

# LE MICRO- A ECRAN

**CANON X 07 :  
BRANCHEZ VOTRE MICRO-ORDINATEUR  
SUR VOTRE TELEVISEUR.**

**IMPRESSIONNANT, LE CANON X 07 POUR UN MICRO-PORTABLE ! UNE INTERFACE OPTIONNELLE VOUS PERMET DE LE BRANCHER SUR VOTRE TELEVISEUR ET DE VISUALISER AINSI TOUTES LES OPERATIONS INSCRITES SUR VOTRE X 07.**

**MAIS LE CANON X 07 N'EST PAS SEULEMENT LE PREMIER MICRO-PORTABLE A ECRAN, IL EST AUSSI LE PREMIER MICRO-MULTICARTES.**

**SA FORCE ? DES PETITES CARTES EXTRAORDINAIRES POUR REALISER ET CONSERVER VOS PROPRES PROGRAMMES, COMME VOUS L'ENTENDEZ... A LA CARTE.**

**PRATIQUE, IL PARLE EN BASIC, LE LANGAGE ORDINATEUR FACILE A APPRENDRE.**

**AVEC SES NOMBREUSES CASSETTES ET CARTES A PROGRAMMES AUSSI ELABORES QUE LA GESTION DE STOCK, LA FACTURATION, LA PAYE, LE TABLEUR,... CANON X 07 A EGALLEMENT BIEN D'AUTRES ATOUTS.**

**GRACE A SES MULTIBRANCHEMENTS : MACHINE A ECRIRE, IMPRIMANTE, ORDINATEUR, MODEM ET MEME VOTRE TELEVISEUR... CE TOUT PETIT ORDINATEUR A TROUVE PLUS D'UN MOYEN POUR DEVENIR GRAND.**

**JE SOUHAITERAIS RECEVOIR VOTRE DOCUMENTATION COMPLETE SUR LE MICRO-ORDINATEUR X 07.**

**VOICI MON NOM, MON ADRESSE ET MON TELEPHONE :**

**NOM \_\_\_\_\_**

**SOCIETE \_\_\_\_\_**

**N° \_\_\_\_\_ RUE \_\_\_\_\_**

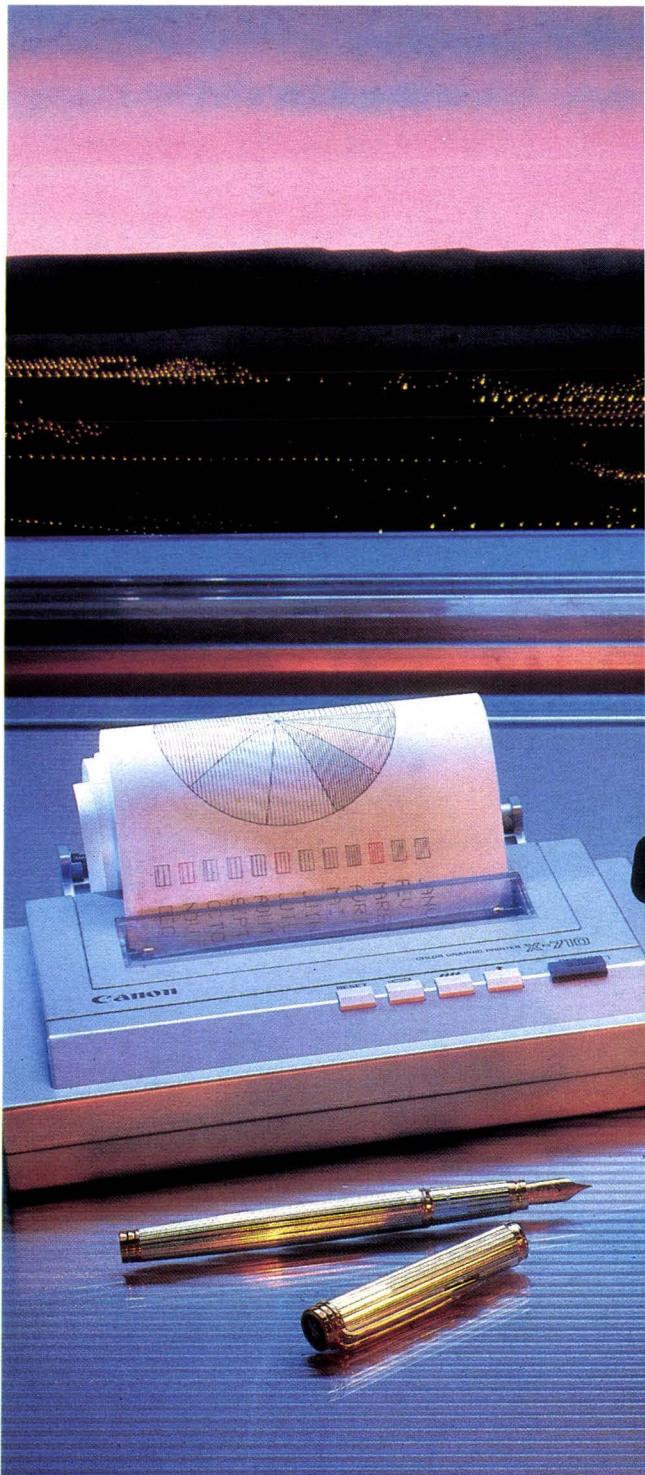
**VILLE \_\_\_\_\_**

**CODE POSTAL \_\_\_\_\_ TELEPHONE \_\_\_\_\_**

**DEMANDE D'INFORMATION A RENVOYER A CANON FRANCE,  
93154 LE BLANC-MESNIL CEDEX, TELEPHONE 865.42.23.**

## Canon

**CANON, HAUTE TECHNICITE, HAUTE SIMPLICITE**  
**CANON EST PRESENT AU SICOB : ZONE 4A, STAND 4101**



# POR~~T~~ABLE RAN.



# « Voyeur » pour Apple II

Quel drôle de nom pour un utilitaire. Et pourtant, avec « Voyeur », vous allez pouvoir localiser un programme ou tout autre fichier sur la disquette, changer le nom ou le type du programme autostart (binaire, texte), rendre des programmes « invisibles » lors d'un catalog, et bien d'autres choses...

Sachez toutefois que ce programme se limite au travail sur des disquettes personnelles et formatées en 16 secteurs. Nous avons volontairement évité de pouvoir lire sur des disquettes dont les octets de repérage adresses et données ont été modifiés, ce qui est pratiquement le cas de toutes les disquettes protégées du commerce. Il en va de même pour celles écrites par demi-piste.

Avant de dévoiler certains « trucs », nous allons commencer par expliquer le fonctionnement et l'utilisation du programme.

« Voyeur » repose sur l'utilisation de la routine RWTS (Read Write Track Sector) comprise dans le DOS. Pour plus d'informations concernant cette routine, il suffit de se reporter au manuel du DOS où elle est clairement décrite.

Le programme a été écrit en Basic car le cahier des charges ne nécessitait nullement la programmation en langage machine. Toutefois, nous avons eu recours à un petit sous-programme utilisant deux routines du moniteur qui, lui, sera en code de base.

Ces deux routines permettent de sortir sur écran le contenu de l'accumulateur du 6502 soit sous la forme de sa valeur hexadécimale, soit sous la forme du caractère correspondant. L'index utilisé dans cette routine est situé à l'adresse \$3CF.

La zone de travail est longue de 256 octets et débute à l'adresse \$2000 (HGR1).

C'est dans ce buffer que l'on va trouver les informations contenues sur le secteur en cours d'étude.

En ce qui concerne la table IOB dont a besoin la routine RWTS, elle débute à l'adresse \$2100 et est initialisée pour une lecture sur le drive I.

## Utilisation du « Voyeur »

Après le lancement, un menu s'affiche sur l'écran. L'utilisateur dispose alors de trois choix :

- l'utilitaire,
- le catalog,
- sortie du programme.

Sachez simplement que l'option « sortie » effectue un NEW, libérant ainsi la mémoire pour l'écriture d'un autre programme (mais détruisant de ce fait les lignes de « Voyeur » qui devront être rechargées pour une nouvelle utilisation).

Si vous avez demandé l'utilitaire, vous devez maintenant entrer les numéros de piste et de secteur sur lesquels vous désirez travailler.

Toute erreur est signalée par un bip sonore et entraîne un renouvellement de la demande. A tout moment, le fait d'entrer « Q » (Quitter) vous renvoie le menu précédent.

Lorsque vous avez entré piste et secteur, et que ceux-ci sont validés, le lecteur se met en route et l'écran vous renvoie l'état du couple piste/secteur sélectionné. Les chiffres en inversé sur la gauche représentent les numéros des octets de 0 à 255.

C'est à ce niveau que le programme devient intéressant. En effet, vous avez maintenant le choix entre dix possibilités qui sont activées par les touches suivantes :

- → : déplacement du curseur vers la gauche ;
- ← : déplacement du curseur vers la droite ;
- ↑ : déplacement du curseur vers le haut ;
- ↓ : déplacement du curseur vers le bas ;
- A : (alphanumérique) provoque l'impression des caractères interprétés ;
- H : (hexadécimal) permet

**UTILITAIRE :**  
**d'Arnaud HOULEMARE**  
**Si le manuel DOS 3.3 de l'Apple II est bien étudié, on peut aisément « savoir » où sont les informations intéressantes des disquettes (localisation des noms des programmes, de leur longueur...). Seulement savoir est peut-être bien... Mais pouvoir modifier, c'est beaucoup mieux ! Langages : Basic + Langage machine 6502 Ordinateurs : Apple II+ et IIE**

l'impression des octets en hexa ;

- espace : modification du contenu du secteur ;
- E : réécriture du secteur sur la disquette ;
- Q : retour au menu précédent ;
- S : travailler sur un autre secteur.

La modification (provoquée par une pression sur la barre d'espacement) s'effectue sur l'octet pointé par le curseur. Vous obtenez en haut de l'écran la mention « MODIFICATION : » puis la valeur actuelle de l'octet, une flèche, et le curseur clignotant. Vous devez alors entrer les deux caractères de votre nouvel octet. L'octet entré, une validation vous est demandée.

Si vous tapez « → », l'octet est validé, modifié en mémoire ainsi que sur l'écran, et vous passez à la modification de l'octet suivant. Si vous tapez « return », l'octet est validé, modifié, mais vous quittez le mode « modification ». Enfin, si vous tapez « N », l'octet n'est pas validé et vous devez entrer sa nouvelle valeur.

