clé **SPRITE** :

```
PTRTAB:   DW      TABLE         ;Pointeur TABLE
          JP      SPRITE        ;Affichage du sprite
TABLE:    DB      "SPRIT"
          DB      "E"+80H
          DB      0             ;Fin de table
```

Le programme situé à l'étiquette **SPRITE** débute par la mémorisation des données qui lui sont passées :

```
SPRITE:   EQU     $             ;Point d'entrée
          LD      H, (IX+1)
          LD      L, (IX+0)
          LD      (ORD) ,HL
          LD      H, (IX+3)
          LD      L, (IX+2)
          LD      (ABS) ,HL     ;Sauv abscisse
          LD      A, (IX+4)
          LD      (NUMERO) ,A
```

La variable située en &H8200+NUMERO est ensuite examinée pour déterminer si le sprite est affiché pour la première fois :

```
          LD      HL,PREMAFF
          LD      A, (NUMERO)
          DEC     A
          LD      D,0
          LD      E,A
          ADD     HL,DE
          LD      A, (HL)       ;Indic 1er aff
          OR      A
```

Lorsque le sprite est affiché pour la première fois, la phase d'affichage du décor est sautée :

```
          JP      Z,SAUVDE      ;Sauvegarde decor
```

A partir du second affichage du sprite, le décor mémorisé à l'ancienne position du sprite est réaffiché. Les coordonnées de ce décor se

trouvent dans la zone mémoire qui commence en ANCXY :

– ordonnée en ANCXY + (NUMERO)*4
– abscisse en ANCXY + (NUMERO)*4 + 2

Ces coordonnées sont extraites de la mémoire et stockées dans les registres DE et HL :

```
LD      HL,ANCXY
LD      A,(NUMERO)
DEC     A
ADD     A,A
ADD     A,A       → (NUMERO)*4
LD      D,0
LD      E,A
ADD     HL,DE     → ANCXY + (NUMERO)*4
LD      E,(HL)
INC     HL
LD      D,(HL)    → DE = (ANCXY + (NUMERO)*4)
INC     HL
LD      C,(HL)
INC     HL
LD      B,(HL)    → BC = (ANCXY + (NUMERO)*4 + 2)
LD      H,B
LD      L,C
```

La routine SCR DOT POSITION du Firmware est ensuite appelée pour connaître l'adresse de l'octet qui correspond à ces coordonnées :

```
CALL    DOTPOS
```

Cette routine renvoie l'adresse recherchée dans le registre HL. Le programme stocke cette adresse dans la variable ADR :

```
LD      (ADR),HL
```

L'adresse mémoire du décor est calculée par une simple addition : l'adresse du premier décor débute en SAUVDEC, et chaque décor occupe 16 octets. L'adresse qui nous intéresse est donc calculée de la manière suivante :

adresse = SAUVDEC + (NUMERO)*16

Comme la multiplication ne fait pas partie des opérations de base du Z80, nous avons eu recours à une boucle pour calculer (NUMERO)*16 :

```
        LD HL,SAUVDEC       ;Adresse du premier décor
        LD DE,16
        LD A,(NUMERO)
BIS:    EQU     $
```

```
            DEC     A
            JR      Z,FININC        ;Fin de l'incrémentation
            ADD     HL,DE
            JR      BIS             ; Boucle d'incrémentation
FININC:     EQU     $
            LD      (ESPRIT) ,HL    ;Adresse du décor
```

L'affichage des 16 octets du décor se fait à l'aide de deux boucles. La première affiche les 8 octets correspondant à la partie gauche du décor :

```
            LD      DE, (ESPRIT)
            LD      HL, (ADR)
            LD      A,8
            LD      B,A
BISDEC1 :
            LD      A, (DE)         → octet lu en mémoire
            LD      (HL) ,A         → affiché sur l'écran
            CALL    NEXTLINE        → passage à la ligne
                                      suivante sur l'écran
            INC     DE              → ... et en mémoire
            INC     DE
            DJNZ    BISDEC1         → boucle d'affichage
```

Remarquez :

– la macro **NEXTLINE** du Firmware qui permet de connaître l'adresse de l'octet qui représente les 4 points situés en-dessous des 4 points courants ;

– l'utilisation pratique de l'instruction **DJNZ** qui exécute la boucle 8 fois, jusqu'à ce que le registre B soit nul.

La seconde boucle affiche les 8 octets correspondant à la partie droite du décor. Elle utilise la même logique que la précédente. Nous n'y reviendrons pas.

Le nouveau décor est sauvegardé selon le procédé inverse. Les données sont lues sur l'écran à partir des coordonnées **ABS** et **ORD**, et sauvegardées en mémoire (lignes 140 à 211). Nous n'y reviendrons pas.

Le programme se termine par l'affichage du sprite aux coordonnées **ABS,ORD**. Cet affichage est très similaire à l'affichage du décor. Il occupe les lignes 212 à 254.

Le programme se termine par une instruction **RET** qui redonne le contrôle au Basic.

DÉMONSTRATION DE L'ANIMATION D'UN SPRITE

Le court programme présenté dans ce paragraphe montre comment utiliser la RSX SPRITE dans un programme Basic. Son but est uniquement démonstratif. Aucune action n'est donc demandée à la personne qui l'utilise, si ce n'est l'appui sur une touche quelquonque du clavier qui met fin à son exécution.

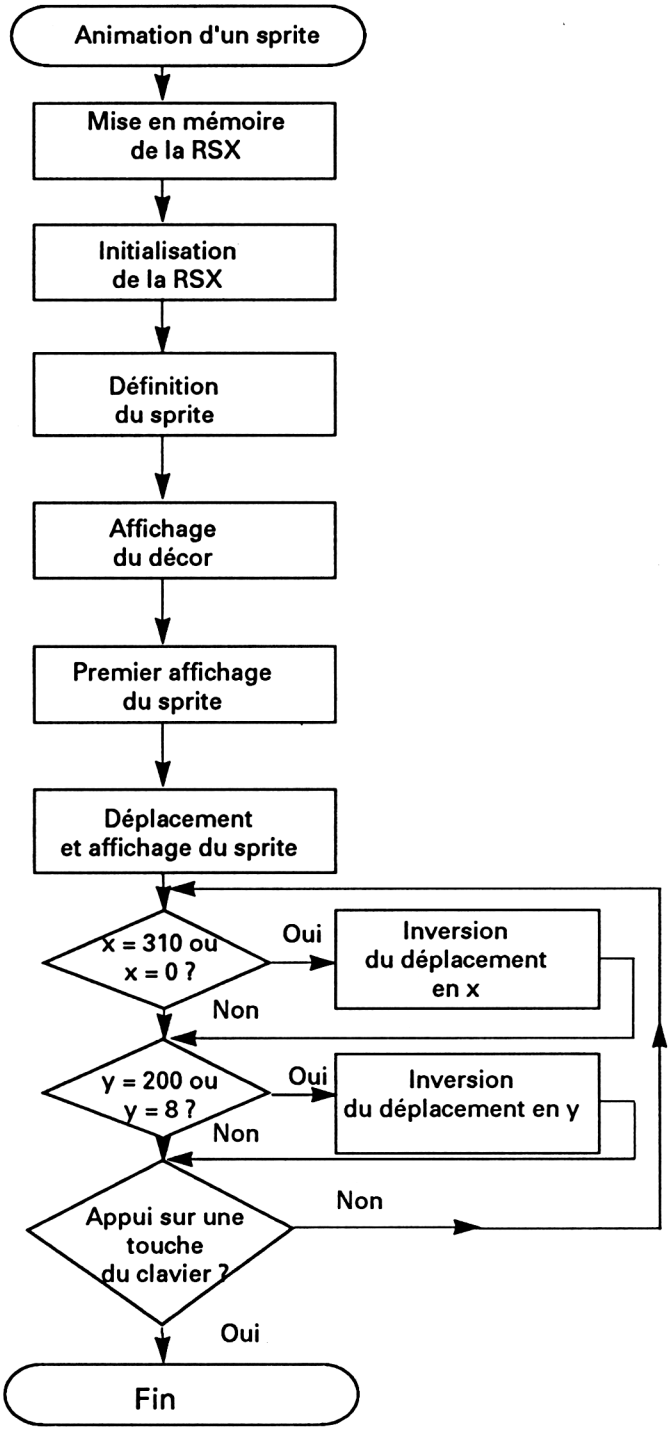
Son listing est le suivant :

```
1000 '=============================================
1010 ' Demonstration de l'animation d'un sprite
1020 '=============================================
1030 '
1040 FOR i=&9000 TO &915E
1050   READ a$
1060   a$="&"+a$
1070   a=VAL(a$)
1080   POKE i,a
1090 NEXT i
1100 '
1110 '---------------------------
1120 ' Initialisation de la RSX
1130 '---------------------------
1140 '
1150 CALL &9039
1160 '
1170 '-------------------------
1180 ' Definition du sprite
1190 '-------------------------
1200 '
1210 FOR i=&8000 TO &8007
1220   IF (i/2)=INT(i/2) THEN POKE i,&FF ELSE POKE i
1230 NEXT i
1240 '
1250 FOR i=&8008 TO &800F
1260   IF (i/2)=INT(i/2) THEN POKE i,0 ELSE POKE i,&I
1270 NEXT i
1280 '-----------
1290 ' Animation
1300 '-----------
1310 '
1320 MODE 1
1330 FOR i=1 TO 1000
1340   PRINT"-";
1350 NEXT i
1360 POKE &8200,0:|SPRITE,1,80,100
1370 POKE &8200,1
```

```
1380 x=80
1390 y=100
1400 dx=2
1410 dy=2
1420 a$=INKEY$
1430 IF (x=310) OR (x=0) THEN dx=-dx
1440 IF (y=200) OR (y=8) THEN dy=-dy
1450 x=x+dx
1460 y=y+dy
1470 |SPRITE,1,x,y
1480 IF a$="" THEN 1420
1490 MODE 2
1500 END
1510 '
1520 '-----------------------------------------------
1530 ' Programme assembleur d'affichage de sprites
1540 '-----------------------------------------------
1550 '
1560 DATA 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
1570 DATA 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
1580 DATA 0,0,0,0,0,0,0,0,0,0,0,0,0,0,32,90,C3
1590 DATA 43,90,53,50,52,49,54,C5,0,1,2D,90,21,29,90,CD
1600 DATA D1,BC,C9,DD,66,1,DD,6E,0,22,3,90,DD,66,3,DD
1610 DATA 6E,2,22,1,90,DD,7E,4,32,0,90,21,0,82,3A,0
1620 DATA 90,3D,16,0,5F,19,7E,B7,CA,C1,90,21,9,90,3A,0
1630 DATA 90,3D,87,87,16,0,5F,19,5E,23,56,23,4E,23,46,60
1640 DATA 69,CD,1D,BC,22,5,90,21,0,81,11,10,0,3A,0,90
1650 DATA 3D,28,3,19,18,FA,22,7,90,ED,5B,7,90,2A,5,90
1660 DATA 3E,8,47,1A,77,CD,26,BC,13,13,10,F7,ED,5B,7,90
1670 DATA 13,2A,5,90,23,3E,8,47,1A,77,CD,26,BC,13,13,10
1680 DATA F7,21,9,90,3A,0,90,3D,87,87,16,0,5F,19,ED,5B
1690 DATA 1,90,73,23,72,ED,5B,3,90,23,73,23,72,ED,5B,1
1700 DATA 90,2A,3,90,CD,1D,BC,22,5,90,21,0,81,11,10,0
1710 DATA 3A,0,90,3D,28,3,19,18,FA,22,7,90,ED,5B,7,90
1720 DATA 2A,5,90,3E,8,47,7E,12,CD,26,BC,13,13,10,F7,ED
1730 DATA 5B,7,90,13,2A,5,90,23,3E,8,47,7E,12,CD,26,BC
1740 DATA 13,13,10,F7,21,0,80,11,10,0,3A,0,90,3D,28,3
1750 DATA 19,18,FA,22,7,90,ED,5B,7,90,2A,5,90,3E,8,47
1760 DATA 1A,77,CD,26,BC,13,13,10,F7,ED,5B,7,90,13,2A,5
1770 DATA 90,23,3E,8,47,1A,77,CD,26,BC,13,13,10,F7,C9,0
```

Le programme en détail

La logique du programme apparaît dans l'ordinogramme suivant :



Le programme débute par la mise en mémoire de la RSX ¦SPRITE à l'aide d'une boucle **FOR NEXT** :

```
1040 FOR i=&9000 TO &915E
1050        READ a$
1060        a$="&"+a$
1070        a=VAL(a$)
1080        POKE i ,a
1090 NEXT i
```

La RSX est ensuite initialisée :

```
1150 CALL &9039
```

Le programme se poursuit par la définition du sprite en mémoire à partir de l'adresse **&8000** :

```
1210 FOR i=&8000 TO &8007
1220        IF (i/2)=INT(i/2) THEN POKE i,&FF ELSE POKE i,0
1230 NEXT i
1240 '
1250 FOR i=&8008 TO &800F
1260        IF (i/2)=INT(i/2) THEN POKE i,0 ELSE POKE i,&FF
1270 NEXT i
```

Pour montrer que les sprites se déplacent sur l'écran sans effacer le décor affiché, une boucle **FOR NEXT** remplit l'écran **MODE 1** de tirets :

```
1320 MODE 1
1330 FOR i=1 TO 1000
1340        PRINT "-";
1350        NEXT i
```

Avant d'afficher le sprite pour la première fois, la mémoire d'adresse **&H8200** est initialisée à **0** :

```
1360 POKE &8200,0: SPRITE,1,80,100
```

La mémoire d'adresse **&H8200** est alors initialisée à **1** pour tout le reste du programme :

```
1370 POKE &H8200,1
```

Les dernières instructions du programme réalisent l'animation du sprite qui semble rebondir sur les coins de l'écran.

La position de départ du sprite est 80,100 :

```
1380 x=80
1390 y=100
```

Les déplacements selon les deux axes sont fixés à 2 pixels :

```
1400 dx=2          → déplacement en x
1410 dy=2          → déplacement en y
```

La fonction **INKEY$** scrute le clavier. Lorsqu'une touche est pressée, le programme prend fin :

```
1420 a$=INKEY$
...
1480 IF a$=" " THEN 1420
1490 MODE 2
1500 END
```

La logique de déplacement du sprite est ultra simple :

– si le sprite se trouve sur l'extrémité droite ou gauche de l'écran, le déplacement en **X** est inversé :

```
1430 IF (x=310) OR (x=0) THEN dx=-dx
```

– si le sprite se trouve sur l'extrémité haute ou basse de l'écran, le déplacement en **Y** est inversé :

```
1440 IF (y=200) OR (y=0) THEN dy=-dy
```

– les coordonnées du sprite sont incrémentées de **dx** et **dy** :

```
1450 x=x+dx
1460 y=y+dy
```

Le sprite est enfin affiché à l'aide d'une instruction ¦**SPRITE** :

```
1470 ¦SPRITE,1,x,y
```

Les dernières lignes du programme (1520 à 1770) contiennent les données hexadécimales de la RSX ¦SPRITE.

Afin d'éviter toute erreur dans la saisie des DATA, voici les données de checksum correspondantes :

```
0 0 86 94 C4 25 A4 7E 57 EE DE FB A1 ED 71 F9 AA B7 24 14 93 7B
```

5/10

Logiciels

Dans cette partie, nous allons étudier et donner des programmes utilitaires traitant de problèmes spécifiques au graphisme.

5/10.1

Programme de dessin

Ce programme permet de créer d'une façon tout à fait conventionnelle des dessins en MODE 1 en utilisant la technique du **TÉLÉCRAN.**

Après avoir choisi les couleurs utilisées dans le dessin, les touches-flèches permettent de déplacer le curseur. Pour dessiner avec la couleur n, il suffit d'appuyer simultanément sur SHIFT et n. Pour obtenir la fonction « gomme », il faut appuyer simultanément sur SHIFT et 0 (car 0 est la couleur de fond).

Quand le dessin est terminé, appuyez sur la touche « ENTER » et répondez « O » (Oui) à la question « **Sauvegarde magnétique ?** ». Si vous possédez un 464 sans lecteur de disquettes, armez-vous de patience car l'écran occupe 16 KO (Kilo Octets) et prend (16/2 + 1) 9 blocs. Vous pouvez accélérer les choses en précisant « SPEED WRITE 1 » avant de lancer l'exécution du programme.

Une fois la sauvegarde effectuée, vous pouvez poursuivre votre travail

en reprenant les choses au point où elles en étaient en répondant « O » (Oui) à la question « Poursuite ? ».

Le listing du programme BASIC est le suivant :

```
1000 REM **********************
1001 REM Trace point par point
1002 REM **********************
1010 '
1020 'Prog. ASM Sauvegarde et affichage ecran
1030 '
1040 FOR I=0 TO &17:READ A:POKE &3000+I,A:NEXT
1050 DATA &21,0,&C0,&11,0,&40,1,&FF,&3F,&ED,&B0,&C9
1060 DATA &21,0,&40,&11,0,&C0,1,&FF,&3F,&ED,&B0,&C9
1070 '
1080 'Initialisation
1090 '
1100 STYLO=0 'Stylo leve
1110 BORDER 0:INK 0,0:INK 1,10:X=0:Y=0
1120 MODE 1
1130 INPUT"Affichage monochrome (O/N)";R$:R$=UPPER$(R$)
1140 IF R$<>"O" AND R$<>"N" THEN 1120
1150 IF R$="O" THEN NBCOUL=1 ELSE NBCOUL=3
1160 PRINT
1170 FOR I=0 TO NBCOUL
1180    PRINT"INK";I;": (0 A 26) ";:INPUT A:INK I,A
1190 NEXT I
1200 PRINT :INPUT"Chargement d'un ecran (O/N) ";R$:R$=UPPER$(R$)
1210 IF R$<>"O" AND R$<>"N" THEN 1200
1220 IF R$="O" THEN PRINT:INPUT"Nom de l'ecran ";ECR$:LOAD ECR$,&C000:GOTO 1240
1230 CLS
1240 PLOT X,Y,2
1250 '
1260 'Boucle principale
```

```
1270 '
1280 A$=INKEY$:IF A$="" THEN 1280 'Attente action
1290 A=ASC(A$)
1300 IF A=55 THEN PLOT X,Y,STYLO:Y=Y+2:X=X-2:GOTO 1240 'En haut a gauche
1310 IF A=57 THEN PLOT X,Y,STYLO:Y=Y+2:X=X+2:GOTO 1240 'En haut a droite
1320 IF A=49 THEN PLOT X,Y,STYLO:Y=Y-2:X=X-2:GOTO 1240 'En bas a gauche
1330 IF A=51 THEN PLOT X,Y,STYLO:Y=Y-2:X=X+2:GOTO 1240 'En bas a droite
1340 IF A=56 THEN PLOT X,Y,STYLO:Y=Y+2:GOTO 1240 'Vers le haut
1350 IF A=50 THEN PLOT X,Y,STYLO:Y=Y-2:GOTO 1240 'Vers le bas
1360 IF A=52 THEN PLOT X,Y,STYLO:X=X-2:GOTO 1240 'Vers la gauche
1370 IF A=54 THEN PLOT X,Y,STYLO:X=X+2:GOTO 1240 'Vers la droite
1380 IF A=95 OR (A>32 AND A<33+NBCOUL) THEN 1420 'Changement de stylo
1390 IF A=13 THEN 1470 'Fin de trace
1400 GOTO 1280 'Boucle de trace
1410 '-----------------------------------------------
1420 REM Changement de couleur stylo
1430 '
1440 IF A=95 THEN STYLO=0 ELSE STYLO=A-32
1450 GOTO 1240
1460 '-----------------------------------------------
1470 REM Fin de trace
1480 '
1490 PLOT X,Y,0:CALL &3000 'Sauvegarde ecran
1500 CLS
1510 LOCATE 1,10:INPUT"Sauvegarde magnetique (O/N) ";R$:R$=UPPER$(R$)
1520 IF R$<>"O" AND R$<>"N" THEN 1510
1530 IF R$="N" THEN CALL &300C:GOTO 1240 'Restitution ecran
1540 LOCATE 1,12:INPUT"Nom de l'ecran ";N$
1550 SAVE N$,B,&4000,&3FFF
1560 LOCATE 1,14:INPUT"Poursuite (O/N) ";R$:R$=UPPER$(R$)
1570 IF R$="O" THEN CALL &300C:GOTO 1240 'Restitution ecran
1580 END
```

Lignes 1010 à 1060 : Chargement des sous-programmes ASSEMBLEUR.

Lignes 1070 à 1230 : Initialisation du programme.

Lignes 1240 : Affichage du curseur graphique.

Lignes 1280 : Lecture du clavier.

Lignes 1290 à 1400 : Action en fonction de la touche pressée.

Ligne 1440 : Changement de la couleur du tracé.

Lignes 1470 à 1580 : Fin de tracé.

Ligne 1550 : avec sauvegarde magnétique.

Ligne 1530 à 1570 : et/ou retour au tracé.

Le programme BASIC défini ci-dessus utilise deux sous-programmes écrits en ASSEMBLEUR. Ces sous-programmes permettent :

1°) de sauvegarder l'écran en mémoire RAM
(transfert du bloc #C000 à #FFFF en #4000),

2°) de restituer la sauvegarde RAM sur l'écran
(transfert du bloc #4000 à #7FFF en #C000).

Ces deux sous-programmes utilisent une mnémonique qui a fait la renommée du Z80 : il s'agit de « LDR ». Son utilisation est très simple : avant l'appel à LDIR, il faut charger les registres HL, DE et BC avec les valeurs suivantes :

HL = Adresse de la première mémoire à déplacer,

DE = Adresse de la première mémoire où va s'effectuer le déplacement,

BC = Longueur du bloc déplacé.

Ce qui donne lieu aux programmes suivants :

```
1                       ORG  3000H
2                       LOAD 3000H
3               ;----------------------
4               ;Sauvegarde de l'ecran
5               ;----------------------
6  3000 2100C0          LD   HL,0C000H      ;@ Depart
7  3003 110040          LD   DE,4000H       ;@ Sauvegarde
8  3006 01FF3F          LD   BC,3FFFH       ;Longueur
9  3009 EDB0            LDIR                ;Transfert
```

```
10 300B C9                    RET
11                ;----------------------
12                ;Restitution de l'ecran
13                ;----------------------
14 300C 210040                LD   HL,4000H          ;@ Depart
15 300F 1100C0                LD   DE,0C000H         ;@ Ecran
16 3012 01FF3F                LD   BC,3FFFH          ;Longueur
17 3015 EDB0                  LDIR                   ;Transfert
18 3017 C9                    RET
19                            END
```

5/10.2

Utilitaires de manipulation de dessins

Lors de la création d'un écran graphique, vous pouvez être confronté au problème suivant : *un objet doit être représenté plusieurs fois sur la même image. Selon la complexité de l'objet, sa duplication peut prendre de quelques minutes à quelques heures.*

Pour éviter cette tâche rébarbative, nous vous proposons trois utilitaires de copie d'objets :

— le premier reproduit un bloc rectangulaire que l'on a défini par ses extrémités bas-gauche et haut-droite ;

— les deuxième et troisième permettent d'obtenir un effet de miroir respectivement par rapport à un axe vertical et par rapport à un axe horizontal.

5/10.2.1

Reproduction de blocs graphiques

Après avoir entré les noms d'écran (avant et après duplication) et les couleurs du dessin, un curseur graphique apparaît en bas et à gauche de l'écran. Dirigez-le avec les touches du bloc numérique pour l'amener en bas et à gauche du rectangle délimitant le bloc à recopier. Appuyez ensuite sur la touche « ENTER ». Déplacez à nouveau le curseur pour l'amener en haut et à droite du même rectangle. Appuyez sur la touche « ENTER ». Un rectangle apparaît. Si l'encadrement est correct (Ques-

tion « OK ? », Réponse « O » (Oui)) déplacez le curseur pour l'amener en bas et à gauche de l'endroit où le bloc doit être dupliqué. Appuyez sur « ENTER ». La copie s'effectue instantanément. A ce niveau, vous pouvez appuyer sur :

« R » pour revenir à l'écran précédent,

« S » pour obtenir une sauvegarde magnétique,

une autre touche pour poursuivre la duplication.

Le programme BASIC de duplication est le suivant :

```
1000 REM Recopie d'un objet sur l'ecran
1010 REM ================================================================
1020 REM 1) Determiner le coin INF GAUCHE de l'objet a deplacer
1030 REM 2) Determiner le coin SUP DROIT  de l'objet a deplacer
1040 REM 3) L'objet est encadre
1050 REM 4) Repondre "O" si l'encadrement est correct,"N" sinon
1060 REM 5) Determiner le coin superieur gauche de la nouvelle position
1070 REM 6) L'objet apparait a sa nouvelle position
1080 REM ================================================================
1090 INK 0,0:BORDER 0:INK 1,10:INK 3,10,6:PEN 1:MODE 1:C=1 'Initialisation
1100 '
1110 'Prog. ASM Sauvegarde et affichage ecran
1120 '
1130 FOR I=0 TO &17:READ A:POKE &9000+I,A:NEXT
1140 DATA &21,0,&C0,&11,0,&40,1,&FF,&3F,&ED,&B0,&C9
1150 DATA &21,0,&40,&11,0,&C0,1,&FF,&3F,&ED,&B0,&C9
1160 '
1170 'Prog ASM Interface entre MBG et ABG
1180 '
1190 FOR I=0 TO 28:READ A:POKE &9018+I,A:NEXT
1200 DATA 1,&64,0,&11,&64,0,&26,&A,&2E,&A,&3E,0,&CD,&35,&90,&C9
1210 DATA 1,&64,0,&11,&64,0,&21,0,&80,&CD,&A2,&90,&C9
1220 '
1230 'Programme MBG (Memorisation de blocs graphiques)
1240 '
1250 FOR I=0 TO 108:READ A:POKE &9035+I,A:NEXT
```

```
1260 DATA &18,&B,&73,&20,&58,&20,&61,&75,&20,&64,&65,&70,&61,&ED,&43,&37,&30,&ED
,&53,&39,&30,&32,&3D,&30,&7C,&32,&3B,&30,&7D,&32,&3C

,&30,&CD,&3,&B9,&21,&0,&80,&11,&94,&2,&3A,&3D,&30,&47,&B7,&28,&4,&ED,&5A,&10,&FC
,&3A,&3B,&30,&77,&23,&22,&3E,&30,&ED,&5B

1270 DATA &37,&30,&2A,&39,&30,&CD,&1D,&BC,&22,&40,&30,&ED,&5B,&3E,&30,&3A,&3B,&3
0,&47,&7E,&12,&23,&13,&10,&FA,&2A,&40,&30,&CD,&26,&B

C,&22,&40,&30,&3A,&3C,&30,&3D,&32,&3C,&30,&20,&E4,&3E,&CF,&12,&C9

1280 '

1290 'Programme ABG (Affichage de blocs graphiques)

1300 '

1310 FOR I=0 TO 72:READ A:POKE &90A2+I,A:NEXT

1320 DATA&18,&A,&41,&42,&47,&D,&A,&3B,&D,&A,&4F,&52,&ED,&43,&A4,&30,&ED,&53,&A6,
&30,&7E,&32,&AA,&30,&23,&22,&A8,&30,&CD,&3,&B9,&ED,&

5B,&A4,&30,&2A,&A6,&30,&CD,&1D,&BC,& 22,&AC,&30,&ED,&5B,&A8,&30,&3A,&AA,&30,&47,
&1A,&77,&23,&13,&10,&FA,&2A,&AC,&30,&CD,&26

1330 DATA &BC,&22,&AC,&30,&1A,&FE,&CF,&20,&E8,&C9

1340 '

1350 PRINT"         COPIE DE BLOCS GRAPHIQUES":PRINT:PRINT

1360 LOCATE 1,6:INPUT"Affichage monochrome (O/N)";R$:R$=UPPER$(R$)

1370 IF R$<>"O" AND R$<>"N" THEN 1360

1371 IF R$="O" THEN NBCOUL=1 ELSE NBCOUL=3

1380 PRINT

1390 FOR I=0 TO NBCOUL

1400    PRINT"INK";I;": (0 A 26) ";:INPUT A:INK I,A

1410 NEXT I

1420 PRINT:PRINT

1421 INPUT"Nom de l'ecran a charger ";N$

1430 INPUT"Nom de l'ecran a sauver  ";N2$

1440 PRINT:PRINT"Une fois toutes les modifications faites, appuyez sur 'S' pour
obtenir une sau- -vegarde de l'ecran sur K7 ou disqu

ette"

1450 PRINT"Appuyez sur 'R' pour retourner a l'ecran precedent et sur une autre t
ouche pour continuer le deplacement des blocs."

1460 PRINT:PEN 3:PRINT"          Appuyez sur une touche"

1470 a$=INKEY$:IF a$="" THEN 1470

1480 LOAD N$,&C000

1490 CALL &9000 'Sauvegarde ecran charge
```

```
1500 B=TEST(X,Y) 'Memo. du pt ou se trouve le curseur
1510 PLOT X,Y,3
1520 '
1530 'Boucle principale
1540 '
1550 A$=INKEY$:IF A$="" THEN 1550 'Attente action
1560 A=ASC(A$)
1570 IF A=55 THEN PLOT X,Y,B:Y=Y+2:X=X-2:GOTO 1500 'En haut a gauche
1580 IF A=57 THEN PLOT X,Y,B:Y=Y+2:X=X+2:GOTO 1500 'En haut a droite
1590 IF A=49 THEN PLOT X,Y,B:Y=Y-2:X=X-2:GOTO 1500 'En bas a gauche
1600 IF A=51 THEN PLOT X,Y,B:Y=Y-2:X=X+2:GOTO 1500 'En bas a droite
1610 IF A=56 THEN PLOT X,Y,B:Y=Y+2:GOTO 1500 'Vers le haut
1620 IF A=50 THEN PLOT X,Y,B:Y=Y-2:GOTO 1500 'Vers le bas
1630 IF A=52 THEN PLOT X,Y,B:X=X-2:GOTO 1500 'Vers la gauche
1640 IF A=54 THEN PLOT X,Y,B:X=X+2:GOTO 1500 'Vers la droite
1650 IF A=13 THEN 1670 'Appui sur ENTER
1660 GOTO 1510
1670 REM Appui sur ENTER
1680 ON C GOTO 1690,1710,1800
1690 'Memo coin superieur gauche
1700 P1=X:P2=Y:C=2:PLOT X,Y,3:GOTO 1500
1710 'Memo coin inferieur droit et trace rectangle
1720 P3=X:P4=Y
1730 PLOT P1,P2:DRAW X,P2:DRAW X,Y:DRAW P1,Y:DRAW P1,P2
1740 IF X>P1 THEN SX=X ELSE SX=P1 'SX=SUP(X,P1)
1750 IF Y>P2 THEN SY=Y ELSE SY=P2 'SY=SUP(Y,P2)
1760 LOCATE ABS(SX/15+2),ABS(26-SY/15):INPUT "OK (O/N) ";R$:R$=UPPER$(R$)
1770 IF R$<>"O" AND R$<>"N" THEN 1760
1780 IF R$="N" THEN CALL &900C:C=1:X=0:Y=0:GOTO 1550
1790 C=3:GOTO 1550
1800 'Tranfert a la position indiquee
1810 'Memo bloc graphique
1820 CALL &900C 'Restitution ecran initial
```

```
1830 A=P1/2:GOSUB 1970:POKE &9019,B:POKE &901A,C
1840 A=P4/2:GOSUB 1970:POKE &901C,B:POKE &901D,C
1850 POKE &901F,INT(ABS((P3-P1)/8))+1:POKE &9021,ABS((P4-P2)/2)+1
1860 CALL &9018 'MBG
1870 'Transfert
1880 A=X/2:GOSUB 1970:POKE &9029,B:POKE &902A,C
1890 A=Y/2:GOSUB 1970:POKE &902C,B:POKE &902D,C
1900 CALL &9028 'ABG
1910 C=1:X=0:Y=0 'Repositionnement curseur
1920 A$=INKEY$:IF A$="" THEN 1920
1930 A$=UPPER$(A$) 'Conversion majuscule
1940 IF A$="R" THEN CALL &900C:GOTO 1500 'Retour a l'ecran precedent
1950 IF A$="S" THEN SAVE N2$,B,&C000,&3FFF
1960 GOTO 1490 'Nouvelle intervention sur le dessin
1970 'Extraction MSB/LSB d'un nombre 16 bits
1980 C=INT(A/256):B=A-C*256
1990 RETURN
```

Lignes 1110 à 1330 : Chargement des sous-programmes ASSEMBLEUR.

Lignes 1350 à 1470 : Initialisation du programme.

Ligne 1480 : Chargement de l'image.

Ligne 1490 : Sauvegarde de l'image pour permettre l'option « R » par la suite.

Lignes 1550 à 1660 : Gestion du curseur.

Lignes 1670 à 1790 : Mémorisation des positions du curseur suite à l'appui sur la touche « ENTER ».

Lignes 1820 à 1900 : Duplication.

Lignes 1910 à 1960 : Attente d'une action au clavier (R, S ou autre).

Le programme BASIC défini ci-dessus utilise des programmes écrits en **ASSEMBLEUR.**

Les sous-programmes de mémorisation et de restitution d'écrans décrits précédemment sont repris ici. De plus, pour permettre les sauvegarde et restitution de portions d'écrans de tailles plus modestes définies par l'utilisateur, deux sous-programmes supplémentaires sont nécessaires : nous les appelerons **MBG** (Mémorisation de Blocs Graphiques) et **ABG** (Affichage de Blocs Graphiques).

MBG :

Avant de décrire la structure du sous-programme MBG, il est nécessaire de faire un rappel sur la structure de l'écran.

L'écran peut être considéré comme une mémoire RAM implantée entre les adresses #C000 et #FFFF.

Reportez-vous à la partie 5/7 pour avoir le détail du codage de l'écran au niveau pixel.

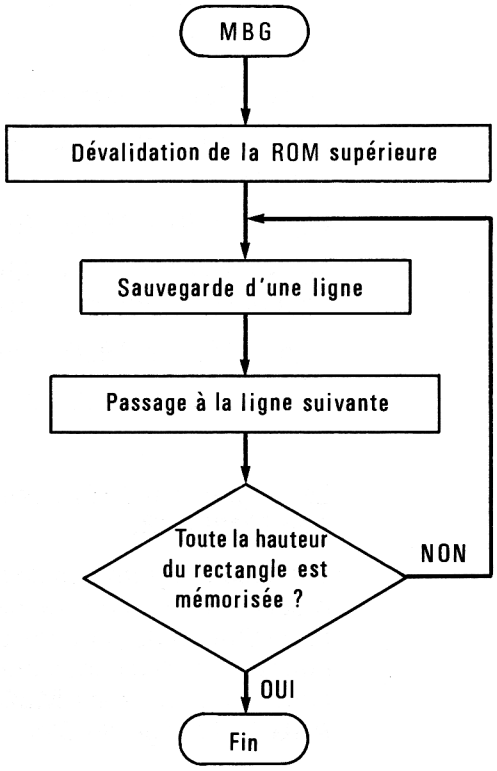
Le problème suivant se pose :

Soit un rectangle de dimensions quelconques dessiné sur l'écran. Comment mémoriser le dessin qui s'y trouve inscrit ?

Vu la complexité structurale de la mémoire d'écran, les constructeurs des CPC ont eu le tact d'agrémenter le FIRMWARE de la routine « DOTPOS » qui, lorsqu'on lui fournit des coordonnées d'écran absolues X et Y, est capable de donner l'adresse de l'octet correspondant en mémoire RAM.

Partant de cette constatation, notre tâche est grandement simplifiée. Connaissant la largeur du rectangle, et vu qu'un octet contient quatre pixels en MODE 1, il sera très simple de sauver une ligne élémentaire. Sachant quelle est la hauteur du rectangle et munis de la routine DOTPOS, le problème devient enfantin. Cependant, précisons un autre petit détail : il ne faut pas oublier de dévalider la ROM supérieure qui occupe également les octets #C000 à #FFFF.

La logique de MBG suit donc l'organigramme suivant :



Voici le listing ASSEMBLEUR correspondant occupant les lignes 1250 à 1270 dans le programme BASIC de recopie d'objets.

```
1                ;
2                ;Memorisateur de blocs graphiques
3                ;
4                ;Entree: BC=Pos X au depart
5                ;        DE=Pos Y au depart
6                ;        H =Largeur en octets
7                ;        L =Hauteur en pixels
8                ;        A =No de bloc a memoriser
9                ;Sortie: Memorisation du bloc
10               ;Pt d'entree: MBG
11               ;
12                            ORG  9C6FH
13                            LOAD 9C6FH
14               MBG:         EQU  $              ;Point d'entree
15 9C6F 180B                  JR   MBGDP          ;Debut du programme
16               ;
17               ;Variables locales
18               ;
19               PX:          DS   2              ;Pos Abs X au depart
20               PY:          DS   2              ;Pos Abs Y au depart
21               LA:          DS   1              ;Largeur d'image(octets)
22               HA:          DS   1              ;Hauteur d'image(pixels)
23               NB:          DS   1              ;No de bloc a memoriser
24               AS:          DS   2              ;@ Sauvegarde bloc
25               SAD:         DS   2              ;Sauvegarde @ mem ecran
26               APS:         EQU  7000H          ;@ 1ere Sauvegarde
27               DOTPOS:      EQU  0BC1DH
28               NTLINE:      EQU  0BC26H
29               ;
```

```
30                      ;Initialisation
31                      ;
32                      MBGDP:      EQU  $
33 9C7C ED43719C                    LD   (PX),BC        ;Pos X au depart
34 9C80 ED53739C                    LD   (PY),DE        ;Pos Y au depart
35 9C84 32779C                      LD   (NB),A         ;No de bloc
36 9C87 7C                          LD   A,H
37 9C88 32759C                      LD   (LA),A         ;Largeur
38 9C8B 7D                          LD   A,L
39 9C8C 32769C                      LD   (HA),A         ;Hauteur
40 9C8F CD03B9                      CALL 0B903H         ;UROM DIS
41                      ;
42 9C92 210070                      LD   HL,APS         ;1ere sauvegarde
43 9C95 119402                      LD   DE,294H        ;Espace entre 2 sauveg.
44 9C98 3A779C                      LD   A,(NB)
45 9C9B 47                          LD   B,A
46 9C9C B7                          OR   A
47 9C9D 2804                        JR   Z,MBGOO
48                      MBGO:       EQU  $              ;Cacul @ Sauvegarde
49 9C9F ED5A                        ADC  HL,DE
50 9CA1 10FC                        DJNZ MBGO
51                      MBGOO:      EQU  $
52 9CA3 3A759C                      LD   A,(LA)
53 9CA6 77                          LD   (HL),A
54 9CA7 23                          INC  HL             ;Largeur en octets
55 9CA8 22789C                      LD   (AS),HL        ;@ Sauvegarde
56                      ;
57 9CAB ED5B719C                    LD   DE,(PX)
58 9CAF 2A739C                      LD   HL,(PY)
59 9CB2 CD1DBC                      CALL DOTPOS         ;OUT HL=@ Mem ecran
60 9CB5 227A9C                      LD   (SAD),HL       ;Sauv mem ecran
```

```
61 9CB8 ED5E789C            LD   DE,(AS)

62              ;
63              MBG1:       EQU  $
64 9CBC 3A759C              LD   A,(LA)
65 9CBF 47                  LD   B,A
66              MBG2:       EQU  $
67 9CC0 7E                  LD   A,(HL)          ;Mem ecran
68 9CC1 12                  LD   (DE),A          ;Mem sauvegarde
69 9CC2 23                  INC  HL
70 9CC3 13                  INC  DE
71 9CC4 10FA                DJNZ MBG2
72 9CC6 2A7A9C              LD   HL,(SAD)
73 9CC9 CD26BC              CALL NTLINE
74 9CCC 227A9C              LD   (SAD),HL
75 9CCF 3A769C              LD   A,(HA)          ;Hauteur
76 9CD2 3D                  DEC  A
77 9CD3 32769C              LD   (HA),A
78 9CD6 20E4                JR   NZ,MBG1
79 9CD8 3ECF                LD   A,0CFH
80 9CDA 12                  LD   (DE),A          ;Terminateur
81 9CDB C9                  RET
82                          END                  ;Zi eind arf arf arf...
```

ABG :

Puisque nous savons sauvegarder en mémoire un objet situé sur l'écran, nous pouvons penser qu'il est simple d'afficher un objet sauvegardé en RAM à une position quelconque sur l'écran. La démarche est effectivement simple : elle consiste à faire l'opposé de la démarche précédente, à savoir :

1°) Lecture d'une ligne élémentaire en mémoire,

2°) Affichage de la ligne sur l'écran,

3°) Passage à la ligne suivante,

4°) Si toute la hauteur du rectangle n'est pas parcourue, retour en 1.

D'où le listing du sous-programme ABG dont les données hexadécimales occupent les lignes 1310 à 1330 dans le programme BASIC de recopie d'objet.

```
1                       ;
2                       ; Afficheur de blocs graphiques
3                       ;
4                       ; Entree: BC=Pos X depart
5                       ;         DE=Pos Y depart
6                       ;         HL=@ RAM depart
7                       ; Sortie: Affichage du bloc
8                       ; Pt d'entree: ABG
9                       ;
10                              ORG  9CDCH
11                              LOAD 9CDCH
12                      ABG:    EQU  $              ;Point d'entree
13 9CDC 180A                    JR   ABGDP          ;Debut programme
14                      ;
15                      ;Variables locales
16                      ;
17                      PX:     DS   2              ;Pos abs X au deaprt
18                      PY:     DS   2              ;Pos abs Y au depart
19                      MI:     DS   2              ;@ de depart RAM image
20                      LA:     DS   2              ;Largeur image
21                      SAD:    DS   2              ;Sauveg @ mem ecran
22                      DOTPOS: EQU  0BC1DH
23                      NTLINE: EQU  0BC26H
24                      ;
25                      ;Initialisation
26                      ;
27                      ABGDP:  EQU  $
28 9CE8 ED43DE9C                LD   (PX),BC        ;Pos X depart
29 9CEC ED53E09C                LD   (PY),DE        ;Pos Y depart
```

```
30 9CF0 7E              LD   A,(HL)
31 9CF1 32E49C          LD   (LA),A          ;Largeur image
32 9CF4 23              INC  HL
33 9CF5 22E29C          LD   (MI),HL         ;@ Mem image
34 9CF8 CD03B9          CALL 0B903H          ;UROM DIS
35              ;
36 9CFB ED5BDE9C        LD   DE,(PX)
37 9CFF 2AE09C          LD   HL,(PY)
38 9D02 CD1DBC          CALL DOTPOS          ;OUT HL=@ Mem ecran
39 9D05 22E69C          LD   (SAD),HL        ;Sauveg @ mem ecran
40 9D08 ED5BE29C        LD   DE,(MI)
41              ABG0:   EQU  $
42 9D0C 3AE49C          LD   A,(LA)
43 9D0F 47              LD   B,A             ;Largeur image
44              ABG1:   EQU  $
45 9D10 1A              LD   A,(DE)
46 9D11 77              LD   (HL),A
47 9D12 23              INC  HL
48 9D13 13              INC  DE
49 9D14 10FA            DJNZ ABG1            ;Affichage 1 ligne elem.
50 9D16 2AE69C          LD   HL,(SAD)
51 9D19 CD26BC          CALL NTLINE          ;Next line
52 9D1C 22E69C          LD   (SAD),HL
53 9D1F 1A              LD   A,(DE)
54 9D20 FECF            CP   0CFH            ;Fin?
55 9D22 20E8            JR   NZ,ABG0         ;Non
56 9D24 C9              RET                  ;Zi einde.
57                      END
```

Remarque :

Pour faciliter l'interfaçage entre le programme BASIC de recopie d'objets et MBG/ABG, un sous-programme interface a été écrit pour MBG et un autre pour ABG.

Le premier donne les informations nécessaires à MBG dans les registres A, BC, DE et HL :

BC est l'abcisse absolue du rectangle source en haut et à gauche.
DE est l'ordonnée absolue du rectangle source en haut et à gauche.

H est la largeur en octets (1 octet = 4 pixels) du rectangle.
L est la hauteur en pixels du rectangle.
A donne le numéro de bloc à sauvegarder (0 si un seul bloc est sauvegardé à la fois. C'est le cas ici.)

Le deuxième donne les informations nécessaires à ABG dans les registres BC, DE et HL :

BC est l'abscisse absolue du rectangle à recopier en haut et à gauche.
DE est l'ordonnée absolue du rectangle à recopier en haut et à gauche.
HL est l'adresse en RAM où se trouve le pavé à recopier (largeur et hauteur sont contenues dans le bloc de mémoire rempli par MBG : il est donc inutile de les préciser ici).

```
1                       ORG  3018H
2                       LOAD 3018H
3               MBG:    EQU  3035H          ;Memo bloc graph
4               ABG:    EQU  30A2H          ;Affich bloc graph
5               ;------------------
6               ;Interface avec MBG
7               ;------------------
8  3018 016400          LD   BC,100         ;X en haut a gauche
9  301B 116400          LD   DE,100         ;Y en haut a gauche
10 301E 260A            LD   H,10           ;Largeur en octets
11 3020 2E0A            LD   L,10           ;Hauteur en pixels
12 3022 3E00            LD   A,0            ;Numero de bloc
13 3024 CD3530          CALL MBG
14 3027 C9              RET
15              ;------------------
16              ;Interface avec ABG
17              ;------------------
18 3028 016400          LD   BC,100         ;X en haut a gauche
19 302B 116400          LD   DE,100         ;Y en haut a gauche
20 302E 2100B0          LD   HL,0B000H      ;@ RAM bloc
21 3031 CDA230          CALL ABG
22 3034 C9              RET
23                      END
```

5/10.2.2

Miroir par rapport à un axe vertical

Supposons qu'un dessin soit inscrit dans un rectangle.

Quelle démarche utiliser pour reproduire le contenu du rectangle réfléchi dans un miroir vertical ?

Pour chaque ligne élémentaire haute d'un pixel , il suffit de prendre les pixels successifs et de les inverser (le plus à gauche va à l'extrême droite et le plus à droite va à l'extrême gauche). Pour simplifier l'approche du problème, nous vous présentons ici un programme de symétrisation écrit en BASIC.

Le listing du programme est le suivant :

```
1000 REM Symetrie d'un objet sur l'ecran %0y
1010 REM ============================================================
1020 REM 1) Determiner le coin INF GAUCHE de l'objet a deplacer
1030 REM 2) Determiner le coin SUP DROIT  de l'objet a deplacer
1040 REM 3) L'objet est encadre
1050 REM 4) Repondre "O" si l'encadrement est correct,"N" sinon
1060 REM 5) Determiner le coin INFERIEUR GAUCHE de la nouvelle position
1070 REM 6) L'objet symetrique %0y apparait a sa nouvelle position
1080 REM ============================================================
1090 INK 0,0:BORDER 0:INK 1,10:INK 3,10,6:PEN 1:MODE 1:C=1 'Initialisation
1100 '
1110 'Prog. ASM Sauvegarde et affichage ecran
1120 '
1130 FOR I=0 TO &17:READ A:POKE &9000+I,A:NEXT
1140 DATA &21,0,&C0,&11,0,&40,1,&FF,&3F,&ED,&B0,&C9
1150 DATA &21,0,&40,&11,0,&C0,1,&FF,&3F,&ED,&B0,&C9
1160 '
1170 PRINT"     SYMETRIE DE BLOCS GRAPHIQUES":PRINT:PRINT
```

```
1180 LOCATE 1,6:INPUT"Affichage monochrome (O/N)";R$:R$=UPPER$(R$)
1190 IF R$<>"O" AND R$<>"N" THEN 1180
1200 IF R$="O" THEN NBCOUL=1 ELSE NBCOUL=3
1210 PRINT
1220 FOR I=0 TO NBCOUL
1230    PRINT"INK";I;": (0 A 26) ";:INPUT A:INK I,A
1240 NEXT I
1250 PRINT:PRINT
1260 INPUT"Nom de l'ecran a charger ";N$
1270 INPUT"Nom de l'ecran a sauver  ";N2$
1280 PRINT:PRINT"Une fois toutes les modifications faites, appuyez sur 'S' pour
obtenir une sau- -vegarde de l'ecran sur K7 ou disqu
ette"
1290 PRINT"Appuyez sur 'R' pour retourner a l'ecran precedent et sur une autre t
ouche pour continuer la symetrisation."
1300 PRINT:PEN 3:PRINT"          Appuyez sur une touche"
1310 a$=INKEY$:IF a$="" THEN 1310
1320 LOAD N$,&C000
1330 CALL &9000 'Sauvegarde ecran charge
1340 B=TEST(X,Y) 'Memo. du pt ou se trouve le curseur
1350 PLOT X,Y,3
1360 '
1370 'Boucle principale
1380 '
1390 A$=INKEY$:IF A$="" THEN 1390 'Attente action
1400 A=ASC(A$)
1410 IF A=55 THEN PLOT X,Y,B:Y=Y+2:X=X-2:GOTO 1340 'En haut a gauche
1420 IF A=57 THEN PLOT X,Y,B:Y=Y+2:X=X+2:GOTO 1340 'En haut a droite
1430 IF A=49 THEN PLOT X,Y,B:Y=Y-2:X=X-2:GOTO 1340 'En bas a gauche
1440 IF A=51 THEN PLOT X,Y,B:Y=Y-2:X=X+2:GOTO 1340 'En bas a droite
1450 IF A=56 THEN PLOT X,Y,B:Y=Y+2:GOTO 1340 'Vers le haut
1460 IF A=50 THEN PLOT X,Y,B:Y=Y-2:GOTO 1340 'Vers le bas
1470 IF A=52 THEN PLOT X,Y,B:X=X-2:GOTO 1340 'Vers la gauche
```

```
1480 IF A=54 THEN PLOT X,Y,B:X=X+2:GOTO 1340 'Vers la droite
1490 IF A=13 THEN 1510 'Appui sur ENTER
1500 GOTO 1350
1510 REM Appui sur ENTER
1520 ON C GOTO 1530,1550,1640
1530 'Memo coin superieur gauche
1540 P1=X:P2=Y:C=2:PLOT X,Y,3:GOTO 1340
1550 'Memo coin inferieur droit et trace rectangle
1560 P3=X:P4=Y
1570 PLOT P1;P2:DRAW X,P2:DRAW X,Y:DRAW P1,Y:DRAW P1,P2
1580 IF X>P1 THEN SX=X ELSE SX=P1 'SX=SUP(X,P1)
1590 IF Y>P2 THEN SY=Y ELSE SY=P2 'SY=SUP(Y,P2)
1600 LOCATE ABS(SX/15+2),ABS(26-SY/15):INPUT "OK (O/N) ";R$:R$=UPPER$(R$)
1610 IF R$<>"O" AND R$<>"N" THEN 1600
1620 IF R$="N" THEN CALL &900C:C=1:X=0:Y=0:GOTO 1390
1630 C=3:GOTO 1390
1640 'Symetrie %Oy
1650 CALL &900C 'Effacement cadre
1660 XF=X+P3-P1:C=0 'Initialisation
1670 FOR I=P2 TO P4 STEP 2
1680   B=0:C=C+2
1690   FOR J=P1 TO P3 STEP 2
1700     B=B+2:A=TEST(J,I):PLOT XF-B,Y+C,A
1710   NEXT J
1720 NEXT I
1730 '
1740 C=1:X=0:Y=0 'Repositionnement curseur
1750 A$=INKEY$:IF A$="" THEN 1750
1760 A$=UPPER$(A$) 'Conversion majuscule
1770 IF A$="R" THEN CALL &900C:GOTO 1340  Retour a l'ecran precedent
1780 IF A$="S" THEN SAVE N2$,B,&C000,&3FFF
1790 GOTO 1330 'Nouvelle intervention sur le dessin
```

Lignes 1130 à 1150 : Chargement des sous-programmes assembleur.

Lignes 1170 à 1310 : Initialisation du programme.

Lignes 1320 : Chargement d'une image.

Lignes 1330 : Sauvegarde pour permettre l'option « R » par la suite.

Lignes 1390 à 1500 : Gestion du curseur.

Lignes 1510 à 1590 : Mémorisation des positions du curseur suite à un appui sur la touche « ENTER ».

Lignes 1640 à 1720 : Fonction miroir vertical.

Lignes 1740 à 1790 : Attente d'une action au clavier (R, S ou autre).

Les sous-programmes ASSEMBLEUR utilisés sont les sous-programmes de mémorisation et de restitution de mémoire d'écran décrits au chap. 10.1 de cette partie.

5/10.2.3

Miroir par rapport à un axe horizontal

Le problème est le même que celui exposé au chapitre 10.2.2 de cette partie mais le miroir est vertical.

Supposons qu'un dessin soit inscrit dans un rectangle.

Quelle démarche utiliser pour reproduire le contenu du rectangle réfléchi dans un miroir horizontal ?

Appelons « rectangle source » le rectangle qui va être réfléchi et « rectangle image » le rectangle réfléchi. Si le rectangle source comporte n lignes élémentaires (en hauteur), il suffit de le parcourir et de faire :

Ligne n – i du rectangle image = Ligne i du rectangle source pour i variant entre 1 et n – 1.

Comme précédemment, pour simplifier l'approche du problème, nous vous proposons un programme écrit en BASIC.

Les sous-programmes écrits en ASSEMBLEUR sont les mêmes que ceux du programme précédent.

Le listing du programme est le suivant :

```
1000 REM Symetrie d'un objet sur l'ecran
1010 REM ================================================================
1020 REM 1) Determiner le coin INF GAUCHE de l'objet a deplacer
1030 REM 2) Determiner le coin SUP DROIT  de l'objet a deplacer
1040 PEM 3) L'objet est encadre
1050 REM 4) Repondre "O" si l'encadrement est correct,"N" sinon
1060 REM 5) Determiner le coin SUPERIEUR GAUCHE de la nouvelle position
1070 REM 6) L'objet symetrique %Oy apparait a sa nouvelle position
1080 REM ================================================================
1090 INK 0,0:BORDER 0:INK 1,10:INK 3,10,6:PEN 1:MODE 1:C=1 'Initialisation
1100 '
1110 'Prog. ASM Sauvegarde et affichage ecran
```

```
1120 '
1130 FOR I=0 TO &17:READ A:POKE &9000+I,A:NEXT
1140 DATA &21,0,&C0,&11,0,&40,1,&FF,&3F,&ED,&B0,&C9
1150 DATA &21,0,&40,&11,0,&C0,1,&FF,&3F,&ED,&B0,&C9
1160 '
1170 PRINT"     SYMETRIE DE BLOCS GRAPHIQUES":PRINT:PRINT
1180 LOCATE 1,6:INPUT"Affichage monochrome (O/N)";R$:R$=UPPER$(R$)
1190 IF R$<>"O" AND R$<>"N" THEN 1180
1200 IF R$="O" THEN NBCOUL=1 ELSE NBCOUL=3
1210 PRINT
1220 FOR I=0 TO NBCOUL
1230    PRINT"INK";I;": (0 A 26) ";:INPUT A:INK I,A
1240 NEXT I
1250 PRINT:PRINT
1260 INPUT"Nom de l'ecran a charger ";N$
1270 INPUT"Nom de l'ecran a sauver  ";N2$
1280 PRINT:PRINT"Une fois toutes les modifications faites, appuyez sur 'S' pour
obtenir une sau- -vegarde de l'ecran sur K7 ou disqu

ette"
1290 PRINT"Appuyez sur 'R' pour retourner a l'ecran precedent et sur une autre t
ouche pour continuer la symetrisation."
1300 PRINT:PEN 3:PRINT"          Appuyez sur une touche"
1310 a$=INKEY$:IF a$="" THEN 1310
1320 LOAD N$,&C000
1330 CALL &9000 'Sauvegarde ecran charge
1340 B=TEST(X,Y) 'Memo. du pt ou se trouve le curseur
1350 PLOT X,Y,3
1360 '
1370 'Boucle principale
1380 '
1390 A$=INKEY$:IF A$="" THEN 1390 'Attente action
1400 A=ASC(A$)
1410 IF A=55 THEN PLOT X,Y,B:Y=Y+2:X=X-2:GOTO 1340 'En haut a gauche
1420 IF A=57 THEN PLOT X,Y,B:Y=Y+2:X=X+2:GOTO 1340 'En haut a droite
```

```
1430 IF A=49 THEN PLOT X,Y,B:Y=Y-2:X=X-2:GOTO 1340 'En bas a gauche
1440 IF A=51 THEN PLOT X,Y,B:Y=Y-2:X=X+2:GOTO 1340 'En bas a droite
1450 IF A=56 THEN PLOT X,Y,B:Y=Y+2:GOTO 1340 'Vers le haut
1460 IF A=50 THEN PLOT X,Y,B:Y=Y-2:GOTO 1340 'Vers le bas
1470 IF A=52 THEN PLOT X,Y,B:X=X-2:GOTO 1340 'Vers la gauche
1480 IF A=54 THEN PLOT X,Y,B:X=X+2:GOTO 1340 'Vers la droite
1490 IF A=13 THEN 1510 'Appui sur ENTER
1500 GOTO 1350
1510 REM Appui sur ENTER
1520 ON C GOTO 1530,1550,1640
1530 'Memo coin superieur gauche
1540 P1=X:P2=Y:C=2:PLOT X,Y,3:GOTO 1340
1550 'Memo coin inferieur droit et trace rectangle
1560 P3=X:P4=Y
1570 PLOT P1,P2:DRAW X,P2:DRAW X,Y:DRAW P1,Y:DRAW P1,P2
1580 IF X>P1 THEN SX=X ELSE SX=P1 'SX=SUP(X,P1)
1590 IF Y>P2 THEN SY=Y ELSE SY=P2 'SY=SUP(Y,P2)
1600 LOCATE SX/15+2,26-SY/15:INPUT "OK (O/N) ";R$:R$=UPPER$(R$)
1610 IF R$<>"O" AND R$<>"N" THEN 1600
1620 IF R$="N" THEN CALL &900C:C=1:X=0:Y=0:GOTO 1390
1630 C=3:GOTO 1390
1640 'Symetrie %0x
1650 CALL &900C 'Effacement cadre
1660 C=0 'Initialisation
1670 FOR I=P2 TO P4 STEP 2
1680   B=0:C=C+2
1690   FOR J=P1 TO P3 STEP 2
1700     B=B+2:A=TEST(J,I):PLOT X+B,Y-C,A
1710   NEXT J
1720 NEXT I
1730 '
1740 C=1:X=0:Y=0 'Repositionnement curseur
```

```
1750 A$=INKEY$:IF A$="" THEN 1750
1760 A$=UPPER$(A$) 'Conversion majuscule
1770 IF A$="R" THEN CALL &900C:GOTO 1340 'Retour a l'ecran precedent
1780 IF A$="S" THEN SAVE N2$,B,&C000,&3FFF
1790 GOTO 1330 'Nouvelle intervention sur le dessin
```

Lignes 1130 à 1150 : Chargement des sous-programmes assembleur.

Lignes 1170 à 1310 : Initialisation du programme.

Ligne 1320 : Chargement d'une image.

Ligne 1330 : Sauvegarde pour permettre l'option « R » par la suite.

Lignes 1390 à 1500 : Gestion du curseur.

Lignes 1510 à 1590 : Mémorisation des positions du curseur suite à un appui sur la touche « ENTER ».

Lignes 1640 à 1720 : Fonction miroir vertical.

Lignes 1740 à 1790 : Attente d'une action au clavier (R, S ou autre).

Remarque sur l'utilisation du programme :

Le troisième appui sur la touche « ENTER » doit se faire lorsque le curseur est positionné en haut et à gauche du rectangle image (et non pas en bas et à gauche comme dans le programme précédent).

5/10.2.4

RSX de manipulation d'images

Afin de compléter la bibliothèque des instructions graphiques du Basic, nous allons vous proposer diverses RSX faciles à intégrer dans un programme écrit en Locomotive Basic.

I. Copie de blocs : RSX ¦COPYBL

L'écran des CPC est composé de 640 pixels horizontaux sur 400 pixels verticaux. Selon la résolution (Mode 0, 1 ou 2), les points élémentaires peuvent être composés de 2, 4 ou 8 pixels. A moins de spécifier le contraire à l'aide d'une instruction **ORIGIN**, le point de coordonnées 0,0 se trouve dans le coin inférieur gauche de l'écran. Les abscisses sont donc croissantes de la gauche vers la droite, et les ordonnées du bas vers le haut. Cette petite introduction a pour but de vous remémorer le fonctionnement du système de coordonnées graphiques des CPC.

La RSX ¦**COPYBL** permet de copier une partie de l'écran délimitée par un rectangle dont deux des extrémités sont spécifiées :

– bord inférieur droit de coordonnées X1, Y1 ;
– bord supérieur gauche de coordonnées X2, Y2 ;

à la position spécifiée : coordonnées X3, Y3.

COMMENT UTILISER LA RSX

Le listing de la RSX est le suivant :

```
1                       ORG  9000H
2                       LOAD 9000H
3               ;
4               ;-------------------------------
5               ; RSX COPYBL
6               ; Format :
7               ;   |COPYBL,X1,Y1,X2,Y2,X3,Y3
8               ; Entree :
9               ;   X1,Y1=Coord gauche source
10              ;   X2,Y2=Coord droite source
11              ;   X3,Y3=Coord gauche dessin
12              ; Sortie : Copie du bloc
13              ;-------------------------------
14              ;
15              ;---------------------------------
16              ; Declaration des constantes
17              ; et des variables du programme
18              ;---------------------------------
19              ;
20              LOGEXT:     EQU  0BCD1H          ;KL LOG EXT
21              TESTABS:    EQU  0BBF0H          ;GRA TEST ABS
22              SETPEN:     EQU  0DBDEH          ;GRA SET PEN
23              PLOTABS:    EQU  0BBEAH          ;GRA PLOT ABS
24              BUF:        DS   4               ;Zone RAM pour LOG EXT
25 9004 0990    PTRTAB:     DW   TABLE           ;Pointeur TABLE
26 9006 C32E90              JP   COPYBL          ;Copie du bloc
27 9009 434F5059 TABLE:     DB   "COPYB"
27 900D 42
28 900E CC                  DB   "L"+80H
29 900F 00                  DB   0               ;Fin de table
30              X1:         DS   2               ;Abs coin sup gauche SRC
```

```
31                      Y1:         DS   2              ;Ord coin sup gauche SRC
32                      X2:         DS   2              ;Abs coin inf gauche SRC
33                      Y2:         DS   2              ;Ord coin inf gauche SRC
34                      X3:         DS   2              ;Abs coin sup gauche BUT
35                      Y3:         DS   2              ;Ord coin sup gauche BUT
36                      XS:         DS   2              ;Abs source courante
37                      YS:         DS   2              ;Ord source courante
38                      XB:         DS   2              ;Abs but courante
39                      YB:         DS   2              ;Ord but courante
40                      ;
41                      ;-----------------------------------
42                      ; Definition de la RSX
43                      ;-----------------------------------
44                      ;
45                      DEFRSX:     EQU  $              ;Point d'entree
46 9024 010490                      LD   BC,PTRTAB      ;Ptr table definition
47 9027 210090                      LD   HL,BUF         ;Buffer pour LOG EXT
48 902A CDD1BC                      CALL LOGEXT         ;Definition de la RSX
49 902D C9                          RET
50                      ;
51                      ;-----------------------------------
52                      ; Traitement de COPYBL
53                      ;-----------------------------------
54                      ;
55                      COPYBL:     EQU  $              ;Point d'entree
56                      ;
57                      ;- - - - - - - - - - - - - - - -
58                      ; Memorisation des parametres
59                      ;- - - - - - - - - - - - - - - -
```

```
60                ;
61 902E DD6601              LD   H,(IX+1)
62 9031 DD6E00              LD   L,(IX+0)
63 9034 221A90              LD   (Y3),HL          ;Ordonnee but
64 9037 DD6603              LD   H,(IX+3)
65 903A DD6E02              LD   L,(IX+2)
66 903D 221890              LD   (X3),HL          ;Abscisse but
67 9040 DD6605              LD   H,(IX+5)
68 9043 DD6E04              LD   L,(IX+4)
69 9046 221690              LD   (Y2),HL          ;Ordonnee droite
70 9049 DD6607              LD   H,(IX+7)
71 904C DD6E06              LD   L,(IX+6)
72 904F 221490              LD   (X2),HL          ;Abscisse droite
73 9052 DD6609              LD   H,(IX+9)
74 9055 DD6E08              LD   L,(IX+8)
75 9058 221290              LD   (Y1),HL          ;Ordonnee gauche
76 905B DD660B              LD   H,(IX+11)
77 905E DD6E0A              LD   L,(IX+10)
78 9061 221090              LD   (X1),HL          ;Abscisse gauche
79                ;
80                ;- - - - - - - - - - - - - - - -
81                ; Initialisation des variables
82                ;- - - - - - - - - - - - - - - -
83                ;
84 9064 2A1090              LD   HL,(X1)
85 9067 221C90              LD   (XS),HL
86 906A 2A1290              LD   HL,(Y1)
87 906D 221E90              LD   (YS),HL
88 9070 2A1890              LD   HL,(X3)
89 9073 222090              LD   (XB),HL
90 9076 2A1A90              LD   HL,(Y3)
```

```
 91 9079 222290              LD    (YB),HL
 92              ;
 93              ;- - - - - - - - - - - -
 94              ; Copie du bloc
 95              ;- - - - - - - - - - - -
 96              ;
 97              BOUCLE:     EQU   $
 98 907C 2A1E90              LD    HL,(YS)
 99 907F ED5B1C90            LD    DE,(XS)
100 9083 CDF0BB              CALL  TESTABS          ;Test coul point
101 9086 CDDEBB              CALL  SETPEN           ;Init couleur
102 9089 2A2290              LD    HL,(YB)
103 908C ED5B2090            LD    DE,(XB)
104 9090 CDEABB              CALL  PLOTABS          ;Copie
105 9093 2A1490              LD    HL,(X2)
106 9096 ED5B1C90            LD    DE,(XS)
107 909A 37                  SCF
108 909B 3F                  CCF
109 909C ED52                SBC   HL,DE
110 909E 7C                  LD    A,H
111 909F B5                  OR    L
112 90A0 2011                JR    NZ,PTSUIV        ;Point suivant
113             ;
114 90A2 2A1690              LD    HL,(Y2)
115 90A5 ED5B1E90            LD    DE,(YS)
116 90A9 37                  SCF
117 90AA 3F                  CCF
118 90AB ED52                SBC   HL,DE
119 90AD 7C                  LD    A,H
```

```
120 90AE B5                 OR   L
121 90AF 2014               JR   NZ,LIGSUIV        ;Ligne suivante
122 90B1 1832               JR   FIN
123                ;
124                ;- - - - - - - - -
125                ; Point suivant
126                ;- - - - - - - - -
127                ;
128                PTSUIV:     EQU  $
129 90B3 37                 SCF
130 90B4 3F                 CCF
131 90B5 2A1C90             LD   HL,(XS)
132 90B8 23                 INC  HL
133 90B9 221C90             LD   (XS),HL           ;XS+1
134 90BC 2A2090             LD   HL,(XB)
135 90BF 23                 INC  HL
136 90C0 222090             LD   (XB),HL           ;XB+1
137 90C3 18B7               JR   BOUCLE
138                ;
139                ;- - - - - - - - - -
140                ; Ligne suivante
141                ;- - - - - - - - - -
142                ;
143                LIGSUIV:    EQU  $
144 90C5 37                 SCF
145 90C6 3F                 CCF
146 90C7 2A1090             LD   HL,(X1)
147 90CA 221C90             LD   (XS),HL           ;XS=X1
148 90CD 2A1E90             LD   HL,(YS)
149 90D0 23                 INC  HL
150 90D1 23                 INC  HL
```

```
151 90D2 221E90              LD    (YS),HL          ;YS+2
152 90D5 2A1890              LD    HL,(X3)
153 90D8 222090              LD    (XB),HL          ;XB=X3
154 90DB 2A2290              LD    HL,(YB)
155 90DE 23                  INC   HL
156 90DF 23                  INC   HL
157 90E0 222290              LD    (YB),HL          ;YB+2
158 90E3 1897                JR    BOUCLE
159                ;
160                FIN:      EQU   $                ;Fin du programme
161 90E5 C9                  RET
162                          END
```

```
BUF         9000 BOUCLE     907C COPYBL     902E DEFRSX     9024
FIN         90E5 LOGEXT     BCD1 LIGSUIV    90C5 PLOTABS    BBEA
PTRTAB      9004 PTSUIV     90B3 SETPEN     BBDE TESTABS    BBF0
TABLE       9009 X1         9010 X2         9014 X3         9018
XS          901C XB         9020 Y1         9012 Y2         9016
Y3          901A YS         901E YB         9022
```

Sa version Assembleur est intéressante pour comprendre son fonctionnement, mais son utilisation réelle se fera sous la forme de données hexadécimales insérées dans un programme Basic.

Pour utiliser la RSX, il faut l'installer à l'aide d'une instruction **CALL** :

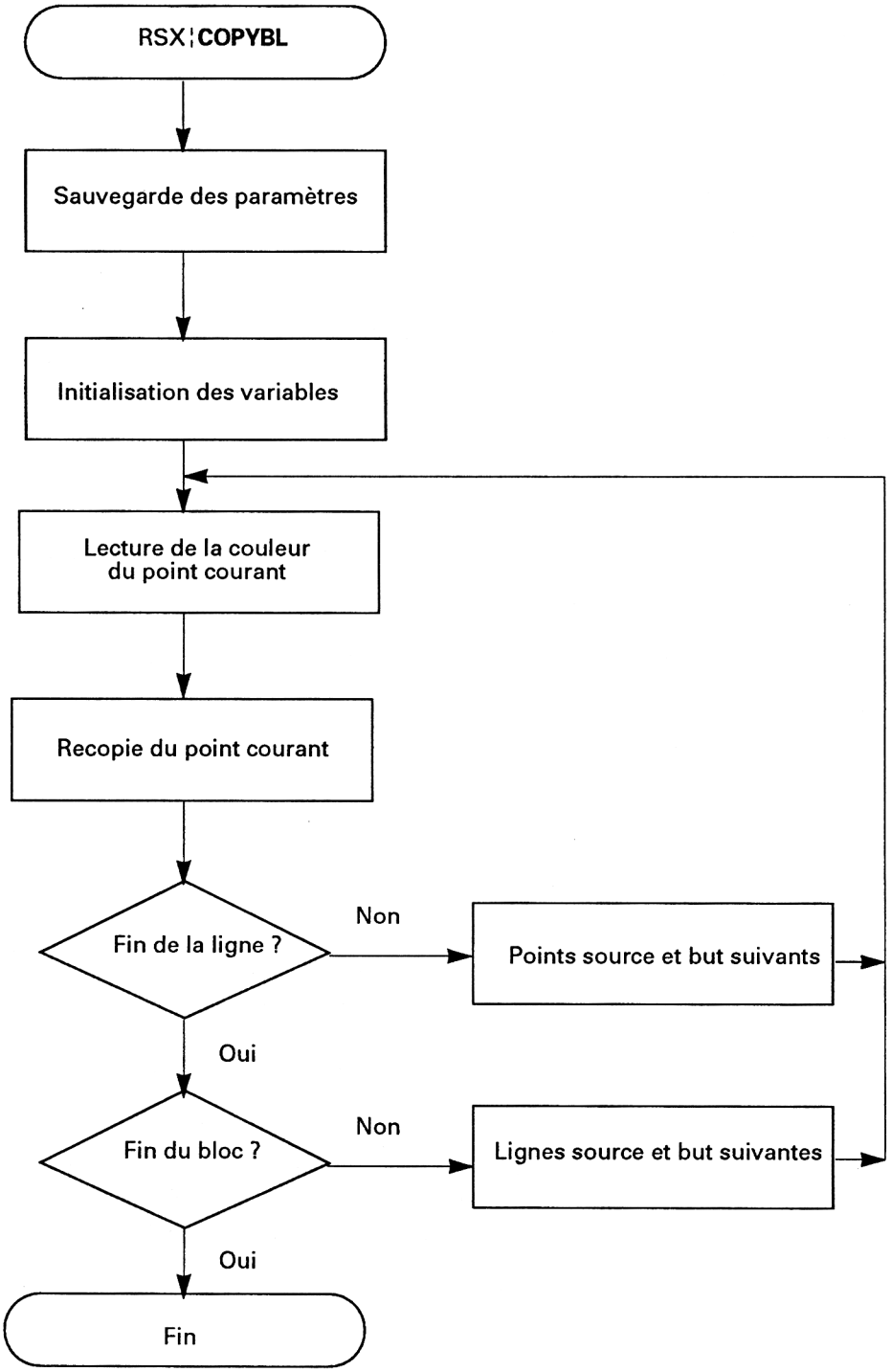
CALL &9024

puis l'appeler en spécifiant les coordonnées extrémales du rectangle à copier (X1, Y1 et X2, Y2) et le coin inférieur gauche à partir duquel doit être affiché la copie (X3, Y3) :

|COPYBL, x1, y1, x2, y2, x3, y3

LA RSX EN DÉTAIL

La logique de la RSX apparaît dans l'ordinogramme suivant :



La RSX débute par la déclaration des constantes et variables utilisées par le programme :

– la primitive **LOGEXT** permet de définir la RSX **|COPYBL** ;
– la primitive **TESTABS** renvoie la couleur d'un point de l'écran ;
– la primitive **SETPEN** définit la couleur de tracé ;
– la primitive **PLOTABS** permet d'allumer un point sur l'écran :

```
LOGEXT:   EQU   0BCD1H   ;KL LOG EXT
TESTABS:  EQU   0BBF0H   ;GRA TEST ABS
SETPEN:   EQU   0BBDEH   ;GRA SET PEN
PLOTABS:  EQU   0BBEAH   ;GRA PLOT ABS
```

Les données suivantes concernent la définition de la RSX. Remarquez en particulier le pointeur **PTRTAB** qui fait référence à l'adresse de traitement de la RSX **(JP COPYBL)**, et la table de définition de la RSX qui contient le mot clé **COPYBL** :

```
BUF:          DS   4          ;Zone RAM pour LOGEXT
PTRTAB:       DW   TABLE      ;Pointeur table
              JP   COPYBL     ;Copie du bloc
TABLE:        DB   "COPYB"
              DB   "L"+80H
              DB   0          ;Fin de table
```

La zone de déclaration des variables se termine par diverses coordonnées d'écran :

```
X1:           DS   2          ;Abs coin sup gauche SRC
...
YB:           DS   2          ;Ord but courante
```

La RSX est définie à l'aide du petit programme situé entre les lignes 45 et 49 :

```
DEFRSX:       EQU  $          ;Point d'entrée
              LD   BC, PTRTAB ;Ptr table définition
              LD   HL, BUF    ;Buffer pour LOG EXT
              CALL LOGEXT     ;Définition de la RSX
              RET
```

Lorsque le mot **|COPYBL** sera reconnu par le Basic, le programme situé à l'étiquette **COPYBL** sera exécuté (le tableau **PTRTAB** pointe sur l'étiquette **COPYBL** et contient le mot clé **COPYBL**).

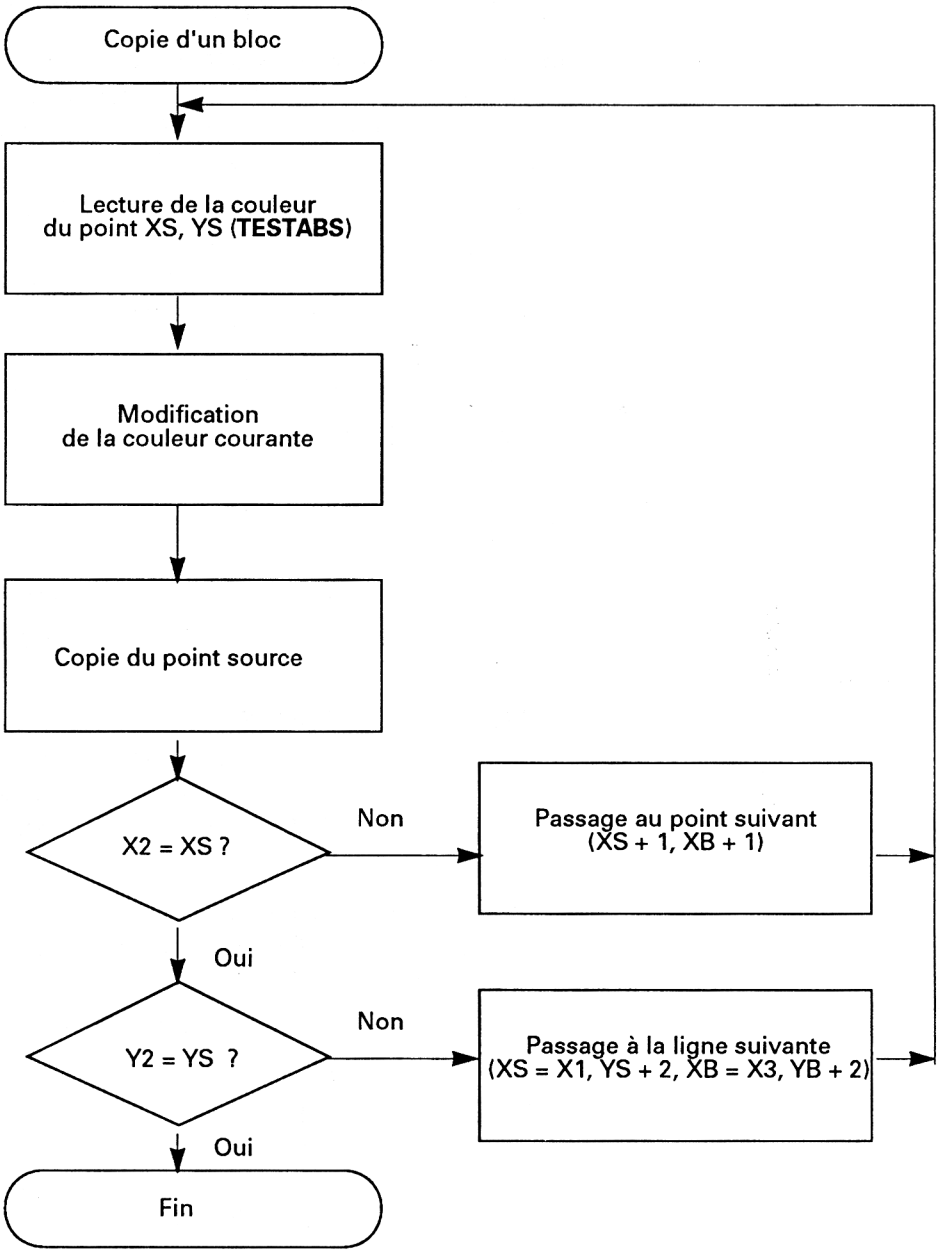
Le programme situé à l'étiquette **COPYBL** débute par la mémorisation des données qui lui sont passées :

```
COPYBL:       EQU  $
              LD   H, (IX+1)
              ...
              LD   (X1), HL
```

Il se poursuit par l'initialisation des coordonnées source (Xs et Ys) et but (Xb et Yb) :

```
LD      HL, (X1)
...
LD      (YB), HL
```

Les lignes qui suivent représentent le cœur du programme. Deux boucles imbriquées copient le rectangle source dans le rectangle but selon la logique suivante :



La couleur du point situé en XS, YS est lue à l'aide de la macro TESTABS :

```
BOUCLE :    EQU  $
            LD   HL, (YS)
            LD   DE, (XS)
            CALL TESTABS     ; Test coul point
```

Cette couleur devra être reproduite dans le bloc but. Aussi est-elle passée à la macro SETPEN qui initialise la couleur de tracé :

```
            CALL SETPEN      ;Init couleur
```

Le point de coordonnées XB, YB est allumé avec la même couleur que le point source :

```
            LD   HL, (YB)
            LD   DE, (XB)
            CALL PLOTABS     ;Copie
```

L'abscisse source est alors comparée à l'extrémité droite du bloc source, dans le but de déterminer si une des lignes du bloc source a été décrite. L'indicateur de retenue est préalablement mis à zéro pour éviter toute erreur dans la soustraction 16 bits entre HL et DE :

```
            LD   HL, (X2)
            LD   DE, (XS)
            SCF
            CCF
            SBC  HL, DE
            LD   A, H
            OR   L
```

Lorsque toute la ligne n'a pas été décrite, le programme donne le contrôle à l'étiquette PTSUIV qui donne accès aux points source et but suivants :

```
            JR   NZ, PTSUIV  ;Point suivant
```

Dans le cas où toute la ligne a été décrite, le programme teste si tout le bloc source a été décrit en comparant les ordonnées Y2 et YS :

```
            LD   HL, (Y2)
            LD   DE, (YS)
            SCF
            CCF
            SBC  HL, DE
            LD   A, H
            OR   L
```

Lorsque tout le bloc source n'a pas été décrit, le programme donne le contrôle à l'étiquette LIGSUIV qui donne accès au premier point de la ligne suivante dans les blocs source et but :

```
        JR    NZ, LIGSUIV      ;Ligne suivante
```

Dans le cas où tout le bloc source a été décrit, le programme redonne le contrôle au Basic :

```
        JR    FIN
        ...
FIN:    EQU   $
        RET
```

Le programme se termine par les deux routines qui permettent de passer au point suivant et à la ligne suivante dans les blocs source et but.

Le passage au point suivant se fait par la simple incrémentation de variables XS et XB :

```
PTSUIV: EQU   $
        SCF
        CCF
        LD    HL, (XS)
        INC   HL
        LD    (XS), HL      ;XS+1
        LD    HL, (XB)
        INC   HL
        LD    (XB), HL
```

Le passage à la ligne suivante est plus complexe :

– l'abscisse source est initialisée à X1 (XS=X1) ;
– l'ordonnée source est incrémentée de 2 (YS=YS+2) ;
– l'abscisse but est initialisée à X3 (XB=X3) ;
– l'ordonnée but est incrémentée de 2 (YB=YB+2) :

```
        SCF
        CCF
        LD    HL, (X1)
        LD    (XS), HL      ;XS=X1
        LD    HL, (YS)
        INC   HL
        INC   HL
        LD    (YS), HL      ;YS+2
        LD    HL, (X3)
        LD    (XB), HL      ;XB=X3
        LD    HL, (YB)
        INC   HL
        INC   HL
        LD    (YP), HL      ;YB+2
```

PROGRAMME D'EXEMPLE EN BASIC

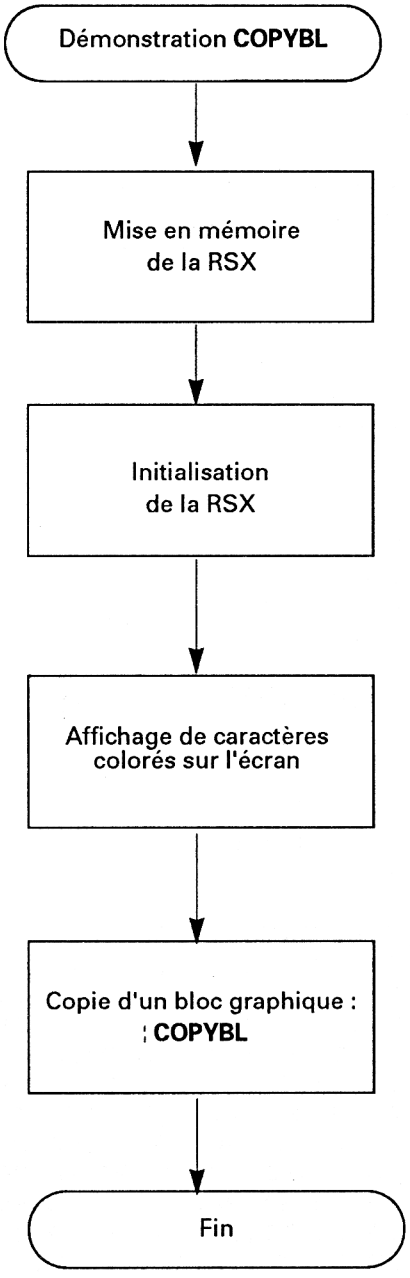
Voici un court programme qui illustre l'utilisation de la RSX |COPYBL en Basic :

```
1000 '================================
1010 ' Demonstration de la RSX COPYBL
1020 '================================
1030 '
1040 FOR i=&9000 TO &90E5
1050   READ a$
1060   a$="&"+a$
1070   a=VAL(a$)
1080   POKE i,a
1090 NEXT i
1100 '
1110 '--------------------------
1120 ' Initialisation de la RSX
1130 '--------------------------
1140 '
1150 CALL &9024
1160 '
1170 '---------------
1180 ' Demonstration
1190 '---------------
1200 '
1210 MODE 1
1220 FOR J=1 TO 7
1230   FOR I=1 TO 40
1240     PEN i MOD 4
1250     PRINT CHR$(J+64);
1260   NEXT i
1270 NEXT j
1280 |COPYBL,0,369,639,399,0,100
1290 PEN 1
1300 END
1310 '
1320 '--------------------------
1330 ' Donnees de la RSX COPYBL
1340 '--------------------------
1350 '
1360 DATA 0,0,0,0,9,90,C3,2E,90,43,4F,50,59,42,CC,0
1370 DATA 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
1380 DATA 0,0,0,0,1,4,90,21,0,90,CD,D1,BC,C9,DD,66
1390 DATA 1,DD,6E,0,22,1A,90,DD,66,3,DD,6E,2,22,18,90
1400 DATA DD,66,5,DD,6E,4,22,16,90,DD,66,7,DD,6E,6,22
1410 DATA 14,90,DD,66,9,DD,6E,8,22,12,90,DD,66,B,DD,6E
1420 DATA A,22,10,90,2A,10,90,22,1C,90,2A,12,90,22,1E,90
1430 DATA 2A,18,90,22,20,90,2A,1A,90,22,22,90,2A,1E,90,ED
1440 DATA 5B,1C,90,CD,F0,BB,CD,DE,BB,2A,22,90,ED,5B,20,90
1450 DATA CD,EA,BB,2A,14,90,ED,5B,1C,90,37,3F,ED,52,7C,B5
1460 DATA 20,11,2A,16,90,ED,5B,1E,90,37,3F,ED,52,7C,B5,20
1470 DATA 14,18,32,37,3F,2A,1C,90,23,22,1C,90,2A,20,90,23
1480 DATA 22,20,90,18,B7,37,3F,2A,10,90,22,1C,90,2A,1E,90
1490 DATA 23,23,22,1E,90,2A,18,90,22,20,90,2A,22,90,23,23
1500 DATA 22,22,90,18,97,C9,0,0,0,0,0,0,0,0,0,0
```

Les données de checksum correspondantes sont les suivantes :

```
67 0 B1 7A 22 A6 4 16 C1 22 3 9B 8B DF 4E
```

La logique de fonctionnement de ce programme apparaît dans l'ordinogramme suivant :



La ligne 1150 initialise la RSX :

1150 CALL &9024

La ligne 1280 copie le bloc de coordonnées extrémales 0,369 et 639,399 en 0,100 :

1280 |COPYBL, 0, 369, 639, 399, 0, 100

Dans la suite logique de la RSX |**COPYBL**, nous allons maintenant étudier trois autres RSX qui effectuent la copie d'un bloc graphique avec :

– symétrie par rapport à un axe horizontal (|**SYMOX**) ;
– symétrie par rapport à un axe vertical (|SYMOY) ;
– symétrie par rapport à un axe horizontal et à un axe vertical (|SYMOXOY).