Lorsque vous choisissez « E » (mode écriture), la mention « ECRITURE : » apparaît en haut de l'écran suivie de :

« P:? »

Vous devez alors entrer le numéro de piste sur laquelle vous désirez écrire. Si vous souhaitez réécrire sur la même piste, faites simplement « return ». La procédure est similaire pour le choix du secteur.

Ce type de choix permet de mettre en « réserve » ou de déplacer un secteur sur la disquette.

Une fois les paramètres d'écriture saisis, il est nécessaire de valider afin d'éviter toute maladresse qui pourrait se révéler désastreuse. « return » valide et lance l'écriture alors

que « N » a pour effet de solliciter une nouvelle entrée de paramètres.

Vous savez maintenant tout sur le mode d'emploi de « Voyeur ». Il est grand temps de l'appliquer à des cas concrets. C'est ce que nous allons faire dès maintenant.

## La visualisation des secteurs du catalog

Sachez simplement que ces secteurs se situent piste \$11, secteurs \$F et suivants (\$E à \$1). Une lecture de secteurs sous le mode « A » visualise directement les secteurs avec les noms de programme en clair, alors que sous le mode « H » vous obtenez simplement les octets tels qu'ils sont inscrits sur la disquette (tout au moins après « denibbling » !!! mais ça, c'est une autre histoire...).

On retrouve d'ailleurs éventuellement des noms de programmes « fantômes ». En effet, lors d'un DELETE, le programme n'est pas réellement effacé, mais il est désactivé logiquement par l'écriture d'un octet \$FF dans l'octet de type de programme et ses secteurs sont remis à disposition du système d'exploitation.

## Pour rendre un programme invisible...

Pour l'exemple il vous faudra initialiser une disquette sous le nom « I ». Le catalog est donc :

DISK VOLUME 254  
A 002 I

La lecture de la piste \$11 secteur \$F sous le mode « A » vous permet de situer l'écriture du « I ». Pour « effacer » le programme du catalog, il suffit

d'écrire à la suite du nom de programme « n » fois \$88 (n est égal à 7 plus la longueur du nom à effacer). Dans notre exemple, la longueur du nom « I » est 1, donc n = 7 + 1 = 8.

Cette modification effectuée, réécrivez le secteur et faites un catalog de votre disquette. Le programme « I » a disparu !

Attention, le programme existe toujours mais il n'est plus accessible par un simple LOAD ou RUN.

### *... ou changer de programme autostart*

Vous avez peut-être déjà fait un RENAME du programme autostart. Le résultat est implausible : la disquette refuse de « BOOTER » le nouveau programme. Le message d'erreur qui découle est FILE NOT FOUND ou alors, dans certains cas, on récupère la main sans qu'aucun programme n'ait été chargé.

Cette situation s'explique très bien, en effet, lors de l'initialisation d'une disquette, le nom du programme autostart est stocké sur les pistes du

| NOMENCLATURE DES VARIABLES : |   |
|------------------------------|---|
| IOB                          | ADRESSE DE LA TABLE IOB                               |
| SB                           | ADRESSE DU SOUS PROGRAMME MACHINE                     |
| R\$                          | REPONSE A UNE QUESTION ET VARIABLE D'ENTREE DE CONVER |
| R                            | VAL (R\$)   |
| C                            | VARIABLE DE RETOUR DU SOUS-P CONVER                   |
| P                            | NUMERO DE PISTE                                       |
| P1                           | N° PISTE TRANSITOIRE                                  |
| P\$                          | VARIABLE D'ENTREE POUR LE N° DE PISTE                 |
| S                            | NUMERO DE SECTEUR                                     |
| S1                           | N° SECTEUR TRANSITOIRE                                |
| S\$                          | VARIABLE D'ENTREE POUR LE N° DE SECTEUR               |
| B                            | POINTEUR DU BUFFER                                    |
| B1                           | POINTEUR TRANSITOIRE                                  |
| BUF                          | ADRESSE DU BUFFER                                     |
| VT                           | VALEUR DE VTAB  |
| V1                           | VTAB TRANSITOIRE                                      |
| HT                           | VALEUR DE HTAB  |
| H1                           | HTAB TRANSITOIRE                                      |
| MOD                          | FLAG DE MODIFICATION                                  |
| M\$                          | VALEUR INTERMEDIAIRE DE MODIFICATION                  |
| N,I                          | VARIABLES DE BOUCLES                                  |

```

0300- AE CF 03 LDX $03CF
0303- BD 00 20 LDA $2000,X
0306- 20 DA FD JSR $FDDA
0309- 60 RTS
030A- AE CF 03 LDX $03CF
030D- BD 00 20 LDA $2000,X
0310- 20 F0 FD JSR $FDF0
0313- 60 RTS

```

*La seule routine en langage machine est destinée à l'emploi de routines du moniteur.*

DOS. Or, à l'occasion d'un RENAME, seul est modifié le nom sur le catalog.

Ici encore, « Voyeur » va nous aider. D'après la loi de Murphy, vous aurez forcément oublié quel était le nom du programme autostart avant le RENAME, mais si l'on sait que le nom du programme autostart est écrit piste 1 secteur 9 à partir de l'octet numéro 117, la situation perd de son drame.

Reprenons la disquette précédente : si l'on écrit à partir de l'octet 118 du secteur 9 de la piste 1 les 8 octets \$88, notre disquette « BOOTERA » sur notre programme invisible.

Voilà quelques exemples d'utilisation de « Voyeur ». Cette liste n'est nullement exhaustive et l'auteur l'utilise dans bien d'autres domaines. Les utilisations ne dépendent en fait que de l'imagination de celui qui l'utilise.

Nous espérons que vous prendrez autant de plaisir à l'utiliser que nous en avons pris nous même à le concevoir. Et si vous vous décidez à partir à la chasse des secteurs perdus... Bonne chasse ! ■

```

0 REM *** VOYEUR ***
10 REM *** INITI ***
15 IOB = 8448:I$ = "H":SB = 767
20 RESTORE :D$ = CHR$(4)
25 PRINT D$;"PR£0"
30 FOR N = IOB TO IOB + 35
35 READ I: POKE N,I: NEXT N
40 FOR N = SB TO SB + 20
45 READ I: POKE N,I: NEXT N
50 DATA 169,33,160,10,32,217,3,96
55 DATA 0,0,1,96,1,0,0,0,32,33,0,32
60 DATA 0,0,1,0,0,96,1,0,0,0,0,0,0,1,239,216,
65 DATA 174,207,3,189,0,32,32,218,253,96,174,207,3,189,0,32,32,240,253,9
   6
100 REM *** MENU ***
105 NORMAL : HOME
110 PRINT "*** VOYEUR ***": PRINT : PRINT "copyright 1984 Arnaud HOULEM
ARE"
115 PRINT : FLASH : PRINT "PLACER LA DISQUETTE SUR LAQUELLE ON VEUT TRAVA
ILLER DANS LE DRIVE 1": NORMAL
120 PRINT : PRINT : PRINT " 1: LECTURE/ECRITURE": PRINT
125 PRINT " 2: CATALOG": PRINT
130 PRINT " 3: SORTIR": PRINT
135 VTAB 22: PRINT "ENTRER VOTRE CHOIX:"; GET R$: PRINT
140 R = VAL (R$): IF R = 0 OR R > 3 THEN 135
145 ON R GOTO 200,1000,5000
200 REM *** LECTURE ***
205 HOME : INVERSE
210 PRINT "RETOUR AU MENU: Q" : PRINT
215 R$ = ""

```

*Listing du programme Basic.*

```

220 INPUT "ENTRER LE NO DE PISTE EN HEXA:";R$
225 IF R$ = "" THEN C = 100: GOTO 240
230 IF R$ = "Q" THEN GOTO 100
235 GOSUB 10000
240 IF C > 34 THEN VTAB 4: HTAB 1: PRINT "";: GOTO 220
245 LET P = C: LET P$ = R$
250 IF R$ = "Q" THEN 120
255 NORMAL : PRINT : PRINT : INVERSE
260 INPUT "ENTRER LE NO DE SECTEUR EN HEXA:";R$
265 IF R$ = "" THEN C = 100: GOTO 280
270 IF R$ = "Q" THEN GOTO 100
275 GOSUB 10000
280 IF C > 16 THEN VTAB 7: HTAB 1: PRINT "";: GOTO 260
285 LET S = C: LET S$ = R$
290 R$ = ""
295 NORMAL : VTAB 22: PRINT "VALIDATION?";: GET R$: PRINT : IF R$ = "" THEN
295
300 IF R$ = "N" THEN 200
305 POKE 8462,P: POKE 8463,S
310 POKE 8470,1
315 CALL 8448
320 HOME
325 B = 0:BUF = 8192:VT = 2:HT = 5
330 VTAB 2: HTAB 1: INVERSE : PRINT "0": NORMAL
335 LET VA = PEEK (BUF + B)
340 IF B > 255 THEN 400
345 IF HT > 40 THEN VT = VT + 1: VTAB VT: HTAB 1: INVERSE : PRINT B: NORMAL
:HT = 5
350 VTAB VT: HTAB HT
355 POKE 975,B
360 IF I$ = "H" THEN CALL 768
365 IF I$ = "A" THEN CALL 778
370 B = B + 1:HT = HT + 3
375 GOTO 335
400 REM ** SUITE **
405 VT = 2:HT = 4:B = 0
410 VTAB VT: HTAB HT
415 INVERSE : PRINT ">";: NORMAL : HTAB HT + 3
420 VTAB 1: HTAB 32: PRINT "P:";P$;: HTAB 37: PRINT "S:";S$;
425 V1 = VT:H1 = HT
430 R$ = ""
435 GET R$: IF R$ = "" THEN 435
440 R = ASC (R$)
445 IF R = 21 THEN H1 = HT + 3:BI = B + 1: GOTO 500
450 IF R = 8 THEN H1 = HT - 3:BI = B - 1: GOTO 500
455 IF R = 11 THEN V1 = VT - 1:BI = B - 12: GOTO 500
460 IF R = 10 THEN V1 = VT + 1:BI = B + 12: GOTO 500
465 IF R = 65 THEN LET I$ = "A": GOTO 320
470 IF R = 72 THEN LET I$ = "H": GOTO 320
475 IF R = 83 THEN 200
480 IF R = 81 THEN 100
485 IF R = 32 THEN MOD = 1: GOTO 500
490 IF R = 69 THEN GOSUB 800
495 GOTO 430
500 IF BI > 255 OR BI < 0 THEN 410
505 VTAB VT: HTAB HT: PRINT " "
510 IF H1 > 39 THEN V1 = V1 + 1:H1 = 4
515 IF H1 < 4 THEN V1 = V1 - 1:H1 = 37
520 B = BI:HT = H1:VT = V1
525 IF MOD = 1 THEN GOSUB 600
530 GOTO 410
600 REM *** MODIF ***
605 VTAB VT: HTAB HT: INVERSE : PRINT ">": NORMAL
610 VTAB 1: HTAB 1: PRINT "MODIFICATION: ";
615 POKE 975,B: CALL 768: VTAB 1: HTAB 17: PRINT "-> ";
620 M$ = ""

```

```

625 FOR N = 1 TO 2
630 R$ = ""
635 HTAB 18 + N: GET R$:R = ASC (R$): IF R < = 47 OR R = > 71 OR R =
    > 58 AND R < = 64 THEN PRINT ""; GOTO 630
640 M$ = M$ + R$
645 HTAB 18 + N: PRINT R$;
650 NEXT N
655 PRINT " VALID.";
660 R$ = ""
665 GET R$: IF R$ = "" THEN 665
670 IF R$ = "N" THEN 610
675 IF ASC (R$) = 21 THEN R1 = 1
680 VTAB 1: HTAB 1: PRINT "
685 R$ = M$: GOSUB 10000
690 VTAB VT: HTAB HT + 1: IF I$ = "H" THEN PRINT M$
695 IF I$ = "A" THEN PRINT CHR$ (C)
700 POKE BUF + B,C
705 IF R1 = 1 THEN R1 = 0:R = 21: GOTO 445
710 MOD = 0: RETURN
800 REM *** ECRIT ***
805 VTAB 1: HTAB 1: PRINT "ECRITURE";
810 P1 = P:S1 = S
815 HTAB 10: PRINT "P:";: INPUT R$: IF R$ = "" THEN P1$ = P$: GOTO 840
820 IF R$ = "Q" THEN 915
825 GOSUB 10000
830 IF C = 999 THEN 805
835 LET P1 = C: LET P1$ = R$
840 VTAB 1: HTAB 12: PRINT P1$;" "
845 VTAB 1: HTAB 15: PRINT "S:";: INPUT R$: IF R$ = "" THEN S1$ = S$: GOTO
    870
850 IF R$ = "Q" THEN 915
855 GOSUB 10000
860 IF C = 999 THEN 845
865 LET S1 = C: LET S1$ = R$
870 VTAB 1: HTAB 17: PRINT S1$;" ";
875 HTAB 21: PRINT "VALID.";
880 GET R$: IF R$ = "" THEN 880
885 IF R$ = "N" THEN 805
890 IF R$ = "Q" THEN 915
895 P = P1:S = S1:P$ = P1$:S$ = S1$
900 POKE 8470,2: POKE 8462,P: POKE 8463,S
905 CALL 8448
910 POKE 8470,1
915 VTAB 1: HTAB 1: PRINT "
920 RETURN                                ";: HTAB 39
1000 REM *** CATALOG ***
1005 HOME
1010 PRINT : PRINT CHR$ (4); "CATALOG"
1015 PRINT : PRINT "FRAPPER UNE TOUCHE...";: R$ = ""
1020 GET R$: IF R$ = "" THEN 1020
1025 GOTO 100
1030 END
5000 NEW
10000 REM *** CONVER ***
10010 LET C = 0
10020 FOR N = LEN (R$) TO 1 STEP - 1
10030 LET O = ASC (MID$ (R$,N,N))
10040 IF O > 64 AND O < 71 THEN C = C + (O - 55) * 16 ^ (LEN (R$) - N):
    GOTO 10060
10050 IF O < 58 AND O > 47 THEN C = C + (O - 48) * 16 ^ (LEN (R$) - N):
    GOTO 10060
10055 LET C = 999: RETURN
10060 NEXT N
10070 RETURN

```

SPR£0

# Un jeu d'aventure pour le Canon X07

Le jeu proprement dit est un parcours dans un labyrinthe où sont répartis divers monstres et armes. Le joueur, qui ne voit ce labyrinthe qu'à travers une fenêtre de quatre cases sur quatre doit parvenir à un trésor (placé aléatoirement) et le ramener à son point de départ.

Écrit essentiellement en langage machine, ce logiciel intègre toutefois quelques lignes en Basic, dont le rôle est de créer les caractères spéciaux utilisés (pour représenter les différents éléments du labyrinthe) ainsi que de dessiner le labyrinthe lui-même et d'implanter le programme binaire (les DATAs des lignes 240 à 264).

De même, la scrutation du clavier étant assez délicate sur le Canon X07, il a semblé plus simple de contrôler les touches pressées par un sous-programme Basic qui exploite la variable système d'adresse  $1A28_H$ .

## Le jeu

Le jeu en lui-même consiste à se déplacer dans un labyrinthe pour trouver un trésor, puis à ressortir dudit labyrinthe. Sur l'écran, vous ne vous déplacerez pas : c'est le labyrinthe qui bougera autour de vous ; en effet seule une fenêtre de  $4 \times 4$  du labyrinthe – qui, lui, fait  $25 \times 25$  – est visible. Durant ces déplacements, vous pourrez rencontrer divers obstacles. Tout d'abord un mur : dans ce cas, vous ne pourrez plus avancer et serez obligé de choisir une autre direction (on se déplace grâce aux quatre touches du curseur). Vous pourrez également rencontrer un espace, il sera possible alors d'avancer sans problème et un point sera affiché à votre ancienne position, ce qui vous permettra de savoir où vous êtes passé. Vous pourrez tout aussi facilement repasser sur un point. De toute manière, avancer vous coûtera un pas. Tout au long de la partie vous verrez affiché en bas à droite le nombre de pas qu'il vous reste pour trouver le trésor et ressortir.

Ce nombre de pas, fixé au début du jeu, dépend de la distance qu'il y a entre le trésor et vous.

Durant les déplacements, vous serez confronté à divers éléments. Tout d'abord, vous pourrez trouver une épée.

Si vous en possédez déjà une, aucun résultat ne sera à attendre. Par contre, si vous tentez de « marcher » dans une case contenant une arme, vous la prendrez et cette case deviendra inaccessible (cette action n'est réalisée que si vous n'êtes pas en possession du trésor : si c'est le cas, ce trésor remplace l'épée).

Vous pourrez faire de plus fâcheuses rencontres : un « Mazog ». Le fait de tenter d'occuper la case d'un mazog entraînera un combat. Si vous possédez une épée, le monstre sera anéanti sans problème (mais vous perdrez aussi votre épée). Par contre, si vous n'êtes pas armé, votre décès sera garanti et un message laconique s'affichera en guise d'oraison funèbre.

Notons que le fait de détruire un mazog rapporte 10 points, ce qui est gratifiant pour le score (et remonte toujours le moral lorsqu'on a perdu).

Enfin, vous pourrez trouver le trésor. Pour vous en emparer, il vous suffira de tenter d'occuper la case dans laquelle il se trouve. Notons dans ce cas que si vous tenez une épée, celle-ci prendra la place du trésor.

A partir du moment où celui-ci est acquis, il vous faudra retourner à votre point de départ.

Mais pour l'ensemble de la partie, un crédit de « pas » est calculé par le programme. Et si le parcours est trop long ou que la recherche a été hésitante, la

**JEU :  
Mazog  
d'Alain RITOUX**  
**Perdu dans un labyrinthe, saurez-vous y retrouver le trésor et le ramener à votre point de départ sans être dévoré par les divers monstres qui y rôdent ?**  
**Langages: Basic + code machine Z 80**  
**Ordinateur : Canon X07**

nullité du compteur de pas (en bas de l'affichage) entraînera votre décès pour cause de famine.

Lorsque toutes ces embûches auront été contournées, votre victoire sera confirmée par un petit message de félicitations.

## Le programme

Nous l'avons vu plus haut, une partie du programme est écrite en Basic (**tableau 1**). L'utilisation de ce langage, somme toute plutôt lent, est destinée à simplifier le travail d'animation.

Ainsi, les caractères du Canon X07 pouvant être redéfinis par une instruction spéciale (FONTS), les personnages peuvent être manipulés simplement

sous la forme d'un ou plusieurs codes.

L'utilisation s'en fait comme suit :

FONTS (code) : «  $\ell_1, \ell_2, \ell_3, \ell_4, \ell_5, \ell_6, \ell_7, \ell_8$  ».

La valeur **code** représente le code du caractère à redéfinir et les valeurs  $\ell_1, \dots, \ell_8$ , les codes des huit lignes symbolisant la nouvelle forme du caractère.

Un autre rôle du programme est d'affecter au trésor l'une de ses 5 positions potentielles.