II. Copie de blocs avec symétrie horizontale : RSX ¦SYMOX

Le listing de la RSX ¦SYMOX est le suivant :

```
1                               ORG  9000H
2                               LOAD 9000H
3                       ;
4                       ;-------------------------------
5                       ; RSX SYMOX
6                       ; Format :
7                       ;   ¦SYMOX,X1,Y1,X2,Y2,X3,Y3
8                       ; Entree :
9                       ;   X1,Y1=Coord gauche source
10                      ;   X2,Y2=Coord droite source
11                      ;   X3,Y3=Coord gauche dessin
12                      ; Sortie : Symetrie OX
13                      ;-------------------------------
14                      ;
15                      ;-------------------------------
16                      ; Declaration des constantes
17                      ; et des variables du programme
18                      ;-------------------------------
19                      ;
20                      LOGEXT:     EQU  0BCD1H         ;KL LOG EXT
21                      TESTABS:    EQU  0BBF0H         ;GRA TEST ABS
22                      SETPEN:     EQU  0BBDEH         ;GRA SET PEN
23                      PLOTABS:    EQU  0BBEAH         ;GRA PLOT ABS
24                      BUF:        DS   4              ;Zone RAM pour LOG EXT
25 9004 0990            PTRTAB:     DW   TABLE          ;Pointeur TABLE
26 9006 C32D90                      JP   SYMOX          ;Symetrie OX
27 9009 53594D4F TABLE:             DB   "SYMO"
28 900D D8                          DB   "X"+80H
29 900E 00                          DB   0              ;Fin de-table
30                      X1:         DS   2              ;Abs coin sup gauche SRC
```

```
31                  Y1:         DS   2              ;Ord coin sup gauche SRC
32                  X2:         DS   2              ;Abs coin inf gauche SRC
33                  Y2:         DS   2              ;Ord coin inf gauche SRC
34                  X3:         DS   2              ;Abs coin sup gauche BUT
35                  Y3:         DS   2              ;Ord coin sup gauche BUT
36                  XS:         DS   2              ;Abs source courante
37                  YS:         DS   2              ;Ord source courante
38                  XB:         DS   2              ;Abs but courante
39                  YB:         DS   2              ;Ord but courante
40                  ;
41                  ;-----------------------------------
42                  ; Definition de la RSX
43                  ;-----------------------------------
44                  ;
45                  DEFRSX:     EQU  $              ;Point d'entree
46 9023 010490                  LD   BC,PTRTAB      ;Ptr table definition
47 9026 210090                  LD   HL,BUF         ;Buffer pour LOG EXT
48 9029 CDD1BC                  CALL LOGEXT         ;Definition de la RSX
49 902C C9                      RET
50                  ;
51                  ;-----------------------------------
52                  ; Traitement de SYMOX
53                  ;-----------------------------------
54                  ;
55                  SYMOX:      EQU  $              ;Point d'entree
56                  ;
57                  ;- - - - - - - - - - - - - - - -
58                  ; Memorisation des parametres
59                  ;- - - - - - - - - - - - - - - -
60                  ;
```

```
61 902D DD6601              LD   H,(IX+1)
62 9030 DD6E00              LD   L,(IX+0)
63 9033 221990              LD   (Y3),HL          ;Ordonnee but
64 9036 DD6603              LD   H,(IX+3)
65 9039 DD6E02              LD   L,(IX+2)
66 903C 221790              LD   (X3),HL          ;Abscisse but
67 903F DD6605              LD   H,(IX+5)
68 9042 DD6E04              LD   L,(IX+4)
69 9045 221590              LD   (Y2),HL          ;Ordonnee droite
70 9048 DD6607              LD   H,(IX+7)
71 904B DD6E06              LD   L,(IX+6)
72 904E 221390              LD   (X2),HL          ;Abscisse droite
73 9051 DD6609              LD   H,(IX+9)
74 9054 DD6E08              LD   L,(IX+8)
75 9057 221190              LD   (Y1),HL          ;Ordonnee gauche
76 905A DD660B              LD   H,(IX+11)
77 905D DD6E0A              LD   L,(IX+10)
78 9060 220F90              LD   (X1),HL          ;Abscisse gauche
79                ;
80                ;- - - - - - - - - - - - - - - - -
81                ; Initialisation des variables
82                ;- - - - - - - - - - - - - - - - -
83                ;
84 9063 37                  SCF
85 9064 3F                  CCF
86 9065 2A1590              LD   HL,(Y2)
87 9068 ED5B1190            LD   DE,(Y1)
88 906C ED52                SBC  HL,DE
89 906E ED5B1990            LD   DE,(Y3)
90 9072 19                  ADD  HL,DE
91 9073 221990              LD   (Y3),HL          ;Transformation de Y3
```

```
 92                ;
 93 9076 2A0F90               LD   HL,(X1)
 94 9079 221B90               LD   (XS),HL
 95 907C 2A1190               LD   HL,(Y1)
 96 907F 221D90               LD   (YS),HL
 97 9082 2A1790               LD   HL,(X3)
 98 9085 221F90               LD   (XB),HL
 99 9088 2A1990               LD   HL,(Y3)
100 908B 222190               LD   (YB),HL
101               ;
102               ;- - - - - - - - - - - - - - -
103               ; Symetrisation OX du bloc
104               ;- - - - - - - - - - - - - - -
105               ;
106               BOUCLE:     EQU  $
107 908E 2A1D90               LD   HL,(YS)
108 9091 ED5B1B90             LD   DE,(XS)
109 9095 CDF0BB               CALL TESTABS          ;Test coul point
110 9098 CDDEBB               CALL SETPEN           ;Init couleur
111 909B 2A2190               LD   HL,(YB)
112 909E ED5B1F90             LD   DE,(XB)
113 90A2 CDEABB               CALL PLOTABS          ;Copie
114 90A5 2A1390               LD   HL,(X2)
115 90A8 ED5B1B90             LD   DE,(XS)
116 90AC 37                   SCF
117 90AD 3F                   CCF
118 90AE ED52                 SBC  HL,DE
119 90B0 7C                   LD   A,H
120 90B1 B5                   OR   L
```

```
121 90B2 2011                JR   NZ,PTSUIV          ;Point suivant
122              ;
123 90B4 2A1590              LD   HL,(Y2)
124 90B7 ED5B1D90            LD   DE,(YS)
125 90BB 37                  SCF
126 90BC 3F                  CCF
127 90BD ED52                SBC  HL,DE
128 90BF 7C                  LD   A,H
129 90C0 B5                  OR   L
130 90C1 2014                JR   NZ,LIGSUIV        ;Ligne suivante
131 90C3 1832                JR   FIN
132              ;
133              ;- - - - - - - - -
134              ; Point suivant
135              ;- - - - - - - - -
136              ;
137              PTSUIV:     EQU  $
138 90C5 37                  SCF
139 90C6 3F                  CCF
140 90C7 2A1B90              LD   HL,(XS)
141 90CA 23                  INC  HL
142 90CB 221B90              LD   (XS),HL           ;XS+1
143 90CE 2A1F90              LD   HL,(XB)
144 90D1 23                  INC  HL
145 90D2 221F90              LD   (XB),HL           ;XB+1
146 90D5 18B7                JR   BOUCLE
147              ;
148              ;- - - - - - - - -
149              ; Ligne suivante
150              ;- - - - - - - - -
151              ;
```

```
152                 LIGSUIV:    EQU  $
153 90D7 37                     SCF
154 90D8 3F                     CCF
155 90D9 2A0F90                 LD   HL,(X1)
156 90DC 221B90                 LD   (XS),HL          ;XS=X1
157 90DF 2A1D90                 LD   HL,(YS)
158 90E2 23                     INC  HL
159 90E3 23                     INC  HL
160 90E4 221D90                 LD   (YS),HL          ;YS+2
161 90E7 2A1790                 LD   HL,(X3)
162 90EA 221F90                 LD   (XB),HL          ;XB=X3
163 90ED 2A2190                 LD   HL,(YB)
164 90F0 2B                     DEC  HL
165 90F1 2B                     DEC  HL
166 90F2 222190                 LD   (YB),HL          ;YB-2
167 90F5 1897                   JR   BOUCLE
168                 ;
169                 FIN:        EQU  $                ;Fin du programme
170 90F7 C9                     RET
171                             END
```

Pour utiliser la RSX, il faut l'installer à l'aide d'une instruction **CALL** :

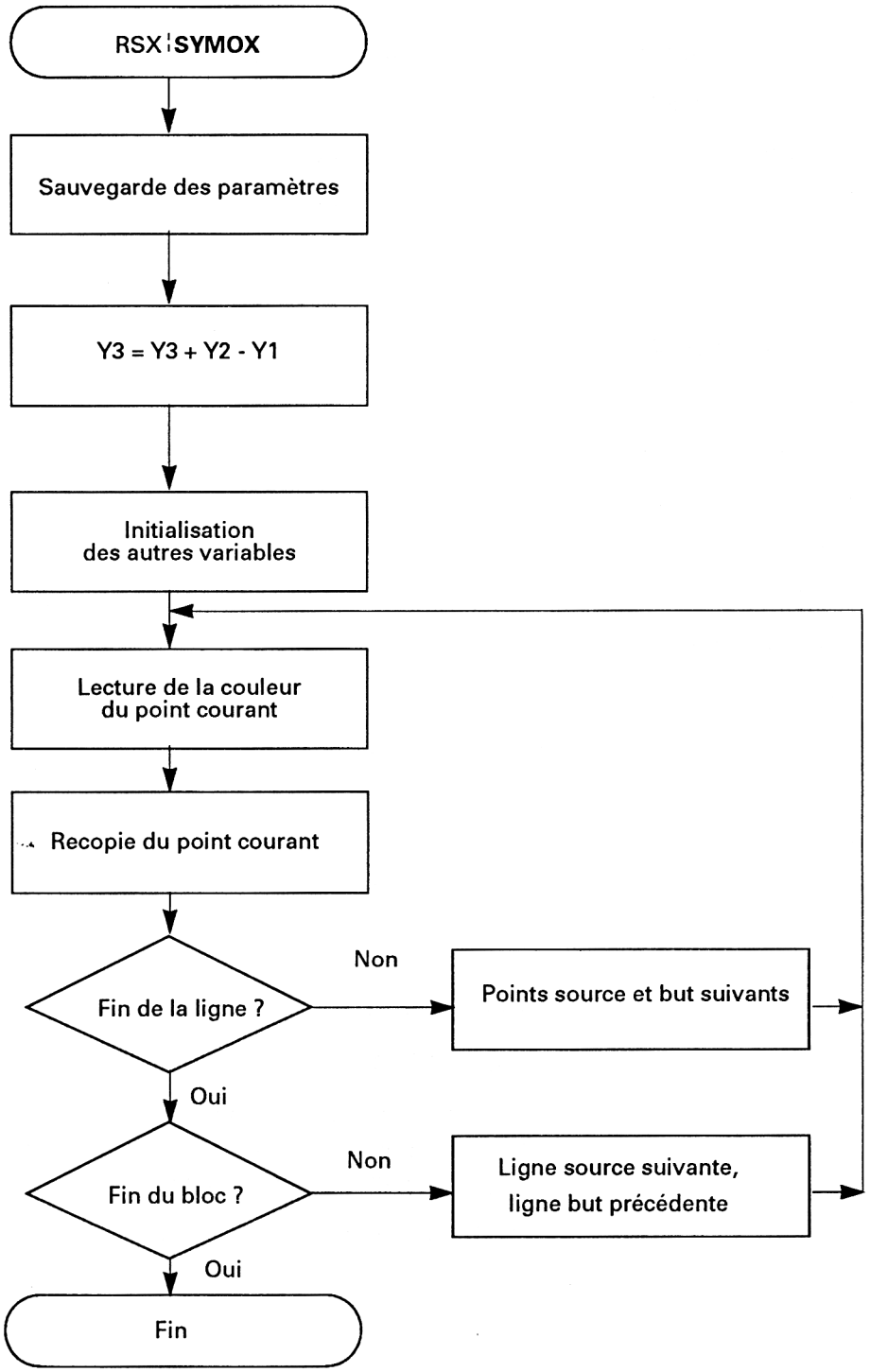
CALL &9023

puis l'appeler en spécifiant les coordonnées extrémales du rectangle à symétriser (X1, Y1 et X2, Y2) et le coin inférieur gauche à partir duquel doit être affichée la copie symétrisée (X3, Y3) :

|SYMOX, x1, y1, x2, y2, x3, y3

LA RSX EN DÉTAIL

La logique de la RSX apparaît dans l'ordinogramme suivant :



Comme vous le voyez, cette RSX est assez proche de la précédente |COPYBL. Les différences majeures sont les suivantes :

- lignes 84 à 91 : la variable Y3 qui représente l'ordonnée gauche du rectangle but est transformée en Y3 + Y2 - Y1 :

```
SCF
CCF
LD    HL, (Y2)
LD    DE, (Y1)
SBC   HL, DE
LD    DE, (Y3)
ADD   HL, DE
LD    (Y3), HL
```

  Grâce à cette modification de la valeur de Y3, la boucle d'affichage permettra d'effectuer la symétrie par rapport à un axe horizontal ;

- lignes 164 à 166 : la routine de passage à la ligne suivante décrémente la variable YB au lieu de l'incrémenter. La copie est donc effectuée vers le bas :

```
LD    HL, (YB)
DEC   HL
DEC   HL
LD    (YB), HL      ;YB-2
```

PROGRAMME D'EXEMPLE EN BASIC

Le programme qui suit illustre l'utilisation de la RSX ¦SYMOX en Basic :

```
1000 '==============================
1010 ' Demonstration de la RSX SYMOX
1020 '==============================
1030 '
1040 FOR i=&9000 TO &90F7
1050   READ a$
1060   a$="&"+a$
1070   a=VAL(a$)
1080   POKE i,a
1090 NEXT i
1100 '
1110 '--------------------------
1120 ' Initialisation de la RSX
1130 '--------------------------
1140 '
1150 CALL &9023
1160 '
1170 '----------------
1180 ' Demonstration
1190 '----------------
1200 '
1210 MODE 1
1220 FOR J=1 TO 7
1230   FOR I=1 TO 40
1240     PEN i MOD 4
1250     PRINT CHR$(J+64);
1260   NEXT i
1270 NEXT j
1280 ¦SYMOX,0,369,639,399,0,100
1290 PEN 1
1300 END
1310 '
1320 '-------------------------
1330 ' Donnees de la RSX SYMOX
1340 '-------------------------
1350 '
1360 DATA 0,0,0,0,9,90,C3,2D,90,53,59,4D,4F,D8,0,0
1370 DATA 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
1380 DATA 0,0,0,1,4,90,21,0,90,CD,D1,BC,C9,DD,66,1
1390 DATA DD,6E,0,22,19,90,DD,66,3,DD,6E,2,22,17,90,DD
1400 DATA 66,5,DD,6E,4,22,15,90,DD,66,7,DD,6E,6,22,13
1410 DATA 90,DD,66,9,DD,6E,8,22,11,90,DD,66,B,DD,6E,A
1420 DATA 22,F,90,37,3F,2A,15,90,ED,5B,11,90,ED,52,ED,5B
1430 DATA 19,90,19,22,19,90,2A,F,90,22,1B,90,2A,11,90,22
1440 DATA 1D,90,2A,17,90,22,1F,90,2A,19,90,22,21,90,2A,1D
1450 DATA 90,ED,5B,1B,90,CD,F0,BB,CD,DE,BB,2A,21,90,ED,5B
1460 DATA 1F,90,CD,EA,BB,2A,13,90,ED,5B,1B,90,37,3F,ED,52
1470 DATA 7C,B5,20,11,2A,15,90,ED,5B,1D,90,37,3F,ED,52,7C
1480 DATA B5,20,14,18,32,37,3F,2A,1B,90,23,22,1B,90,2A,1F
1490 DATA 90,23,22,1F,90,18,B7,37,3F,2A,F,90,22,1B,90,2A
1500 DATA 1D,90,23,23,22,1D,90,2A,17,90,22,1F,90,2A,21,90
1510 DATA 2B,2B,22,21,90,18,97,C9,0,0,0,0,0,0,0,0
```

Les données de checksum correspondantes sont les suivantes :

```
3D 0 B2 55 56 9B 7C 14 40 8D 9D 5D BA 8D 43 A3
```

Sa logique de fonctionnement est similaire à celle du programme Basic illustrant la RSX |**COPYBL**.

III. Copie de blocs avec symétrie verticale : RSX !SYMOY

Le listing de la RSX **!SYMOY** est le suivant :

```
1                       ORG   9000H
2                       LOAD 9000H
3                 ;
4                 ;-------------------------------
5                 ; RSX SYMOY
6                 ; Format :
7                 ;    !SYMOY,X1,Y1,X2,Y2,X3,Y3
8                 ; Entree :
9                 ;    X1,Y1=Coord gauche source
10                ;    X2,Y2=Coord droite source
11                ;    X3,Y3=Coord gauche dessin
12                ; Sortie : Symetrie OY
13                ;-------------------------------
14                ;
15                ;-------------------------------
16                ; Declaration des constantes
17                ; et des variables du programme
18                ;-------------------------------
19                ;
20                LOGEXT:     EQU   0BCD1H          ;KL LOG EXT
21                TESTABS:    EQU   0BBF0H          ;GRA TEST ABS
22                SETPEN:     EQU   0BBDEH          ;GRA SET PEN
23                PLOTABS:    EQU   0BBEAH          ;GRA PLOT ABS
24                BUF:        DS    4               ;Zone RAM pour LOG EXT
25 9004 0990      PTRTAB:     DW    TABLE           ;Pointeur TABLE
26 9006 C32D90                JP    SYMOY           ;Symetrie OY
27 9009 53594D4F  TABLE:      DB    "SYMO"
28 900D D9                    DB    "Y"+80H
29 900E 00                    DB    0               ;Fin de table
30                X1:         DS    2               ;Abs coin sup gauche SRC
31                Y1:         DS    2               ;Ord coin sup gauche SRC
```

```
32                  X2:         DS   2              ;Abs coin inf gauche SRC
33                  Y2:         DS   2              ;Ord coin inf gauche SRC
34                  X3:         DS   2              ;Abs coin sup gauche BUT
35                  Y3:         DS   2              ;Ord coin sup gauche BUT
36                  XS:         DS   2              ;Abs source courante
37                  YS:         DS   2              ;Ord source courante
38                  XB:         DS   2              ;Abs but courante
39                  YB:         DS   2              ;Ord but courante
40                  ;
41                  ;-----------------------------------
42                  ; Definition de la RSX
43                  ;-----------------------------------
44                  ;
45                  DEFRSX:     EQU  $              ;Point d'entree
46 9023 010490                  LD   BC,PTRTAB      ;Ptr table definition
47 9026 210090                  LD   HL,BUF         ;Buffer pour LOG EXT
48 9029 CDD1BC                  CALL LOGEXT         ;Definition de la RSX
49 902C C9                      RET
50                  ;
51                  ;-----------------------------------
52                  ; Traitement de SYMOY
53                  ;-----------------------------------
54                  ;
55                  SYMOY:      EQU  $              ;Point d'entree
56                  ;
57                  ;- - - - - - - - - - - - - - - -
58                  ; Memorisation des parametres
59                  ;- - - - - - - - - - - - - - - -
60                  ;
```

```
61 902D DD6601              LD   H,(IX+1)
62 9030 DD6E00              LD   L,(IX+0)
63 9033 221990              LD   (Y3),HL          ;Ordonnee but
64 9036 DD6603              LD   H,(IX+3)
65 9039 DD6E02              LD   L,(IX+2)
66 903C 221790              LD   (X3),HL          ;Abscisse but
67 903F DD6605              LD   H,(IX+5)
68 9042 DD6E04              LD   L,(IX+4)
69 9045 221590              LD   (Y2),HL          ;Ordonnee droite
70 9048 DD6607              LD   H,(IX+7)
71 904B DD6E06              LD   L,(IX+6)
72 904E 221390              LD   (X2),HL          ;Abscisse droite
73 9051 DD6609              LD   H,(IX+9)
74 9054 DD6E08              LD   L,(IX+8)
75 9057 221190              LD   (Y1),HL          ;Ordonnee gauche
76 905A DD660B              LD   H,(IX+11)
77 905D DD6E0A              LD   L,(IX+10)
78 9060 220F90              LD   (X1),HL          ;Abscisse gauche
79                ;
80                ;- - - - - - - - - - - - - - - -
81                ; Initialisation des variables
82                ;- - - - - - - - - - - - - - - -
83                ;
84 9063 37                  SCF
85 9064 3F                  CCF
86 9065 2A1390              LD   HL,(X2)
87 9068 ED5B0F90            LD   DE,(X1)
88 906C ED52                SBC  HL,DE
89 906E ED5B1790            LD   DE,(X3)
90 9072 19                  ADD  HL,DE
91 9073 221790              LD   (X3),HL          ;Transformation de X3
```

```
 92                 ;
 93 9076 2A0F90                LD    HL,(X1)
 94 9079 221B90                LD    (XS),HL
 95 907C 2A1190                LD    HL,(Y1)
 96 907F 221D90                LD    (YS),HL
 97 9082 2A1790                LD    HL,(X3)
 98 9085 221F90                LD    (XB),HL
 99 9088 2A1990                LD    HL,(Y3)
100 908B 222190                LD    (YB),HL
101                 ;
102                 ;- - - - - - - - - - - - - - -
103                 ; Symetrisation OY du bloc
104                 ;- - - - - - - - - - - - - - -
105                 ;
106                 BOUCLE:     EQU   $
107 908E 2A1D90                LD    HL,(YS)
108 9091 ED5B1B90              LD    DE,(XS)
109 9095 CDF0BB                CALL  TESTABS          ;Test coul point
110 9098 CDDEBB                CALL  SETPEN           ;Init couleur
111 909B 2A2190                LD    HL,(YB)
112 909E ED5B1F90              LD    DE,(XB)
113 90A2 CDEABB                CALL  PLOTABS          ;Copie
114 90A5 2A1390                LD    HL,(X2)
115 90A8 ED5B1B90              LD    DE,(XS)
116 90AC 37                    SCF
117 90AD 3F                    CCF
118 90AE ED52                  SBC   HL,DE
119 90B0 7C                    LD    A,H
120 90B1 B5                    OR    L
```

```
121 90B2 2011                JR   NZ,PTSUIV          ;Point suivant
122              ;
123 90B4 2A1590              LD   HL,(Y2)
124 90B7 ED5B1D90            LD   DE,(YS)
125 90BB 37                  SCF
126 90BC 3F                  CCF
127 90BD ED52                SBC  HL,DE
128 90BF 7C                  LD   A,H
129 90C0 B5                  OR   L
130 90C1 2014                JR   NZ,LIGSUIV         ;Ligne suivante
131 90C3 1832                JR   FIN
132              ;
133              ;- - - - - - - - -
134              ; Point suivant
135              ;- - - - - - - - -
136              ;
137              PTSUIV:     EQU  $
138 90C5 37                  SCF
139 90C6 3F                  CCF
140 90C7 2A1B90              LD   HL,(XS)
141 90CA 23                  INC  HL
142 90CB 221B90              LD   (XS),HL            ;XS+1
143 90CE 2A1F90              LD   HL,(XB)
144 90D1 2B                  DEC  HL
145 90D2 221F90              LD   (XB),HL            ;XB-1
146 90D5 18B7                JR   BOUCLE
147              ;
148              ;- - - - - - - - -
149              ; Ligne suivante
150              ;- - - - - - - - -
151              ;
```

```
152                 LIGSUIV:    EQU  $
153 90D7 37                     SCF
154 90D8 3F                     CCF
155 90D9 2A0F90                 LD   HL,(X1)
156 90DC 221890                 LD   (XS),HL          ;XS=X1
157 90DF 2A1D90                 LD   HL,(YS)
158 90E2 23                     INC  HL
159 90E3 23                     INC  HL
160 90E4 221D90                 LD   (YS),HL          ;YS+2
161 90E7 2A1790                 LD   HL,(X3)
162 90EA 221F90                 LD   (XB),HL          ;XB=X3
163 90ED 2A2190                 LD   HL,(YB)
164 90F0 23                     INC  HL
165 90F1 23                     INC  HL
166 90F2 222190                 LD   (YB),HL          ;YB+2
167 90F5 1897                   JR   BOUCLE
168                 ;
169                 FIN:        EQU  $                ;Fin du programme
170 90F7 C9                     RET
171                             END
```

Pour utiliser la RSX, il faut l'installer à l'aide d'une instruction **CALL** :

CALL &9023

puis l'appeler en spécifiant les coordonnées extrémales du rectangle à symétriser (X1, Y1 et X2, Y2) et le coin inférieur gauche à partir duquel doit être affichée la copie symétrisée (X3, Y3) :

|SYMOY, X1, Y1, X2, Y2, X3, Y3

LA RSX EN DÉTAIL

La logique de la RSX est la même que celle de la RSX |**SYMOX**, à deux détails près :

- lignes 84 à 91 : la variable X3 qui représente l'abscisse gauche du rectangle but est transformée en X3 + X2 - X1 :

```
SCF
CCF
LD   HL, (X2)
LD   DE, (X1)
SBC  HL, DE
LD   DE, (X3)
ADD  HL, DE
LD   (X3), HL
```

  Grâce à cette modification de la valeur de X3, la boucle d'affichage permettra d'effectuer la symétrie par rapport à un axe vertical.

- lignes 143 à 145 : la routine de passage au point suivant diminue la valeur de XB à chaque itération :

```
LD   HL, (XB)
DEC  HL
LD   (XB), HL      ;XB-1
```

- lignes 164 à 166 : la routine de passage à la ligne suivante est identique à la routine de même nom de la RSX |COPYBL.

PROGRAMME D'EXEMPLE EN BASIC

Le programme qui suit est très proche du précédent. Les seules différences se trouvent au niveau :

- de l'appel de la RSX (ligne 1280) :

```
1280 |SYMOY, 0, 369, 639, 399, 0, 100
```

- des données hexadécimales de la RSX (lignes 1360 à 1510) ;

Le listing ci-dessous illustre l'utilisation de la RSX **|SYMOY** en Basic :

```
1000 '================================
1010 ' Demonstration de la RSX SYMOY
1020 '================================
1030 '
1040 FOR i=&9000 TO &90F7
1050   READ a$
1060   a$="&"+a$
1070   a=VAL(a$)
1080   POKE i,a
1090 NEXT i
1100 '
1110 '--------------------------
1120 ' Initialisation de la RSX
1130 '--------------------------
1140 '
1150 CALL &9023
1160 '
1170 '---------------
1180 ' Demonstration
1190 '---------------
1200 '
1210 MODE 1
1220 FOR J=1 TO 7
1230   FOR I=1 TO 40
1240     PEN i MOD 4
1250     PRINT CHR$(J+64);
1260   NEXT i
1270 NEXT j
1280 |SYMOY,0,369,639,399,0,100
1290 PEN 1
1300 END
1310 '
1320 '--------------------------
1330 ' Donnees de la RSX SYMOY
1340 '--------------------------
1350 '
1360 DATA 0,0,0,0,9,90,C3,2D,90,53,59,4D,4F,D9,0,0
1370 DATA 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
1380 DATA 0,0,0,1,4,90,21,0,90,CD,D1,BC,C9,DD,66,1
1390 DATA DD,6E,0,22,19,90,DD,66,3,DD,6E,2,22,17,90,DD
1400 DATA 66,5,DD,6E,4,22,15,90,DD,66,7,DD,6E,6,22,13
1410 DATA 90,DD,66,9,DD,6E,8,22,11,90,DD,66,B,DD,6E,A
1420 DATA 22,F,90,37,3F,2A,13,90,ED,5B,F,90,ED,52,ED,5B
1430 DATA 17,90,19,22,17,90,2A,F,90,22,1B,90,2A,11,90,22
1440 DATA 1D,90,2A,17,90,22,1F,90,2A,19,90,22,21,90,2A,1D
1450 DATA 90,ED,5B,1B,90,CD,F0,BB,CD,DE,BB,2A,21,90,ED,5B
1460 DATA 1F,90,CD,EA,BB,2A,13,90,ED,5B,1B,90,37,3F,ED,52
1470 DATA 7C,B5,20,11,2A,15,90,ED,5B,1D,90,37,3F,ED,52,7C
1480 DATA B5,20,14,18,32,37,3F,2A,1B,90,23,22,1B,90,2A,1F
1490 DATA 90,2B,22,1F,90,18,B7,37,3F,2A,F,90,22,1B,90,2A
1500 DATA 1D,90,23,23,22,1D,90,2A,17,90,22,1F,90,2A,21,90
1510 DATA 23,23,22,21,90,18,97,C9,0,0,0,0,0,0,0,0
```

Les données de checksum correspondantes sont les suivantes :

```
3E 0 B2 55 56 9B 78 10 40 8D 9D 5D BA 95 43 93
```

IV. Copie de blocs avec symétries horizontale et verticale : RSX SYMOXOY

Nous allons terminer ce chapitre avec une RSX qui combine les actions de |SYMOX et |SYMOY .

Le listing de la RSX ¦SYMOXOY est le suivant :

```
1                               ORG  9000H
2                               LOAD 9000H
3                       ;
4                       ;-------------------------------
5                       ; RSX SYMOXOY
6                       ; Format :
7                       ;   ¦SYMOXOY,X1,Y1,X2,Y2,X3,Y3
8                       ; Entree :
9                       ;   X1,Y1=Coord gauche source
10                      ;   X2,Y2=Coord droite source
11                      ;   X3,Y3=Coord gauche dessin
12                      ; Sortie : Symetrie OX OY
13                      ;-------------------------------
14                      ;
15                      ;-------------------------------
16                      ; Declaration des constantes
17                      ; et des variables du programme
18                      ;-------------------------------
19                      ;
20                      LOGEXT:     EQU  0BCD1H         ;KL LOG EXT
21                      TESTABS:    EQU  0BBF0H         ;GRA TEST ABS
22                      SETPEN:     EQU  0BBDEH         ;GRA SET PEN
23                      PLOTABS:    EQU  0BBEAH         ;GRA PLOT ABS
24                      BUF:        DS   4              ;Zone RAM pour LOG EXT
25 9004 0990            PTRTAB:     DW   TABLE          ;Pointeur TABLE
26 9006 C32F90                      JP   SYMOXOY        ;Symetrie OX OY
27 9009 53S94D4F        TABLE:      DB   "SYMOXO"
27 900D 584F
28 900F D9                          DB   "Y"+80H
29 9010 00                          DB   0              ;Fin de table
30                      X1:         DS   2              ;Abs coin sup gauche SRC
```

```
31                      Y1:         DS   2                  ;Ord coin sup gauche SRC
32                      X2:         DS   2                  ;Abs coin inf gauche SRC
33                      Y2:         DS   2                  ;Ord coin inf gauche SRC
34                      X3:         DS   2                  ;Abs coin sup gauche BUT
35                      Y3:         DS   2                  ;Ord coin sup gauche BUT
36                      XS:         DS   2                  ;Abs source courante
37                      YS:         DS   2                  ;Ord source courante
38                      XB:         DS   2                  ;Abs but courante
39                      YB:         DS   2                  ;Ord but courante
40                      ;
41                      ;-----------------------------------
42                      ; Definition de la RSX
43                      ;-----------------------------------
44                      ;
45                      DEFRSX:     EQU  $                  ;Point d'entree
46 9025 010490                      LD   BC,PTRTAB          ;Ptr table definition
47 9028 210090                      LD   HL,BUF             ;Buffer pour LOB EXT
48 902B CDD1BC                      CALL LOBEXT             ;Definition de la RSX
49 902E C9                          RET
50                      ;
51                      ;-----------------------------------
52                      ; Traitement de SYMOXOY
53                      ;-----------------------------------
54                      ;
55                      SYMOXOY:    EQU  $                  ;Point d'entree
56                      ;
57                      ;- - - - - - - - - - - - - - - - -
58                      ; Memorisation des parametres
59                      ;- - - - - - - - - - - - - - - - -
```

```
60
61 902F DD6601              LD   H,(IX+1)
62 9032 DD6E00              LD   L,(IX+0)
63 9035 221B90              LD   (Y3),HL          ;Ordonnee but
64 9038 DD6603              LD   H,(IX+3)
65 903B DD6E02              LD   L,(IX+2)
66 903E 221990              LD   (X3),HL          ;Abscisse but
67 9041 DD6605              LD   H,(IX+5)
68 9044 DD6E04              LD   L,(IX+4)
69 9047 221790              LD   (Y2),HL          ;Ordonnee droite
70 904A DD6607              LD   H,(IX+7)
71 904D DD6E06              LD   L,(IX+6)
72 9050 221590              LD   (X2),HL          ;Abscisse droite
73 9053 DD6609              LD   H,(IX+9)
74 9056 DD6E08              LD   L,(IX+8)
75 9059 221390              LD   (Y1),HL          ;Ordonnee gauche
76 905C DD660B              LD   H,(IX+11)
77 905F DD6E0A              LD   L,(IX+10)
78 9062 221190              LD   (X1),HL          ;Abscisse gauche
79                ;
80                ;- - - - - - - - - - - - - - - - -
81                ; Initialisation des variables
82                ;- - - - - - - - - - - - - - - - -
83                ;
84 9065 37                  SCF
85 9066 3F                  CCF
86 9067 2A1790              LD   HL,(Y2)
87 906A ED5B1390            LD   DE,(Y1)
88 906E ED52                SBC  HL,DE
89 9070 ED5B1B90            LD   DE,(Y3)
90 9074 19                  ADD  HL,DE
```

```
 91 9075 221B90              LD    (Y3),HL          ;Transformation de Y3
 92                ;
 93 9078 37                  SCF
 94 9079 3F                  CCF
 95 907A 2A1590              LD    HL,(X2)
 96 907D ED5B1190            LD    DE,(X1)
 97 9081 ED52                SBC   HL,DE
 98 9083 ED5B1990            LD    DE,(X3)
 99 9087 19                  ADD   HL,DE
100 9088 221990              LD    (X3),HL          ;Transformation de X3
101               ;
102 908B 2A1190              LD    HL,(X1)
103 908E 221D90              LD    (XS),HL
104 9091 2A1390              LD    HL,(Y1)
105 9094 221F90              LD    (YS),HL
106 9097 2A1990              LD    HL,(X3)
107 909A 222190              LD    (XB),HL
108 909D 2A1B90              LD    HL,(Y3)
109 90A0 222390              LD    (YB),HL
110               ;
111               ;- - - - - - - - - - - - - - - -
112               ; Symetrisation OX OY du bloc
113               ;- - - - - - - - - - - - - - - -
114               ;
115               BOUCLE:     EQU   $
116 90A3 2A1F90              LD    HL,(YS)
117 90A6 ED5B1D90            LD    DE,(XS)
118 90AA CDF0BB              CALL  TESTABS          ;Test coul point
119 90AD CDDEBB              CALL  SETPEN           ;Init couleur
```

```
120 90B0 2A2390             LD   HL,(YB)
121 90B3 ED5B2190           LD   DE,(XB)
122 90B7 CDEABB             CALL PLOTABS          ;Copie
123 90BA 2A1590             LD   HL,(X2)
124 90BD ED5B1D90           LD   DE,(XS)
125 90C1 37                 SCF
126 90C2 3F                 CCF
127 90C3 ED52               SBC  HL,DE
128 90C5 7C                 LD   A,H
129 90C6 B5                 OR   L
130 90C7 2011               JR   NZ,PTSUIV        ;Point suivant
131             ;
132 90C9 2A1790             LD   HL,(Y2)
133 90CC ED5B1F90           LD   DE,(YS)
134 90D0 37                 SCF
135 90D1 3F                 CCF
136 90D2 ED52               SBC  HL,DE
137 90D4 7C                 LD   A,H
138 90D5 B5                 OR   L
139 90D6 2014               JR   NZ,LIGSUIV       ;Ligne suivante
140 90D8 1832               JR   FIN
141             ;
142             ;- - - - - - - - -
143             ; Point suivant
144             ;- - - - - - - - -
145             ;
146             PTSUIV:     EQU  $
147 90DA 37                 SCF
148 90DB 3F                 CCF
149 90DC 2A1D90             LD   HL,(XS)
150 90DF 23                 INC  HL
```

```
151 90E0 221D90                 LD    (XS),HL           ;XS+1
152 90E3 2A2190                 LD    HL,(XB)
153 90E6 2B                     DEC   HL
154 90E7 222190                 LD    (XB),HL           ;XB-1
155 90EA 1837                   JR    BOUCLE
156                 ;
157                 ;- - - - - - - - - -
158                 ; Ligne suivante
159                 ;- - - - - - - - - -
160                 ;
161                 LIGSUIV:    EQU   $
162 90EC 37                     SCF
163 90ED 3F                     CCF
164 90EE 2A1190                 LD    HL,(X1)
165 90F1 221D90                 LD    (XS),HL           ;XS=X1
166 90F4 2A1F90                 LD    HL,(YS)
167 90F7 23                     INC   HL
168 90F8 23                     INC   HL
169 90F9 221F90                 LD    (YS),HL           ;YS+2
170 90FC 2A1990                 LD    HL,(X3)
171 90FF 222190                 LD    (XB),HL           ;XB=X3
172 9102 2A2390                 LD    HL,(YB)
173 9105 2B                     DEC   HL
174 9106 2B                     DEC   HL
175 9107 222390                 LD    (YB),HL           ;YB-2
176 910A 1897                   JR    BOUCLE
177                 ;
178                 FIN:        EQU   $                 ;Fin du programme
179 910C C9                     RET
180                             END
```

Pour utiliser la RSX, il faut l'installer à l'aide d'une instruction CALL :

```
CALL &9025
```

puis l'appeler en spécifiant les coordonnées extrémales du rectangle à symétriser (X1, Y1 et X2, Y2) et le coin inférieur gauche à partir duquel doit être affichée la copie symétrisée (X3, Y3) :

```
|SYMOY, x1, y1, x2, y2,x3, y3
```

La RSX en détail

La logique de la RSX combine la logique des deux précédentes. Les variables Y3 (lignes 84 à 91) et X3 (lignes 93 à 100) sont modifiées pour pointer sur le coin supérieur droit du rectangle but. De cette manière la copie du bloc source opèrera une double symétrie.

La routine de passage au point suivant décrémente l'abscisse but (lignes 152 à 154).

La routine de passage à la ligne suivante décrémente l'ordonnée but (lignes 172 à 175).

Programme d'exemple en Basic

Le programme d'exemple est très proche des deux précédents. Les seules différences se trouvent au niveau :

- de l'initialisation de la RSX (ligne 1150) :

  ```
  1150 CALL &9025
  ```

- de l'appel de la RSX (ligne 1280) :

  ```
  1280 |SYMOXOY, 0, 369, 639, 399, 0, 100
  ```

- des données hexadécimales de la RSX (lignes 1360 à 1520).

```
1000 '==============================
1010 ' Demonstration de la RSX SYMOXOY
1020 '==============================
1030 '
1040 FOR i=&9000 TO &910C
1050   READ a$
1060   a$="&"+a$
1070   a=VAL(a$)
1080   POKE i,a
1090 NEXT i
1100 '
1110 '--------------------------
1120 ' Initialisation de la RSX
1130 '--------------------------
1140 '
1150 CALL &9025
1160 '
1170 '---------------
1180 ' Demonstration
1190 '---------------
1200 '
1210 MODE 1
1220 FOR J=1 TO 7
1230   FOR I=1 TO 40
1240     PEN i MOD 4
1250     PRINT CHR$(J+64);
1260   NEXT i
1270 NEXT j
1280 |SYMOXOY,0,369,639,399,0,100
1290 PEN 1
1300 END
1310 '
1320 '--------------------------
1330 ' Donnees de la RSX SYMOXOY
1340 '--------------------------
1350 '
1360 DATA 0,0,0,0,9,90,C3,2F,90,53,59,4D,4F,58,4F,D9
1370 DATA 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
1380 DATA 0,0,0,0,0,1,4,90,21,0,90,CD,D1,BC,C9,DD
1390 DATA 66,1,DD,6E,0,22,1B,90,DD,66,3,DD,6E,2,22,19
1400 DATA 90,DD,66,5,DD,6E,4,22,17,90,DD,66,7,DD,6E,6
1410 DATA 22,15,90,DD,66,9,DD,6E,8,22,13,90,DD,66,B,DD
1420 DATA 6E,A,22,11,90,37,3F,2A,17,90,ED,5B,13,90,ED,52
1430 DATA ED,5B,1B,90,19,22,1B,90,37,3F,2A,15,90,ED,5B,11
1440 DATA 90,ED,52,ED,5B,19,90,19,22,19,90,2A,11,90,22,1D
1450 DATA 90,2A,13,90,22,1F,90,2A,19,90,22,21,90,2A,1B,90
1460 DATA 22,23,90,2A,1F,90,ED,5B,1D,90,CD,F0,BB,CD,DE,BB
1470 DATA 2A,23,90,ED,5B,21,90,CD,EA,BB,2A,15,90,ED,5B,1D
1480 DATA 90,37,3F,ED,52,7C,B5,20,11,2A,17,90,ED,5B,1F,90
1490 DATA 37,3F,ED,52,7C,B5,20,14,18,32,37,3F,2A,1D,90,23
1500 DATA 22,1D,90,2A,21,90,2B,22,21,90,18,B7,37,3F,2A,11
1510 DATA 90,22,1D,90,2A,1F,90,23,23,22,1F,90,2A,19,90,22
1520 DATA 21,90,2A,23,90,2B,2B,22,23,90,18,97,C9,0,0,0
```

Les données de checksum correspondantes sont les suivantes :

```
E7 0 4B 52 91 5C B1 7C B3 AD 89 83 75
D8 2C 48 35
```

5/10.3

Utilitaires de compactage

Devant le peu de mémoire RAM disponible sur les CPC 464, 664 ou même 6128, il nous semble judicieux d'étudier le fonctionnement de routines simples de compactage graphique.