Notons que les amateurs de langage machine (**tableau 2**) pourront remplacer cette partie par une routine donnant n'importe quelle position au trésor... ce qui compliquera le jeu lorsque vous en serez devenu le maître. ■

### La structure interne du programme Basic

|                       |  |
|-----------------------|--|
| <b>Ligne 1-14</b>     | Présentation du programme.   |
| <b>Lignes 15-190</b>  | Redéfinition des caractères graphiques.  |
| <b>Lignes 199-225</b> | Ensemble des DATAs contenant le labyrinthe.  |
| <b>Ligne 230-264</b>  | Ensemble des DATAs contenant le programme en langage machine                                       |
| <b>Ligne 300</b>      | Implantation du labyrinthe en mémoire.   |
| <b>Ligne 301</b>      | Implantation du programme en langage machine.  |
| <b>Lignes 302-309</b> | Initialisation de la position du trésor.   |
| <b>Lignes 320-330</b> | Test du clavier + appel à la routine en langage machine.   |
| <b>Lignes 350-372</b> | Affichage des différents messages.   |
| <b>Lignes 400</b>     | DATA contenant les différentes positions du trésor, ainsi que le nombre de pas crédités au joueur. |

Tableau 1

### La structure interne du programme en langage machine

|  |   |
|--|---|
| <b>Adresses</b>                          |   |
| <b>1A00<sub>H</sub>-1A02<sub>H</sub></b> | Variable indiquant la position du personnage.                                   |
| <b>1A02<sub>H</sub>-1A27<sub>H</sub></b> | Affichage de la partie visible du labyrinthe.                                   |
| <b>1A28<sub>H</sub></b>                  | Variable donnant le code de la dernière touche pressée                          |
| <b>1A29<sub>H</sub></b>                  | Variable indiquant l'état du personnage (armé, porteur du trésor).              |
| <b>1A2A<sub>H</sub>-1A4C<sub>H</sub></b> | Calcul du déplacement du personnage.  |
| <b>1A4F<sub>H</sub>-1A5A<sub>H</sub></b> | Routine sonore effectuant aussi la récupération du code du caractère rencontré. |
| <b>1A5B<sub>H</sub>-1A7D<sub>H</sub></b> | Affichage de la trace du joueur (point).  |
| <b>1A7F<sub>H</sub>-1A83<sub>H</sub></b> | Sous-programme sonore.  |
| <b>1A84<sub>H</sub>-1A85<sub>H</sub></b> | Variable comptant les pas restant au joueur.                                    |
| <b>1A86<sub>H</sub>-1A8A<sub>H</sub></b> | Traitement de la rencontre avec un mur.   |
| <b>1AA7<sub>H</sub>-1AAE<sub>H</sub></b> | Fin de la routine de calcul du déplacement.                                     |

Tableau 2

```

1 REM * MAZOG *
*(C) 1984 By* * A. RITOUX *
2 REM * Version *
   *Canon X-07*
4 Q=RND(0)
13 CLS:CLEAR 50,&H19FF :DEFINTA-X
14 PRINT" Veuillez patienter Je rentre
le laby- rinthe . Merci ."
15 REM ****
16 REM * DEF CHR$ *
17 REM ****
20 FONT$(&H81)="0,0,0,0,0,0,0,0" :FONT$(&
H82)="255,255,255,255,255,255,255,255"
30 FONT$(&H84)="255,128,128,128,188,128,
128,255"
40 FONT$(&HE4)="255,4,4,68,244,68,4,255"
50 FONT$(&H88)="255,128,188,168,180,188,
128,255"
60 FONT$(&HE8)="255,4,244,180,86,244,4,2
55"
70 FONT$(&H80)="0,32,92,24,12,80,32,0" :F
ONT$(&HE0)="0,32,208,192,192,40,16,0"
80 FONT$(&H83)="0,0,0,32,0,0,0,0" :FONT$(
&HE3)="0,0,0,32,0,0,0,0"
90 FONT$(&HE1)="0,0,0,0,0,0,0,0" :FONT$(&
HE2)="255,255,255,255,255,255,255,255"
100 FONT$(&H90)="0,4,4,4,28,4,8,8" :FONT$(
&HF0)="0,128,128,0,192,0,128,128"
110 FONT$(&H91)="0,4,4,4,28,4,8,8" :FONT$(
&HF1)="0,128,160,32,224,32,128,128"
190 FONT$(&H92)="0,4,4,4,60,60,4,8" :FONT$(
&HF2)="0,128,128,0,224,224,0,128"
198 REM ****
199 REM*LABYRINTHE*
200 REM ****
201 DATA2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,
2,2,2,2,2,2,2,2
202 DATA2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,4,
2,2,2,2,2,2,2,2
203 DATA2,2,1,1,1,2,2,2,2,1,1,1,1,1,1,
1,1,1,1,2,0,1,1,2
204 DATA2,2,2,2,1,2,1,1,1,2,1,2,2,2,2,2,
2,1,2,1,2,1,2,1,2
205 DATA2,2,1,1,1,2,1,2,1,1,1,2,1,1,1,1,
2,1,2,1,2,1,2,1,2
206 DATA2,2,0,2,2,2,1,2,4,2,1,2,1,2,1,0,
1,1,2,1,2,1,1,2,2
207 DATA2,2,1,1,1,1,1,1,1,1,2,1,2,2,2,1,
1,2,2,1,2,2,1,2,2
208 DATA2,2,1,2,2,2,0,2,2,2,2,2,1,1,2,2,
```

```

2,4,1,1,1,0,1,1,2
209 DATA2,2,4,1,1,1,1,2,1,1,1,2,2,1,0,2,
2,1,2,2,2,2,2,2,2
210 DATA2,2,2,2,2,2,2,2,1,2,1,1,2,2,1,1,
2,1,1,1,1,1,2,4,2
211 DATA2,2,2,1,1,1,1,1,1,2,2,0,1,2,2,1,
1,1,2,1,2,1,2,1,2
212 DATA2,2,4,1,2,2,2,1,2,2,4,2,1,1,2,2,
2,4,1,1,2,1,2,0,2
213 DATA2,2,1,1,2,4,2,0,2,1,1,1,2,0,1,1,
1,1,1,2,2,0,2,1,2
214 DATA2,2,2,1,2,1,2,1,2,1,2,1,4,2,2,0,
2,2,2,1,1,1,2,1,2
215 DATA2,2,2,0,1,1,2,1,1,1,2,1,4,1,2,1,
4,1,2,1,2,2,2,1,2
216 DATA2,2,2,2,2,2,2,2,2,2,2,1,1,1,2,1,
2,1,1,1,1,1,1,1,2
217 DATA2,2,1,0,1,1,2,2,2,1,2,2,2,2,2,1,
2,2,2,1,2,2,2,1,2
218 DATA2,2,1,2,2,2,2,2,1,1,1,1,1,1,1,1,
1,1,1,2,2,1,1,1,2
219 DATA2,2,1,0,1,2,1,1,2,2,1,2,1,2,2,2,
2,2,1,2,2,1,2,2,2
220 DATA2,2,0,2,2,2,1,2,1,1,1,2,1,2,1,1,
1,1,1,1,1,1,2,1,2
221 DATA2,2,1,2,1,1,1,1,1,2,2,2,1,2,1,2,
2,2,1,2,1,2,1,1,2
222 DATA2,2,1,1,1,1,2,2,2,2,2,1,1,1,2,1,2,
1,1,1,2,1,2,0,2,2
223 DATA2,2,2,2,1,2,1,1,0,1,1,2,2,2,1,2,
0,2,2,2,1,2,1,2,2
224 DATA2,2,1,0,1,2,1,2,2,2,1,1,1,2,1,1,
1,2,1,1,1,4,1,1,2
225 DATA2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,
2,2,2,2,2,2,2,2,2
230 REM ****
231 REM *CODES L/M*
232 REM ****
240 DATA2E,1E,2A,0,1A,11,CC,FF,19,11,15,
0,E,1,6,1,ED,43,B8,0,6,4,7E,EF,C6,60,EF
242 DATA23,10,F8,19,C,3E,5,B9,20,E9,C9,0
,0,0,1,11,0,0,3A,28,1A,FE,1,20,3,11,E7
244 DATAFF,FE,3,20,3,11,1,0,FE,5,20,3,11
,19,0,FE,7,20,3,11,FF,FF,C3,A6,1A,CD,7F
246 DATA1A,CD,2A,1A,2A,0,1A,E5,19,7E,1F,
30,28,D1,EB,ED,A0,1B,2B,36,83,EB,22,0
248 DATA1A,2A,84,1A,2B,22,84,1A,7C,B5,20
,5,3E,3,32,29,1A,CD,2,1A,18,7F,3E,7,EF
250 DATA1C,0,0,0,1F,30,3,D1,18,72,1F,30,
20,EB,E1,CB,46,20,69,3E,82,CB,4E,28,2

```

```

252 DATA3E,88,36,91,EB,77,CD,02,1A,18,58
,7A,B3,C0,0,0,D1,18,50,0,1F,30,13,EB,E1
254 DATA3E,82,CB,46,28,2,3E,84,36,92,EB,
77,CD,2,1A,18,39,D1,22,0,1A,1A,CB,47,20
256 DATA5,3E,3,32,29,1A,3E,83,12,CD,2,1A
,CD,DE,1A,18,20,1E,20,16,80,21,3,5,22
258 DATAB8,0,7A,EF,C6,60,EF,3E,FF,D3,F4,
6,FF,10,FE,AF,D3,F4,7A,EE,10,57,18,2,18
260 DATA17,6,FF,10,FE,1D,20,DB,2A,0,1A,3
6,90,11,A,0,2A,84,1A,19,22,84,1A,C9,21
262 DATA4,D,22,B8,0,2A,84,1A,CD,98,BB,3E
,20,EF,2A,0,1A,CB,4E,28,F,11,2E,1E,A7
264 DATAED,52,7C,B5,20,5,3E,2,32,29,1A,C
9
300 FOR A=&H1D00 TO &H1F70:READ B:POKE A
,B+&H80 :NEXT
301 FOR A=&H1A00 TO &H1B3C:READ Z$:POKE
A,VAL("&H"+Z$):NEXT
302 RESTORE 400:DIM X(5,2):FOR A=1 TO 5:
READ X(A,1),X(A,2):POKE X(A,1),129:NEXT
303 G=INT(RND(0)*4)+1:POKE X(G,1),&H88
304 STE=X(G,2)*5:POKE&H1A84,STEMOD256:PO
KE&H1A85,STE\256:POKE&H1E2E,&H90
308 CLS:FORA=0TO3:LOCATE8,A:PRINT"!";:NE
XT:EXEC&H1A02
309 LOCATE12,0:PRINT"MAZOG";:LOCATE12,2:
PRINT"STEPS:";
320 Q=STICK(0):POKE&H1A28,Q:EXEC&H1A4F
330 ON PEEK(&H1A29) GOTO 320,350,370
350 CLS:PRINT"* Bravo vous avez ** reus
si a sortir ** du labyrinthe . *";
352 PRINT"* Vous voila riche *";
353 IFINKEY$=""THEN353ELSEPAS=256*PEEK(&
H1A85)+PEEK(&H1A84)
354 PRINT"* Il vous reste : *"INT((PAS/
STE)*1000)/10" % de vos pas"
360 LINEINPUT"Voulez-vous essayer de fai
re mieux ?";Z$
363 IFZ$<>"N"THENRUNELSEEND
370 CLS:PRINT" Vous etes decede . Domma
ge dommage... "
371 LINEINPUT" Desirez-vous jouer a nou
veau ?";Z$
372 IFZ$<>"N"THENRUNELSEEND
400 DATA7476,62,7547,74,7829,52,8005,44,
8022,42

```

|                   |                   |
|-------------------|-------------------|
| 1A02 LD hl,(1A00) | 1A59 ADD hl,de    |
| 1A05 LD de,FFCC   | 1A5A LD a,(hl)    |
| 1A08 ADD hl,de    | 1A5B RRA          |
| 1A09 LD de,0015   | 1A5C JR NC,1A86   |
| 1A0C LD c,01      | 1A5E POP de       |
| 1A0E LD b,01      | 1A5F EX de,hl     |
| 1A10 LD (00B8),bc | 1A60 LDI          |
| 1A14 LD b,04      | 1A62 DEC de       |
| 1A16 LD a,(hl)    | 1A63 DEC hl       |
| 1A17 RST 28       | 1A64 LD (hl),83   |
| 1A18 ADD a,60     | 1A66 EX de,hl     |
| 1A1A RST 28       | 1A67 LD (1A00),hl |
| 1A1B INC hl       | 1A6A LD hl,(1A84) |
| 1A1C DJNZ 1A16    | 1A6D DEC hl       |
| 1A1E ADD hl,de    | 1A6E LD (1A84),hl |
| 1A1F INC c        | 1A71 LD a,h       |
| 1A20 LD a,05      | 1A72 OR l         |
| 1A22 CP c         | 1A73 JR NZ,1A7A   |
| 1A23 JR NZ,1A0E   | 1A75 LD a,03      |
| 1A25 RET          | 1A77 LD (1A29),a  |
| 1A26 NOP          | 1A7A CALL 1A02    |
| 1A27 NOP          | 1A7D JR 1AFE      |
| 1A28 NOP          | 1A7F LD a,07      |
| 1A29 NOP          | 1A81 RST 28       |
| 1A2A LD de,0000   | 1A82 RET          |
| 1A2D LD a,(1A28)  | 1A83 NOP          |
| 1A30 CP 01        | 1A84 NOP          |
| 1A32 JR NZ,1A37   | 1A85 NOP          |
| 1A34 LD de,FFE7   | 1A86 RRA          |
| 1A37 CP 03        | 1A87 JR NC,1A8C   |
| 1A39 JR NZ,1A3E   | 1A89 POP de       |
| 1A3B LD de,0001   | 1A8A JR 1AFE      |
| 1A3E CP 05        | 1A8C RRA          |
| 1A40 JR NZ,1A45   | 1A8D JR NC,1AAF   |
| 1A42 LD de,0019   | 1A8F EX de,hl     |
| 1A45 CP 07        | 1A90 POP hl       |
| 1A47 JR NZ,1A4C   | 1A91 BIT 0,(hl)   |
| 1A49 LD de,FFFF   | 1A93 JR NZ,1AFE   |
| 1A4C JP 1AA6      | 1A95 LD a,82      |
| 1A4F CALL 1A7F    | 1A97 BIT 1,(hl)   |
| 1A52 CALL 1A2A    | 1A99 JR Z,1A9D    |
| 1A55 LD hl,(1A00) | 1A9B LD a,88      |
| 1A58 PUSH hl      | 1A9D LD (hl),91   |

Fig. 2. – La routine en langage machine est fournie ici pour les amateurs désirant en utiliser toutes les finesse.

|      |              |
|------|--------------|
| 1A9F | EX de,hl     |
| 1AA0 | LD (hl),a    |
| 1AA1 | CALL 1A02    |
| 1AA4 | JR 1AFE      |
| 1AA6 | LD a,d       |
| 1AA7 | OR e         |
| 1AA8 | RET NZ       |
| 1AA9 | NOP          |
| 1AAA | NOP          |
| 1AAB | POP de       |
| 1AAC | JR 1AFE      |
| 1AAE | NOP          |
| 1AAF | RRA          |
| 1AB0 | JR NC,1AC5   |
| 1AB2 | EX de,hl     |
| 1AB3 | POP hl       |
| 1AB4 | LD a,82      |
| 1AB6 | BIT 0,(hl)   |
| 1AB8 | JR Z,1ABC    |
| 1ABA | LD a,84      |
| 1ABC | LD (hl),92   |
| 1ABE | EX de,hl     |
| 1ABF | LD (hl),a    |
| 1AC0 | CALL 1A02    |
| 1AC3 | JR 1AFE      |
| 1AC5 | POP de       |
| 1AC6 | LD (1A00),hl |
| 1AC9 | LD a,(de)    |
| 1ACA | BIT 0,a      |
| 1ACC | JR NZ,1AD3   |
| 1ACE | LD a,03      |
| 1AD0 | LD (1A29),a  |
| 1AD3 | LD a,83      |
| 1ADS | LD (de),a    |
| 1AD6 | CALL 1A02    |
| 1AD9 | CALL 1ADE    |
| 1ADC | JR 1AFE      |
| 1ADE | LD e,20      |
| 1AE0 | LD d,80      |
| 1AE2 | LD hl,0503   |
| 1AE5 | LD (00B8),hl |
| 1AE8 | LD a,d       |
| 1AE9 | RST 28       |
| 1AEA | ADD a,60     |
| 1AEC | RST 28       |
| 1AED | LD a,FF      |
| 1AEF | OUT (F4),a   |
| 1AF1 | LD b,FF      |
| 1AF3 | DJNZ 1AF3    |
| 1AF5 | XOR a        |
| 1AF6 | OUT (F4),a   |
| 1AF8 | LD a,d       |
| 1AF9 | XOR 10       |
| 1AFB | LD d,a       |
| 1AFC | JR 1B00      |
| 1AFE | JR 1B17      |
| 1B00 | LD b,FF      |
| 1B02 | DJNZ 1B02    |
| 1B04 | DEC e        |
| 1B05 | JR NZ,1AE2   |
| 1B07 | LD hl,(1A00) |
| 1B0A | LD (hl),90   |
| 1B0C | LD de,000A   |
| 1B0F | LD hl,(1A84) |
| 1B12 | ADD hl,de    |
| 1B13 | LD (1A84),hl |
| 1B16 | RET          |
| 1B17 | LD hl,0004   |
| 1B1A | LD (00B8),hl |
| 1B1D | LD hl,(1A84) |
| 1B20 | CALL BB98    |
| 1B23 | LD a,20      |
| 1B25 | RST 28       |
| 1B26 | LD hl,(1A00) |
| 1B29 | BIT 1,(hl)   |
| 1B2B | JR Z,1B3C    |
| 1B2D | LD de,1E2E   |
| 1B30 | AND a        |
| 1B31 | SBC hl,de    |
| 1B33 | LD a,h       |
| 1B34 | OR l         |
| 1B35 | JR NZ,1B3C   |
| 1B37 | LD a,02      |
| 1B39 | LD (1A29),a  |
| 1B3C | RET          |

**Abonnez-vous  
à**

# MICRO-SYSTÈMES

**1 AN  
11 numéros**

**190 F\***

(\* Étranger: 250 F)

*Ne manquez plus votre rendez-vous avec  
MICRO-SYSTÈMES.  
Abonnez-vous dès maintenant et profitez de  
cette réduction qui vous est offerte en nous  
retournant la carte-réponse "abonnement",  
en dernière page.*



**MICRO SYSTEMES**

*Le sérieux d'un journal  
au service d'une technique.*

# Un désassembleur 6809

## écrit en Basic

Chacun peut un jour avoir envie de débuter dans la programmation en langage machine. Il est alors pratique de disposer d'utilitaires tels qu'un assembleur ou un désassembleur. Le but de cet article est de vous proposer un désassembleur pour 6809 qui guidera vos premiers pas.

Ce programme, bien qu'écrit en Basic, permet l'analyse de routines formées de codes hexadécimaux, en traduisant ces derniers sous forme mnémonique. En effet, l'utilisation des mnémoniques rend plus aisée la lecture d'un programme objet.

L'intérêt d'un désassembleur vient du fait qu'il contribue à la compréhension de sous-programmes intégrés à un logiciel existant (interpréteur Basic, moniteur), ce qui permet un développement rationnel de ses propres outils en langage machine.

Ce désassembleur a été initialement conçu sur un Dragon 32. Il peut être utilisé directement sur un TRS 80 Color ou adapté avec quelques modifications (notamment pour la fonction HEX\$ qui convertit un nombre décimal en une chaîne de caractères hexadécimaux) sur tout autre micro-ordinateur fonctionnant avec un 6809.

De plus, le programme a été conçu selon les principes de la programmation structurée (fig. 1), ce qui entraîne, pour le lecteur, une approche plus facile de son fonctionnement.

La richesse du 6809 vient du fait qu'il possède un grand nombre d'instructions pouvant être adressées de beaucoup de manières différentes. Ce microprocesseur possède plusieurs registres 16 bits, ce qui lui permet d'indexer toute la mémoire et d'autoriser des branchements « longs ». Le tableau 1 présente les diverses notations utilisées pour représenter les différents modes d'adressage existants. Les lecteurs qui désireraient connaître la liste complète des différentes mnémoniques utilisables sur ce microprocesseur peuvent consulter l'article de Micro-Systèmes n° 20, sur le 6809.

### La méthode de décodage utilisée

La structure générale de ce logiciel est présentée dans l'organigramme de la figure 2. Le corps du programme se trouve entre les lignes 100 et 700. Le nombre total d'octets que comporte une instruction varie selon le mode d'adressage. C'est pourquoi le programme va chercher (ligne 200) dans la mémoire 5 octets consécutifs (ce qui correspond aux instructions les plus longues) à partir

**UTILITAIRE :  
Un désassembleur 6809  
de T. DURAND, D. HAINAUT  
et E. CHEVALIER**

Analysez les routines inscrites dans la mémoire morte de votre ordinateur afin de les exploiter pour votre propre compte avec cet utilitaire qui représente, en outre, un excellent outil d'initiation au langage machine.

**Langage : Basic  
Ordinateur : Dragon 32**

de l'adresse de départ (entrée en ligne 100), puis les range dans le tableau « M ». On procède alors par décomposition en 16 pages de 16 codes. Par exemple, la mnémonique ayant pour code 9E sera dirigée vers

la page 9 (représentée par la variable D). C'est le 14<sup>e</sup> code de cette page (représenté par la variable « E »).