Ces routines vous permettront :

— de stocker plusieurs images en mémoire RAM sans avoir à faire de nombreux accès disquette ou cassette ;

— d'afficher rapidement les images compactées.

Sachant qu'un écran occupe 16 kilo-octets de mémoire RAM, il sera possible de stocker 2 ou 3 écrans sur CPC 464 et 664 et 5 ou 6 sur CPC 6128. Si vous voulez (par exemple) créer des jeux d'aventures, il vous faudra faire des accès disque pour charger chaque nouvel écran, à moins que vous n'utilisiez un décompacteur/afficheur.

Nous étudierons trois types de compactage/décompactage/affichage. Chacun s'applique à des cas très particuliers pour être efficace, et le lecteur pourra créer de nouvelles routines de compactage en partant des propos développés ici.

5/10.3.1

Compactage filiforme

Nous allons étudier :

1) Le principe de compactage d'objets filiformes.

2) Le principe de décompactage et d'affichage de fichiers filiformes.

I. Le compacteur

Les dessins monochromes définis par leurs contours (dessins du genre « fil de fer ») pourront être énormément compactés (jusqu'à un facteur 20 !) par le programme qui va suivre.

Le procédé de compactage est très simple. Il consiste en :

— la définition d'un point quelconque appartenant au dessin,

— le codage de la direction dans laquelle il faut se déplacer pour atteindre le point suivant.

Le programme de compactage filiforme défini est écrit en BASIC et occupe les lignes 1200 à 1450.

Entrez le nom de l'écran à compacter. Cet écran aura été créé par le programme de tracé défini au chapitre 10.1 de la partie 5.

Déplacez ensuite le curseur graphique (grâce aux touches-flèches) jusqu'à rencontrer un point allumé sur le dessin.

Appuyez sur la touche « ENTER ». Le programme trace alors d'une autre couleur (PEN 2) le contour de la forme jusqu'à aboutir à une discontinuité. Arrivé à ce point, le tracé s'arrête. Il faut alors déplacer le curseur pour « sauter » la discontinuité. Dès que le curseur se trouve sur un autre point (voisin) du dessin, appuyez sur la touche « ENTER », et ainsi de suite jusqu'à ce que tout l'objet ait changé de couleur.

Appuyez alors deux fois sur la touche « ESC ». Le programme indique la place occupée par le fichier compacté et propose une sauvegarde magnétique ou un retour au compactage (appui sur « ENTER »).

Le programme de compactage est le suivant :

```
1000 REM ********************
1010 REM Codage de formes
1020 REM ********************
1030 '
1040 'Initialisation
1050 '
1060 'Prog. ASM Sauvegarde et affichage ecran
1070 FOR I=0 TO &17:READ A:POKE &2F00+I,A:NEXT
1080 DATA &21,0,&C0,&11,0,&40,1,&FF,&3F,&ED,&B0,&C9
1090 DATA &21,0,&40,&11,0,&C0,1,&FF,&3F,&ED,&B0,&C9
1100 '
1110 ON BREAK GOSUB 1670 'Sortie du programme
1120 INK 0,0:INK 1,10:INK 3,6,25:BORDER 0:MODE 1
1130 INPUT "Nom de l'ecran a coder ";N$:LOAD N$,&C000
1140 W=10:AG=&3000 'Adresse graphique
1150 GOTO 1490 'Positionnement en debut de Forme
1160 AG=&3000:V1=INT(X/256):V2=X-V1*256:V3=INT(Y)/256:V4=Y-V3*256:POKE AG,V1:POK
E AG+1,V2:POKE AG+2,V3:POKE AG+3,V4:AG=&3003 'Entete
1170 '
1180 ' Codage de la forme en memoire
1190 '
1200 mx=x:my=y
1210 PLOT X,Y,2
1220 W=9 'Init du calcul
1230 IF TEST(X+2,Y)=1 THEN U=X+2:V=Y:D=0:GOSUB 1430
1240 IF TEST(X+2,Y+2)=1 THEN U=X+2:V=Y+2:D=1:GOSUB 1430
1250 IF TEST(X,Y+2)=1 THEN U=X:V=Y+2:D=2:GOSUB 1430
1260 IF TEST(X-2,Y+2)=1 THEN U=X-2:V=Y+2:D=3:GOSUB 1430
1270 IF TEST(X-2,Y)=1 THEN U=X-2:V=Y:D=4:GOSUB 1430
1280 IF TEST(X-2,Y-2)=1 THEN U=X-2:V=Y-2:D=5:GOSUB 1430
1290 IF TEST(X,Y-2)=1 THEN U=X:V=Y-2:D=6:GOSUB 1430
1300 IF TEST(X+2,Y-2)=1 THEN U=X+2:V=Y-2:D=7:GOSUB 1430
1310 IF W=9 THEN 1490
1320 X=WX:Y=WY:PT1=0
```

```
1330 IF G=1 THEN G=0 ELSE G=1

1340 IF G=1 THEN OCT=16*wD ELSE OCT=OCT+wD:AG=AG+1:POKE AG,OCT

1350 IF W=0 THEN PLOT x,y,2:GOTO 1370 'Saut de plume

1360 GOTO 1210

1370 IF G=1 THEN OCT=OCT OR &80 ELSE OCT=OCT OR 8

1380 IF g=1 THEN ag=ag+1

1390 POKE AG,OCT:GOTO 1490

1400 '

1410 'Calcul optimal du prochain point

1420 '

1430 W1=-(TEST(U+2,V)=1)-(TEST(U+2,V+2)=1)-(TEST(U,V+2)=1)-(TEST(U-2,V+2)=1)-(TE
ST(U-2,V)=1)-(TEST(U-2,V-2)=1)-(TEST(U,V-2)=1)-(TEST

(U+2,V-2)=1)

1440 IF W1<W THEN W=W1:WX=U:WY=V:WD=D

1450 RETURN

1460 '

1470 'Positionnement du curseur

1480 '

1490 WX=0:WY=0

1500 IF PT1=1 THEN AG=AG+1:POKE AG,&88 'Point unique

1510 cou=TEST(x,y):PLOT x,y,3

1520 A$=INKEY$:IF A$="" THEN 1520

1530 A=ASC(A$)

1540 PLOT X,Y,COU

1550 IF A=240 THEN Y=Y+2:WY=WY+2

1560 IF A=241 THEN Y=Y-2:WY=WY-2

1570 IF A=242 THEN X=X-2:WX=WX-2

1580 IF A=243 THEN X=X+2:WX=WX+2

1590 IF a<>13 THEN 1510

1600 IF WX>=0 THEN AX=WX/2 ELSE AX=(-WX/2) OR &80

1610 IF WY>=0 THEN AY=WY/2 ELSE AY=(-WY/2) OR &80

1620 AG=AG+1:POKE AG,AX:AG=AG+1:POKE AG,AY:g=0:ag=ag+1

1630 PT1=1:GOTO 1200
```

```
1640 '
1650 'Sortie du programme
1660 '
1670 CALL &2F00 'Sauvegarde ecran
1680 CLS:PRINT"La forme occupe la memoire situee"
1690 PRINT"entre &3000 et ";HEX$(AG+1);"."
1700 PRINT:INPUT"Nom de la sauvegarde (ou ENTER) ";R$
1710 IF LEN(R$)=0 THEN CALL &2F0C:RETURN 'Pas de sauvegarde
1720 POKE AG+1,&FF:SAVE R$,B,&3000,AG+2-&3000 'Sauvegarde
1730 END
```

Lignes 1070 à 1090 : Chargement des sous-programmes Assembleur

Lignes 1110 à 1160: Initialisation du programme

Lignes 1200 à 1390: Codage de la forme

Lignes 1430 à 1450: Calcul optimal du prochain point

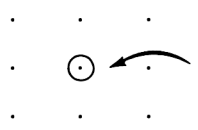
Lignes 1490 à 1630: Positionnement du curseur graphique sur le prochain départ

Ligne 1710 : Retour au compactage si appui sur ENTER

Ligne 1720 : ou sortie avec sauvegarde du fichier compacté

Une technique intéressante employée dans ce programme est *la recherche du prochain point à allumer en créant le moins possible de discontinuités.* Cette technique est basée sur le principe suivant :

Tout point de l'écran peut être entouré de huit façons différentes :



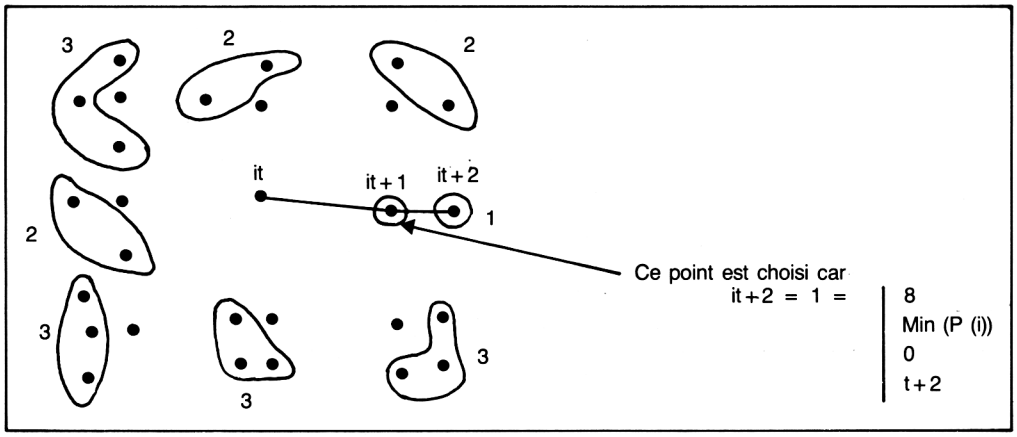
un point quelconque de l'écran

Appelons « t » le contexte actuel, « t+1 » le contexte après un déplacement, et « t+n » le contexte après n déplacements.

Si, pour chacun de ces huit points, nous calculons le nombre de points immédiatement contigus P(i) (et donc le nombre de déplacements possibles en t+2), il apparaît que :

$$\min_{i=0}^{8} (P(i))$$

donnera le point i ayant le moins de chance de provoquer une discontinuité (Calcul de Min (P(i)) effectué ligne 1440, et calcul de P(i) effectué ligne 1430).

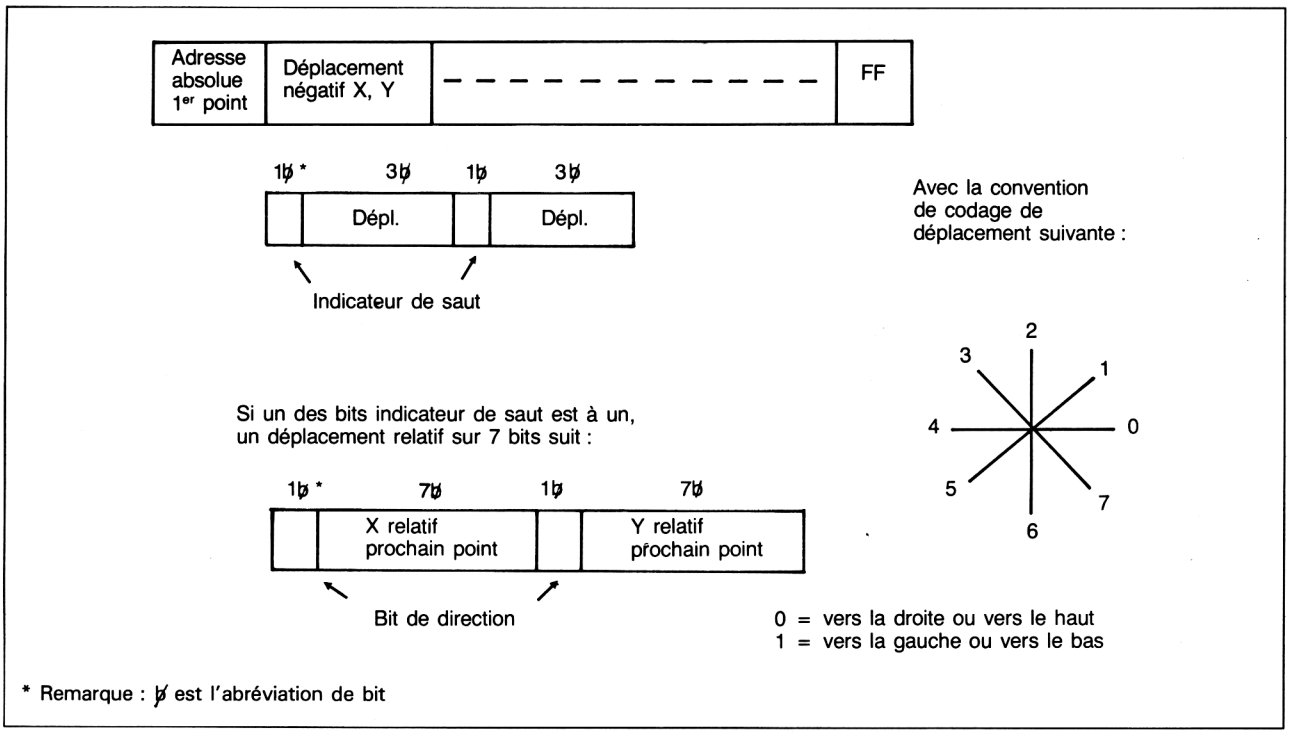


Remarque importante :

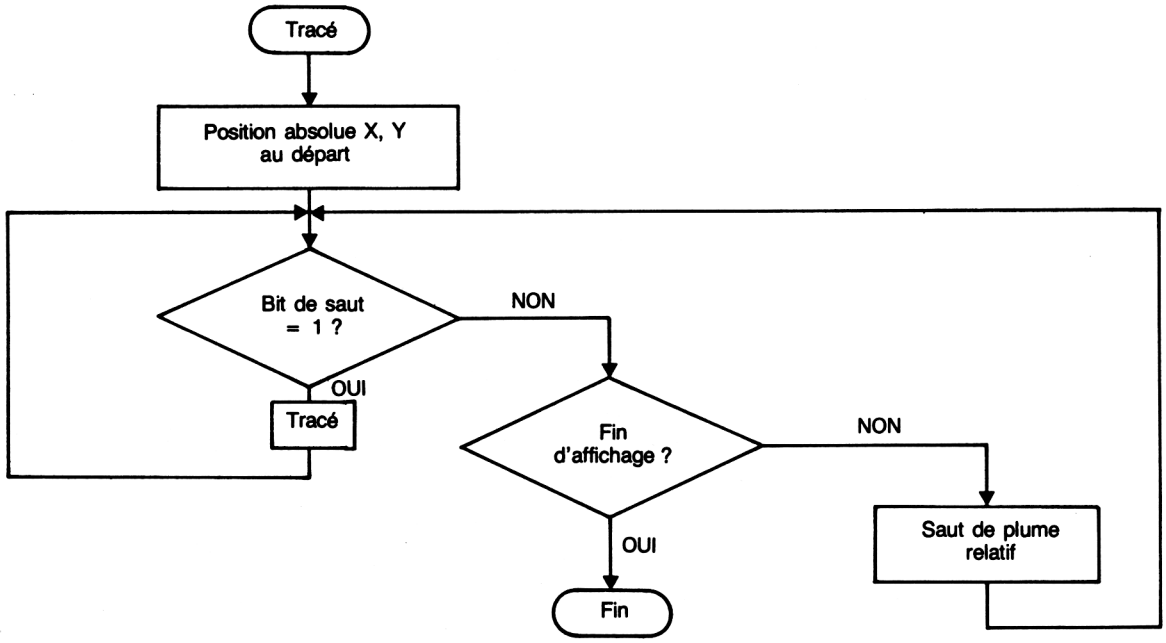
La forme devra OBLIGATOIREMENT être dessinée en PEN 1. Référez-vous aux lignes 1230 à 1300 du listing qui explique le pourquoi de la chose : le calcul du prochain point est effectué en comparant le résultat de la fonction TEST (qui donne la couleur d'un point) à 1.

Le programme défini ci-dessus utilise des sous-programmes écrits en langage d'assemblage pour l'affichage des dessins filiformes.

L'utilisation du langage d'assemblage est quasi obligatoire pour réduire le temps d'affichage qui, malgré tout, n'est pas négligeable (une seconde pour 2 000 points). Le principe est simple. Le fichier compacté généré par le programme précédent possède la structure suivante :



Partant de la structure de ce fichier, voyons comment écrire le programme de tracé.



d'où le programme d'assemblage suivant :

```
1                ;
2                ; Afficheur de formes 4 bits
3                ;
4                ; Entree: HL=à de la forme
5                ; Sortie: Tts registres ecrases
6                ; Pt d'entree: AF0
7                ;
8                          ORG  9000H
9                          LOAD 9000H
10               ;
11               FORM:     DS   2              ;à Forme
12               RT:       DS   1
13               TSAUT:    EQU  $
```

```
14 9003 D090                DW   DIR0
15 9005 D890                DW   DIR1
16 9007 E090                DW   DIR2
17 9009 E890                DW   DIR3
18 900B F090                DW   DIR4
19 900D F890                DW   DIR5
20 900F 0091                DW   DIR6
21 9011 0891                DW   DIR7
22                ;
23                AF0:      EQU  $              ;Point d'entree
24 9013 220090              LD   (FORM),HL      ;Sauvegarde à forme
25 9016 FD2A0090            LD   IY,(FORM)
26 901A FD5600              LD   D,(IY+0)
27 901D FD5E01              LD   E,(IY+1)
28 9020 FD6602              LD   H,(IY+2)
29 9023 FD6E03              LD   L,(IY+3)
30 9026 CDEABB              CALL 0BBEAH         ;Plot absolu
31                ;
32 9029 DD2A0090            LD   IX,(FORM)
33 902D DD23                INC  IX
34 902F DD23                INC  IX
35 9031 DD23                INC  IX
36 9033 DD23                INC  IX             ;IX=à 1er octet forme
37                ;
38                AF000:    EQU  $
39 9035 DD7E00              LD   A,(IX+0)
40                AF00:     EQU  $
41 9038 FE88                CP   88H
42 903A 2841                JR   Z,AF01
43 903C E670                AND  70H
44 903E CB2F                SRA  A
45 9040 CB2F                SRA  A
46 9042 CB2F                SRA  A
```

```
47 9044 CB2F                SRA  A
48 9046 87                  ADD  A,A
49 9047 87                  ADD  A,A
50 9048 87                  ADD  A,A              ;Quartet fort X 8
51 9049 2A0390              LD   HL,(TSAUT)
52 904C 0600                LD   B,0
53 904E 4F                  LD   C,A
54 904F 09                  ADD  HL,BC
55 9050 3E00                LD   A,0
56 9052 320290              LD   (RT),A           ;Memo retour
57 9055 E9                  JP   (HL)             ;Trait quartet fort
58              AF02:       EQU  $
59 9056 DD7E00              LD   A,(IX+0)
60 9059 CB7F                BIT  7,A
61 905B 2020                JR   NZ,AF01          ;Saut de plume
62 905D E607                AND  7
63 905F 87                  ADD  A,A
64 9060 87                  ADD  A,A
65 9061 87                  ADD  A,A              ;Quartet faible X 8
66 9062 2A0390              LD   HL,(TSAUT)
67 9065 0600                LD   B,0
68 9067 4F                  LD   C,A
69 9068 09                  ADD  HL,BC
70 9069 3E01                LD   A,1
71 906B 320290              LD   (RT),A           ;Memo retour
72 906E E9                  JP   (HL)             ;Trait quartet faible
73              ;
74              AF03:       EQU  $
75 906F DD7E00              LD   A,(IX+0)
76 9072 CB5F                BIT  3,A
77 9074 2007                JR   NZ,AF01          ;Saut de plume
78 9076 DD23                INC  IX
```

```
 79 9078 DD7E00              LD    A,(IX+0)
 80 907B 18BB                JR    AF00
 81              ;
 82              AF01:       EQU   $
 83 907D DD23                INC   IX
 84 907F DD7E00              LD    A,(IX+0)
 85 9082 FEFF                CP    0FFH
 86 9084 C8                  RET   Z
 87 9085 CB7F                BIT   7,A
 88 9087 200B                JR    NZ,AF011          ;X negatif
 89 9089 6F                  LD    L,A
 90 908A 2600                LD    H,0
 91 908C 37                  SCF
 92 908D 3F                  CCF
 93 908E ED6A                ADC   HL,HL
 94 9090 54                  LD    D,H
 95 9091 5D                  LD    E,L               ;Sauvegarde
 96 9092 1812                JR    AF012
 97              AF011:      EQU   $
 98 9094 E67F                AND   7FH
 99 9096 6F                  LD    L,A
100 9097 2600                LD    H,0
101 9099 37                  SCF
102 909A 3F                  CCF
103 909B ED6A                ADC   HL,HL
104 909D 54                  LD    D,H
105 909E 5D                  LD    E,L
106 909F 210000              LD    HL,0
107 90A2 ED52                SBC   HL,DE
108 90A4 54                  LD    D,H
109 90A5 5D                  LD    E,L               ;Sauvegarde depl. X rel
110             ;
```

```
111                 AF012:      EQU  $
112 90A6 DD7E01                 LD   A,(IX+1)
113 90A9 CB7F                   BIT  7,A
114 90AB 2009                   JR   NZ,AF013          ;Y negatif
115 90AD 6F                     LD   L,A
116 90AE 2600                   LD   H,0
117 90B0 37                     SCF
118 90B1 3F                     CCF
119 90B2 ED6A                   ADC  HL,HL
120 90B4 1810                   JR   AF014
121                 AF013:      EQU  $
122 90B6 E67F                   AND  7FH
123 90B8 6F                     LD   L,A
124 90B9 2600                   LD   H,0
125 90BB 37                     SCF
126 90BC 3F                     CCF
127 90BD ED6A                   ADC  HL,HL
128 90BF 44                     LD   B,H
129 90C0 4D                     LD   C,L
130 90C1 210000                 LD   HL,0
131 90C4 ED42                   SBC  HL,BC
132                 AF014:      EQU  $
133 90C6 CDEDBB                 CALL 0BBEDH            ;Plot relatif
134 90C9 DD23                   INC  IX
135 90CB DD23                   INC  IX
136 90CD C33590                 JP   AF000             ;Passage a la suite
137                 ;
138                 DIR0:       EQU  $
139 90D0 110200                 LD   DE,2
140 90D3 210000                 LD   HL,0
141 90D6 1836                   JR   DIR8
142                 ;
```

```
143                DIR1:       EQU  $
144 90D8 110200                LD   DE,2
145 90DB 210200                LD   HL,2
146 90DE 182E                  JR   DIR8
147                ;
148                DIR2:       EQU  $
149 90E0 110000                LD   DE,0
150 90E3 210200                LD   HL,2
151 90E6 1826                  JR   DIR8
152                ;
153                DIR3:       EQU  $
154 90E8 11FEFF                LD   DE,0FFFEH
155 90EB 210200                LD   HL,2
156 90EE 181E                  JR   DIR8
157                ;
158                DIR4:       EQU  $
159 90F0 11FEFF                LD   DE,0FFFEH
160 90F3 210000                LD   HL,0
161 90F6 1816                  JR   DIR8
162                ;
163                DIR5:       EQU  $
164 90F8 11FEFF                LD   DE,0FFFEH
165 90FB 21FEFF                LD   HL,0FFFEH
166 90FE 180E                  JR   DIR8
167                ;
168                DIR6:       EQU  $
169 9100 110000                LD   DE,0
170 9103 21FEFF                LD   HL,0FFFEH
171 9106 1806                  JR   DIR8
172                ;
173                DIR7:       EQU  $
174 9108 110200                LD   DE,2
```

```
175 910B 21FEFF              LD   HL,0FFFEH
176              ;
177              DIR8:       EQU  $
178 910E CDEDBB              CALL 0BBEDH
179 9111 3A0290              LD   A,(RT)
180 9114 CB47                BIT  0,A
181 9116 CA5690              JP   Z,AF02
182 9119 C36F90              JP   AF03
183                          END
```

Remarque :

Ce programme utilise la routine du FIRMWARE « PLOT RELATIVE ». Reportez-vous en au chapitre 2.7 de la partie 4 pour avoir plus de détails.

II. Le Décompacteur/Afficheur

Ce programme écrit en BASIC intègre le sous-programme précédent et permet d'afficher simplement une forme à l'écran.

Pour l'utiliser, entrez le nom de la forme à afficher et son implantation en mémoire.

Cette implantation sera toujours &3000 pour une utilisation standard du programme de codage. Pour pouvoir afficher plusieurs formes, il faudra leur donner des adresses d'implantation différentes. Dans ce cas, modifiez le programme de codage lignes 1160, 1690 et 1720 en conséquence. Après avoir chargé la ou les formes, leur dessin apparaît à l'écran.

Le programme d'affichage est le suivant :

```
1000 REM ****************************
1010 REM Affichage de formes codees
1020 REM ****************************
1030 '
1040 'Initialisation
1050 '
1060 MEMORY &3000:MODE 1:INK 0,0:INK 1,10:BORDER 0
1070 PRINT"Ce programme permet d'afficher une ou"
1080 PRINT"plusieurs formes a la sortie du"
1090 PRINT"programme 'Compacteur filiforme'
```

```
1100 '

1110 'Chargement du S/P ASM afficheur de formes

1120 '

1130 FOR I=&9003 TO &911B:READ A:POKE I,A:NEXT I

1140 DATA &D0,&90,&D8,&90,&E0,&90,&E8,&90,&F0,&90,&F8,&90,&0,&91,&8,&91

1150 DATA &22,&0,&90,&FD,&2A,&0,&90,&FD,&56,&0,&FD,&5E,&1,&FD,&66,&2,&FD,&6E,&3,
&CD,&EA,& BB,&DD,&2A,&0,&90,&DD,&23,&DD,&23,&DD,&23,

&DD,&23,&DD,&7E,&0,&FE,&88,&28,&41,&E6,&70,&CB,&2F,&CB,&2F,&CB,&2F,&CB,&2F,&87,&
87,&87,&2A,&3,&90,&6,&0,&4F,&9,&3E,&0,&32

1160 DATA &2,&90,&E9,&DD,&7E,&0,&CB,&7F,&20,&20,&E6,&7,&87,&87,&87,&2A,&3,&90,&6
,&0,&4F,&9,&3E,&1,&32,&2,&90,&E9,&DD,&7E,&0,&CB,&5F,

&20,&7,&DD,&23,&DD,&7E,&0,&18,&BB,&DD,&23,&DD,&7E,&0,&FE,&FF,&C8,&CB,&7F,&20,&B,
&6F,&26,&0,&37,&3F,&ED,&6A,&54,&5D,&18

1170 DATA &12,&E6,&7F,&6F,&26,&0,&37,&3F,&ED,&6A,&54,&5D,&21,&0,&0,&ED,&52,&54,&
5D,&DD,&7E,&1,&CB,&7F,&20,&9,&6F,&26,&0,&37,&3F,&ED,

&6A,&18,&10,&E6,&7F,&6F,&26,&0,&37,&3F,&ED,&6A,&44,&4D,&21,&0,&0,&ED,&42,&CD,&ED
,&BB,&DD,&23,&DD,&23,&C3,&35,&90,&11,&2

1180 DATA &0,&21,&0,&0,&18,&36,&11,&2,&0,&21,&2,&0,&18,&2E,&11,&0,&0,&21,&2,&0,&
18,&26,&11,&FE,&FF,&21,&2,&0,&18,&1E,&11,&FE,&FF,&21

,&0,&0,&18,&16,&11,&FE,&FF,&21,&FE,&FF,&18,&E,&11,&0,&0,&21,&FE,&FF,&18,&6,&11,&
2,&0,&21,&FE,&FF,&CD,&ED,&BB,&3A,&2,&90

1190 DATA &CB,&47,&CA,&56,&90,&C3,&6F,&90

1200 FOR I=0 TO 6:READ A:POKE &9200+I,A:NEXT 'Commande de l'afficheur

1210 DATA &21,0,&30,&CD,&13,&90,&C9

1220 '

1230 'Saisie des formes a afficher

1240 '

1250 NF=NF+1 'Numero de forme

1260 LOCATE 1,10:INPUT"Nom de la forme";N$

1270 INPUT"Implantation memoire";IM(NF)

1280 LOAD N$,IM(NF) 'Chargement forme

1290 INPUT"Position sur l'ecran: X=";X

1300 INPUT"                      Y=";Y

1310 A1=INT(X/256):A2=X-A1*256:A3=INT(Y/256):A4=Y-A3*256

1320 POKE IM(NF)+1,A2:POKE IM(NF)+3,A4:POKE IM(NF),A1:POKE IM(NF)+2,A3

1330 PRINT:INPUT"Une autre forme (O/N) ";R$:R$=UPPER$(R$)
```

```
1340 IF R$<>"O" AND R$<>"N" THEN 1330
1350 IF R$="O" THEN 1250
1360 '
1370 'Affichage de la (des) forme(s)
1380 '
1390 CLS:NF=1 'Affichage forme 1
1400 WHILE IM(NF)<>0
1410   MSB=INT(IM(NF)/256):LSB=IM(NF)-MSB*256 'à sur 8 bits
1420   POKE &9201,LSB:POKE &9202,MSB 'Interface afficheur
1430   CALL &9200 'Affichage
1440   NF=NF+1
1450 WEND
1460 END
```

Lignes 1060 à 1090 : Présentation

Lignes 1130 à 1190 : Chargement du sous-programme afficheur

Lignes 1200 à 1210 : Chargement de l'interface BASIC/ASSEMBLEUR

Lignes 1250 à 1350 : Saisie des formes à afficher

Lignes 1390 à 1460 : Affichage des formes.

Les sous-programmes écrits en langage d'assemblage utilisés sont :

— le programme d'affichage défini dans le compacteur filiforme,

— un programme d'interfaçage avec le BASIC qui consiste à donner dans HL la première adresse de la forme à afficher. Cette adresse est décomposée en poids fort et poids faible Ligne 1300.

5/10.3.2

Compacteurs monochromes en mode 1

Nous allons étudier :

— Le compactage d'objets monochromes selon deux principes :

- le tassement d'octets,
- la recherche de répétition.

— Le décompactage et l'affichage d'objets compactés selon les deux principes.

I. Les compacteurs

TASSEMENT D'OCTETS

Nous avons vu plus haut (reportez-vous à la description du sous-programme ASSEMBLEUR MBG, chapitre 10.2 de la partie 5) quelle était la structure de l'écran en MODE 1.

Si un objet affiché sur l'écran possède seulement deux couleurs :

La couleur de fond et une couleur de motif, il apparaît que l'on peut réduire son occupation en mémoire.

Le principe adopté est le suivant :

Quatre bits sur les huit utilisés pour définir un groupe de 4 pixels sont essentiels pour définir totalement le dessin monochrome.

Remarque :

Dans la suite, nous parlerons de ce compacteur en le désignant par le nom « type 1 ».

Si PEN 1 est la couleur du dessin, les 4 bits seront ceux de poids faible,

Si PEN 2 est la couleur du dessin, les 4 bits seront ceux de poids fort,

Si PEN 3 est la couleur du dessin, les 4 bits seront au choix, ceux de poids faible ou ceux de poids fort.

Cette distinction des couleurs apparaît lignes 1450 à 1480 :

Ligne 1450 : Extraction d'un octet du dessin,

Ligne 1460 : PEN 1,

Ligne 1470 : PEN 2,

Ligne 1480 : PEN 3.

REPETITION DE MOTIFS

Un deuxième type de compactage intéressant et facilement mis en œuvre consiste à compter le nombre d'octets (groupe de 4 pixels en MODE 1) identiques sur une ligne élémentaire, et à coder cette répétition :

Nombre de répétitions, Octet, Terminateur.

Remarque :

Dans la suite, nous parlerons de ce compacteur en le désignant par le nom « type 2 ».

L'écriture du sous-programme BASIC correspondant à ce compactage est immédiate.

Pour chaque ligne du bloc à compacter, une recherche de répétition au niveau octet est effectuée puis codée :

Lignes 1610 à 1630 : Recherche de répétition

Ligne 1640 : Mémorisation

Ligne 1660 : Terminateur

Entrez le nom de l'image à compacter, puis délimitez les coins supérieur gauche et inférieur droit de la portion d'image à compacter en vous servant des touches-flèches (la validation des deux coins se fait par la touche « ENTER »).

Un certain temps, proportionnel aux dimensions du dessin à compacter et à la lenteur du BASIC (!) est nécessaire pour le calcul des fichiers compactés.

Le programme affiche alors la place occupée par le compactage de type 1 et 2.

Le programme de compactage des types 1 et 2 est le suivant :

```
1000 REM Compacteur graphique monochrome simple
1010 '*******************************************************
1020 REM Chargement de l'image d'ecran concernee
1030 '
1040 INK 0,0:INK 1,10:BORDER 0:MODE 1
1050 PRINT"Entrez le nom de l'image a traiter,"
```

```
1060 PRINT"puis servez-vous des touches-fleches"
1070 PRINT"pour delimiter les coins superieur"
1080 PRINT"gauche et inferieur droit de l'image"
1090 PRINT:INPUT "Nom de l'image a traiter ";N$
1100 LOAD N$
1110 '*********************************************************
1120 'Chargement de la routine ASM DOTPOS
1130 '
1140 FOR I=&9006 TO &9013
1150   READ A:POKE I,A
1160 NEXT I
1170 DATA &ED,&5B,&0,&90,&2A,&2,&90,&CD,&1D,&BC,&22,&4,&90,&C9
1180 '*********************************************************
1190 REM Positionnement des coins superieur gauche et inferieur droit
1200 '
1210 X=0: Y=0 'Initialisation du curseur graphique
1220 COU=TEST(X,Y):PLOT X,Y,3
1230 A$=INKEY$:IF A$="" THEN 1230 'Saisie deplacement
1240 A=ASC(A$):PLOT X,Y,COU 'Affichage curseur
1250 IF A=240 THEN Y=Y+2 'Vers le haut
1260 IF A=241 THEN Y=Y-2 'Vers le bas
1270 IF A=242 THEN X=X-2 'Vers la gauche
1280 IF A=243 THEN X=X+2 'Vers la droite
1290 IF A<>13 THEN 1220 'Boucle de saisie
1300 '
1310 P=P+1 'Nombre de saisies
1320 IF P=1 THEN X1=X/2:Y1=Y/2:GOTO 1220 'Saisie coin inferieur droit
1330 IF P=2 THEN X2=X/2:Y2=Y/2 'Fin de saisie
1340 POKE &8004,INT((X2-X1)/4) 'Nombre d'octets horizontalement
1350 '*********************************************************
1360 'Compactage 1
1370 '
```

```
1380 AG=&8005 'Adresse memoire du debut des donnees compactees
1390 FOR Y=Y2 TO Y1
1400   XH=INT(X1/256):XL=X1-XH*256:YH=INT(Y/256):YL=Y-YH*256
1410   POKE &9000,XL:POKE &9001,XH:POKE &9002,YL:POKE &9003,YH
1420   CALL &9006 'DOTPOS
1430   AD=PEEK(&9004)+PEEK(&9005)*256 'Adresse ecran debut de ligne
1440   FOR X=X1 TO X2 STEP 4
1450     A=PEEK(AD+(X-X1)/4) 'Octet graphique
1460     IF A<16 THEN AF=AF+A*(16^(1-PA))
1470     IF A>15 AND INT(A/16)=A/16 THEN AF=AF+(A/16)*(16^(1-PA))
1480     IF A>15 AND INT(A/16)<>A/16 THEN AF=AF+((A AND &F0)/16)*(16^(1-PA))
1490     IF PA=1 THEN PA=0 ELSE PA=1
1500     IF PA=0 THEN AG=AG+1:POKE AG,AF:AF=0 '1 Octet compacte mis en memoire
1510   NEXT X
1520 NEXT Y
1530 IF PA=1 THEN AG=AG+1:POKE AG,AF
1540 AG=AG+1:POKE AG,&FF:AG=AG+1:POKE AG,&AA 'Terminateur
1550 '*******************************************************
1560 ' Compactage 2
1570 '
1580 P1=&8006:P2=&9006 'Pointeurs sur fichiers compactes
1590 A=PEEK(P1):I=1:P1=P1+1:B=PEEK(P1)
1600 IF A=&FF THEN IF B=&AA THEN 1660
1610 WHILE B=A
1620   I=I+1:P1=P1+1:B=PEEK(P1)
1630 WEND
1640 POKE P2,I:POKE P2+1,A:P2=P2+2
1650 GOTO 1590 'Boucle de compactage
1660 POKE P2,A:POKE P2+1,B:P2=P2+2 'Terminateur
1670 '*******************************************************
2000 'Informations sur les longueurs des fichiers compactes
2010 'et eventuelle sauvegarde magnetique
```

```
2020 '
2030 CLS:PRINT"Fichier compacte 1) de &8000 a ";HEX$(AG)
2040 FOR I=&8000 TO &8005
2050   POKE I+&1000,PEEK(I)
2060 NEXT I
2070 POKE &9004,PEEK(&9004)+1
2080 PRINT:PRINT"Fichier compacte 2) de &9000 a ";HEX$(P2)
2090 PRINT:PRINT"Sauvegarde magnetique (O/N) ?"
2100 A$=INKEY$:IF A$="" THEN 2100
2110 A$=UPPER$(A$)
2120 IF A$<>"O" AND A$<>"N" THEN SOUND 1,100,30:GOTO 2100
2130 IF A$="N" THEN 2230 'Fin du programme
2140 PRINT:PRINT"1)Type 1, 2)Type 2 ou 3)Les deux types"
2150 A$=INKEY$:IF A$="" THEN 2150
2160 IF A$<>"1" AND A$<>"2" AND A$<>"3" THEN SOUND 1,100,30:GOTO 2150
2170 PRINT
2180 IF A$="1" THEN INPUT"Nom du fichier type 1: ";N1$
2190 IF A$="2" THEN INPUT"Nom du fichier type 2: ";N2$
2200 IF A$="3" THEN INPUT"Nom du fichier type 1: ";N1$:INPUT"Nom du fichier type
 2: ";N2$
2210 IF N1$<>"" THEN SAVE N1$,B,&8000,AG+1-&8000
2220 IF N2$<>"" THEN SAVE N2$,B,&9000,P2+1-&9000
2230 END
```

Lignes 1040 à 1100 : Initialisation et présentation

Ligne 1100 : Chargement de l'image à traiter

Lignes 1140 à 1170 : Interface Assembleur avec la routine « DOT-POS » (Reportez-vous au chap. 10.2 de la partie 5 pour avoir plus de détails.)

Lignes 1190 à 1240 : Définition des coins supérieur gauche et inférieur droit du bloc à compacter.

Lignes 1400 à 1430 : Interfaçage avec DOTPOS

Lignes 1440 à 1510 : Compactage

Ligne 1530 : Mémorisation

Ligne 1540 : Terminateur (&FFAA)

Lignes 1580 à 1660 : Compacteur de type 2

Lignes 2000 à 2230 : Sauvegarde d'un des compactages

La routine du FIRMWARE « DOTPOS » est utilisée dans ce programme. Reportez-vous au chapitre 2.7 de la partie 4 pour avoir plus de détails à ce sujet.

II. Les décompacteurs-afficheurs

Le programme de compactage « type 1 et 2 » ne serait pas complet sans ce petit programme d'affichage de fichiers compactés de type 1 et 2.

Entrez le nom du fichier compacté par le programme précédent, le type de compactage (1 ou 2) et la position graphique sur l'écran en bas et à gauche du bloc à afficher (0<X<329 et 0<Y<200). Le fichier est chargé et l'affichage est immédiat.