Certaines instructions nécessitent un pré-octet (\$10 ou \$11). Elles sont alors décodées

### Description du programme

|              |  |
|--------------|--|
| 10-20        | PRESENTATION DU PROGRAMME                            |
| 40-90        | DEFINITION DES CHAINES DE CARACTERES                 |
| 100          | ENTREE DE L'ADRESSE DE DEPART EN HEXADECIMAL         |
| 200          | DECOMPOSITION DU PREMIER OCTET                       |
| 300-310      | CAS PARTICULIER DU PRE-OCTET (\$10 ET \$11)          |
| 500          | INDIRECTION SUIVANT LA PAGE : 0 A 15                 |
| 600-700      | PROGRAMME D'AFFICHAGE                                |
| 1000 PAGE 0  | ROTATIONS ET INCREMENTATIONS EN ADRESSAGE DIRECT     |
| 1500 PAGE 1  | INSTRUCTIONS ARITHMETIQUES ET TRANSFERTS             |
| 2000 PAGE 2  | BRANCHEMENTS COURTS                                  |
| 2500 PAGE 3  | INSTRUCTIONS ARITHMETIQUES, EMPILEMENTSETDEPILEMENTS |
| 3000 PAGE 4  | ROTATIONS ET INCREMENTATIONS SUR A                   |
| 3500 PAGE 5  | ROTATIONS ET INCREMENTATIONS SUR B                   |
| 4000 PAGE 6  | ROTATIONS ET INCREMENTATIONS EN ADRESSAGE INDEXE     |
| 4500 PAGE 7  | ROTATIONS ET INCREMENTATIONS EN ADRESSAGE ETENDU     |
| 5000 PAGE 8  | OPERATIONS ET TESTS SUR A EN ADRESSAGE IMMEDIAT      |
| 5500 PAGE 9  | OPERATIONS ET TESTS SUR A EN ADRESSAGE DIRECT        |
| 6000 PAGE 10 | OPERATIONS ET TESTS SUR A EN ADRESSAGE INDEXE        |
| 6500 PAGE 11 | OPERATIONS ET TESTS SUR A EN ADRESSAGE ETENDU        |
| 7000 PAGE 12 | OPERATIONS ET TESTS SUR B EN ADRESSAGE IMMEDIAT      |
| 7500 PAGE 13 | OPERATIONS ET TESTS SUR B EN ADRESSAGE DIRECT        |
| 8000 PAGE 14 | OPERATIONS ET TESTS SUR B EN ADRESSAGE INDEXE        |
| 8500 PAGE 15 | OPERATIONS ET TESTS SUR B EN ADRESSAGE ETENDU        |
| 10000        | CAS DU PRE-OCTET (\$10)                              |
| 11000        | CAS DU PRE-OCTET (\$11)                              |
| 12000        | ANALYSE DE L'ADRESSAGE INDEXE                        |
| 13000        | CONVERSION DECIMAL-HEXADECIMAL 8 BITS                |
| 14000        | CONVERSION DECIMAL-HEXADECIMAL 16 BITS               |
| 15000        | SOUSS-PROGRAMME D'ERREUR                             |
| 16000        | CALCUL DES ADRESSES DE BRANCHEMENT                   |
| 17000        | AFFICHAGE DE LA NOTICE                               |
| 18000        | CONVERSION HEXADECIMAL-DECIMAL                       |
| 19000        | DECOMPOSITION BINAIRE D'UN OCTET                     |

Fig. 1. — Le détail de la structure du programme met en évidence son fonctionnement et le rôle de chaque groupe de lignes.

| MODE D'ADRESSAGE |                  | EXEMPLE DE NOTATION     |
|------------------|------------------|-------------------------|
| INHERENT         |                  | RTS                     |
| IMMEDIAT         |                  | LDS # A000              |
| DIRECT           |                  | LDA # 2E                |
| ETENDU           |                  | LDX < 3F                |
| ETENDU           |                  | JSR > BFFF              |
| INDIRECT         |                  | JSR > [BFFF]            |
| RELATIF          |                  | BNE 7000                |
| RELATIF LONG     |                  | LBEQ 8000               |
| INDEXE           | SUR 5 BITS       | LDD-OB,X<br>LDD [-OB,X] |
|                  | INDIRECT         | LDA,X                   |
|                  | DEPL. NUL        | LDA [X]                 |
|                  | INDIRECT         | LDA X+                  |
|                  | AUTO-INCREmente  | LDA X++                 |
|                  | INDIRECT         | LDA [X++]               |
|                  | AUTO-DECRe mente | LDA -X                  |
|                  | INDIRECT         | LDA --X                 |
| PAR A            | INDIRECT         | LDA [-X]                |
| PAR B            | INDIRECT         | LDY A,X                 |
|                  | INDIRECT         | LDY [A,X]               |
|                  | INDIRECT         | LDY B,X                 |
| SUR 8 BITS       | INDIRECT         | LDY [B,X]               |
| SUR 16 BITS      | INDIRECT         | LDA -A0,Y               |
|                  | INDIRECT         | LDA [-A0,Y]             |
|                  | INDIRECT         | LDA +B000,Y             |
| PAR D            | INDIRECT         | LDA [+B000,Y]           |
| SUR 8 BITS/PC    | INDIRECT         | LDA D,X                 |
|                  | INDIRECT         | LDA -A0,PC              |
| SUR 16 BITS/PC   | INDIRECT         | LDA [-A0,PC]            |
|                  | INDIRECT         | LDA 9000,PC             |
|                  | INDIRECT         | LDA [9000,PC]           |

Tableau 1. – Les différents types d'adressage du 6809 font de ce microprocesseur le « 8 bits » le plus puissant du marché.

|                        |   |
|------------------------|---|
| <b>A\$, A1\$, A2\$</b> | VARIABLES UNIVERSELLES                                  |
| <b>B\$</b>             | MNEMONIQUE  |
| <b>C\$</b>             | ADRESSE OU DONNÉE                                       |
| <b>D\$ à I\$</b>       | CONSTANTES CONTENANT LES MNEMONIQUES                    |
| <b>T\$</b>             | INKEY\$   |
| <b>U\$</b>             | LISTE DES REGISTRES INDEXABLES                          |
| <b>AA</b>              | ADRESSE DU PREMIER OCTET DECODE DANS L'INSTRUCTION      |
| <b>C</b>               | CONTIENT L'ADRESSE RELATIVE AUX BRANCHEMENTS            |
| <b>D</b>               | NUMERO DE PAGE  |
| <b>E</b>               | NUMERO DANS LA PAGE                                     |
| <b>DE</b>              | LONGUEUR TOTALE DE L'INSTRUCTION (DEPLACEMENT)          |
| <b>I,N</b>             | VARIABLES DE BOUCLES                                    |
| <b>K(0) à K(7)</b>     | DECOMPOSITION BINAIRE D'UN OCTET                        |
| <b>KC</b>              | NUMERO DU REGISTRE D'INDEXATION                         |
| <b>KM</b>              | REPRESENTE LES 4 BITS DE POIDS FAIBLE DU POST-OCTET     |
| <b>MN</b>              | LONGUEUR MNEMONIQUE + POST-OCTET EN INDEXE              |
| <b>M(1) à M(5)</b>     | REPRESENTE LES OCTETS DE L'INSTRUCTION                  |
| <b>M</b>               | EGALE à M(2)  |
| <b>U</b>               | UTILISEE DANS LA DECOMPOSITION BINAIRE                  |
| <b>O</b>               | INDIQUE LE NOMBRE D'INSTRUCTIONS RESTANT A DESASSEMBLER |

Tableau 2. – Liste des variables principales du programme.

|                    |               |
|--------------------|---------------|
| 7000:3B            | RTI           |
| 7001:36 FF         | PSHU PSXYDBAC |
| 7003:1E 89         | EXG A ;B      |
| 7005:1F 03         | TFR D ;U      |
| 7007:26 0B         | BNE 7014      |
| 7009:7E 70 FF      | JMP >70FF     |
| 700C:6E 9F 70 FF   | JMP >[70FF]   |
| 7010:9A A0         | ORA <A0       |
| 7012:11A3 99 A0 00 | CMPUC-6000X ] |
| 7017:CC 40 00      | LDD #4000     |
| 701A:AD E6         | JSR A;S       |
| 701C:AD E1         | JSR S ++      |
| 701E:AD E3         | JSR --S       |
| 7020:1027 01 A0    | LBEQ 71C4     |
| 7024:13            | SYNC          |
| 7025:1C FF         | ANDCC#FF      |
| 7027:10EF B4       | STS EY ]      |
| 702A:FF 00 FF      | STU >00FF     |
| 702D:00 FF         | NEG <FF       |
| 702F:00 FF         | NEG <FF       |

Fig. 3. – Un exemple de désassemblage montre la structure du listing généré.

à part, respectivement à partir des lignes 10000 et 11000. Le test d'indirection est réalisé en lignes 300 et 310.

Pour les autres codes (ceux qui ne nécessitent qu'un octet), l'indirection est réalisée en ligne 500, à l'aide de la fonction ON... GOSUB. Les lignes 600 à 700 assurent l'affichage du résultat, ainsi que la prise en compte des ordres au clavier.

Les mnémoniques étant en général communes à plusieurs pages, il est pratique de les définir en début de programme. C'est le rôle des chaînes de caractères D\$ et I\$ définies en lignes 40 à 90. Les contenus des différentes pages sont précisés dans le plan du programme.

Grâce à la variable E et à la fonction MID\$, le choix de la mnémonique est rendu aisé. Il correspond à une sous-chaîne de la chaîne relative à la page. Par exemple, si le code est 39, on a D = 3 et E = 9. On prend la chaîne de caractères correspondant à la page 3, c'est-à-dire I\$. La sous-chaîne correspond alors à RTS, qui est la 9<sup>e</sup> mnémonique de cette page. Tous les sous-programmes entre les lignes 1000 et 8500 fonctionnent selon ce principe (sauf en ce qui concerne l'adressage indexé qui est traité en 12000). Dans le cas où l'instruction est un branchement, on fait appel au sous-programme

des lignes 16000. Pour les empiilements et les dépilements, les registres PC, DP et CCR sont respectivement notés P, D et C (comme l'illustre l'exemple de la figure 3).