Le programme d'affichage est le suivant :

```
1000 'Affichage de fichiers compactes de type 1 ou 2

1010 '

1020 'Decompacteur de type 1

1030 '

1040 FOR I=&A00F TO &A0DF:READ A:POKE I,A:NEXT

1050 DATA &E5,&21,&0,&A0,&6,&F,&3E,&0,&77,&23,&10,&FC,&E1,&11,&4,&A0,&1,&6,&0,&E
D,&B0,&22,&0,&A0,&FD,&21,&3,&A0,&DD,&2A,&0,&A0,&3A,&

9,&A0,&3D,&28,&D,&3D,&28,&5,&21,&A9,& A0,&18,&8,&21,&93,&A0,&18,&3,&21,&81,&A0,&
22,&8,&A0,&ED,&5B,&4,&A0,&2A,&6,&A0

1060 DATA & CD,&1D,&BC,&3A,&8,&A0,&47,&DD,&7E,&0,&FE,&FF,&20,&8,&DD,&7E,&1,&FE,&
AA,&20,&1,& C9,&DD,&7E,&0,&E6,&F0,&CB,&F,&CB,&F,&CB,

&F,&CB,&F,&57,&DD,&7E,&0,&E6,&F,&5F,&DD,&E5,&DD,&2A,&B,&A0,&DD,&E9,&DD,&E1,&72,&
23,&5,&CC,&CB,&A0,&73,&23,&DD,&23,&5

1070 DATA &CC,&CB,&A0,&18,&C3,&7A,&CB,&7,&CB,&7,&CB,&7,&CB,&7,&57,&7B,&CB,&7,&CB
,&7,&CB,&7,& CB,&7,&5F,&18,&D8,&7A,&FD,&77,&0,&CB,&7

,&CB,&7,&CB,&7,&CB,&7,&FD,&B6,&0,&57,&7B,&FD,&77,&0,&CB,&7,&CB,&7,&CB,&7,&CB,&7,
&FD,&B6,&0,&5F,&18,&B6,&D5,&2A,&6,&A0

1080 DATA &23,&22,&6,&A0,&ED,&5B,&4,&A0,&CD,&1D,&BC,&3A,&8,&A0,&47,&D1,&C9

1090 '------------------------------------------------------------------------

1100 'Decompacteur de type 2

1110 '
```

```
1120 FOR I=&A10B TO &A1DF: READ A:POKE I,A:NEXT

1130 DATA &11,&0,&A1,&1,&6,&0,&ED,&B0,&22,&9,&A1,&FD,&21,&8,&A1,&DD,&2A,&9,&A1,&
2A,&2,&A1,&ED,&5B,&0,&A1,&CD,&1D,&BC,&3A,&4,&A1,&47,

&DD,&7E,&0,&FE,&FF,&20,&8,&DD,&7E,&1,&FE,&AA,&20,&1,&C9,&DD,&4E,&0,&DD,&7E,&1,&E
6,&F0,&CB,&F,&CB,&F,&CB,&F,&CB,&F

1140 DATA &57,&DD,&7E,&1,&E6,&F,&5F,&3A,&5,&A1,&3D,&28,&39,&3D,&28,&22,&7A,&FD,&
77,&0,&CB,&7,&CB,&7,&CB,&7,&CB,&7,&FD,&B6,&0,&57,&7B

,&FD,&77,&0,&CB,&7,&CB,&7,&CB,&7,&CB,&7,& FD,&B6,&0,&5F,&18,&14,&7A,&CB,&7,&CB,&
7,&CB,&7,&CB,&7,&57,&7B,&CB,&7,&CB,&7,&CB

1150 DATA &7,&CB,&7,&5F,&72,&23,&5,&28,&E,&73,&23,&5,&28,&27,&D,&20,&F3,&DD,&23,
&DD,&23,& 18,&88,&C5,&D5,&2A,&2,&A1,&23,&22,&2,&A1,&

ED,&5B,&0,&A1,&CD,&1D,&BC,&22,&6,&A1,&D1,&C1,&2A,&6,&A1,&3A,&4,&A1,&47,&18,&D4,&
C5,&D5,&2A,&2,&A1,&23,&22,&2,&A1,&ED

1160 DATA &5B,&0,&A1,&CD,&1D,&BC,&22,&6,&A1,&D1,&C1,&2A,&6,&A1,&3A,&4,&A1,&47,&1
8,&BB

1170 '------------------------------------------------------------------------------

1180 'Interface BASIC/ASM avec les S/P decompacteur 1 et 2

1190 '

1200 FOR I=&A200 TO &A206:READ A:POKE I,A:NEXT

1210 DATA &21,&0,&30,&CD,&F,&90,&C9

1220 '------------------------------------------------------------------------------

1230 MEMORY &4000:MODE 1:INK 0,0:INK 1,10:BORDER 0:PEN 2:PRINT"   AFFICHEUR DE F
ICHIERS COMPACTES":PEN 1

1240 LOCATE 1,10

1250 INPUT"Nom du fichier ";F$

1260 INPUT"Type de compactage (1 ou 2) ";R

1270 IF R<>1 AND R<>2 THEN SOUND 1,100,30:GOTO 1260

1280 INPUT"Position d'affichage: X=";X

1290 INPUT"                      Y=";Y

1300 '

1310 LOAD F$ 'Chargement du fichier compacte

1320 IF R=1 THEN AD=&8000 ELSE AD=&9000 'Adresse d'implantation du fichier

1330 A1=INT(X/256):A2=X-A1*256:A3=INT(Y/256):A4=Y-A3*256

1340 POKE AD,A2:POKE AD+1,A1:POKE AD+2,A4:POKE AD+3,A3 'Coordonnees X,Y

1350 POKE AD+5,C  'Couleur d'encre

1360 IF R=1 THEN POKE &A202,&80:POKE &A204,&F:POKE &A205,&A0 'Interface DECOM1
```

```
1370 IF R=2 THEN POKE &A202,&90:POKE &A204,&B:POKE &A205,&A1 'Interface DECOM2
1380 POKE &8004,11
1390 CLS:CALL &A200 'Affichage
1400 END
```

Lignes 1040 à 1160 : Chargement des décompacteurs

Lignes 1200 à 1210 : Programme d'interfaçage entre le BASIC et les décompacteurs

Lignes 1250 à 1300 : Entrée du type de compactage et chargement du fichier compacté

Lignes 1320 à 1380 : Interfaçage ASSEMBLEUR

Ligne 1390 : Affichage

Deux sous-programmes sont écrits en ASSEMBLEUR pour minimiser leur temps d'exécution. Il s'agit des programmes de décompactage/affichage.

Décompactage de type 1 :

Ce décompacteur occupe les lignes 1040 à 1080 dans le listing BASIC. Le principe de décompactage est simple. Il consiste à isoler les quartets de poids fort puis faible de chaque octet et à les afficher en tant qu'octets, jusqu'à la rencontre du code terminateur &FFAA qui marque la fin du fichier graphique compacté.

La routine du FIRMWARE « DOTPOS » est utilisée pour calculer l'adresse de la ligne élémentaire suivante. Reportez-vous au chap. 2.7 de la partie 4 pour avoir plus de détails.

```
1               ;
2               ; Afficheur de fichiers graphiques
3               ; de type 1
4               ;
5               ; Entree : HL=à Fichier
6               ; Pt d'entree : TYPE1
7               ; Sortie : Tts registres ecrases
8               ;
9                       ORG  9AAFH
10                      LOAD 9AAFH
11              ;
12              ;Reservation de zones
```

```
13                      ;
14                      DATA:       EQU  $
15                      PTRAD:      DS   2              ;à courante ecran
16 9AB1 00              GEN1:       DB   0              ;Usage general
17                      GEN2:       DS   1              ;Usage general
18                      SX:         DS   2              ;Sauvegarde en X
19                      SY:         DS   2              ;Sauvegarde en Y
20                      LA:         DS   1              ;Larg. en oct. du dessin
21                      CE:         DS   1              ;Couleur d'encre
22                      LC:         DS   1              ;Oct. largeur courante
23                      TR:         DS   2              ;à de trait=F(couleur)
24                      DEFI:       DS   2              ;Debut fichier graph.
25                      ;
26                      DOTPOS:     EQU  0BC1DH         ;Dot Position
27                      ;
28                      ;Initialisation
29                      ;
30                      TYPE1:      EQU  $
31 9ABE E5                          PUSH HL             ;Sauv à data
32 9ABF 21AF9A                      LD   HL,DATA        ;Debut de RAZ
33 9AC2 060F                        LD   B,15           ;Lgr
34 9AC4 3E00                        LD   A,0
35                      RAZ:        EQU  $
36 9AC6 77                          LD   (HL),A
37 9AC7 23                          INC  HL
38 9AC8 10FC                        DJNZ RAZ
39 9ACA E1                          POP  HL             ;Restitution à data
40                      ;
41                      ;HL=à Source
42 9ACB 11B39A                      LD   DE,SX          ;Destination
43 9ACE 010600                      LD   BC,6           ;Longueur du transfert
44 9AD1 EDB0                        LDIR                ;Transfert
```

```
45 9AD3 22AF9A              LD   (PTRAD),HL        ;Pointeur debut DATA
46 9AD6 FD21B29A            LD   IY,GEN2
47 9ADA DD2AAF9A            LD   IX,(PTRAD)
48              ;
49              ;Initialisation couleur d'encre
50              ;
51 9ADE 3AB89A              LD   A,(CE)            ;Couleur d'encre
52 9AE1 3D                  DEC  A
53 9AE2 280D                JR   Z,COU1            ;Couleur 1
54 9AE4 3D                  DEC  A
55 9AE5 2805                JR   Z,COU2            ;Couleur 2
56              COU3:       EQU  $                 ;Couleur 3
57 9AE7 21589B              LD   HL,TR3            ;à de traitement 3
58 9AEA 1808                JR   COUFIN
59              COU2:       EQU  $
60 9AEC 21429B              LD   HL,TR2            ;à de traitement 2
61 9AEF 1803                JR   COUFIN
62              COU1:       EQU  $                 ;Couleur 1
63 9AF1 21309B              LD   HL,TR1            ;à de traitement 1
64              COUFIN:     EQU  $
65 9AF4 22BA9A              LD   (TR),HL           ;à de trait. choisie
66              ;
67              ;
68              AFO:        EQU  $                 ;Debut affichage
69              ;Initialisation
70 9AF7 ED5BB39A            LD   DE,(SX)
71 9AFB 2AB59A              LD   HL,(SY)
72 9AFE CD1DBC              CALL DOTPOS            ;Position Ecran
73 9B01 3AB79A              LD   A,(LA)
74 9B04 47                  LD   B,A               ;B=Nbre Octets largeur
75              ;
76              ;Decompactage
```

```
77                ;
78                AFO:        EQU   $
79  9B05 DD7E00               LD    A,(IX+0)
80  9B08 FEFF                 CP    0FFH
81  9B0A 2008                 JR    NZ,AF00
82  9B0C DD7E01               LD    A,(IX+1)
83  9B0F FEAA                 CP    0AAH
84  9B11 2001                 JR    NZ,AF00
85  9B13 C9                   RET                 ;Terminateur rencontre
86                AF00:       EQU   $
87  9B14 DD7E00               LD    A,(IX+0)
88  9B17 E6F0                 AND   0F0H
89  9B19 CB0F                 RRC   A
90  9B1B CB0F                 RRC   A
91  9B1D CB0F                 RRC   A
92  9B1F CB0F                 RRC   A
93  9B21 57                   LD    D,A           ;Quartet fort
94  9B22 DD7E00               LD    A,(IX+0)
95  9B25 E60F                 AND   0FH
96  9B27 5F                   LD    E,A           ;Quartet faible
97  9B28 DDE5                 PUSH  IX
98  9B2A DD2ABA9A             LD    IX,(TR)
99  9B2E DDE9                 JP    (IX)          ;Traitement d'un octet
100               ;
101               TR1:        EQU   $             ;Affichage
102 9B30 DDE1                 POP   IX
103 9B32 72                   LD    (HL),D
104 9B33 23                   INC   HL
105 9B34 05                   DEC   B
106 9B35 CC7A9B               CALL  Z,LISUI       ;Ligne suivante
107 9B38 73                   LD    (HL),E
108 9B39 23                   INC   HL
```

```
109 9B3A DD23               INC  IX
110 9B3C 05                 DEC  B
111 9B3D CC7A9B             CALL Z,LISUI          ;Ligne suivante
112 9B40 18C3               JR   AFO              ;Boucle d'affichage
113              ;
114              TR2:       EQU  $                ;Affichage couleur 2
115 9B42 7A                 LD   A,D
116 9B43 CB07               RLC  A
117 9B45 CB07               RLC  A
118 9B47 CB07               RLC  A
119 9B49 CB07               RLC  A
120 9B4B 57                 LD   D,A
121 9B4C 7B                 LD   A,E
122 9B4D CB07               RLC  A
123 9B4F CB07               RLC  A
124 9B51 CB07               RLC  A
125 9B53 CB07               RLC  A
126 9B55 5F                 LD   E,A
127 9B56 18D8               JR   TR1              ;Affichage
128              TR3:       EQU  $                ;Affichage couleur 3
129 9B58 7A                 LD   A,D
130 9B59 FD7700             LD   (IY+0),A
131 9B5C CB07               RLC  A
132 9B5E CB07               RLC  A
133 9B60 CB07               RLC  A
134 9B62 CB07               RLC  A
135 9B64 FDB600             OR   (IY+0)
136 9B67 57                 LD   D,A
137 9B68 7B                 LD   A,E
138 9B69 FD7700             LD   (IY+0),A
139 9B6C CB07               RLC  A
140 9B6E CB07               RLC  A
```

```
141 9B70 CB07              RLC  A
142 9B72 CB07              RLC  A
143 9B74 FDB600            OR   (IY+0)
144 9B77 5F                LD   E,A
145 9B78 18B6              JR   TR1            ;Affichage
146              ;
147              LISUI:    EQU  $              ;Affich. ligne suiv.
148 9B7A D5                PUSH DE
149 9B7B 2AB59A            LD   HL,(SY)
150 9B7E 23                INC  HL
151 9B7F 22B59A            LD   (SY),HL
152 9B82 ED5BB39A          LD   DE,(SX)
153 9B86 CD1DBC            CALL DOTPOS         ;HL=à mem ecran
154 9B89 3AB79A            LD   A,(LA)
155 9B8C 47                LD   B,A
156 9B8D D1                POP  DE
157 9B8E C9                RET                 ;Ligne suivante
158                        END
```

Décompacteur de type 2 :

Il occupe les lignes 1120 à 1160 du listing BASIC.

Le principe est différent du précédent. Il consiste en l'isolement de deux octets : le deuxième représente le motif à répéter, le premier le nombre de répétitions à effectuer. Le deuxième octet est affiché n fois et l'opération se répète jusqu'à la rencontre du terminateur &FFAA.

Comme dans le sous-programme précédent, la routine du FIRMWARE « DOTPOS » est utilisée pour calculer l'adresse de la ligne élémentaire suivante lors de l'affichage.

```
1                ;
2                ;Afficheur de fichiers graphiques
3                ; de type 2 compactes par COMPAC2
4                ;
5                ; Entree : HL=à Fichier
6                ; Pt d'entree : TYPE2
```

```
 7                ; Sortie : Ts registres ecrases
 8                ;
 9                                ORG  91FCH
10                                LOAD 91FCH
11                ;
12                XABS:           EQU  9000H
13                YABS:           EQU  9002H
14                LAOC:           EQU  9004H
15                COEN:           EQU  9005H
16                DEDA:           EQU  9006H
17                DOTPOS:         EQU  0BC1DH
18                ;
19                DATA:           EQU  $                  ;Debut des DS
20                SX:             DS   2
21                SY:             DS   2
22                LA:             DS   1
23                CE:             DS   1
24                POS:            DS   2
25                GEN1:           DS   1
26                PTRAD:          DS   2
27                ;
28                ;
29                ;Initialisation
30                ;
31                ;HL=à Source
32                TYPE2:          EQU  $
33 9207 11FC91                    LD   DE,SX              ;Destination
34 920A 010600                    LD   BC,6               ;Longueur
35 920D EDB0                      LDIR                    ;Transfert
36 920F 220592                    LD   (PTRAD),HL
37 9212 FD210492                  LD   IY,GEN1
38 9216 DD2A0592                  LD   IX,(PTRAD)
```

```
39                      ;
40 921A 2AFE91          LD    HL,(SY)
41 921D ED5BFC91        LD    DE,(SX)
42 9221 CD1DBC          CALL  DOTPOS
43                      ;
44                      ;Decompactage
45                      ;
46 9224 3A0092          LD    A,(LA)
47 9227 47              LD    B,A
48                      AF0:  EQU   $
49 9228 DD7E00          LD    A,(IX+0)
50 922B FEFF            CP    0FFH
51 922D 2008            JR    NZ,AF00
52 922F DD7E01          LD    A,(IX+1)
53 9232 FEAA            CP    0AAH
54 9234 2001            JR    NZ,AF00
55 9236 C9              RET                 ;Terminateur rencontre
56                      ;
57                      AF00: EQU   $
58 9237 DD4E00          LD    C,(IX+0)      ;Nb de repetitions
59 923A DD7E01          LD    A,(IX+1)
60 923D E6F0            AND   0F0H
61 923F CB0F            RRC   A
62 9241 CB0F            RRC   A
63 9243 CB0F            RRC   A
64 9245 CB0F            RRC   A
65 9247 57              LD    D,A           ;Octet gauche isole
66 9248 DD7E01          LD    A,(IX+1)
67 924B E60F            AND   0FH
68 924D 5F              LD    E,A           ;Octet droit isole
69                      ;
70 924E 3A0192          LD    A,(CE)
```

```
 71 9251 3D                     DEC   A
 72 9252 2839                   JR    Z,AF3             ;Encre demandee=1
 73 9254 3D                     DEC   A
 74 9255 2822                   JR    Z,INK2            ;Encre demandee=2
 75              INK3:          EQU   $
 76 9257 7A                     LD    A,D
 77 9258 FD7700                 LD    (IY+0),A
 78 925B CB07                   RLC   A
 79 925D CB07                   RLC   A
 80 925F CB07                   RLC   A
 81 9261 CB07                   RLC   A
 82 9263 FDB600                 OR    (IY+0)
 83 9266 57                     LD    D,A
 84 9267 7B                     LD    A,E
 85 9268 FD7700                 LD    (IY+0),A
 86 926B CB07                   RLC   A
 87 926D CB07                   RLC   A
 88 926F CB07                   RLC   A
 89 9271 CB07                   RLC   A
 90 9273 FDB600                 OR    (IY+0)
 91 9276 5F                     LD    E,A
 92 9277 1814                   JR    AF3               ;Suite
 93              ;
 94              INK2:          EQU   $
 95 9279 7A                     LD    A,D
 96 927A CB07                   RLC   A
 97 927C CB07                   RLC   A
 98 927E CB07                   RLC   A
 99 9280 CB07                   RLC   A
100 9282 57                     LD    D,A
101 9283 7B                     LD    A,E
102 9284 CB07                   RLC   A
```

```
103 9286 CB07                RLC   A
104 9288 CB07                RLC   A
105 928A CB07                RLC   A
106 928C 5F                  LD    E,A
107              ;
108              ;Affichage
109              ;
110              AF3:        EQU   $
111 928D 72                  LD    (HL),D
112 928E 23                  INC   HL
113 928F 05                  DEC   B
114 9290 280E                JR    Z,AF1
115              AF31:       EQU.  $
116 9292 73                  LD    (HL),E
117 9293 23                  INC   HL
118 9294 05                  DEC   B
119 9295 2827                JR    Z,AF2
120              AF32:       EQU   $
121 9297 0D                  DEC   C
122 9298 20F3                JR    NZ,AF3
123 929A DD23                INC   IX
124 929C DD23                INC   IX
125 929E 1888                JR    AF0
126              AF1:        EQU   $
127 92A0 C5                  PUSH  BC
128 92A1 D5                  PUSH  DE
129 92A2 2AFE91              LD    HL,(SY)
130 92A5 23                  INC   HL
131 92A6 22FE91              LD    (SY),HL
132 92A9 ED5BFC91            LD    DE,(SX)
133 92AD CD1DBC              CALL  DOTPOS
134 92B0 220292              LD    (POS),HL
```

```
135 92B3 D1                    POP   DE
136 92B4 C1                    POP   BC
137 92B5 2A0292                LD    HL,(POS)
138 92B8 3A0092                LD    A,(LA)
139 92BB 47                    LD    B,A
140 92BC 18D4                  JR    AF31
141              AF2:          EQU   $
142 92BE C5                    PUSH  BC
143 92BF D5                    PUSH  DE
144 92C0 2AFE91                LD    HL,(SY)
145 92C3 23                    INC   HL
146 92C4 22FE91                LD    (SY),HL
147 92C7 ED5BFC91              LD    DE,(SX)
148 92CB CD1DBC                CALL  DOTPOS)
149 92CE 220292                LD    (POS),HL
150 92D1 D1                    POP   DE
151 92D2 C1                    POP   BC
152 92D3 2A0292                LD    HL,(POS)
153 92D6 3A0092                LD    A,(LA)
154 92D9 47                    LD    B,A
155 92DA 18BB                  JR    AF32
156                            END
```

5/10.4

Graphicomanies

Ce chapitre est une galerie ouverte à tous, petits et grands pourront y proposer les plus beaux fleurons de leur logithèque personnelle. Les critères qui feront que votre œuvre sera retenue et publiée sont simples. Beau sera le dessin, court sera le programme. Ces programmes pourront être écrits en tous langages tournant sur CPC.

5/10.4.1

Jeux de points

L'idée directrice de ces quelques petits programmes est de faire correspondre à tout point de l'écran un nombre z fonction plus ou moins compliquée des coordonnées x et y du point écran et dont la valeur absolue autorisera le tracé si elle est divisible par 2 et ne l'autorisera pas si elle n'est pas multiple de 2.

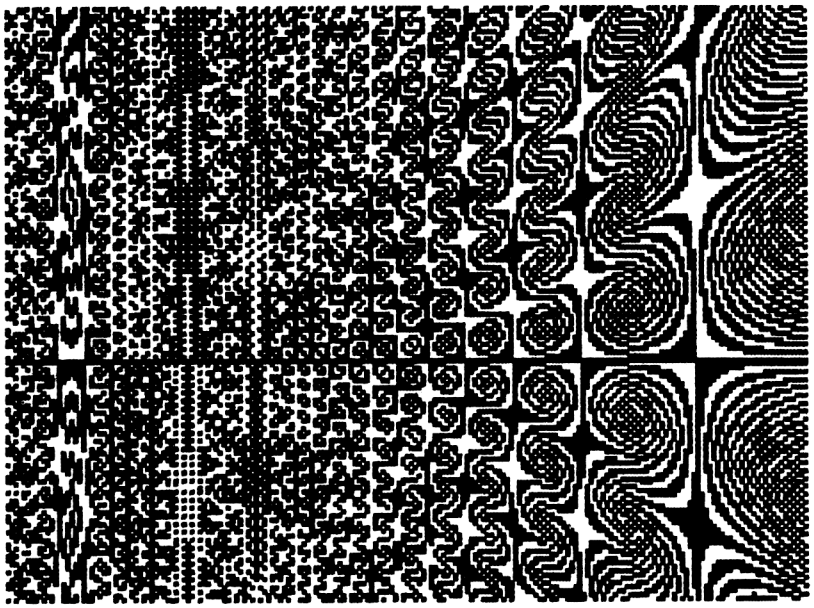
Ce critère apparaît en ligne 100 dans chacun des programmes.

Trois fonctions différentes vous sont proposées ici, et comme vous pouvez le constater, chacune de ces fonctions conduit à un graphisme différent et pour le moins original. Les valeurs de « a » et « b » sont en fait les points de départ du balayage d'une portion d'un plan fictif. Le nombre « cote » est en fait la longueur du côté d'un carré de ce plan et dont le coin inférieur gauche se situe en [a, b].

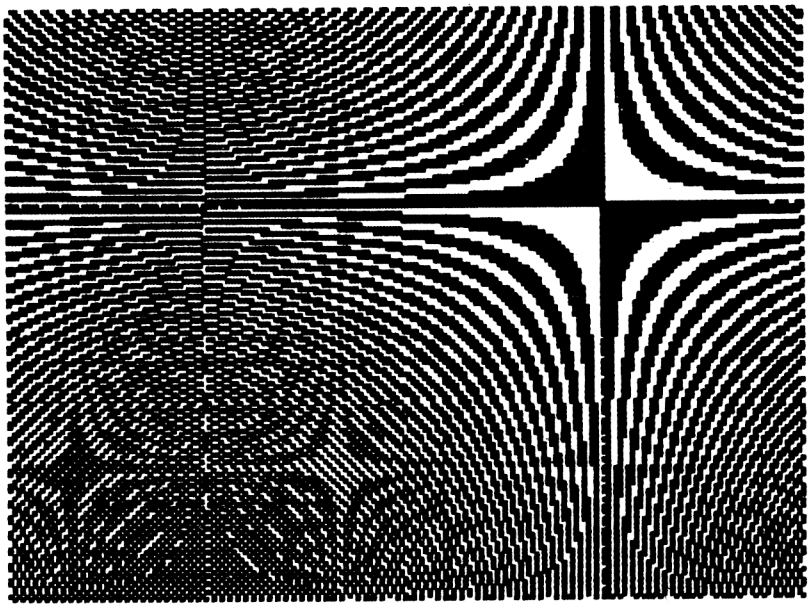
A chacun des points de ce carré, on fait correspondre un point de votre écran, celui-là bien réel. Les coordonnées du plan-écran étant ici « i » et « j ». Le balayage de l'écran se fait aux lignes 40 et 50, la transformation en 60 et 70 et le calcul de la fonction « z » en 80.

Graph 1

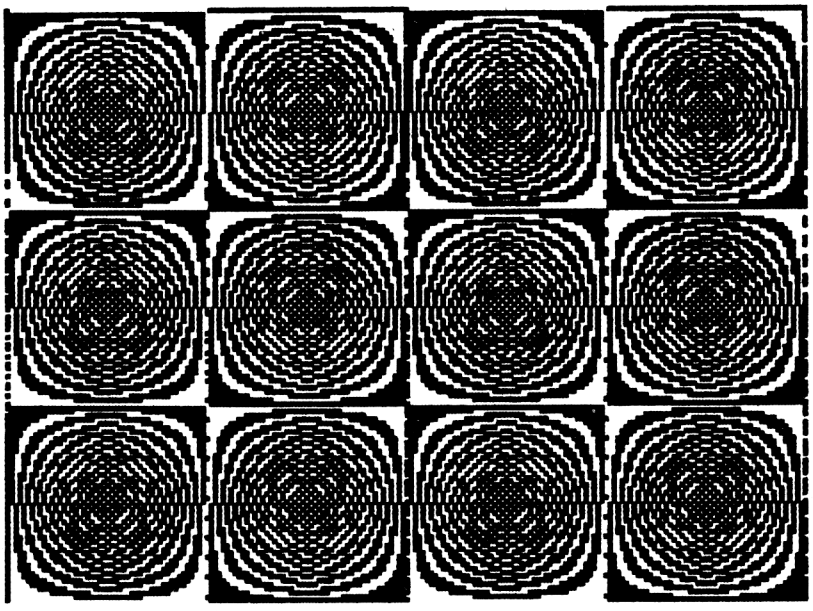
```
1 REM GRAPH1
5 MODE 2:CLS
10 INPUT"a,b";a,b
20 INPUT"cote";cote
30 CLS:LOCATE 1,1
40 FOR i=0 TO 300
50 FOR j=0 TO 200
60 x=a+cote*i/100
70 y=b+cote*j/100
80 z=(1.1)^x*y
90 c=INT(z)
100 IF c MOD 2= 0 THEN PLOT i,j
110 NEXT j,i
```



Variations sur Graph 2
diverses valeurs de a, b et côté

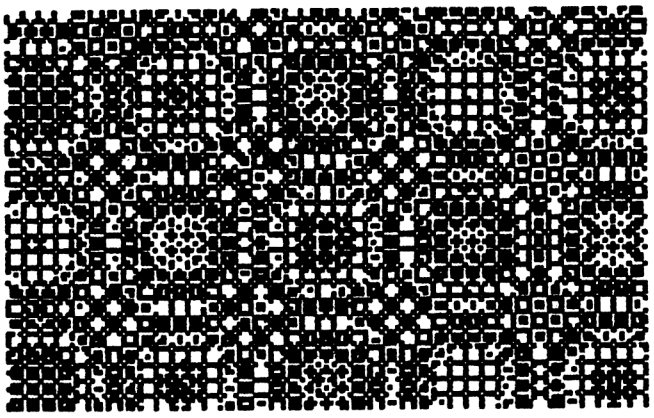


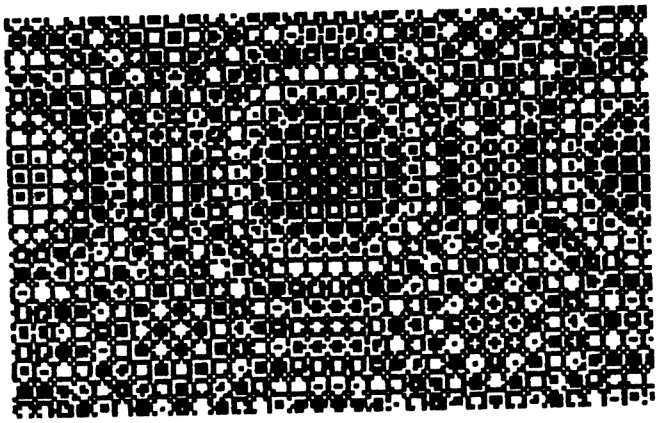
```
1 REM GRAPH2
5 MODE 2:CLS
10 INPUT"a,b";a,b
20 INPUT"cotÉ";cote
30 CLS:LOCATE 1,1
40 FOR i=0 TO 300
50 FOR j=0 TO 200
60 x=a+cote*i/100
70 y=b+cote*j/100
80 z=x*y
90 c=INT(z)
100 IF c MOD 2= 0 THEN PLOT i,j
110 NEXT j,i
```

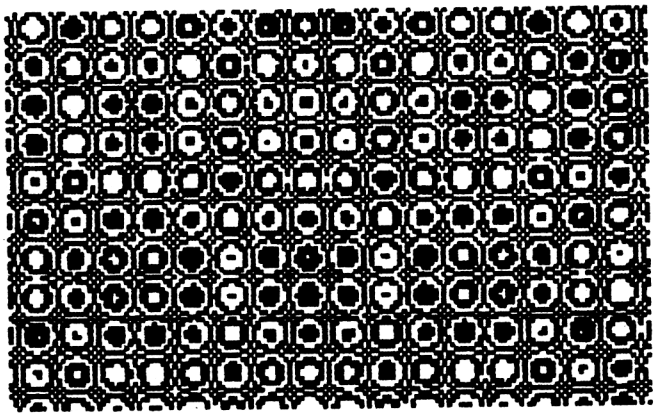


Graph 3

```
1 REM GRAPH3
5 MODE 2:CLS
10 INPUT"a,b";a,b
20 INPUT"cote";cote
30 CLS:LOCATE 1,1
40 FOR i=0 TO 300
50 FOR j=0 TO 200
60 x=a+cote*i/100
70 y=b+cote*j/100
80 z=(x^2+y^2)
90 c=INT(z)
100 IF c MOD 2= 0 THEN PLOT i,j
110 NEXT j,i
```

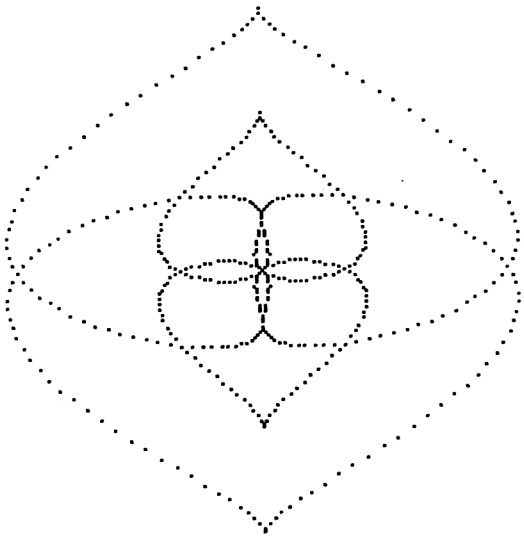




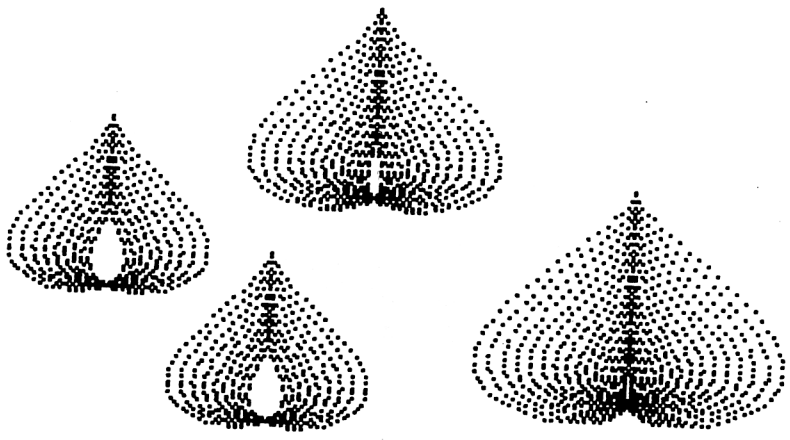


Entrelacs

```
1 REM ENTRELACS
10 CLS:MODE 2
20 H=90
30 FOR i=PI TO 5*PI+0.1 STEP PI/30
40 a=COS(i/2):b=SIN(i/2)
50 x=320+H*a*a:y=200+H*b*0.65:x1=320-H*a*a:y1=y:x2=x:y2=200-H*b*0.65
60 x3=x1:y3=y2:PLOT x,y:PLOT x1,y1:PLOT x2,y2:PLOT x3,y3:H=H-2:NEXT i
```

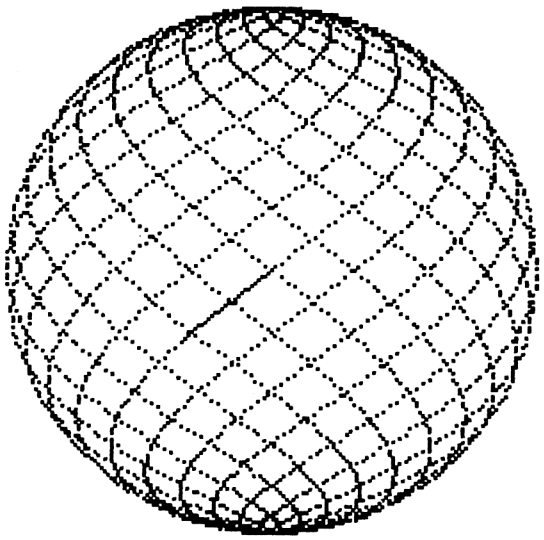


```
10 REM Bagdad
20 MODE 2:CLS
30 r=90:s=100:t=80:u=80:GOSUB 60:r=85:s=95:t=75:u=75:GOSUB 60:r=80:s=90:t=70:u=7
0:GOSUB 60:r=75:s=85:t=65:u=65:GOSUB 60
40 r=70:s=80:t=60:u=60:GOSUB 60
50 r=65:s=75:t=60:u=60:GOSUB 60:r=60:s=70:t=60:u=60:GOSUB 60:r=60:s=65:t=60:u=60
:GOSUB 60:r=60:s=60:t=60:u=60:GOSUB 60:END
60 FOR i=PI TO 3*PI  STEP PI/30:a=COS(i/2)*COS(i/2):b=SIN(i/2)*0.65
70 x=160+r*a:y=100+r*b:v=160-r*a:w=y:PLOT x,y:PLOT v,w:r=r-2:c=240+s*a:d=160+s*b
:e=240-s*a:f=d:PLOT c,d:PLOT e,f:s=s-2
80 g=80+t*a:h=130+t*b:j=80-t*a:l=h:PLOT g,h:PLOT j,l:t=t-2:m=50+g:n=40+h:o=50+j:
p=40+l:PLOT m,n:PLOT o,p:u=u-2
90 NEXT i:RETURN
```



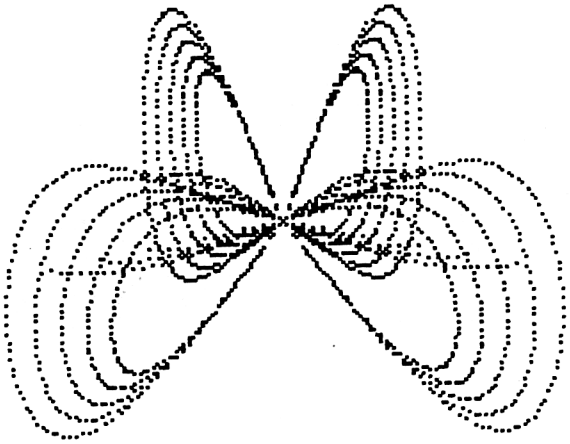
Balle

```
1 REM Balle sur une ligne !...
10 MODE 1:CLS:FOR P=0 TO 31.55 STEP.01:X
=320+160*COS(2*P)*SIN(2.6*P):Y=200+160*S
IN(2*P):PLOT X,Y,3:NEXT
```



Butterfly

```
10 REM Butterfly
20 MODE 1:CLS
30 r=90:GOSUB 40:r=85:GOSUB 40:r=80:GOSUB 40:r=75:GOSUB 40:r=70:GOSUB 40:END
40 FOR i=PI TO 2*PI-0.3 STEP 0.03
50 x=160+r*COS(I/2)*3:y=100-r*SIN(2*i)
60 x1=160-r*COS(i/2)*3:y1=y:x2=160+r*COS(i/2)*1.5:y2=100+r*SIN(2*i)
70 x3=160-r*COS(i/2)*1.5:y3=y2:PLOT x,y:PLOT x1,y1:PLOT x2,y2:PLOT x3,y3
80 r=r-1:NEXT i:RETURN
```

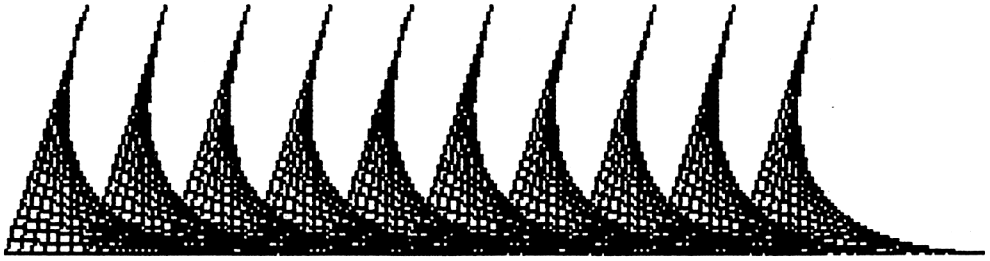


5/10.4.2

Jeux de lignes

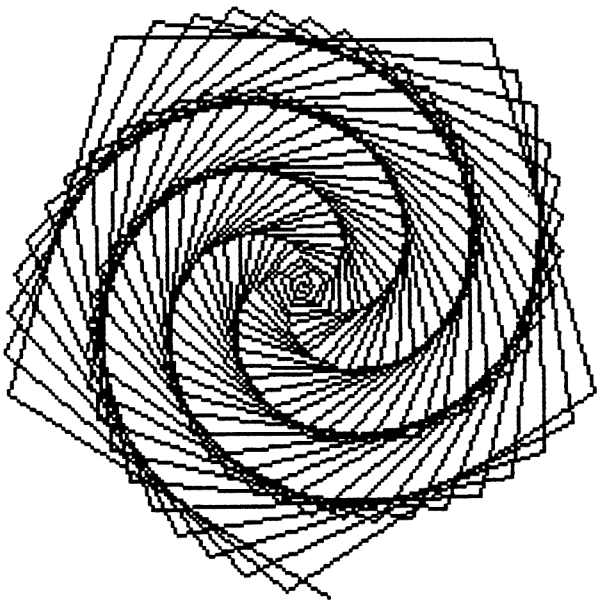
La digue de la Rochelle

```
10 REM La digue de La Rochelle
20 MODE 2:CLS
30 x0=10:y0=150:r=160
40 a=2*PI/5*i:b=2*PI/5:x=x0+r*COS(a-b):y=y0-r*SIN(a-b):PLOT x0+r,y0:DRAW x0,y0:D
RAW x,y
50 FOR j=1 TO 10:FOR i=1 TO 20
60 x1=x0+r-(r/20)*i*COS(a):y1=y0:x2=x0+(r/20)*i*COS(a-b):y2=y0-(r/20)*i*SIN(a-b)
:PLOT x1,y1:DRAW x2,y2:NEXT i
70 x0=x0+r*COS(a-b):NEXT j
```



Colimagone

```
1 REM Colimagone
10 CLS:MODE 1:PLOT 320,200:FOR X=0 TO 18
00 STEP 5:A=A+.55:C=COS(X)*A+320:L=SIN(X
)*A+200:DRAW C,L:NEXT
```



5/10.4.3

Les espaces inconnus

Notions mathématiques de base

Quelques notions sont nécessaires pour découvrir ces nouveaux êtres mathématiques errant dans des espaces immatériels.

Que les forts en maths, pour qui ces notions sont triviales, nous excusent, il faut bien que tout le monde comprenne...

Tout commence avec la question :

L'équation $x^2 = -1$ possède-t-elle une solution ?

A la réponse NON, partant du principe qu'on ne peut extraire la racine carrée d'un nombre négatif, on peut opposer la réponse OUI, pourquoi pas ?

Cette réponse affirmative, que nous vous demandons simplement d'admettre soulève le voile sur l'immensité étrange et passionnante des nombres complexes.

Admettons donc qu'il existe un nombre que nous noterons **i** et dont le carré est égal à -1.

Ce nombre d'un type nouveau appartient à l'ensemble des nombres complexes. L'ensemble des réels n'est qu'une partie infime de ce dernier.

Tout nombre complexe est repéré par son affixe. L'affixe **z** de tout nombre complexe est composée de 2 parties.
L'une appelée *partie réelle* sera notée **x** si vous le voulez bien alors que l'autre, nommée *partie imaginaire*, sera notée **y**.

Ces trois nombres sont alors reliés par la relation :

$$z = x + iy$$

Cela posé, on peut définir sur cet ensemble toutes les opérations classiques connues sur l'ensemble des réels, à savoir :

— *L'addition*

$z_1 = x_1 + iy_1$ et $z_2 = x_2 + iy_2$ entraînera

$$z_1 + z_2 = (x_1 + x_2) + i\,(y_1 + y_2)$$

La somme de deux nombres complexes est un nombre complexe dont la partie réelle est la somme des parties réelles et la partie imaginaire, la somme de deux parties imaginaires.

— De même, *la soustraction* de deux nombres complexes s'écrira

$$z_1 = x_1 + iy_1 \text{ et } z_2 = x_2 + iy_2$$

$$z_1 - z_2 = (x_1 - x_2) + i\,(y_1 - y_2)$$

— La *multiplication* de deux nombres complexes est un peu plus compliquée, en effet :

$$Z = X + iY = z_1\, z_2 = (x_1 + iy_1)\,(x_2 + iy_2) = x_1\, x_2 + ix_1\, y_2 + ix_2\, y_1 + i^2 y_1 y_2$$

souvenons-nous d'avoir décidé que $\mathbf{i^2 = -1}$ alors :

$$Z = X + iY = (x_1\, x_2 - y_1\, y_2) + i\,(x_1\, y_2 + x_2\, y_1)$$

étonnant non ?

Les notations x et y ne sont pas sans rappeler les notations des coordonnées d'un point dans un **plan** muni d'un système d'axes Ox,Oy. Ce choix n'est en effet pas innocent puisqu'il va nous permettre de représenter un nombre complexe par un point dans un plan bien réel celui-ci.

Il suffira de représenter les parties imaginaires et réelles, respectivement par des points sur les axes d'ordonnée Oy et d'abscisse Ox.

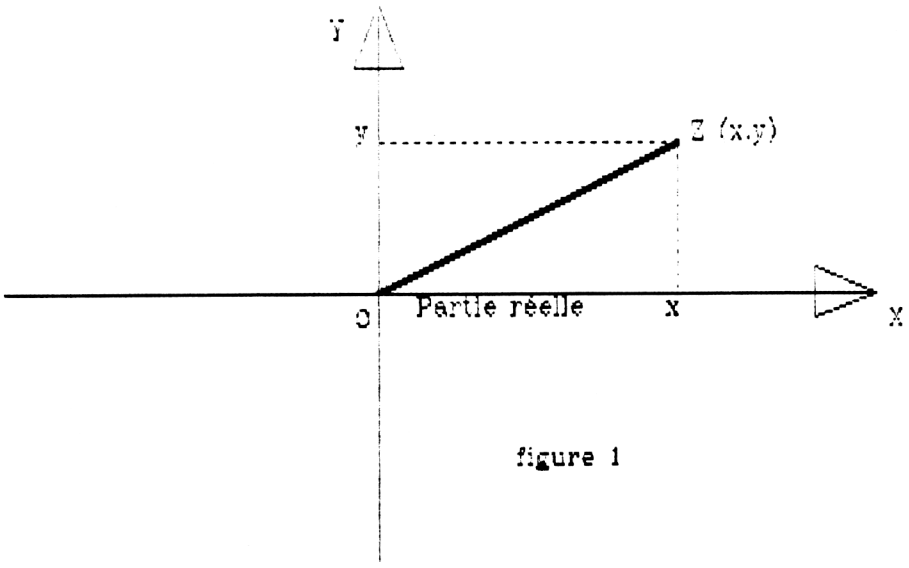


figure 1

Tout nombre complexe **z** sera alors représenté par un point **Z** de coordonnées **x** et **y**.