La confusion qui pourrait exister entre « D » désignant le registre double « A » + « B » et « D », identifiant DP, le registre de page, est levée par le contexte, le premier faisant 16 bits, tandis que le second n'en a que 8. Lorsqu'il y a un pré-octet, on traite toujours selon la méthode des pages, mais ici c'est le second octet qui est décomposé.

Le décodage des instructions relatives au mode indexé étant assez complexe, il est analysé séparément à partir de la ligne 12000. Tout d'abord, on décompose le post-octet en binaire en appelant le sous-programme des lignes 19000. On recherche le registre d'indexation avec les bits 5 et 6 du post-octet. Puis on calcule le nombre d'octets total de l'instruction. Le post-octet peut être suivi de 0, 1 ou 2 octets. La variable MN représente le nombre d'octets mnémonique + post-octet.

Enfin, la variable DE est égale au nombre d'octets que comporte l'instruction. On ajoute DE à l'adresse de départ (AA), ce qui donne l'adresse de l'instruction suivante.

Le listing du programme est

en figure 4 et le tableau 2 fournit le rôle des principales variables.

## L'utilisation

Après avoir tapé RUN, le programme affiche le titre, puis la notice d'utilisation. Il demande (en ligne 100) l'adresse à partir de laquelle vous désirez déAssembler. Cette adresse doit être entrée en hexadécimal, sinon un signal sonore vous le rappellera.

Le programme analyse alors 12 instructions consécutives. Vous avez le choix entre 4 options, suivant la touche appuyée :

- **SPACE** affiche 12 nouvelles instructions,
- **ENTER** ajoute une ligne supplémentaire sur l'écran,
- **@** vous permet de reprendre une nouvelle adresse de départ,
- **S** vous fait quitter le programme.

L'appui sur toute autre touche provoque l'affichage de la liste des commandes acceptées par le programme.

Lors du déAssemblyage, l'ordinateur peut afficher trois points d'interrogation à la place de la mnémonique. Cela signifie que le code ne correspond à aucune instruction. De même, l'affichage de TFR ?? ou de EXG ?? vous indique une erreur de format dans les registres.

## Extension du programme

Il est possible d'améliorer et de compléter ce programme en lui ajoutant quelques fonctions nouvelles. On peut par exemple lui adjoindre un programme de « DUMP » qui visualisera l'état de la mémoire. D'autre part, on peut lui ajouter une fonction permettant de modifier la valeur d'un octet, sans arrêter le programme.

Pour augmenter l'intérêt de ce programme, les adeptes du langage machine pourront créer une sous-routine permettant de visualiser les registres internes du microprocesseur. Ceci peut s'avérer utile pour bien suivre les séquences d'un programme machine.

Enfin, les possesseurs d'un ordinateur capable d'afficher plus de 32 caractères par ligne peuvent améliorer l'affichage en modifiant la routine des lignes 600-700. ■

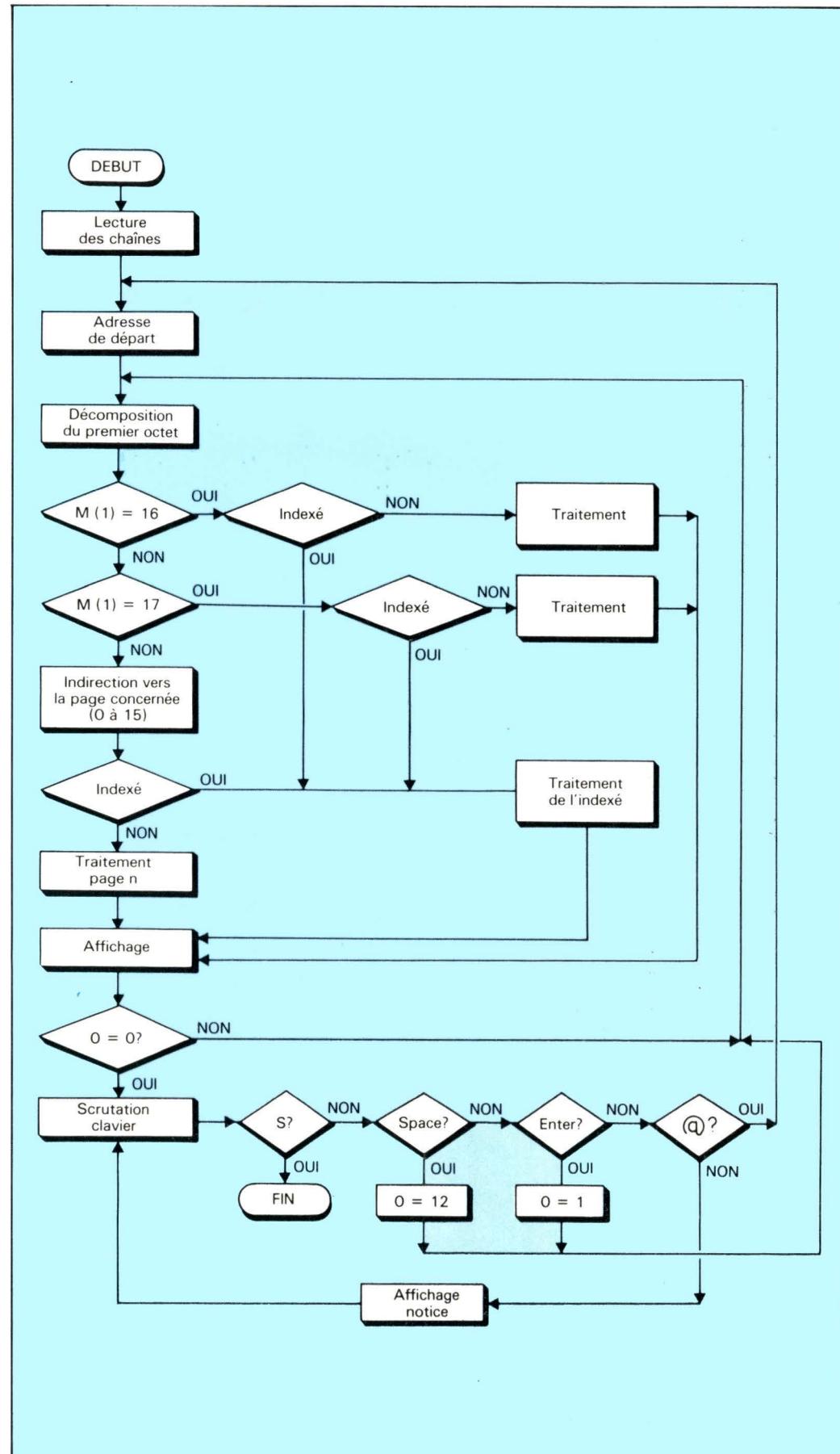


Fig. 2. – L'organigramme proposé ici permettra à tous les possesseurs d'un ordinateur basé sur un 6809 d'effectuer son adaptation.

```

3 REM ****
4 REM * DESASSEMBLEUR 6809 *
5 REM * AUTEURS: T. DURAND *
6 REM * D. HAINAUT *
7 REM * E. CHEVALIER*
8 REM ****
9 REM
10 CLS0 PRINT@193, " DESASSEMBLEUR 6809" : SCREEN0,1:FORI=1TO5000:NEXT
20 GOSUB17000
37 REM
38 REM DEFINITION DES CHAÎNES
39 REM
40 IS="LEARNLERYLESLEAUPSHSPULSPSHUFLU??" RTS ABX RTI CHAYMUL ??? SWI "
41 HS="??? ??? NDP SYNC??? ??? LBRALBSR?? DAA ORCC?? ANDCSEX EXG TFR "
42 GS="SUBCMPPBSCBADDDBANDBITBLDB STB EOPBADCBORB ADDBLDD STD LDU STU "
43 FS="SUBCMPPBSCBASUBDABITBLDB STA EORRADCDARA ADDACMPXJSR LDX STX "
44 ES="RARRHILSCLCSNEEDQVCVSPLMIGELTGTL"
45 DS="NEG?????COMLBR??RORASRSLROLDEC??INCTSTJMFCLR"
100 CLS PRINT29E INPUT"ADRESSE DE DEPART":RA$ GOSUB1000:CLS
110 D=12
200 FORI=1TO5.MK1)=PEEK(RA+I-1):NEXT:D=INT(MK1)/16:E=MK1-16*D
289 REM
290 REM TESTS PRE-OCTET $10,$11
291 REM
300 IFMK1>16THENM=MK2:GOSUB1000:GOTO600
310 IFMK1>17THENM=MK2:GOSUB11000:GOTO600
483 REM
490 REM AIGUILAGE VERS PAGE3
491 REM
500 DND=1GOSUB1000,1500,2000,2500,3000,3500,4000,4500,5000,5500,6000,6500,7000,7
500,8000,8500
589 REM
590 REM AFFICHAGE D'UNE INSTRUC.
591 REM
600 AS="000"+HEX$(RA)+":":A$=RIGHT$(A$,5)
610 IF MK1=160RM1=17THENRA1$=HEX$MK1>>ELSEA1$="0"+HEX$(MK1)+":":A1$=RIGHT$(A
1$,3)
615 IFDE=1THEN630
620 FORI=2TODE:A2$="0"+HEX$(MK1)):A1$=A1$+RIGHT$(A2$,2)+":":NEXT
630 A1$=A1$+"":A1$=LEFT$(A1$,14)
640 A$=A$+A1$+BS+C$+":":A$=LEFT$(A$,32)
650 PRINTA$:RA=RA+DE
660 D=0-1 IFOK>>0THEN200
670 T$=INKEY$:IFT$=""THEN670
680 IFT$="THEEND=12:CLS:GOTO200
690 IFT$=CHR$(13)THEN0=1:GOTO200
700 IFT$=""THEEND100
710 IFT$="S"THENCLS:END
720 GOSUB17000:GOTO670
589 REM
590 REM ROTATIONS DECALAGES
591 REM INCREMENTATIONS
592 REM MODE DIRECT
593 REM
1000 BS=MID$(D$,3*E+1,3)
1010 IF E=10RE=20RE=50RE=1:THEN15000ELSEBS=BS+" "<:DE=2:GOSUB13000:RETURN
1489 REM
1490 REM INSTRUC. ARITHMETIQUES
1491 REM LOGIQUES
1492 REM ECHANGES TRANSFERTS
1493 REM
1500 BS=MID$(H$,4*E+1,4)
1510 IFE=10RE=40RE=50RE=60RE=110RE=200THEN15000
1520 IFE=10THENBS=BS+" ">:DE=2:GOSUB13000:RETURN
1530 IFE=12THENBS=BS+C$+":":DE=2:GOSUB13000:RETURN
1540 IFE=60RE=7THENBS=E$+":":DE=3:C=256*M(2)+M(3):GOTO10120
1550 IFE=140RE=15THEN1600
1560 C$="">:DE=1:RETURN
1600 AS="D X Y U S PC A B CCDF":X=INT(M(2)/16):Y=M(2)-16*X
1610 IFOX>7ANDY<8THENDE=2:C$=""?>:RETURN
1620 IFOX>8ANDY>7THENDE=2:C$=""?>:RETURN
1630 A1$=MID$(A$,2*X+1,2):A2$=MID$(A$,2*X+1,2):IF A1$=" "ORA2$=" "THENDE=1:C$=
":":RETURN
1640 C$=A1$+"":A2$>:DE=2:RETURN
1889 REM
1990 REM BRANCHEMENTS COURTS
1991 REM
2000 BS=MID$(E$,2*E+1,2)
2005 BS="E"+BS
2010 BS=B$+" ">:DE=2:IFMK2>>127THENC=M(2)-256ELSEC=M(2)
2020 GOTO16000
2489 REM
2490 REM INSTRUC. ARITHMETIQUES
2491 REM EMPILEMENTS DEPILEMENTS
2492 REM
2500 BS=MID$(I$,4*E+1,4)
2505 IFE=60RE=14THEN15000
2510 IFE=90RE=110RE=130RE=15THENDE=1:C$="">:RETURN
2520 IFE>4THENM=2:GOTO12000
2525 IFE=12THENBS=BS+" ">:
2530 IFE=40RE=5THEN2600
2540 IFE=60RE=7THEN2610
2550 DE=2:GOSUB12000:RETURN
2600 AS="PUXYDECAC":GOTO2615
2610 AS="PSXYDECAC"
2615 14MC2:GOSUB19000
2620 C$="">:FORN=7TQ08STEP-1:IF(KN)=0THENC$=C$+" "ELSEC$=C$+MID$(A$,8-N,1)
2630 NEXT