On peut cette fois définir le module d'un nombre complexe comme la longueur du vecteur **OZ.** Notons ce module **|z|.**

$$|z| = \sqrt{(x^2 + y^2)}$$

d'après Pythagore appliqué au triangle rectangle OZx.

Ces quelques notions sont pour l'instant suffisantes pour aborder ces espaces inconnus. N'hésitons plus et entrons alors dans le monde étonnant des nombres complexes.

Passage du plan complexe à l'écran de l'Amstrad

Le principe des deux programmes qui suivent est très simple. Considérons une portion du plan complexe délimitée par le rectangle formé par les points Z_{11}, Z_{22}, Z_{12}, et Z_{21}.

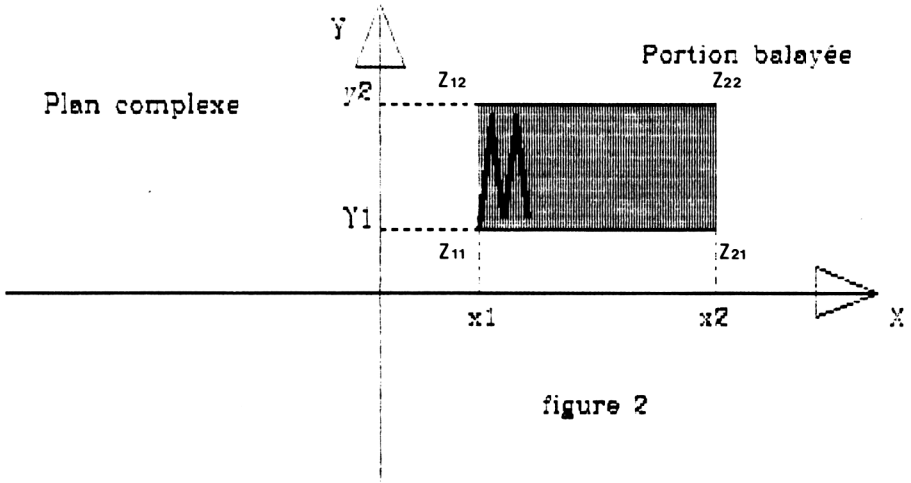


figure 2

Balayons systématiquement cette portion de bas en haut et de gauche à droite après avoir défini un accroissement **dx** sur l'axe des réels et **dy** sur l'axe imaginaire.

Faisons correspondre à chaque point ainsi balayé, un point de l'écran du CPC en écrivant que :

$$dx = (x_2 - x_1)/640 \text{ et } dy = (y_2 - y_1)/400$$

Le point courant dans le plan complexe correspondra alors au point-écran **XP, YP.**

Ces principes de balayage et de correspondance entre plan complexe et écran étant définis, nous pouvons alors faire subir à tout point du plan complexe toute sorte de torture et obtenir des graphismes assez merveilleux.

Coucher de soleil imaginaire

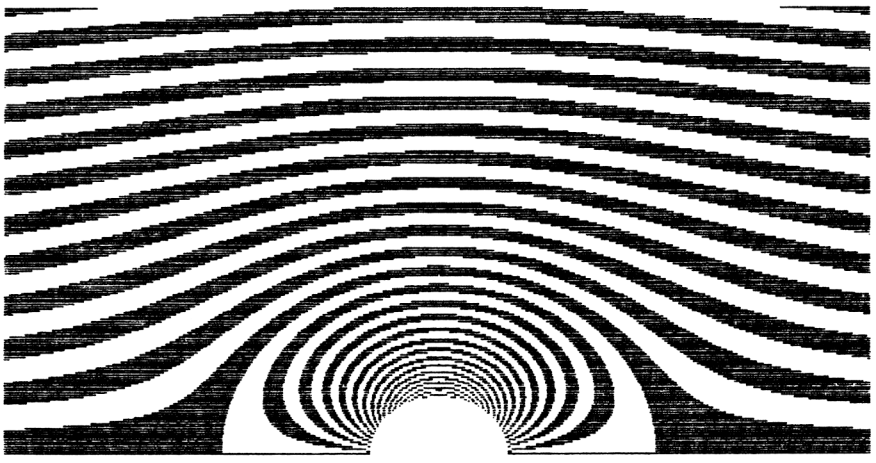
Décidons que pour tout point de la portion de plan complexe définie plus haut, nous calculions le module (ici **xx** en ligne 120) et que si ce module est inférieur à 0,1 nous passons au point suivant.

Dans le cas contraire, nous formons le nombre **f = y − y/xx**.

Si la partie entière de **f** multipliée par 10 est paire, nous traçons le point-écran correspondant, sinon nous passons au point suivant (ligne 150).

```
10 CLS
20 MODE 2
30 XP=0:YP=0
40 x1=-2:x2=2
50 y1=0:y2=(y1+(x2-x1)*400/640)
60 dx=(x2-x1)/640
70 dy=(y2-y1)/400
80 FOR x=x1 TO x2 STEP dx
90 XP=XP+1
100 FOR y=y1 TO y2 STEP dy
110 YP=YP+1
120 xx=x*x+y*y
130 IF xx<0.1 THEN 160
140 f=y-y/xx
150 IF (INT(f*10)MOD 2=0) THEN PLOT XP,YP
160 NEXT y
165 YP=0
170 NEXT x
```

Ce traitement barbare, dissociant les nombres complexes très particuliers obéissant à la règle, des autres, pauvres êtres informes ne méritant pas de figurer sur notre écran, va générer le coucher du soleil le plus bizarre que vous ayez jamais vu...



Coucher de soleil imaginaire.

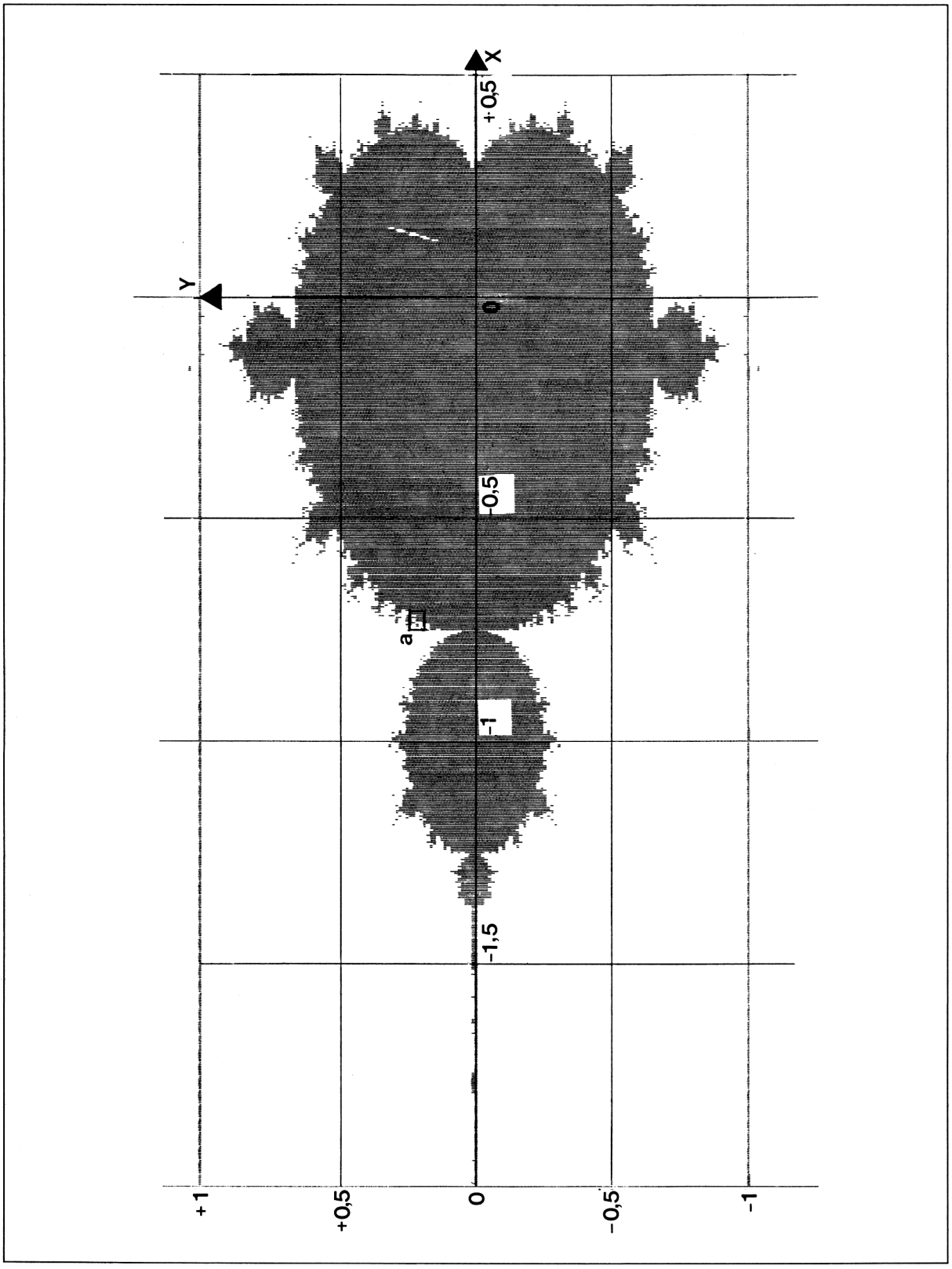
Variations sur Mandelbrot

Benoît Mandelbrot, mathématicien français qui s'est illustré au centre de recherches T. Watson d'IBM, par ses travaux sur la géométrie fractale, ouvre pour nous la porte d'une immensité curieuse et magnifique : l'ensemble de Mandelbrot. Cet ensemble est constitué des nombres complexes **c** qui, quel que soit le nombre d'itérations appliquées à l'algorithme $z = z^2 + c$ restent finis.

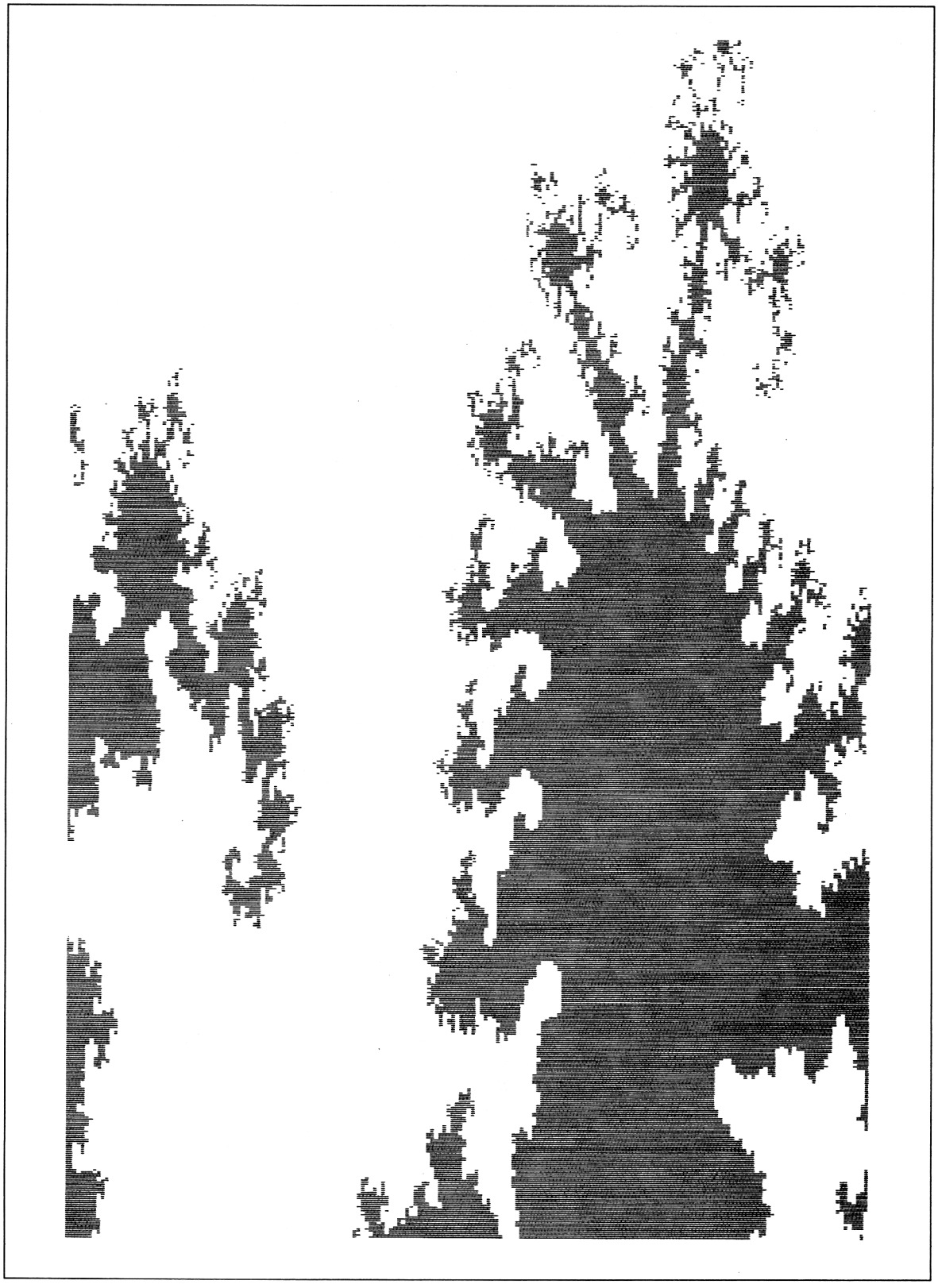
```
10 REM (c) WEKA (jpc 1988)
15 INK 1,13:INK 0,0:BORDER 0
20 CLS
30 MODE 2
40 XP=0:YP=0
50 INPUT"x1,x2";x1,x2
60 INPUT"y1,y2";y1,y2
65 INPUT"Nombre d'iterations N";N
66 CLS
70 dx=(x2-x1)/640
80 dy=(y2-y1)/400
90 FOR x=x1 TO x2 STEP dx
100 XP=XP+1
110 FOR y=y1 TO y2 STEP dy
120 x0=0:y0=0
130 YP=YP+1
140 FOR i=1 TO N
150 xx=x0*x0-y0*y0+x
160 y0=2*x0*y0+y:x0=xx
170 yy=(x0*x0+y0*y0)
180 IF yy>4 THEN 210
190 NEXT i
200 PLOT XP,YP
210 NEXT y
220 YP=0
230 NEXT x
```

Choisissons parmi ceux-ci, ceux dont le module reste inférieur à 2. Et représentons-les sur notre écran en suivant le principe défini plus haut.

Ces nombres sont alors représentés sur la figure suivante. Cet ensemble a été dessiné avec seulement 50 itérations : plus ce nombre sera grand, plus les détails seront finis (dans la limite de définition de votre écran) mais aussi plus le temps de calcul sera long, passant de quelques heures à quelques... jours.



L'ensemble de Mandelbrot dans le plan complexe (50 itérations).



Détail de l'ensemble de Mandelbrot dans le plan complexe $x_1 = -0,745 - x_2 = -0,72 - y_1 = 0,216 - y_2 = 0,25$.

Chaque zone est en principe intéressante à explorer mais la frontière de l'ensemble est plus riche lorsque le nombre d'itérations est faible.

La figure de détail en page 9 correspond à un agrandissement de la zone notée **[a]** sur la figure précédente. On pourrait agrandir encore cette zone et y découvrir de nouvelles formes. Il n'y a aucune limite à cela pas même celle de la dimension atomique. Seul votre Amstrad imposera ses limites de précision et de temps de calcul à votre vertigineuse soif d'entrer dans l'infiniment petit.

Les algorithmes de Barry Martin

D'autres mathématiciens comme Barry Martin se sont inspirés de la démarche de B. Mandelbrot. A savoir, faire des itérations successives sur une même fonction. Mais cette fois, cette fonction s'appliquait sur des nombres réels et sur une seule valeur initiale. On ne balaye pas un plan comme précédemment, mais on part d'un point unique pour en calculer une série d'autres.

Dans les deux programmes qui suivent, ce principe est appliqué sur deux fonctions différentes.

Le premier programme « Hopalong » utilise les fonctions suivantes :

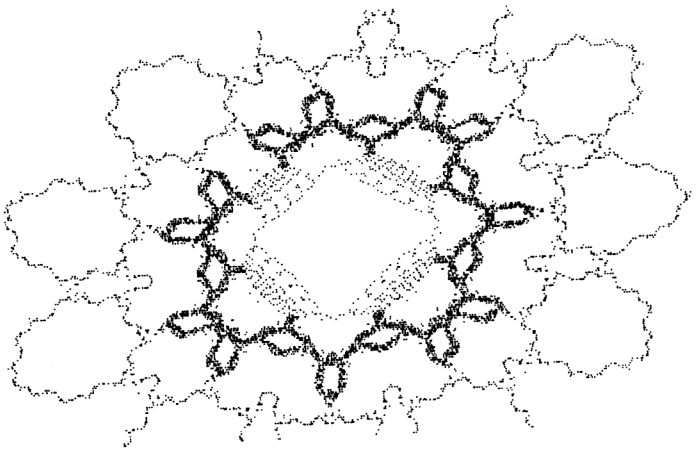
$$x = y - SGN(x) * \sqrt{|(b*x - c)|}$$
$$y = a - x$$

La fonction SGN(x) (signe de x) prend la valeur -1 si **x** est négatif et 1 si **x** est positif. L'étonnant pour ce type de programme est que le graphisme obtenu varie énormément suivant les valeurs des paramètres **a, b** et **c**.

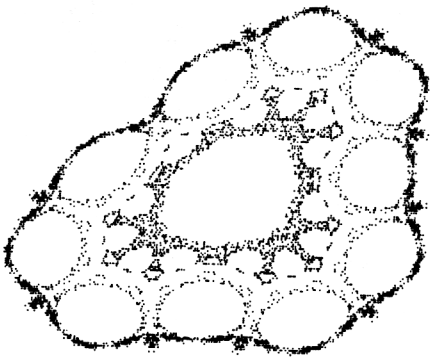
Le nombre d'itérations influe sur le niveau de détail du graphisme (voir figures page 9). Là encore, vous devrez attendre quelques heures avant de voir votre écran se couvrir de ces merveilles bizarres.

```
10 REM Algorithme   HOPALONG
20 MODE 2:CLS
30 INPUT"a,b,c";a,b,c
31 INPUT"N",N
32 INPUT"Echelle",E
35 INPUT"coordonnees du centre x,y";cx,cy
50 LOCATE 1,1:x=0:y=0
60 FOR i=1 TO N
70 PLOT x*E+cx,y*E+cy
80 xx=y-SGN(x)*(ABS(b*x-c))^0.5
90 yy=a-x
100 x=xx:y=yy
110 NEXT i
```

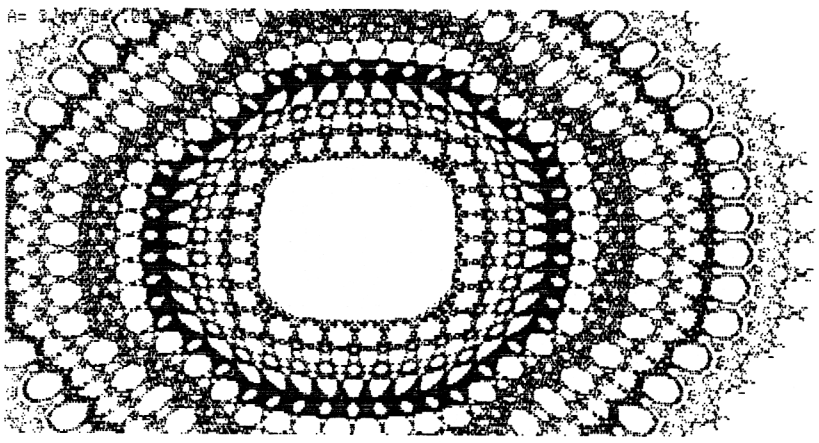
Diversités d'Hopalong :



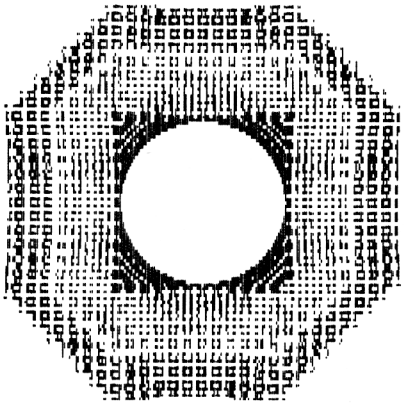
A = 3,14 - B = 0,3 - C = 0,3 - N = 10 000



A = 3,14 - B = – 1 - C = 0,03 - N = 10 000

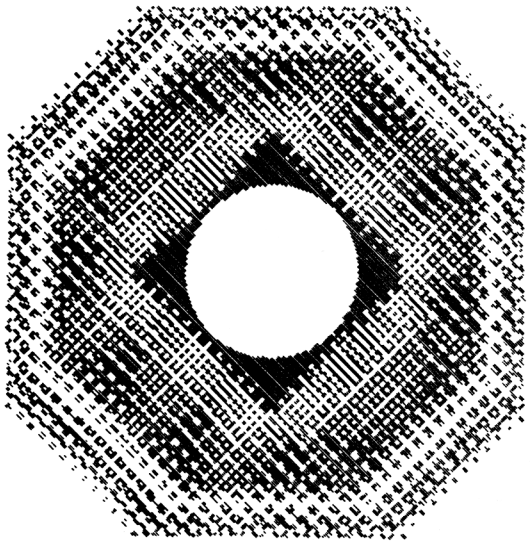


A = 3,14 - B = 0,08 - C = 0,03 - N = 60 000

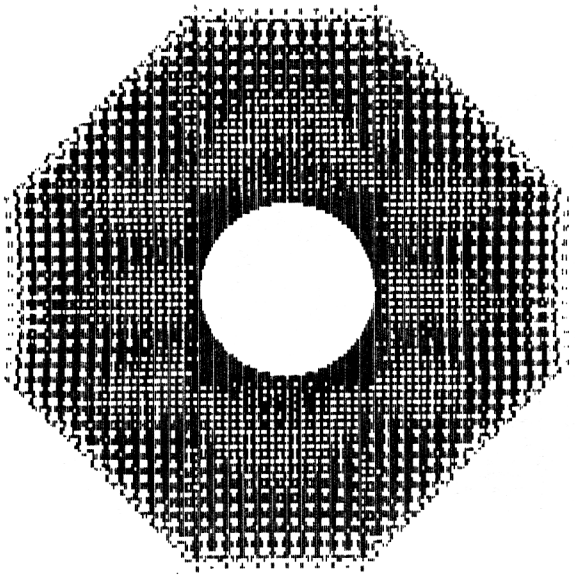


Evolution du graphisme en fonction du nombre d'itérations N

a = – 200 - b = 0,1 - c = – 80 - N = 120 135



Hopalong :
a = – 200 - b = 0,1 - c = – 80 - N = 423 424



Hopalong :
a = – 200 - b = 0,1 - c = – 80 - N = 1 359 620

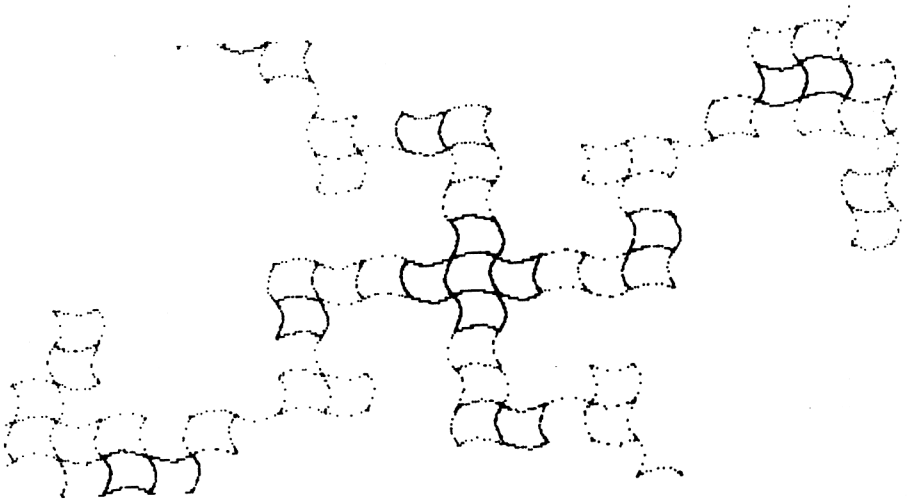
Le dernier programme proposé est composé comme le précédent ; mais utilise les fonctions :

$$x = y - SIN\ (x)$$
$$y = a - x$$

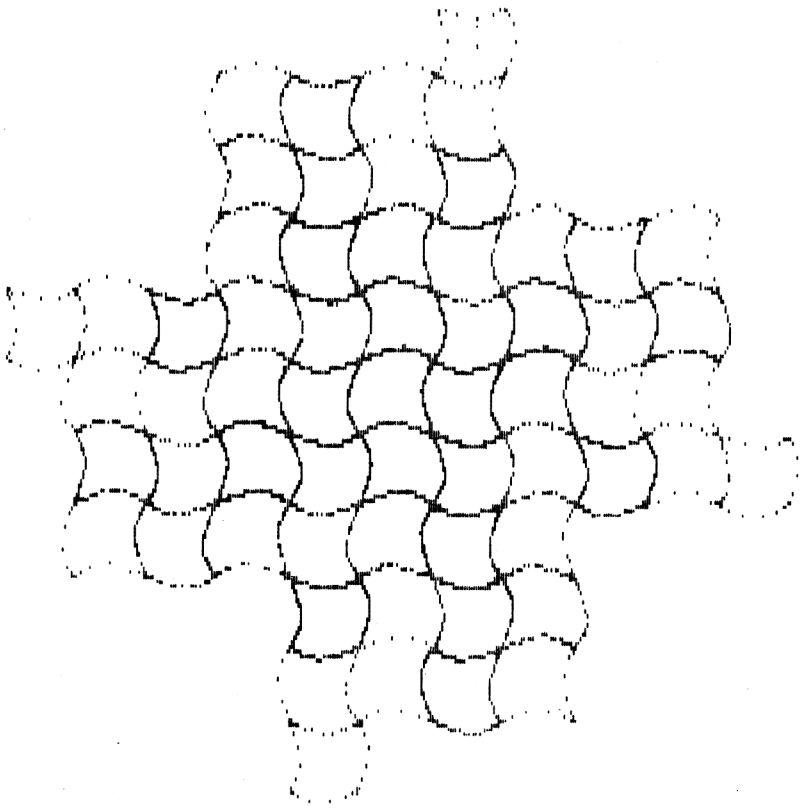
Un seul paramètre entre en jeu ici **a**. Les meilleurs résultats seront obtenus pour **a** très voisin de π.

Vous pourrez choisir sur ce dernier programme, l'échelle de votre dessin ainsi que le point de celui-ci à afficher au centre de votre écran.

```
10 REM Algorithme de Barry MARTIN
20 MODE 2:CLS
30 INPUT"a";a
40 INPUT"N",N
50 INPUT"Echelle",E
60 INPUT"coordonnees du centre x,y";cx,cy
70 CLS:PRINT"Algorithme de Barry MARTIN"
80 LOCATE 1,1:x=0:y=0
90 FOR i=1 TO N
100 PLOT x*E+cx,y*E+cy
110 xx=y-SIN(x)
120 yy=a-x
130 x=xx:y=yy
140 NEXT i
```



Algorithme de Barry Martin : a = 3,12 N = 100 000



Algorithme de Barry Martin : a = 3,09 N = 300 000

5/10.4.4

Graphisme sur variations sinusoïdales

Comme vous avez pu le remarquer dans cette partie, les graphismes harmonieux sont difficilement exploitables sans l'utilisation des mathématiques et de leurs équations.

Nous allons dans ce chapitre vous initier aux courbes qu'il est possible de tracer sous forme paramétrique à l'aide des fonctions sinus et cosinus.

La variété d'équations sous formes sinusoïdales étant à la mesure de l'imagination humaine, nous nous arrêterons dans les différents paragraphes à certaines équations, et nous nous attacherons plutôt à vous donner l'envie d'aller plus loin, en vous apportant la réflexion préliminaire à tout tracé. Le lecteur profane relira avec profit les chapitres 13/2.3 et 13/2.4.

Equations les plus couramment utilisées :

$\cos^2(x) + \sin^2(x) = 1$
$\cos(-x) = \cos(x)$
$\sin(-x) = -\sin(x)$
$\cos(x + y) = \cos(x) * \cos(y) - \sin(x) * \sin(y)$
$\sin(x+ y) = \sin(x) * \cos(y) + \cos(x) * \sin(y)$

Variations sur sinus et cosinus

Nous allons nous intéresser à une formule générale d'équation paramétrique, qui à un premier abord semble ardue, mais, nous le verrons sera d'une étude relativement simple, et permettra le tracé de toute une variété de courbes.

Un algorithme général

Voyons d'abord l'algorithme général que nous donnerons à nos programmes pour tracer nos courbes paramétriques.

Il nous faudra d'abord entrer les 8 valeurs constantes, puis effectuer les calculs correspondants à différentes valeurs du paramètre t.

Après chaque calcul, on procèdera à l'affichage du point correspondant.

- Début
    - Entrer les valeurs des constantes
    - POUR t DE valeurinitiale A valeurfinale
        - Calculer l'abscisse x
        - Calculer l'ordonnée y
        - PROCEDURE AFFICHAGE
    - FINPOUR
- FIN

Vous pourrez remarquer que cet algorithme est relativement simple, sauf pour la procédure d'affichage qui va demander une certaine réflexion sur les valeurs que pourront atteindre les coordonnées x et y.

La procédure d'affichage

Afin d'afficher nos courbes, nous allons utiliser l'écran en MODE 2, c'est-à-dire dans sa résolution la plus grande, pour afficher le maximum de pixels.

Dans cette résolution, le nombre de points affichables et égale à 640 x 400.

En étudiant un peu plus notre équation paramétrique générale, les fonctions SIN et COS nous générerons des valeurs comprises entre - 1 et +1, d'où certaines valeurs de x et y comprises entre un minima négatif et un maxima positif.

Il est donc logique de prendre pour origine un point central de l'écran, par exemple le point de coordonnées : x = 320 et y = 200.

Les points que nous afficherons à partir de cette origine ne devrons ensuite pas dépasser les limites de l'écran, ni être trop près du centre si l'on veut obtenir des courbes intéressantes à regarder.

Nous utiliserons ainsi un facteur multiplicatif d'agrandissement tel que l'écran quasi complet soit utilisé. Ce facteur multiplicatif sera calculé en fonction des valeurs minimales et maximales de x et y, et dépendra ainsi des 4 paramètres a1, b1, c1 et d1.

En x, le minimum sera atteint quand x sera égal à - a1 - b1, le maximum pour a1 + b1.

De même en y, nous aurons pour minimum - c1 - d1, et pour maximum c1 + d1.

Le rapport de proportionnalité pourra être ainsi :

100 / (c1 + d1) en x
et **300 / (c1 + d1)** en y.

Toute valeur inférieure pourra convenir, et nous vous conseillerons en fait de prendre la plus petite de ces valeurs en tant que rapport commun, afin de garder une courbe représentée dans un repère orthonormé, pour éviter de l'étirer selon l'une des coordonnées.

Le programme

Certains d'entre vous se sont peut-être déjà lancé sur leur Amstrad pour appliquer l'algorithme précédemment décrit. Qu'ils ne se précipitent pas, nous allons ici leur fournir un programme plus complet.

Ce programme proposera par menu de tracer une courbe, ou éventuellement de revoir une courbe précédemment sauvegardée sur disquette.

En effet, lors du tracé d'une courbe, il vous sera possible de la sauvegarder. Nous avons même ajouté un choix d'impression en « hard-copy » sur imprimante compatible Epson (la série DMP 2000 en fait partie).

En résumé, voici les différentes possibilités du programme :

• *Choix 1 du menu :* tracé de courbes par entrée des huit paramètres.

Pendant ce tracé, vous aurez trois possibilités :

– un appui sur la touche **I** pour lancer l'impression sur l'imprimante ;

– un appui sur la touche **S** pour sauvegarder le tracé sur disquette. Il faudra fournir un nom de huit caractères maximum. Le programme se chargera de créer deux fichiers en ajoutant l'extension **.PAR** pour le fichier paramètre, et l'extension **.PIC** pour l'écran graphique comportant uniquement la courbe. Il vous sera, par exemple, possible d'utiliser cette courbe dans une application personnelle ultérieure ;

– un appui sur la touche **F** met fin au tracé et renvoit au menu.

• *Choix 2 du menu :* lecture d'une courbe préalablement sauvegardée sur disquette. Les paramètres sont de plus affichés, pour éventuellement reprendre une étude sur une série de courbes aux paramètres proches.

A partir de ce cahier des charges, nous pouvons vous présenter le programme ci-après listé :

```
10 REM ************************************
20 REM *                                  *
30 REM * TRACE DE COURBES D'EQUATIONS     *
40 REM * PARAMETRIQUES SOUS LA FORME      *
50 REM * X= A1 COS(A2 t)+B1 SIN(B2 t)     *
60 REM * Y= C1 COS(C2 t)+D1 SIN(D2 t)     *
70 REM *                                  *
80 REM ************************************
90 REM
100 MODE 2
110 PRINT "Nous vous proposons de tracer
 les courbes d'equations parametriques"
120 PRINT "ayant les formes suivantes:"
130 PRINT
140 PRINT
150 PRINT "a1 * COS(a2 * t) + b1 * SIN (
b2 * t)
160 PRINT "c1 * COS(c2 * t) + d1 * SIN (
d2 * t)"
170 GOSUB 1580
180 LOCATE 1,10
190 FOR i = 0 TO 8
200 LOCATE 1,10+i
210    PRINT STRING$(80,CHR$(32))
220 NEXT i
230 LOCATE 1,10
240 PRINT "Veuillez entrer les different
s parametres ci-dessous:"
250 PRINT
260 INPUT "terme a1 ";a1
270 INPUT "terme a2 ";a2
280 INPUT "terme b1 ";b1
290 INPUT "terme b2 ";b2
300 INPUT "terme c1 ";c1
310 INPUT "terme c2 ";c2
320 INPUT "terme d1 ";d1
330 INPUT "terme d2 ";d2
340 REM
350 REM RAPPEL DE L'EQUATION PROPOSEE
360 REM
370 MODE 2
380 GOSUB 1400
390 REM
400 REM CALCUL DES RAPPORT DE
410 REM PROPORTIONALITE POUR
420 REM  L'OCCUPATION ECRAN
430 REM
440 rapportx=ABS(a1)+ABS(b1)
450 rapporty=ABS(c1)+ABS(d1)
460 rapport = MAX(rapportx,rapporty)
```

```
470 REM
480 REM ORIGINE DE L'ECRAN GRAPHIQUE
490 REM
500 ORIGIN 320,220
510 REM
520 REM AFFICHAGE PREMIER POINT
530 REM
540 t=0
550 x = a1 * COS(a2 * t) + b1 * SIN (b2
* t)
560 y = c1 * COS(c2 * t) + d1 * SIN (d2
* t)
570 coordonneex = x * 170 / rapport
580 coordonneey = y * 170 / rapport
590 PLOT coordonneex,coordonneey
600 REM
610 REM CALCUL COMPLET DE LA COURBE
620 REM PAR BOUCLE SUR t
630 REM
640 WHILE 1
650     x = a1 * COS(a2 * t) + b1 * SIN (
b2 * t)
660     y = c1 * COS(c2 * t) + d1 * SIN (
d2 * t)
670     DRAW x/rapport*170,y/rapport*170
680        REM incrementation de t par
690        REM pas fixe mais modifiable
700     t=t+0.01
710        REM test touche
720     a$=INKEY$
730     IF a$ <> INKEY$ THEN GOSUB 790
740        REM
750 WEND
760 REM
770 REM VERIFICATION TOUCHE FRAPPEE
780 REM
790 a = 0
800 IF a$ = "I" OR a$ = "i" THEN a = 1
810 IF a$ = "S" OR a$ = "s" THEN a = 2
820 IF a$ = "F" OR a$ = "f" THEN RUN
830 ON a GOSUB 880,1190
840 RETURN
850 REM
860 REM COPY GRAPHIQUE SUR IMPRIMANTE
870 REM
880 DATA cd,ba,bb,cd,e7,bb,32,bf,a0,cd
890 DATA 6b,a0,21,8f,01,22,c0,a0,11,00
900 DATA 00,3e,06,32,c2,a0,cd,7b,a0,0e
910 DATA 00,3a,c2,a0,47,e5,d5,c5,cd,f0
920 DATA bb.c1,d1,21,bf,a0,be,e1,37,20
```

```
930 DATA 01,a7,cb,11,2b,10,ea,cd,b3,a0
940 DATA 79,cd,aa,a0,13,e5,21,7f,02,37
950 DATA ed,52,e1,38,05,2a,c0,a0,18,cd
960 DATA 23,7c,b5,c8,2b,11,00,00,22,c0
970 DATA a0,3e,03,bd,20,ba,7c,b4,20,b6
980 DATA 3e,04,32,c2,a0,18,af,3e,1b,cd
990 DATA aa,a0,3e,33,cd,aa,a0,3e,10,cd
1000 DATA aa,a0,c9,e5,3e,42,cd,1e,bb,e1
1010 DATA 28,02,e1,c9,3e,0d,cd,aa,a0,3e
1020 DATA 0a,cd,aa,a0,3e,1b,cd,aa,a0,3e
1030 DATA 2a,cd,aa,a0,3e,04,cd,aa,a0,3e
1040 DATA 7f,cd,aa,a0,3e,02,cd,aa,a0,c9
1050 DATA cd,2e,bd,38,fb,cd,2b,bd,c9,3a
1060 DATA c2,a0,fe,06,c8,af,cb,11,cb,11
1070 DATA c9,00,00,00,00
1080 :
1090 MEMORY &9FFF:total=0
1100 FOR i=&A000 TO &A0C2
1110 READ a$:a=VAL("&"+a$):POKE i,a
1120 total=total+a
1130 NEXT
1140 IF total<>24125 THEN PRINT"erreur":
STOP
1150 CALL &A000
1155 PLOT coordonneex,coordonneey
1160 RETURN
1170 REM
1180 REM SAUVEGARDE DE L'IMAGE
1190 LOCATE 1,24
1200 PRINT STRING$(70,CHR$(32));
1210 LOCATE 1,25
1220 PRINT STRING$(70,CHR$(32));
1230 REM
1240 LOCATE 1,24
1250 INPUT "Nom de la courbe (8 lettres
sans extension) ";nom$
1260 IF LEN(nom$) > 8 THEN GOTO 1190
1270 courbe$ = nom$ + ".PIC"
1280 parametre$ = nom$+ ".PAR"
1290 OPENOUT parametre$
1300 WRITE #9,a1,a2,b1,b2,c1,c2,d1,d2
1310 CLOSEOUT
1320 LOCATE 1,24
1330 PRINT STRING$(70,CHR$(32));
1340 SAVE courbe$,b,&C000,&4000
1350 '
1360 RETURN
1370 REM
1380 REM AFFICHAGE CARACTERISTIQUES
```

```
1390 REM
1400 LOCATE 1,24
1410 PRINT "x=" ; USING "#####.###" ; a1
;
1420 PRINT "*COS(" ; USING "#####.###" ;
a2 ;
1430 PRINT "*t)" ; USING "+#####.###" ; b
1 ;
1440 PRINT "*SIN(" ; USING "#####.###" ;
b2 ;
1450 PRINT "*t)"
1460 LOCATE 1,25
1470 PRINT "y=" ; USING "#####.###" ; c1
;
1480 PRINT "*COS(" ; USING "#####.###" ;
c2 ;
1490 PRINT "*t)" ; USING "+#####.###" ; d
1 ;
1500 PRINT "*SIN(" ; USING "#####.###" ;
d2 ;
1510 PRINT "*t)";
1520
1530 RETURN
1540 REM
1550 REM
1560 REM AFFICHAGE DU MENU
1570 REM
1580 LOCATE 1,10
1590 PRINT " 1 -> Trace d'une courbe"
1600 PRINT:PRINT
1610 PRINT " 2 -> Chargement d'une courb
e"
1620 REM
1630 WHILE a$ <> "1" AND a$ <> "2"
1640     a$ = INKEY$
1650 WEND
1660 REM
1670 ON VAL(a$) GOSUB 180,1720
1680 RUN
1690 REM
1700 REM CHARGEMENT D'UNE COURBE
1710 REM
1720 LOCATE 1,24
1730 INPUT "Nom de la courbe (8 lettres
sans extension) ";nom$
1740 courbe$ = nom$ + ".PIC"
1750 parametre$ = nom$ + ".PAR"
```

```
1760 OPENIN parametre$
1770 INPUT #9,a1,a2,b1,b2,c1,c2,d1,d2
1780 CLOSEIN
1790 MODE 2
1800 LOAD courbe$
1810 GOSUB 1400
1820 CALL &BB06
1830 RETURN
```

Lignes 10 à 160 : présentation du programme.

Ligne 170 : envoi au sous-programme du menu.

Lignes 180 à 220 : effacement du menu uniquement.

Lignes 230 à 330 : acquisition des paramètres.

Ligne 380 : envoi au sous-programme d'affichage des deux équations paramétriques.

Lignes 440 à 460 : sélection du rapport d'affichage.

Ligne 500 : on fixe l'origine du tracé du centre de la partie restante de l'écran.

Lignes 540 à 590 : un premier point est affiché. C'est le point de départ (pour t = 0), qui sert d'origine pour le tracé à l'aide de l'instruction **DRAW** qui suivra.

Ligne 640 : ue boucle infinie est ici réalisée car la condition 1 (de WHILE 1) est toujours réalisée.

Lignes 650 à 670 : tracé d'une portion de droite positionnant le point suivant.

Ligne 700 : incrémentation du compteur de temps t.

Lignes 720 et 730 : test d'appui sur une touche pour sortir de la boucle.

Lignes 790 à 840 : traitement de la touche frappée, pour orientation vers les sous-programmes correspondants.

Lignes 880 à 1150 : les données hexadécimales pour la copie graphique sont placées en mémoire, puis la copie est lancée par **CALL &A000**.

Ligne 1155 : on positionne à nouveau le point de tracé où il s'était arrêté, car la copie graphique le déplace.

Lignes 1190 à 1360 : l'image est sauvegardée à partir d'un nom donné, en deux fichiers précédemment expliqués.

Lignes 1400 à 1530 : affichage dans le bas de l'écran des deux équations paramétriques.

Lignes 1580 à 1680 : affichage et gestion des choix du menu.

Lignes 1720 à 1830 : acquisition du nom d'une courbe déjà sauvegardée et affichage de celle-ci.

Le retour du cercle

Commençons par donner la valeur 0 à b1 et d1, et une valeur R identique à a1 et c1. Prenons aussi 1 pour valeur des coefficients a2, b2, c2 et d2.

Nous obtenons ainsi le système d'équations suivant :

x = R * COS(t)
y = R * SIN(t)

En appliquant l'algorithme précédent, il vous sera facile de dessiner un cercle de rayon R.

Cette équation est donc l'équation paramétrique du cercle tant étudiée en mathématiques.

Le cercle étiré

Reprenons les équations précédentes en modifiant le paramètre R dans la deuxième, afin qu'il soit différent de celui de la première équation.

x = R1 * COS(t)
y = R2 * COS(t)

Ce nouveau système vous permettra de tracer une ellipse.

Electronique et courbes remarquables

Les électroniciens sont peut-être ceux qui utilisent le plus les fonctions trigonométriques et surtout la fonction SINUS.

En effet, toutes les tensions sinusoïdales ont des équations de la forme :

x = Xmax * SIN(w * t + PHI)

avec : Xmax la valeur maximale de la tension ;
w la pulsation, qui peut encore s'écrire :

w = 2 * PI * fréquence
ou w = 2 * PI / période

et qui s'exprime en radian par seconde (rad/s) ;

et PHI le déphasage, que l'on peut considérer comme étant le retard ou l'avance du signal, et qui s'exprime en radian.

Ainsi, on peut considérer deux signaux sinusoïdaux, que l'on désire représenter l'un par rapport à l'autre ; c'est ce que l'on effectue sur un oscilloscope en utilisant un balayage appelé X/Y. On obtient ce que l'on nomme dans le jargon électronicien les courbes de Lissajous.

x = Xmax * SIN(w1 * t + PHI1)
y = Ymax * SIN(w2 * t + PHI2)

Pour intégrer ces équations aux données de notre programme, il nous faut les transformer à l'aide des formules citées plus haut :

Xmax * SIN(w1 * t + PHI1)
= Xmax * SIN(w1 * t) * COS(PHI1)
+ Xmax * COS(w1 * t) * SIN(PHI1)
= Xmax * SIN(PHI1) * COS(w1 * t)
+ Xmax * COS(PHI1) * SIN(w1 * t)

De même pour y, on obtient :

Ymax * SIN(w2 * t + PHI2)
= Ymax * SIN(PHI2) * COS(w2 * t)
+ Ymax * COS(PHI2) * SIN(w2 * t)

Nous obtenons ainsi nos termes :

a1 = Xmax * SIN(PHI1) a2 = w1
b1 = Xmax * COS(PHI1) b2 = w1
c1 = Ymax * SIN(PHI2) c2 = w2
d1 = Ymax * COS(PHI2) d2 = w2

Si nous prenons par exemple deux signaux, de valeur maximale identique 1, de pulsation identique 1, mais dont l'un est déphasé de $\pi/6$, nous obtenons les équations « électroniques » suivantes :

x = SIN (t)
y = SIN(t + PI/6)

Ce qui nous donne le système d'équations :

x = SIN(t)
y = 0.5 COS(t) + .866 SIN(t)

Les valeurs à donner au programme sont ainsi :

a1 = 0 a2 = 0
b1 = 1 b2 = 1
c1 = 0.5 c2 = 1
d1 = 0.866 d2 = 1

Vous obtenez ainsi la courbe représentée en figure 1.

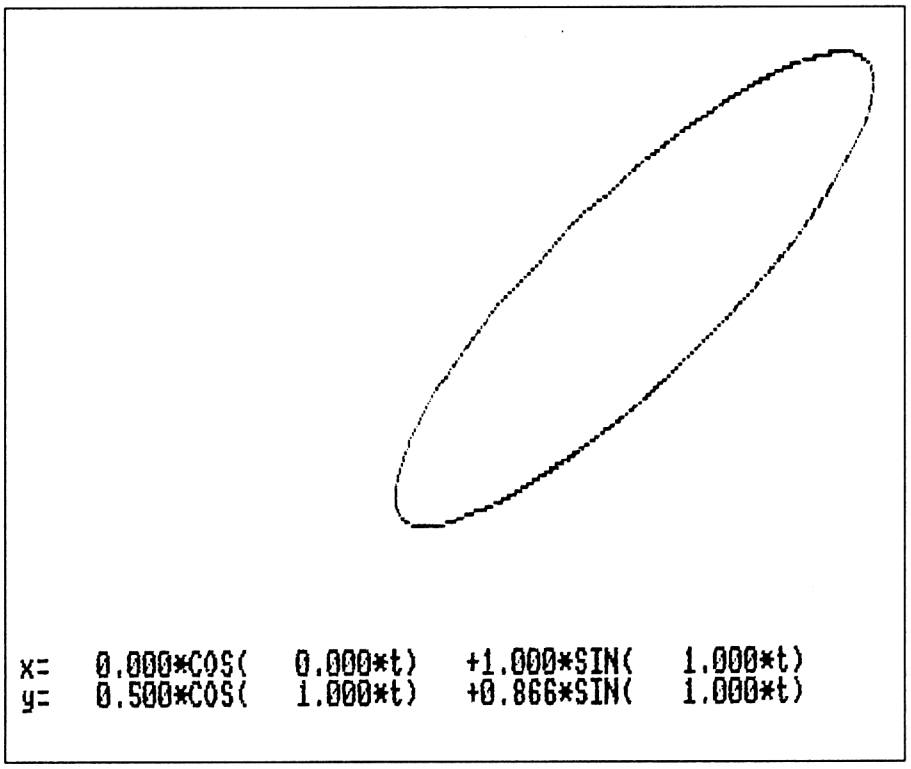


Fig. 1 : Lissajous sur signaux déphasés,

$$x = \text{SIN } t \; ; \; y = \text{SIN } t + \frac{\pi}{3}$$

Il est aussi possible, non plus de modifier le déphasage, mais la valeur des pulsations des deux signaux (ce qui revient à modifier les fréquences).

Par exemple, en « pulsant » l'un des signaux deux fois plus rapidement que l'autre, on obtient les équations suivantes :

x = SIN(t) → Xmax = 1 ; w1 = 1 ; PHI1 = 0
y = SIN(2 * t) → Ymax = 1 ; w2 = 2 ; PHI2 = 0

Les paramètres de nos équations sont simples :

a1 = a2 = c1 = c2 = 0
b1 = b2 = d1 = 1
d2 = 2

et l'on obtient le nœud papillon de la figure 2.

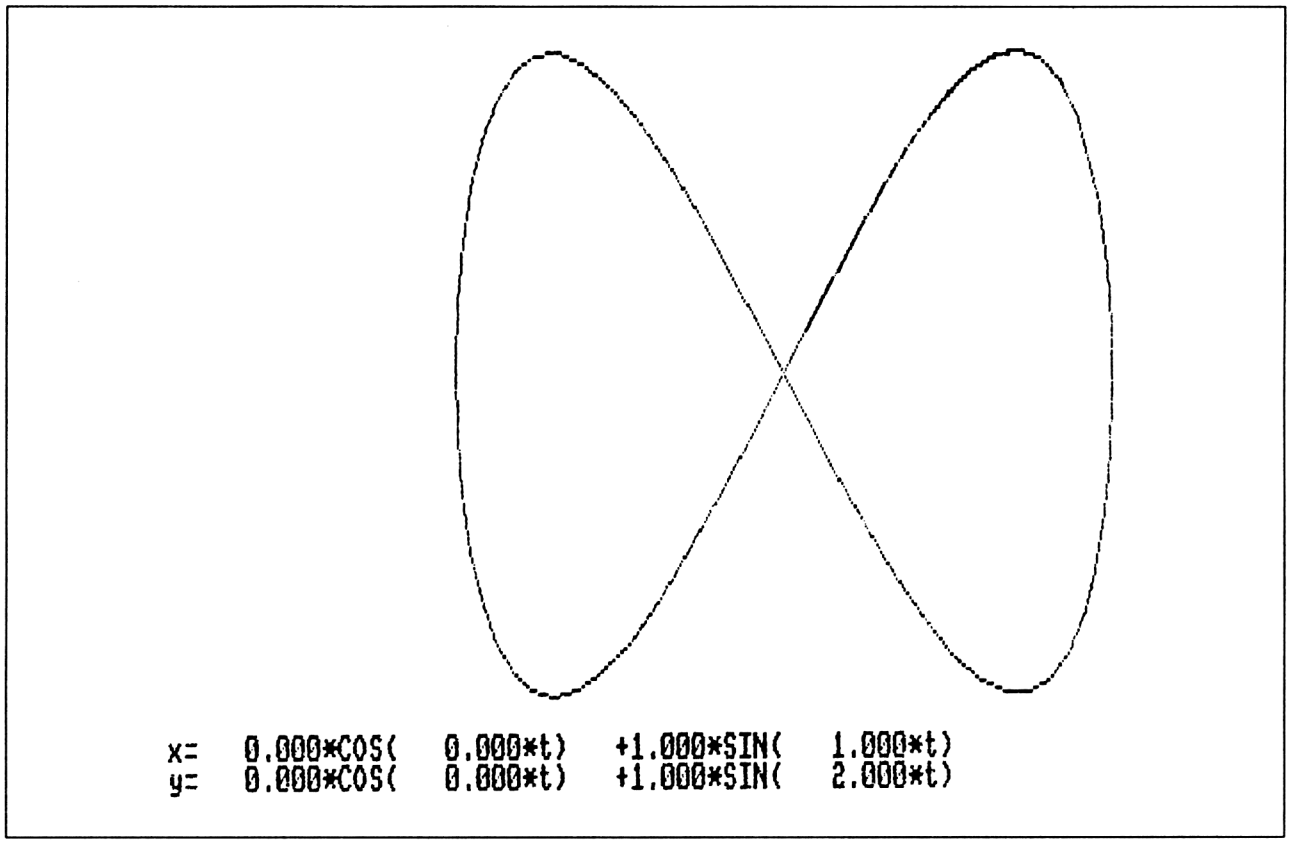


Fig. 2 : Lissajous sur la fréquence,
x = SIN(t) ; y = SIN(2t).

Si vous inversez les paramètres b2 et d2, ce qui revient à doubler la fréquence de x par rapport à y, vous obtenez le même nœud papillon, mais placé par son propriétaire un lendemain de fête bien arrosée.

Et si nous quadruplions la fréquence ? Dans ce cas, le paramètre d2 devient égal à 4, et nous obtenons la courbe de la figure 3.

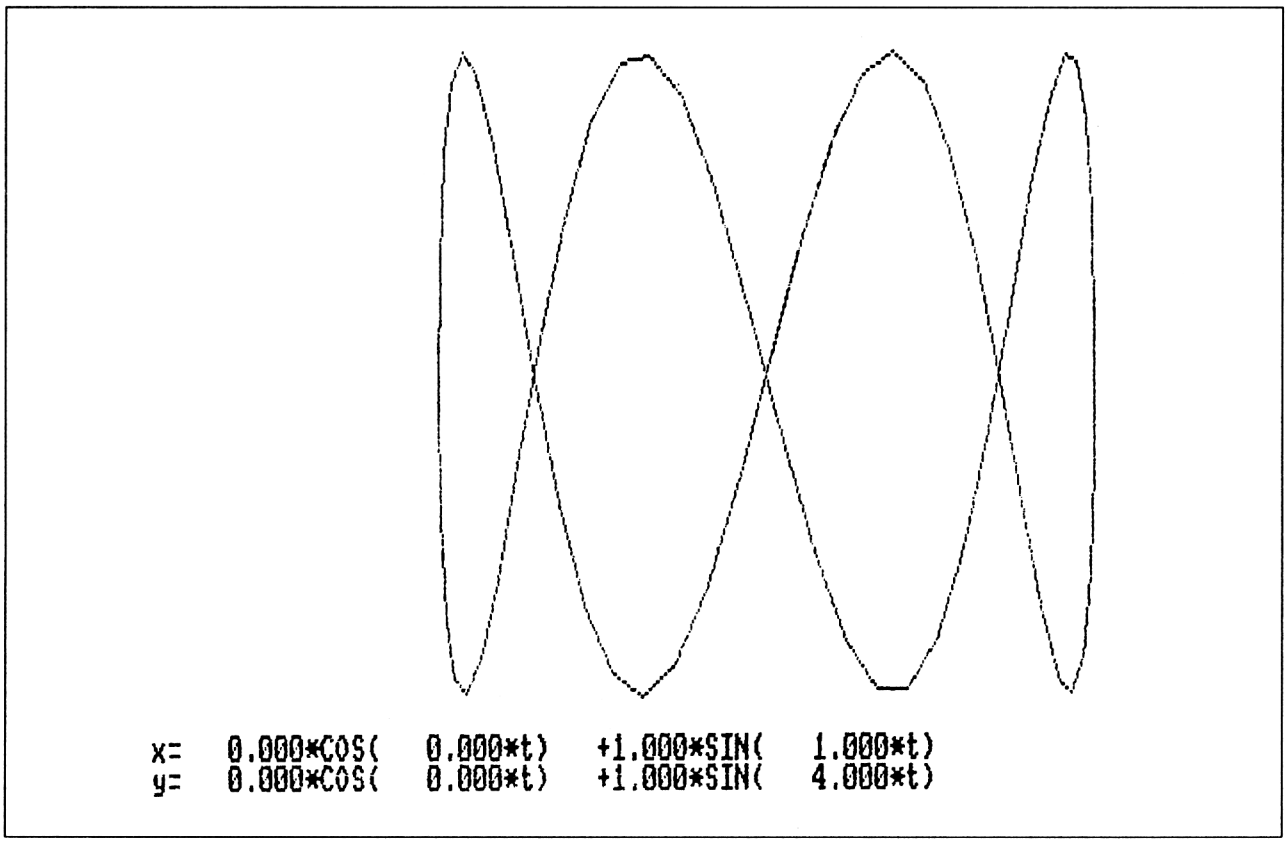


Fig. 3 : x = SIN(t) ; y = SIN(4 t).

Pour rester dans le domaine de Lissajous, il est possible d'imaginer des signaux complètement farfelus (ce que vous dirons les électroniciens, car il leur sera certainement très difficile, voir impossible de les réaliser).

Ces signaux, si vous restez tout de même dans les limites raisonnables de la folie sinusoïdale, vous donnerons probablement des figures proches de relevés réels de Lissajous, et vous permettrons certainement d'empêcher de dormir quelque électronicien pointilleux, en ne lui révélant pas les équations. La figure 4 représente une représentation en X/Y de deux signaux presque probables, et prend pour paramètres :

a1 = SIN(PI/6) = 0.5 a2 = 5
b1 = COS(2 * PI/3) = - 0.5 b2 = 5
c1 = SIN(PI/4) = 0.707 c2 = 6
d1 = COS(5 * PI/6) = - 0.866 d2 = 6

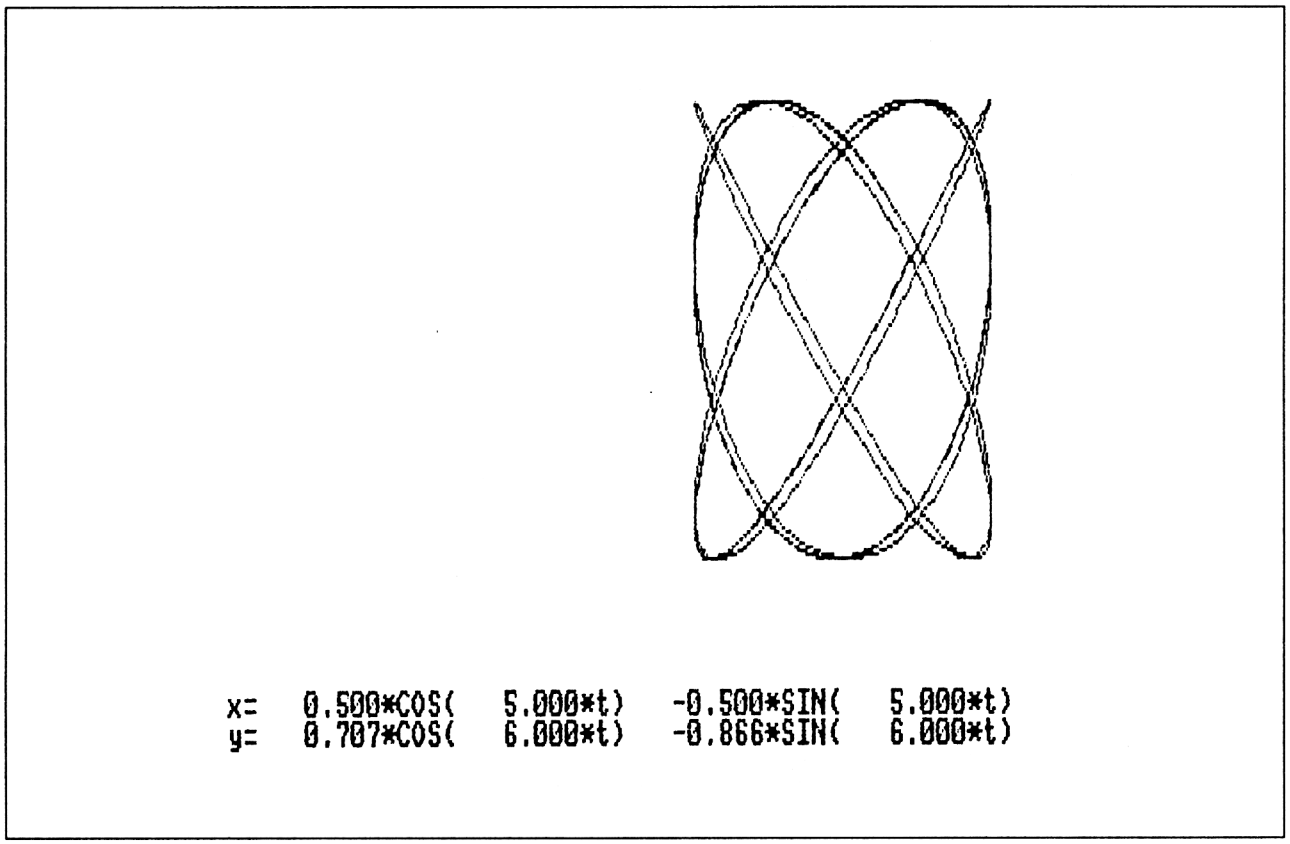


Fig. 4 : Lissajous sur signaux sinusoïdaux bizarres.

Divagations trigonométriques

Notre étude est restée jusque maintenant proche de courbes plus ou moins connues, tracées en fait par des équations remarquables.

Nous vous proposons maintenant de faire voyager votre imagination sur les nombres au sein de notre programme, et de résoudre certains problèmes de représentation.

Notre première courbe folle semble encore proche de Lissajous si nous l'arrêtons au point représenté sur la copie graphique de la figure 5.

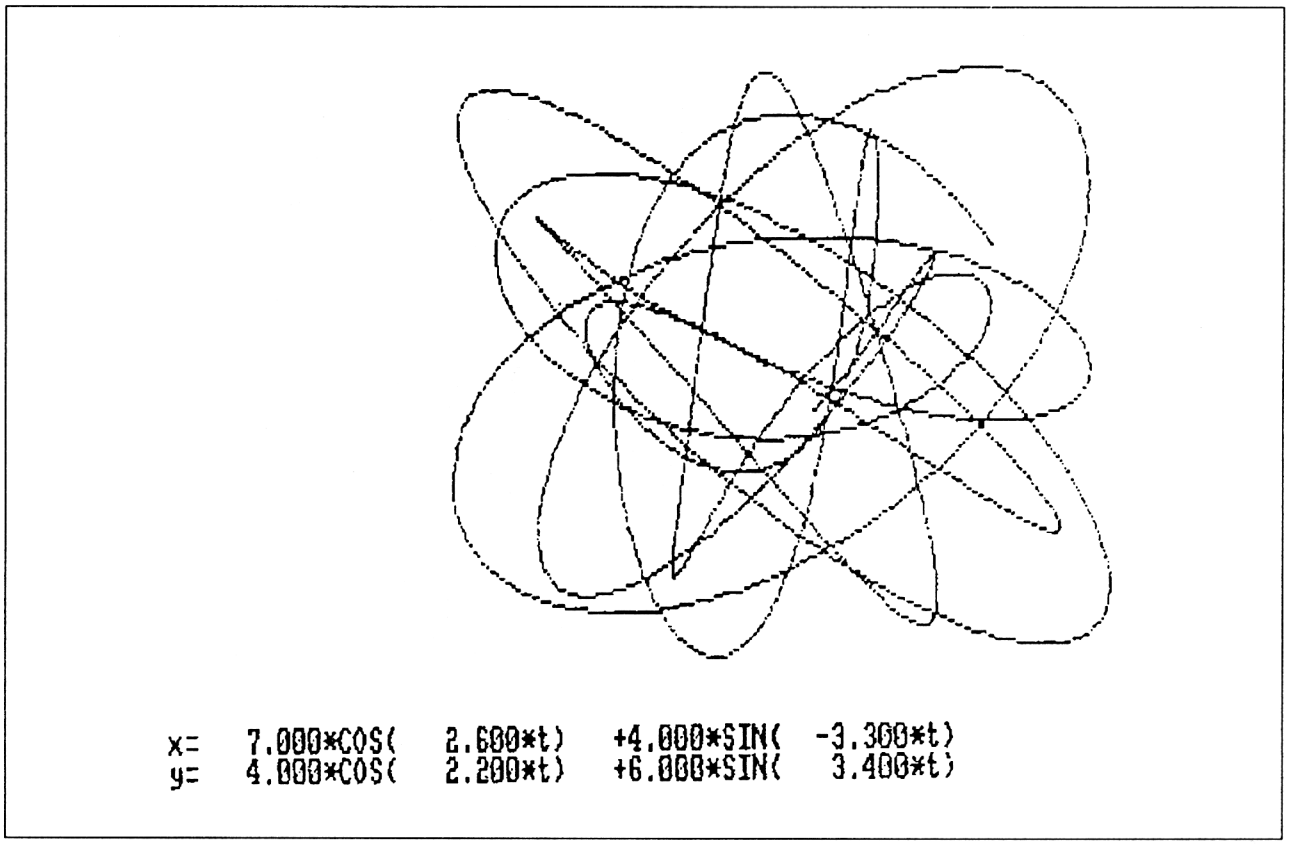


Fig. 5 : Bizarrerie sinusoïdale.

Pas question de la présenter à un technicien électronique, il découvrirait de suite la supercherie.

L'ensemble des nombres étant infini, nous vous proposons d'augmenter, doucement pour l'instant les valeurs des constantes a1, a2, b1, b2, c1, c2, d1 et d2.

Dans les environs des centaines, prenons par exemple :

a1 = 101 a2 = 102
b1 = 103 b2 = 104
c1 = 105 c2 = 106
d1 = 107 d2 = 108

La courbe obtenue avec notre programme est représentée en figure 6.

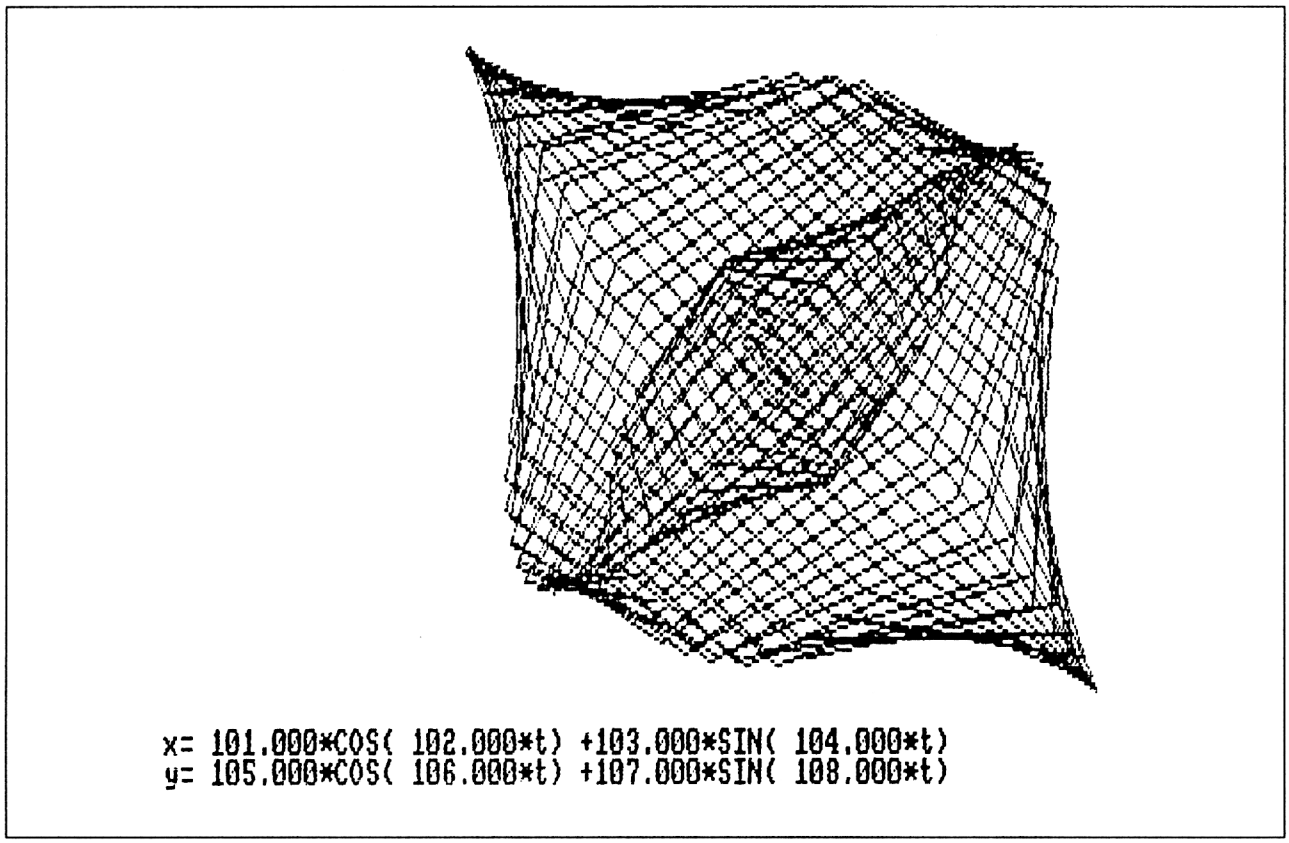


Fig. 6 : Tracé pour t = 0.01.

Bien qu'agréable à regarder, une question se pose sur cette figure : mais où sont donc passées les harmonieuses courbes des fonctions SINUS et COSINUS ?

Le problème provient de la ligne 700 du programme, qui concerne l'incrémentation de la valeur t.

700 t = t + 0.01

La pulsation étant tellement élevée, que la valeur w * t varie beaucoup trop pour obtenir des valeurs de SINUS et COSINUS proches les unes par rapport aux autres.

Comme nous relions nos points à l'aide de l'instruction **DRAW**, nous obtenons des sections de droites reliant deux points calculés très éloignés.

La solution est de diminuer la valeur de l'incrément de t, par exemple dans notre cas, à 0.001 au lieu de 0.01, et tout rentre dans l'ordre, comme le confirme la figure 7.

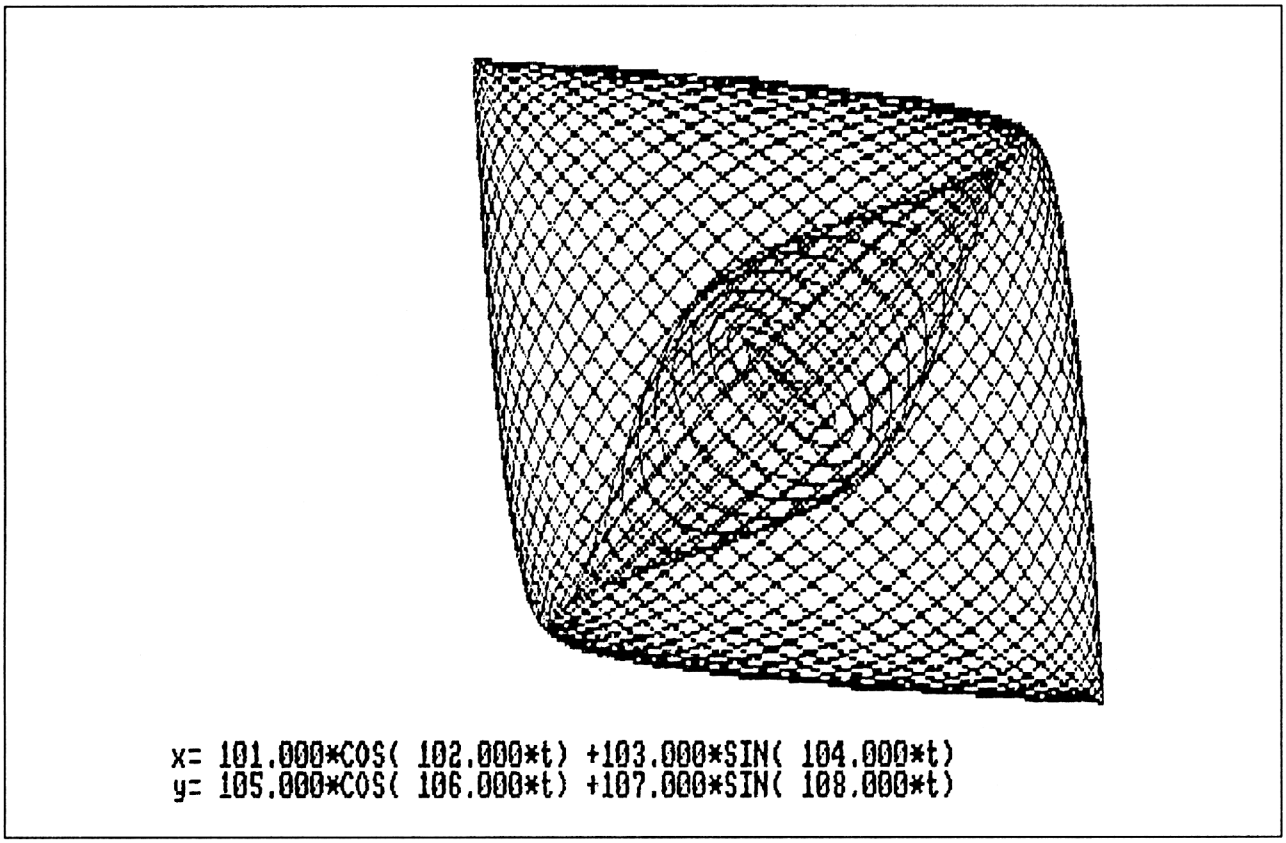


Fig. 7 : Tracé pour t = 0.001.

Toujours autour de la centaine, la figure 8 nous permet d'admirer une rosace sinusoïdale.

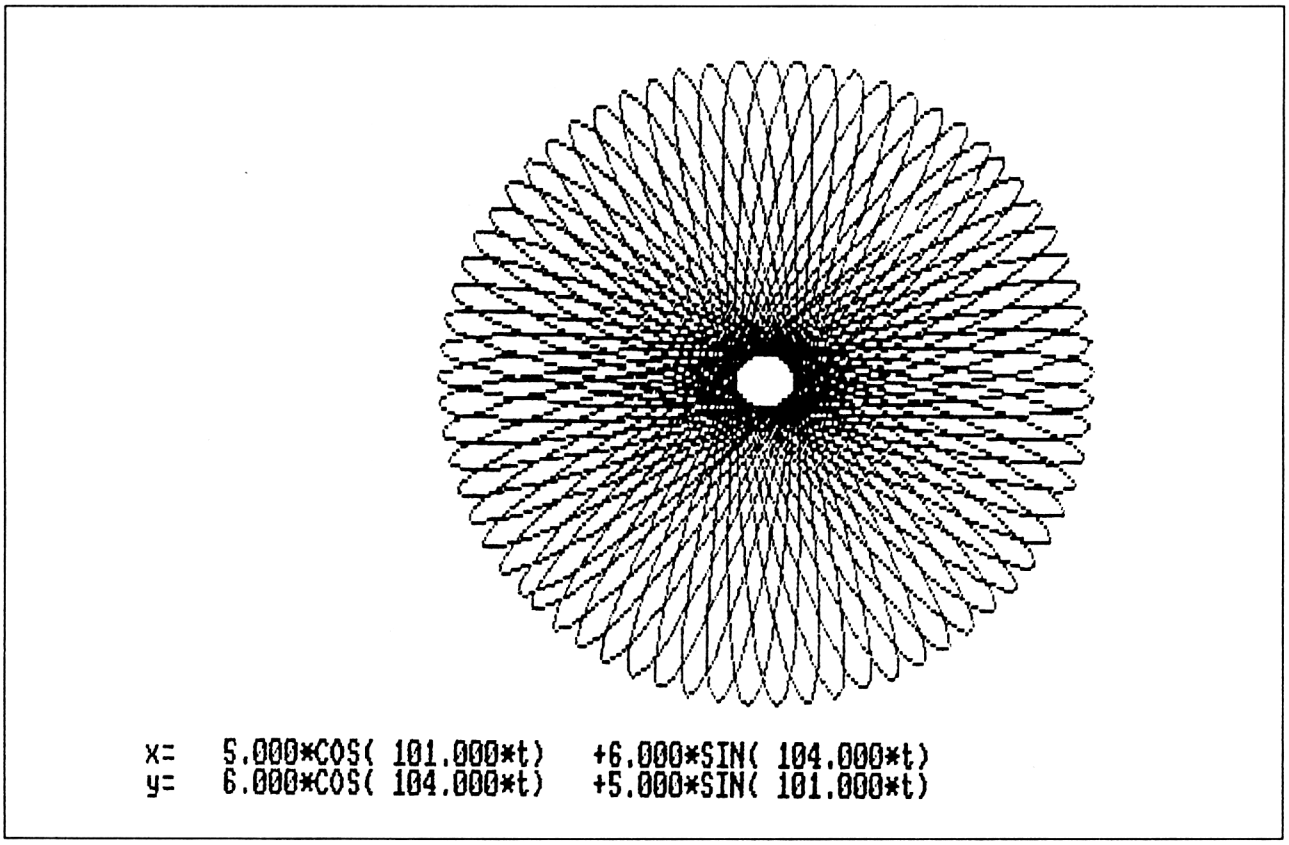


Fig. 8 : Rosace sinusoïdale pour t = 0.001.

Si nous poussions nos investigations plus loin dans les grands nombres, en passant dans l'ordre du millier pour les valeurs de w, nous retrouvrions le même problème, et serions obligés de diminuer l'incrément de t à 0.0001.

De là à faire trouver au programme la valeur adéquate de l'incrément en fonction de la valeur maximale des pulsations données (paramètres a2, b2, c2 et d2), il n'y a qu'un pas que nous vous invitons à franchir pour parfaire vos talents de programmeur. Signalons simplement que cette valeur devra être inversement proportionnelle à la pulsation.

En poussant l'irréel

Pas question ici de faire le tour de toutes les courbes inimaginables, votre bibliothèque ne supporterait pas le poids des feuilles.

Nous faisons confiance à votre imagination, et nous proposons pour conclure les paramètres d'une courbe découverte pas Stanley Miller.

a1 = - 0.7	a2 = 3.01
b1 = 1	b2 = 0.99
c1 = 1	c2 = 1.01
d1 = 0.1	d2 = 15.03

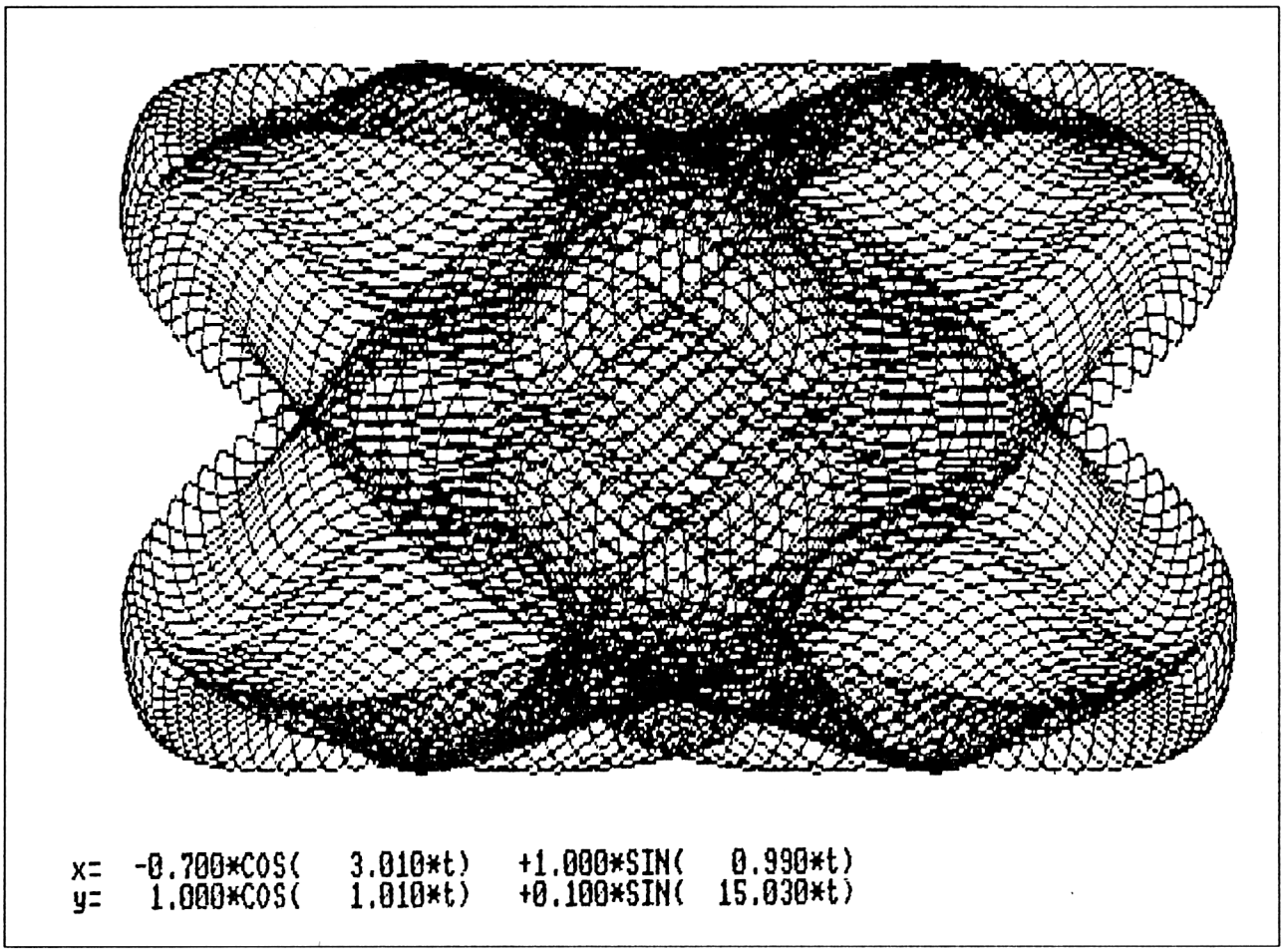


Fig. 9 : La dentelle de S. Miller.

Il vous faudra un peu de patience pour découvrir l'étendue de cette dentelle à faire rougir une bigouden.

5/11

Tracé de cercles

Les CPC ne possèdent pas d'instruction de tracé de cercles en Basic. Nous allons bien vite combler cette lacune à l'aide de la RSX **|CIRCLE**. Tout d'abord, un peu de théorie.

La méthode la plus classique pour tracer un cercle consiste à utiliser ses équations :

$$X = XC + R \times COS(alpha)$$
$$Y = YC + R \times SIN(alpha)$$

où XC et YC sont les coordonnées du centre du cercle, R est le rayon du cercle, et alpha est un angle variant entre 0 et 360 degrés. Cette méthode de tracé est relativement simple. Son seul inconvénient est sa relative lenteur, même en assembleur.

Heureusement, le mathématicien Bresenham a découvert un algorithme qui offre deux avantages :

- les coordonnées sont calculées à l'aide d'opérations entières ;
- les opérations pour calculer chaque point consistent en de simples additions, et sont donc très rapidement effectuées.

Nous n'allons pas retracer le cheminement logique qui a permis à Bresenham de définir l'algorithme. Nous utiliserons simplement le résultat :

Les coordonnées du premier point du cercle sont X=0, Y=R.
La valeur 3-2 x R est affectée à la variable CALC. Si cette valeur est positive, la prochaine valeur de CALC est (X-Y)x4+10+CALC et l'ordonnée du prochain point est incrémentée. Dans le cas contraire, la prochaine valeur de CALC est Xx4+6+CALC. Quelle que soit la valeur de CALC, l'abscisse est incrémentée. Les couples de points suivants appartiennent au cercle : (X, Y), (X, -Y), (-X, Y), (-X, -Y), (Y, X), (Y, -X), (-Y, X), (-Y, -X).
La description du cercle est complète lorsque X-Y devient positif.

Le petit programme Basic suivant applique à la lettre cet algorithme :

```
1000 REM ------------------------------------------------
1010 REM Version BASIC du programme de trace de cercles
1020 REM ------------------------------------------------
1030 REM
1040 CLS
1050 x=100:y=100:r=80
1060 ORIGIN x,y
1070 x=0:y=r
1080 calc=3-r*2
1090 PLOT x,y:PLOT x,-y:PLOT -x,y:PLOT -x,-y
1100 PLOT y,x:PLOT y,-x:PLOT -y,x:PLOT -y,-x
1110 IF x-y>0 THEN END
1120 IF calc>0 THEN calc=(x-y)*4+10+calc:y=y-1 ELSE calc=x*4+6+calc
1130 x=x+1
1140 GOTO 1090
```

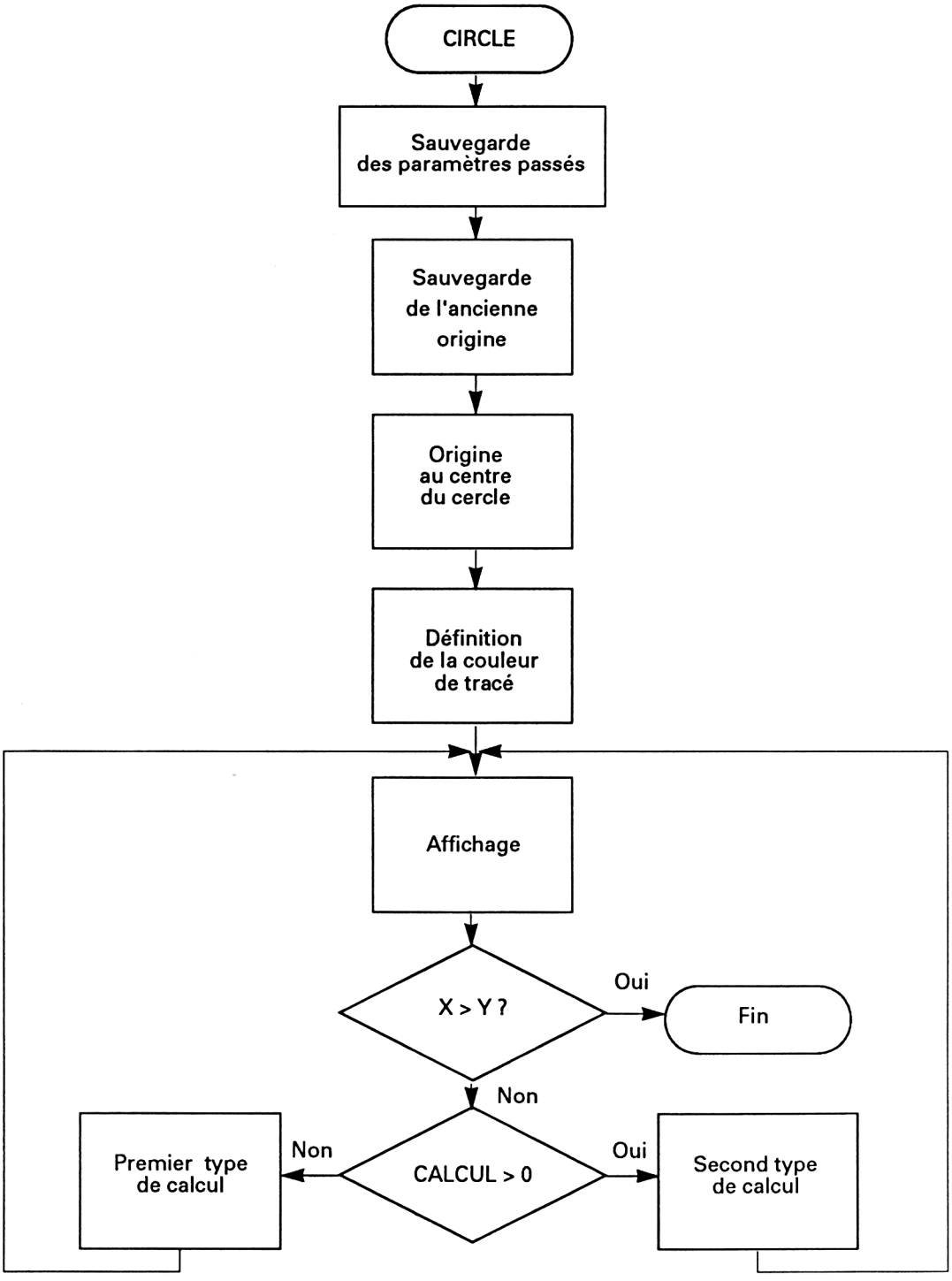
Afin de rendre le tracé de cercles plus accessible a un programme Basic, nous allons définir la RSX **¦CIRCLE** dont la syntaxe est la suivante :

¦CIRCLE,X,Y,R,C

où X et Y sont les coordonnées du cercle, R est le rayon du cercle, et C la couleur de tracé.

LA RSX EN DÉTAIL

La RSX ¦**CIRCLE** est bien entendu écrite en Assembleur. Sa logique apparaît dans l'ordinogramme suivant :



La définition de la RSX (lignes 42 à 46) fait désormais partie des opérations classiques. Reportez-vous si nécessaire aux autres RSX de l'ouvrage pour avoir plus de détails à son sujet.

La routine de traitement de la RSX se trouve à l'adresse **CIRCLE**. Les premières actions effectuées par cette routine consistent en la mémorisation des paramètres qui lui sont passés. Ces paramètres sont pointés par le registre **IX** :

- la couleur de tracé est stockée dans la variable **COUL** (lignes 53 et 54) ;
- le rayon du cercle dans la variable **RAYON** (lignes 55 à 57) ;
- les coordonnées du centre du cercle dans les variables **X** et **Y** (lignes 58 à 63).

L'origine graphique de l'écran étant redéfinie, le programme sauvegarde ensuite l'ancienne origine qui est lue à l'aide de la macro **GRA GET ORIGIN** du FIRMWARE (lignes 69 à 71).

La nouvelle origine est définie au centre du cercle à l'aide de la macro **GRA SET ORIGIN** du FIRMWARE (lignes 77 à 79).

Les coordonnées du point de départ sont initialisées (X=0, Y=R) lignes 85 à 89.

La couleur de tracé est initialisée à l'aide de la variable **COUL** et de la macro **GRA SET PEN** du FIRMWARE (lignes 90 et 91).

La boucle de calcul **BIS** est très proche de celle réalisée en Basic. Notez cependant que les multiplications par 4 ont été effectuées à l'aide de décalages et rotations logiques (lignes 109 et 112 et 130 à 133).

Les points calculés sont affichés à l'aide de la macro **GRA PLOT ABS** du FIRMWARE entre les lignes 149 et 209. De nombreuses instructions **PUSH** et **POP** utilisent la pile pour augmenter la vitesse de calcul.

Le programme se termine par la restitution de l'ancienne origine graphique (lignes 215 à 219).

Le listing de la RSX est le suivant :

```
1                               ORG  9000H
2                               LOAD 9000H
3                ;------------------------------
4                ; RSX CIRCLE
5                ; Format : |CIRCLE,X,Y,R,C
6                ; Entree : X=Abscisse du rayon
7                ;          Y=Ordonnee du rayon
8                ;          R=Rayon
9                ;          C=Couleur
10               ; Sortie : Affichage du cercle
11               ;------------------------------
12               ;
13               ;
14               ;--------------------------------
15               ; Declaration des constantes
16               ; et des variables du programme
17               ;--------------------------------
18               ;
19               LOGEXT:     EQU  0BCD1H          ;KL LOG EXT
20               GETORI:     EQU  0BBCCH          ;GRA GET ORIGIN
21               SETORI:     EQU  0BBC9H          ;GRA SET ORIGIN
22               PLOTABS:    EQU  0BBEAH          ;GRA PLOT ABS
23               SETPEN:     EQU  0BBDEH          ;GRA SET PEN
24               BUF:        DS   4               ;ZONE RAM POUR LOG EXT
25 9004 0990     PTRTAB:     DW   TABLE           ;Pointeur TABLE
26 9006 C32790               JP   CIRCLE          ;Affichage du cercle
27 9009 43495243 TABLE:      DB   "CIRCL"
```

```
27 900D 4C
28 900E C5                      DB    "E"+80H
29 900F 00                      DB    0                   ;Fin de table
30                X:            DS    2                   ;Abscisse centre
31                Y:            DS    2                   ;Urdonnee centre
32                XORI:         DS    2                   ;Abs ancienne origine
33                YORI:         DS    2                   ;Ord ancienne origine
34                RAYON:        DS    2                   ;Rayon du cercle
35                CALC:         DS    2                   ;Var intermediaire
36                COUL:         DS    1                   ;Couleur de trace
37                ;
38                ;-----------------------------------
39                ; Definition de la RSX
40                ;-----------------------------------
41                ;
42                DEFRSX:       EQU   $                   ;Point d'entree
43 901D 010490                 LD    BC,PTRTAB           ;Ptr table definition
44 9020 210090                 LD    HL,BUF              ;Buffer pour LOG EXT
45 9023 CDD1BC                 CALL  LOGEXT              ;Definition de la RSX
46 9026 C9                     RET
47                ;
48                ;-----------------------------------
49                ; Traitement de CIRCLE
50                ;-----------------------------------
51                ;
52                CIRCLE:       EQU   $                   ;Point d'entree
53 9027 DD7E00                 LD    A,(IX+0)
```

```
54 902A 321C90              LD   (COUL),A
55 902D DD6603              LD   H,(IX+3)
56 9030 DD6E02              LD   L,(IX+2)
57 9033 221890              LD   (RAYON),HL        ;Sauv Rayon
58 9036 DD6605              LD   H,(IX+5)
59 9039 DD6E04              LD   L,(IX+4)
60 903C 221290              LD   (Y),HL            ;Sauv Ordonnee
61 903F DD6607              LD   H,(IX+7)
62 9042 DD6E06              LD   L,(IX+6)
63 9045 221090              LD   (X),HL            ;Sauv Abscisse
64                ;
65                ;- - - - - - - - - - - - - - - -
66                ; Sauvegarde ancienne origine
67                ;- - - - - - - - - - - - - - - -
68                ;
69 9048 CDCCBB              CALL GETORI            ;GRA GET ORIGIN
70 904B ED531490            LD   (XORI),DE
71 904F 221690              LD   (YORI),HL
72                ;
73                ;- - - - - - - - - - - - - - - -
74                ; Definition nouvelle origine
75                ;- - - - - - - - - - - - - - - -
76                ;
77 9052 ED5B1090            LD   DE,(X)
78 9056 2A1290              LD   HL,(Y)
79 9059 CDC9BB              CALL SETORI            ;GRA SET ORIGIN
80                ;
```

```
                ;- - - - - - - - - - - - - - -
                ; Initialisations diverses
                ;- - - - - - - - - - - - - - -
                ;
9050 210000             LD    HL,0
9053 221090             LD    (X),HL
9056 221A90             LD    (CALC),HL
9059 2A1890             LD    HL,(RAYON)
905C 221290             LD    (Y),HL
905F 3A1C90             LD    A,(COUL)
9062 CDDEBB             CALL  SETPEN            ;Couleur de trace
                ;
                BIS:    EQU   $                 ;Boucle principale
9065 CDD490             CALL  AFFICHE           ;Aff de 8 points
9068 2A1090             LD    HL,(X)
906B ED5B1290           LD    DE,(Y)
906F AF                 XOR   A
9070 ED52               SBC   HL,DE
9072 F23A91             JP    P,FIN             ;Fin du programme
                ;
9075 2A1A90             LD    HL,(CALC)
9078 110000             LD    DE,0
907B AF                 XOR   A
907C ED52               SBC   HL,DE
907E F2AD90             JP    P,SECOND          ;2eme type de calcul
                ;
                PREMIER:EQU   $                 ;1er type de calcul
9081 2A1090             LD    HL,(X)
```

```
109 9090 CB25                SLA  L
110 9092 CB14                RL   H
111 9094 CB25                SLA  L
112 9096 CB14                RL   H              ;X*4
113 9098 110600              LD   DE,6
114 909B 19                  ADD  HL,DE          ;X*4+6
115 909C ED5B1A90            LD   DE,(CALC)
116 90A0 19                  ADD  HL,DE          ;X*4+6+CALC
117 90A1 221A90              LD   (CALC),HL
118              ;
119              SUITE:      EQU  $
120 90A4 2A1090              LD   HL,(X)
121 90A7 23                  INC  HL
122 90A8 221090              LD   (X),HL         ;X+1
123 90AB 18C4                JR   BIS            ;Suite du traitement
124              ;
125              SECOND:     EQU  $              ;2eme type de calcul
126 90AD 2A1090              LD   HL,(X)
127 90B0 ED5B1290            LD   DE,(Y)
128 90B4 AF                  XOR  A
129 90B5 ED52                SBC  HL,DE
130 90B7 CB25                SLA  L
131 90B9 CB14                RL   H
132 90BB CB25                SLA  L
133 90BD CB14                RL   H              ;(X-Y)*4
134 90BF 110A00              LD   DE,10
135 90C2 19                  ADD  HL,DE          ;(X-Y)*4+10
```

```
136 90C3 ED5B1A90            LD   DE,(CALC)
137 90C7 19                  ADD  HL,DE              ;(X-Y)*4+10+CALC
138 90C8 221A90              LD   (CALC),HL
139 90CB 2A1290              LD   HL,(Y)
140 90CE 2B                  DEC  HL
141 90CF 221290              LD   (Y),HL
142 90D2 18D0                JR   SUITE
143              ;
144              ; - - - - - - - - - - - - - - - -
145              ; Affichage des points calcules
146              ; - - - - - - - - - - - - - - - -
147              ;
148              AFFICHE:    EQU  $
149 90D4 2A1290              LD   HL,(Y)
150 90D7 ED5B1090            LD   DE,(X)
151 90DB D5                  PUSH DE
152 90DC E5                  PUSH HL
153 90DD E5                  PUSH HL
154 90DE D5                  PUSH DE
155              ;
156 90DF 210000              LD   HL,0
157 90E2 ED5B1290            LD   DE,(Y)
158 90E6 AF                  XOR  A
159 90E7 ED52                SBC  HL,DE
160 90E9 ED5B1090            LD   DE,(X)
161 90ED D5                  PUSH DE
162 90EE E5                  PUSH HL
```

```
163 90EF E5                 PUSH HL
164 90F0 D5                 PUSH DE             ;-Y,X
165             ;
166 90F1 210000             LD   HL,0
167 90F4 ED5B1090           LD   DE,(X)
168 90F8 AF                 XOR  A
169 90F9 ED52               SBC  HL,DE
170 90FB ED5B1290           LD   DE,(Y)
171 90FF E5                 PUSH HL
172 9100 D5                 PUSH DE             ;-X,Y
173 9101 D5                 PUSH DE
174 9102 E5                 PUSH HL             ;Y,-X
175             ;
176 9103 E5                 PUSH HL             ;-X
177 9104 210000             LD   HL,0
178 9107 ED5B1290           LD   DE,(Y)
179 910B AF                 XOR  A
180 910C ED52               SBC  HL,DE
181 910E E5                 PUSH HL
182 910F D1                 POP  DE             ;-Y
183 9110 E1                 POP  HL             ;-X
184 9111 E5                 PUSH HL
185 9112 D5                 PUSH DE             ;-X,-Y
186             ;
187 9113 CDEABB             CALL PLOTABS        ;-Y,-X
188 9116 E1                 POP  HL
189 9117 D1                 POP  DE
```

```
190 9118 CDEABB                 CALL PLOTABS            ;-X,-Y
191 911B E1                     POP  HL
192 911C D1                     POP  DE
193 911D CDEABB                 CALL PLOTABS            ;Y,-X
194 9120 E1                     POP  HL
195 9121 D1                     POP  DE
196 9122 CDEABB                 CALL PLOTABS            ;-X,Y
197 9125 E1                     POP  HL
198 9126 D1                     POP  DE
199 9127 CDEABB                 CALL PLOTABS            ;-Y,X
200 912A E1                     POP  HL
201 912B D1                     POP  DE
202 912C CDEABB                 CALL PLOTABS            ;X,-Y
203 912F E1                     POP  HL
204 9130 D1                     POP  DE
205 9131 CDEABB                 CALL PLOTABS            ;Y,X
206 9134 E1                     POP  HL
207 9135 D1                     POP  DE
208 9136 CDEABB                 CALL PLOTABS            ;X,Y
209 9139 C9                     RET
210                 ;
211                 ;- - - - - - - - - - - - - - - -
212                 ; Restitution de l'origine
213                 ;- - - - - - - - - - - - - - - -
214                 ;
215                 FIN:        EQU  $
216 913A ED5B1490               LD   DE,(XORI)
```

```
217 913E 2A1690                 LD   HL,(YORI)
218 9141 CDC9BB                 CALL SETORI              ;GRA SET ORIGIN
219 9144 C9                     RET
220                             END

AFFICHE    90D4 BUF     9000 BIS     9071 CALC    901A
COUL       901C CIRCLE  9027 DEFRSX  901D FIN     913A
GETORI     BBCC LOGEXT  BCD1 PLOTABS BBEA PTRTAB  9004
PREMIER    908D RAYON   9018 SETORI  BBC9 SETPEN  BBDE
SUITE      90A4 SECOND  90AD TABLE   9009 X       9010
XORI       9014 Y       9012 YORI    9016
```

Comme toujours, voici la version chargeur Basic, bien plus pratique à utiliser :

```
1000 REM ------------------------------------------------
1010 REM Chargeur BASIC de la RSX de trace de cercles
1020 REM ------------------------------------------------
1030 REM
1040 FOR i=&9000 TO &9144
1050   READ a$
1060   a$="&"+a$
1070   POKE i,VAL(a$)
1080 NEXT i
1090 CALL &901D
1100 END
1110 REM - - - - - - - - - - - - - - - - - -
1120 REM Donnees du programme de trace
1130 REM - - - - - - - - - - - - - - - - - -
1140 REM
1150 DATA FC,A6,4,90,9,90,C3,27,90,43,49,52,43,4C,C5,0
1160 DATA C,0,B,0,0,0,0,0,11,0,C,0,2,1,4,90
1170 DATA 21,0,90,CD,D1,BC,C9,DD,7E,0,32,1C,90,DD,66,3
1180 DATA DD,6E,2,22,18,90,DD,66,5,DD,6E,4,22,12,90,DD
1190 DATA 66,7,DD,6E,6,22,10,90,CD,CC,BB,ED,53,14,90,22
1200 DATA 16,90,ED,5B,10,90,2A,12,90,CD,C9,BB,21,0,0,22
1210 DATA 10,90,22,1A,90,2A,18,90,22,12,90,3A,1C,90,CD,DE
1220 DATA BB,CD,D4,90,2A,10,90,ED,5B,12,90,AF,ED,52,F2,3A
1230 DATA 91,2A,1A,90,11,0,0,AF,ED,52,F2,AD,90,2A,10,90
1240 DATA CB,25,CB,14,CB,25,CB,14,11,6,0,19,ED,5B,1A,90
1250 DATA 19,22,1A,90,2A,10,90,23,22,10,90,18,C4,2A,10,90
1260 DATA ED,5B,12,90,AF,ED,52,CB,25,CB,14,CB,25,CB,14,11
1270 DATA A,0,19,ED,5B,1A,90,19,22,1A,90,2A,12,90,2B,22
1280 DATA 12,90,18,D0,2A,12,90,ED,5B,10,90,D5,E5,E5,D5,21
1290 DATA 0,0,ED,5B,12,90,AF,ED,52,ED,5B,10,90,D5,E5,E5
1300 DATA D5,21,0,0,ED,5B,10,90,AF,ED,52,ED,5B,12,90,E5
1310 DATA D5,D5,E5,E5,21,0,0,ED,5B,12,90,AF,ED,52,E5,D1
1320 DATA E1,E5,D5,CD,EA,BB,E1,D1,CD,EA,BB,E1,D1,CD,EA,BB
1330 DATA E1,D1,CD,EA,BB,E1,D1,CD,EA,BB,E1,D1,CD,EA,BB,E1
1340 DATA D1,CD,EA,BB,E1,D1,CD,EA,BB,C9,ED,5B,14,90,2A,16
1350 DATA 90,CD,C9,BB,C9,0,0,0,0,0,0,0,0,0,0,0
```

et les données de checksum correspondantes :

```
81 CB 5A 55 E0 F3 98 C2 63 C5 3E 8E 17 DA 67 A2 2C 62 5A 66 AD
```

Le petit programme de démonstration suivant montre qu'il est très simple d'utiliser la RSX |**CIRCLE** :

```
100 REM ----------------------------
110 REM Programme de demonstration
120 REM ----------------------------
130 REM
140 MODE 1
150 FOR I=1 TO 50
160   X=INT(RND(1)*600)
170   Y=INT(RND(1)*400)
180   R=INT(RND(1)*100)
190   C=INT(RND(1)*3)+1
200   |CIRCLE,X,Y,R,C
210 NEXT I
```

Avant de pouvoir utiliser |**CIRCLE**, il faut bien entendu l'avoir installée, par exemple avec le chargeur Basic précédent.

5/12

Tracé de rectangles vides et pleins

Nous vous proposons les RSX ¦BOX et ¦PBOX qui affichent respectivement des rectangles vides et pleins sur l'écran.

RSX ¦ BOX

Comment utiliser la RSX

Le format de la RSX ¦BOX est le suivant :

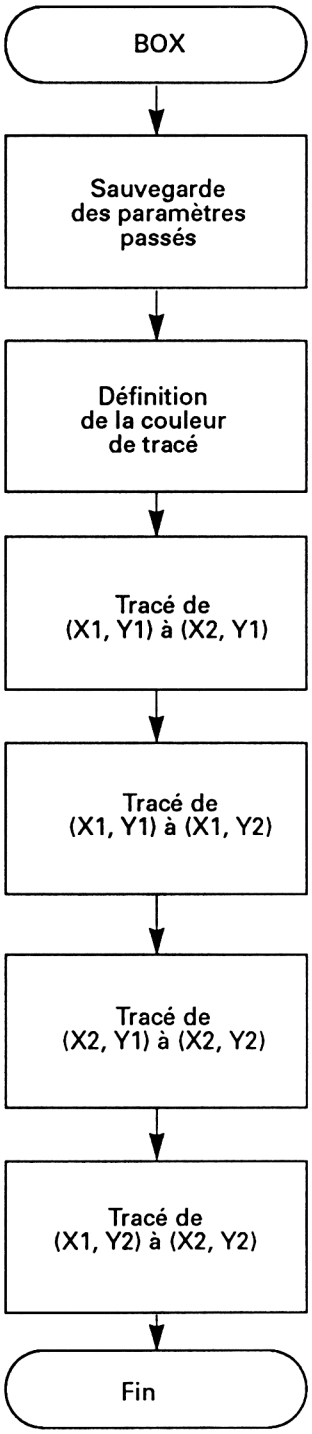
¦BOX, X1, Y1, X2, Y2, C

où X1 et Y1 sont les coordonnées du coin supérieur gauche du rectangle, X2 et Y2 sont les coordonnées du coin inférieur droit du rectangle, et C est la couleur de tracé. Les valeurs possibles pour C dépendent du mode d'affichage :

MODE 0 : 0 à 15,
MODE 1 : 0 à 3,
MODE 2 : 0 ou 1.

La RSX en détail

La logique de la RSX apparaît dans l'ordinogramme suivant :



La routine de traitement se trouve à l'étiquette **BOX**. Les données qui lui sont passées sont stockées dans les variables **COUL** (couleur), **X1**, **Y1**, **X2** et **Y2** (coordonnées extrémales) entre les lignes 56 et 69.

La couleur de tracé est initialisée à l'aide de la macro **GRA SET PEN** du FIRMWARE (lignes 76 et 77).

Le tracé du rectangle vide se fait en quatre étapes.

- Etape 1 :
  - origine du tracé en X1, Y1 (lignes 83 à 85) ;
  - tracé d'une ligne entre X1, Y1 et X2, Y1 (lignes 86 à 88).
- Etape 2 :
  - origine du tracé en X1, Y1 (lignes 90 à 92) ;
  - tracé d'une ligne entre X1, Y1 et X1, Y2 (lignes 93 à 95).
- Etape 3 :
  - origine du tracé en X2, Y1 (lignes 97 à 99) ;
  - tracé d'une ligne entre X2, Y1 et X2, Y2 (lignes 100 à 102).
- Etape 4 :
  - origine du tracé en X1, Y2 (lignes 104 à 106) ;
  - tracé d'une ligne entre X1, Y2 et X2, Y2 (lignes 107 à 109).

Le positionnement de l'origine de chaque tracé se fait avec la macro **GRA MOV ABS** du FIRMWARE, et le tracé des lignes avec la macro **GRA LINE ABS** du FIRMWARE.

Le listing de la RSX est le suivant :

```
                            ORG  9000H
                            LOAD 9000H
                    ;---------------------------------
                    ; RSX BOX
                    ; Format : |BOX,X1,Y1,X2,Y2,C
                    ; Entree : X1=Abs coin sup gauche
                    ;          Y1=Ord coin sup gauche
                    ;          X2=Abs coin inf droit
                    ;          Y2=Ord coin inf droit
                    ;          C=Couleur
                    ; Sortie : Affichage du rectangle
                    ;---------------------------------
                    ;
                    ;
                    ;---------------------------------
                    ; Declaration des constantes
                    ; et des variables du programme
                    ;---------------------------------
                    ;
                    LOGEXT:     EQU  0BCD1H          ;KL LOG EXT
                    SETPEN:     EQU  0BBDEH          ;GRA SET PEN
                    MOVABS:     EQU  0BBC0H          ;GRA MOVE ABS
                    LINEABS:    EQU  0BBF6H          ;GRA LINE ABS
                    BUF:        DS   4               ;ZONE RAM POUR LOG EXT
9004 0990           PTRTAB:     DW   TABLE           ;Pointeur TABLE
9006 C32090                     JP   BOX             ;Affichage du rectangle
9009 424F           TABLE:      DB   "BO"
```

```
28 900B D8                      DB    "X"+80H
29 900C 00                      DB    0                   ;Fin de table
30                 X1:          DS    2                   ;Abscisse coin sup gauche
31                 Y1:          DS    2                   ;Ordonnee coin sup gauche
32                 X2:          DS    2                   ;Abscisse coin inf droit
33                 Y2:          DS    2                   ;Ordonnee coin inf droit
34                 COUL:        DS    1                   ;Couleur de trace
35                 ;
36                 ;---------------------------------
37                 ; Definition de la RSX
38                 ;---------------------------------
39                 ;
40                 DEFRSX:      EQU   $                   ;Point d'entree
41 9016 010490                  LD    BC,PTRTAB           ;Ptr table definition
42 9019 210090                  LD    HL,BUF              ;Buffer pour LOG EXT
43 901C CDD1BC                  CALL  LOGEXT              ;Definition de la RSX
44 901F C9                      RET
45                 ;
46                 ;--------------------
47                 ; Traitement de BOX
48                 ;--------------------
49                 ;
50                 BOX:         EQU   $                   ;Point d'entree
51                 ;
52                 ;- - - - - - - - - - - - - - - -
53                 ; Lecture des donnees passees
54                 ;- - - - - - - - - - - - - - - -
```

```
55                ;
56 9020 DD7E00              LD   A,(IX+0)
57 9023 321590              LD   (COUL),A          ;Couleur de trace
58 9026 DD6603              LD   H,(IX+3)
59 9029 DD6E02              LD   L,(IX+2)
60 902C 221390              LD   (Y2),HL           ;Ord coin inf droit
61 902F DD6605              LD   H,(IX+5)
62 9032 DD6E04              LD   L,(IX+4)
63 9035 221190              LD   (X2),HL           ;Abs coin inf droit
64 9038 DD6607              LD   H,(IX+7)
65 903B DD6E06              LD   L,(IX+6)
66 903E 220F90              LD   (Y1),HL           ;Ord coin sup gauche
67 9041 DD6609              LD   H,(IX+9)
68 9044 DD6E08              LD   L,(IX+8)
69 9047 220D90              LD   (X1),HL           ;Abs coin sup gauche
70                ;
71                ;- - - - - - - - - - - - - - - - -
72                ; Initialisation de la couleur
73                ; de trace du rectangle
74                ;- - - - - - - - - - - - - - - - -
75                ;
76 904A 3A1590              LD   A,(COUL)
77 904D CDDEBB              CALL SETPEN            ;Couleur de trace
78                ;
79                ;- - - - - - - - - - -
80                ; Trace du rectangle
81                ;- - - - - - - - - - -
```

```
 82                ;
 83 9050 ED5B0D90          LD    DE,(X1)
 84 9054 2A0F90            LD    HL,(Y1)
 85 9057 CDC0BB            CALL  MOVABS          ;Coin sup gauche
 86 905A ED5B1190          LD    DE,(X2)
 87 905E 2A0F90            LD    HL,(Y1)
 88 9061 CDF6BB            CALL  LINEABS         ;Ligne du haut
 89                ;
 90 9064 ED5B0D90          LD    DE,(X1)
 91 9068 2A0F90            LD    HL,(Y1)
 92 906B CDC0BB            CALL  MOVABS          ;Coin sup gauche
 93 906E ED5B0D90          LD    DE,(X1)
 94 9072 2A1390            LD    HL,(Y2)
 95 9075 CDF6BB            CALL  LINEABS         ;Ligne de gauche
 96                ;
 97 9078 ED5B1190          LD    DE,(X2)
 98 907C 2A0F90            LD    HL,(Y1)
 99 907F CDC0BB            CALL  MOVABS          ;Coin sup droit
100 9082 ED5B1190          LD    DE,(X2)
101 9086 2A1390            LD    HL,(Y2)
102 9089 CDF6BB            CALL  LINEABS         ;Ligne de droite
103               ;
104 908C ED5B0D90          LD    DE,(X1)
105 9090 2A1390            LD    HL,(Y2)
106 9093 CDC0BB            CALL  MOVABS          ;Coin inf gauche
107 9096 ED5B1190          LD    DE,(X2)
108 909A 2A1390            LD    HL,(Y2)
109 909D CDF6BB            CALL  LINEABS         ;Ligne du bas
```

```
110                ;
111 90A0 C9                    RET
112                            END
```

BUF	9000	BOX	9020	COUL	9015	DEFRSX	9016
LOGEXT	BCD1	LINEABS	BBF6	MOVABS	BBC0	PTRTAB	9004
SETPEN	BBDE	TABLE	9009	X1	900D	X2	9011
Y1	900F	Y2	9013				

Comme toujours, voici le chargeur Basic correspondant :

```
1000 REM ------------------------------------------------------------
1010 REM Chargeur BASIC de la RSX de trace de rectangles vides
1020 REM ------------------------------------------------------------
1030 REM
1040 FOR i=&9000 TO &90A0
1050   READ a$
1060   a$="&"+a$
1070   POKE i,VAL(a$)
1080 NEXT i
1090 CALL &9016
1100 END
1110 REM - - - - - - - - - - - - - - - - - -
1120 REM Donnees du programme de trace
1130 REM - - - - - - - - - - - - - - - - - -
1140 REM
1150 REM BOX
1160 DATA 0,0,0,0,9,90,C3,20,90,42,4F,D8,0,0,0,0
1170 DATA 0,0,0,0,0,0,1,4,90,21,0,90,CD,D1,BC,C9
1180 DATA DD,7E,0,32,15,90,DD,66,3,DD,6E,2,22,13,90,DD
1190 DATA 66,5,DD,6E,4,22,11,90,DD,66,7,DD,6E,6,22,F
1200 DATA 90,DD,66,9,DD,6E,8,22,D,90,3A,15,90,CD,DE,BB
1210 DATA ED,5B,D,90,2A,F,90,CD,C0,BB,ED,5B,11,90,2A,F
1220 DATA 90,CD,F6,BB,ED,5B,D,90,2A,F,90,CD,C0,BB,ED,5B
1230 DATA D,90,2A,13,90,CD,F6,BB,ED,5B,11,90,2A,F,90,CD
1240 DATA C0,BB,ED,5B,11,90,2A,13,90,CD,F6,BB,ED,5B,D,90
1250 DATA 2A,13,90,CD,C0,BB,ED,5B,11,90,2A,13,90,CD,F6,BB
1260 DATA C9,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
```

et les données de checksum afférentes :

```
78 6D 6D 4E 3A 1F 55 6E 9C 51 C9
```

Le petit programme de démonstration suivant montre combien la RSX |**BOX** est facile à manipuler :

```
100 REM ------------------------------
110 REM Programme de demonstration
120 REM ------------------------------------
130 REM
140 MODE 1
150 FOR I=1 TO 50
160   X1=INT(RND(1)*500)
170   Y1=INT(RND(1)*300)
180   X2=INT(RND(1)*600)
190   Y2=INT(RND(1)*400)
200   C=INT(RND(1)*3)+1
210   |BOX,X1,Y1,X2,Y2,C
220 NEXT I
```

Avant de pouvoir utiliser |**BOX**, il faut bien entendu l'avoir installée, par exemple avec le chargeur Basic précédent.

RSX | PBOX

Comment utiliser la RSX

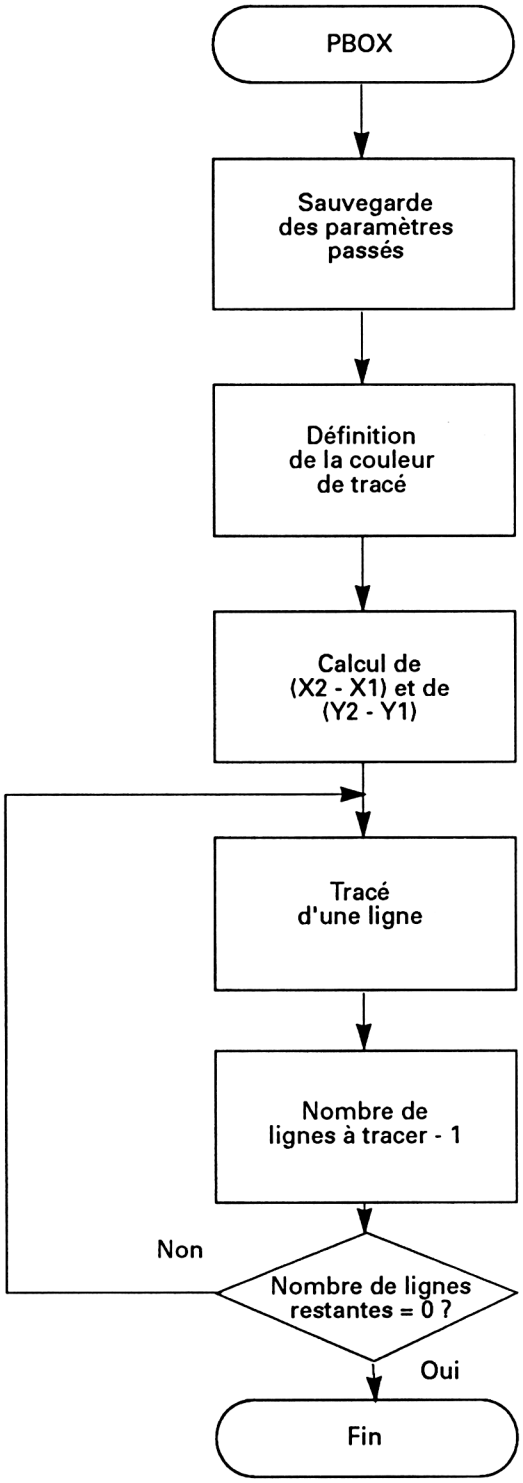
Le format de la RSX |**PBOX** est le même que celui de la RSX |**BOX** :

|BOX, X1, Y1, X2, Y2, C

où X1 et Y1 sont les coordonnées du coin supérieur gauche du rectangle, X2 et Y2 sont les coordonnées du coin inférieur droit du rectangle, et C est la couleur de tracé.

La RSX en détail

La logique de la RSX apparaît dans l'ordinogramme suivant :



La routine de traitement se trouve à l'étiquette **PBOX**. Les données qui lui sont passées sont stockées dans les variables **COUL** (couleur), **X1**, **Y1**, **X2** et **Y2** (coordonnées extrémales) entre les lignes 57 et 70.

La couleur de tracé est initialisée à l'aide de la macro **GRA SET PEN** du FIRMWARE (lignes 77 et 78).

Avant de commencer le tracé du rectangle, le programme calcule :

- La longueur du rectangle (lignes 84 à 88). A ce propos, remarquez l'instruction **XOR A** ligne 86 avant d'effectuer la soustraction. Cette instruction met à zéro l'éventuel indicateur de retenue dans le registre des indicateurs. Si cet indicateur était à un, la soustraction ne donnerait pas le résultat attendu.
- La hauteur du rectangle (lignes 90 à 94). La hauteur, c'est-à-dire le nombre de lignes élémentaires du rectangle, est stockée dans le registre B. Ce registre sera utilisé dans la boucle principale du programme pour décompter le nombre de lignes élémentaires tracées.

La boucle de tracé se trouve entre les lignes 98 et 110. La macro **GRA MOVE ABS** positionne l'origine du tracé, et la macro **GRA LINE REL** trace une ligne horizontale de longueur **OFFX** à partir de l'origine du tracé. Une instruction **DJNZ** décrémente le registre B et redonne le contrôle à l'étiquette **BIS** tant que B n'est pas nul, c'est-à-dire tant que tout le rectangle n'est pas tracé.

Le listing de la RSX est le suivant :

```
 1                          ORG  9000H
 2                          LOAD 9000H
 3                ;---------------------------------
 4                ; RSX PBOX
 5                ; Format : |BOX,X1,Y1,X2,Y2,C
 6                ; Entree : X1=Abs coin sup gauche
 7                ;          Y1=Ord coin sup gauche
 8                ;          X2=Abs coin inf droit
 9                ;          Y2=Ord coin inf droit
10                ;          C=Couleur
11                ; Sortie : Affichage du rectangle
12                ;---------------------------------
13                ;
14                ;
15                ;---------------------------------
16                ; Declaration des constantes
17                ; et des variables du programme
18                ;---------------------------------
19                ;
20                LOGEXT:     EQU  0BCD1H          ;KL LOG EXT
21                SETPEN:     EQU  0BBDEH          ;GRA SET PEN
22                MOVABS:     EQU  0BBC0H          ;GRA MOVE ABS
23                LINEREL:    EQU  0BBF9H          ;GRA LINE REL
24                BUF:        DS   4               ;ZONE RAM POUR LOG EXT
25 9004 0990      PTRTAB:     DW   TABLE           ;Pointeur TABLE
26 9006 C32390                JP   PBOX            ;Affichage du rectangle
27 9009 50424F    TABLE:      DB   "PBO"
```

```
28 900C D8                  DB    "X"+80H
29 900D 00                  DB    0                ;Fin de table
30              OFFX:       DS    2                ;Offset en X
31              X1:         DS    2                ;Abscisse coin sup gauch
32              Y1:         DS    2                ;Ordonnee coin sup gauch
33              X2:         DS    2                ;Abscisse coin inf droit
34              Y2:         DS    2                ;Ordonnee coin inf droit
35              COUL:       DS    1                ;Couleur de trace
36              ;
37              ;---------------------------------
38              ; Definition de la RSX
39              ;---------------------------------
40              ;
41              DEFRSX:     EQU   $                ;Point d'entree
42 9019 010490              LD    BC,PTRTAB        ;Ptr table definition
43 901C 210090              LD    HL,BUF           ;Buffer pour LOG EXT
44 901F CDD1BC              CALL  LOGEXT           ;Definition de la RSX
45 9022 C9                  RET
46              ;
47              ;-------------------
48              ; Traitement de BOX
49              ;-------------------
50              ;
51              PBOX:       EQU   $                ;Point d'entree
52              ;
53              ;- - - - - - - - - - - - - - - -
54              ; Lecture des donnees passees
```

```
55                ;- - - - - - - - - - - - - - - -
56                ;
57 9023 DD7E00              LD   A,(IX+0)
58 9026 321890              LD   (COUL),A          ;Couleur de trace
59 9029 DD6603              LD   H,(IX+3)
60 902C DD6E02              LD   L,(IX+2)
61 902F 221690              LD   (Y2),HL           ;Ord coin inf droit
62 9032 DD6605              LD   H,(IX+5)
63 9035 DD6E04              LD   L,(IX+4)
64 9038 221490              LD   (X2),HL           ;Abs coin inf droit
65 903B DD6607              LD   H,(IX+7)
66 903E DD6E06              LD   L,(IX+6)
67 9041 221290              LD   (Y1),HL           ;Ord coin sup gauche
68 9044 DD6609              LD   H,(IX+9)
69 9047 DD6E08              LD   L,(IX+8)
70 904A 221090              LD   (X1),HL           ;Abs coin sup gauche
71                ;
72                ;- - - - - - - - - - - - - - - - -
73                ; Initialisation de la couleur
74                ; du rectangle
75                ;- - - - - - - - - - - - - - - - -
76                ;
77 904D 3A1890              LD   A,(COUL)
78 9050 CDDEBB              CALL SETPEN            ;Couleur rectangle
79                ;
80                ;- - - - - - - - - - - - - - - -
81                ; Trace du rectangle plein
```

```
 82                ;- - - - - - - - - - - - - - -
 83                ;
 84 9053 2A1490                LD    HL,(X2)
 85 9056 ED5B1090              LD    DE,(X1)
 86 905A AF                    XOR   A
 87 905B ED52                  SBC   HL,DE             ;X2-X1
 88 905D 220E90                LD    (OFFX),HL         ;Offset en X
 89                ;
 90 9060 2A1690                LD    HL,(Y2)
 91 9063 ED5B1290              LD    DE,(Y1)
 92 9067 AF                    XOR   A
 93 9068 ED52                  SBC   HL,DE
 94 906A 45                    LD    B,L               ;Y2-Y1
 95                ;
 96 906B ED5B1090              LD    DE,(X1)
 97 906F 2A1290                LD    HL,(Y1)
 98                BIS:        EQU   $
 99 9072 C5                    PUSH  BC
100 9073 D5                    PUSH  DE
101 9074 E5                    PUSH  HL
102 9075 CDC0BB                CALL  MOVABS            ;Coin sup gauche
103 9078 ED5B0E90              LD    DE,(OFFX)
104 907C 210000                LD    HL,0
105 907F CDF9BB                CALL  LINEREL           ;Une ligne
106 9082 E1                    POP   HL
107 9083 D1                    POP   DE
108 9084 C1                    POP   BC
```

```
109 9085 23                     INC  HL
110 9086 10EA                   DJNZ BIS                ;Boucle
111                ;
112 9088 C9                     RET
113                             END
```

BUF	9000	BIS	9072	COUL	9018	DEFRSX	9019
LOGEXT	BCD1	LINEREL	BBF9	MOVABS	BBC0	OFFX	900E
PTRTAB	9004	PBOX	9023	SETPEN	BBDE	TABLE	9009
X1	9010	X2	9014	Y1	9012	Y2	9016

Comme toujours, voici le chargeur Basic correspondant :

```
1000 REM ------------------------------------------------------------------------
1010 REM Chargeur BASIC de la RSX de trace de rectangles pleins
1020 REM ------------------------------------------------------------------------
1030 REM
1040 FOR i=&9000 TO &9088
1050   READ a$
1060   a$="&"+a$
1070   POKE i,VAL(a$)
1080 NEXT i
1090 CALL &9019
1100 END
1110 REM - - - - - - - - - - - - - - - - - -
1120 REM Donnees du programme de trace
1130 REM - - - - - - - - - - - - - - - - - -
1140 REM
1150 DATA 0,0,0,0,9,90,C3,23,90,50,42,4F,D8,0,0,0
1160 DATA 0,0,0,0,0,0,0,0,0,1,4,90,21,0,90,CD
1170 DATA D1,BC,C9,DD,7E,0,32,18,90,DD,66,3,DD,6E,2,22
1180 DATA 16,90,DD,66,5,DD,6E,4,22,14,90,DD,66,7,DD,6E
1190 DATA 6,22,12,90,DD,66,9,DD,6E,8,22,10,90,3A,18,90
1200 DATA CD,DE,BB,2A,14,90,ED,5B,10,90,AF,ED,52,22,E,90
1210 DATA 2A,16,90,ED,5B,12,90,AF,ED,52,45,ED,5B,10,90,2A
1220 DATA 12,90,C5,D5,E5,CD,C0,BB,ED,5B,E,90,21,0,0,CD
1230 DATA F9,BB,E1,D1,C1,23,10,EA,C9,0,0,0,0,0,0,0
```

et les données de checksum qui permettent de vérifier l'exactitude des données entrées :

```
CB 15 47 9E 12 D1 6 45 13
```

Le petit programme de démonstration suivant montre comment utiliser la RSX |**PBOX** :

```
100 REM ------------------------------
110 REM Programme de demonstration
120 REM ------------------------------
130 REM
140 MODE 1
150 FOR I=1 TO 50
160   X1=INT(RND(1)*500)
170   Y1=INT(RND(1)*300)
180   X2=INT(RND(1)*600)
190   Y2=INT(RND(1)*400)
200   C=INT(RND(1)*3)+1
210   |PBOX,X1,Y1,X2,Y2,C
220 NEXT I
```