```

```

2640 BS=MID$(D$,3*E+1,3)
2989 REM
2990 REM ROTATIONS DECALAGES
2991 REM INCREMENTATIONS SUR A
2992 REM
3000 BS=MID$(D$,3*E+1,3)
3005 IF E=10RE=20RE=50RE=110RE=14THEN15000
3010 IFE=4THENBS=BS+" "A"ELSEBS=BS+"B"
3020 C$="">:DE=1:RETURN
3489 REM
3490 REM ROTATIONS DECALAGES
3491 REM INSTRUCTIONS SUR B
3492 REM
3500 GOTO3000
3969 REM
3990 REM ROTATIONS DECALAGES
3991 REM INCREMENTATIONS
3992 REM MODE INDEXE
3993 REM
4000 BS=MID$(D$,3*E+1,3)+" "
4010 IFE=10RE=20RE=50RE=11THEN15000ELSEMN=2:GOTO12000
4489 REM
4490 REM ROTATIONS DECALAGES
4491 REM INCREMENTATIONS
4492 REM MODE ETENDU
4493 REM
4500 BS=MID$(D$,3*E+1,3)+" "
4510 IFE=10RE=20RE=50RE=11THENDE=1:C$="">:RETURN:ELSEGOT08510
4989 REM
4990 REM OPERATIONS, TESTS SUR A
4991 REM MODE IMMEDIAT
4992 REM
5000 BS=MID$(F$,4*E+1,4)
5010 IF E=70RE=15THEN15000
5020 IFE=13THENBS="BS":GOTO2010:ELSEBS=BS+" "
5040 IFE=30RE=120RE=14THENDE=2:GOSUB14000:RETURN
5050 DE=2:GOSUB13000:RETURN
5489 REM
5490 REM OPERATIONS, TESTS SUR A
5491 REM MODE DIRECT
5492 REM
5500 BS=MID$(F$,4*E+1,4)
5510 BS=B$+" ">:DE=2:GOSUB13000:RETURN
5589 REM
5590 REM OPERATIONS, TESTS SUR A
5591 REM MODE INDEXE
5592 REM
5600 BS=MID$(F$,4*E+1,4)
5610 GOTO8510
6389 REM
6390 REM OPERATIONS, TESTS SUR B
6391 REM MODE IMMEDIAT
6392 REM
7000 BS=MID$(G$,4*E+1,4)
7010 IF E=70RE=130RE=15THEN15000ELSE5020
7489 REM
7490 REM OPERATIONS, TESTS SUR B
7491 REM MODE DIRECT
7492 REM
7500 BS=MID$(G$,4*E+1,4)
7510 GOTO5510
7569 REM
7590 REM OPERATIONS, TESTS SUR B
7591 REM MODE INDEXE
7592 REM
8000 BS=MID$(G$,4*E+1,4):MN=2:GOTO12000
8489 REM
8490 REM OPERATIONS, TESTS SUR B
8491 REM MODE ETENDU
8492 REM
8500 BS=MID$(G$,4*E+1,4)
8510 BS=B$+" ">:DE=2:GOSUB14000:RETURN
9989 REM
9990 REM PRE-OCTET $10
9991 REM
10000 D=INT(M/16):E=M-16*D
10010 IFM>32ANDM>48THEN10100
10020 IFE=63THENDE=2:BS="SWI 2":C$="">:RETURN
10030 IFE>130THEN10500ELSE15000
10100 AS=RIGHT$(E$,30):BS="LE"+MID$(A$,2*E-1,2)+" ">:DE=4
10110 C=M(3)*256+M(4)
10120 IFE>32ANDE>12ANDE>14ANDE>>15THEN15000
10500 GOTO16000
10520 IFE>11THEN10600
10530 IFE=3THENBS="CMPL"
10540 IFE=12THENBS="CMPY"
10550 IFE=14THENBS="LDY"
10560 IFE=15ANDD>>8THENBS="STY"
10565 IFE=15ANDD>>8THEN15000
10570 IFE=10THENNN=3:GOTO12000
10580 IFE=9THENBS=BS+" ">:DE=3:GOSUB13000:RETURN

```

Fig. 4. – Listing du programme Basic.

```

10590 IFD=8THENB$="B$+" "#ELSEB$="B$+" >"  

10595 DE=3:GOSUB14000 RETURN  

10600 IFE<14THENE<15THEN15000  

10610 IFE=15ANDD=12THEN15000  

10620 IFE=14THENB$="LDS " ELSEB$="STS "  

10630 IFE=12THENB$="B$+" "#:GOTO10595  

10640 IFD=15THENB$="B$+" >" :GOTO10595  

10650 IFD=14THENMN=3:GOTC12000  

10660 B$=B$+" <":DE=3:GOSUB13000:RETURN  

10989 REM  

10990 REM PRE-OCTET $11  

10991 REM  

10992 IFM="" :IFM=63THENB$="SUI 3 " :C$="":DE=2:RETURN  

11010 IFM=1310RM=1470RM=1630RM=179THENB$="CMPU"  

11020 IFM=140RM=1560RM=1720RM=186THENB$="CMPS"  

11030 IFB$="":THEN15000  

11040 IFM=1310RM=148THENB$="B$+" #:DE=3:GOSUB14000:RETURN  

11050 IFM=1470RM=156THENB$="B$+" <":DE=3:GOSUB13000:RETURN  

11060 IFM=1790RM=186THENB$="B$+" >":DE=3:GOSUB14000:RETURN:ELSEMH=3:  

11061 GOTC12000  

11089 REM  

11090 REM TRAITEMENT DE L' INDEXE  

11091 REM  

12000 M=MKMN>:GOSUB12000  

12004 REM  

12005 REM REGISTRE D' INDEXATION  

12010 US="X Y U S PC":KC=2*KC6)+K(5):IFK(1)>0ANDK(2)=1ANDK(3)=1THENKC=4  

12015 A$=MID$(US,2KC+1,2)  

12020 IFK(7)>0THEN12500  

12030 IFKM>7THEN12600  

12040 IFKM=8ORKM=12THEN12700  

12050 IFKM=9ORKM=13THEN12800  

12060 IFKM=11THEN12900  

12070 IFKM=15ANDKC=0THENB$="B$+" >":DE=MN+1:GOSUB14000:C$="E"+C$+"J"  

12080 GOTC15000  

12081 A$="+-":A$=MID$(A$,KC+1,1)  

12095 IFK(4)>1THENKN=16-KM  

12096 C$="0"+HEX$(KM):C$=C$+C$+":":A1$=" "+B$=B$+" "+:DE=MN:RETURN  

12098 A$=A1$+"- "+A1$+"++ "-+A1$+"- "+A1$+"- "+A1$+"- "+B$+"+A":+A1$+C$=  

M+1,4) MID$(A$,4KC  

12099 IFKM=0ORKM=2THENIFK(4)=1THEN15000  

12101 IFK(4)>1THENC$="C"+C$+"J":ELSEC$=" "+C$  

12120 B$=B$+" ":DE=MN:RETURN  

12120 C=M(MN+1):DE=MN+1:B$=B$+" "  

12120 IFC>127THENC=256-C:C$="-":ELSEC$="+"  

12120 C$=0+HEX$(M(DE)):C$=RIGHT$(C$,2):RETURN  

12120 C$=C$+A1$:DE=MN+2:RETURN  

12120 B$=B$+" ":IFK(4)=1THENC$="ED":+A1$+"J":ELSEC$=" D":+A1$  

12120 DE=MN:RETURN  

12120 REM  

12120 REM DECIMAL->HEXA 8 BITS  

12120 REM  

12120 13000 C$="0"+HEX$(M(DE)):C$=RIGHT$(C$,2):RETURN  

12120 REM  

12120 13990 REM DECIMAL->HEXA 16 BITS  

12120 REM  

12120 14000 C$="000"+HEX$(256*M(DE)+M(DE+1)):C$=RIGHT$(C$,4):DE=DE+1:RETURN  

12120 REM  

12120 14990 REM PROGRAMME D'ERREUR  

12120 REM  

12120 15000 B$="????":C$="":DE=1:RETURN  

12120 REM  

12120 15990 REM ADRESSE DE BRANCHEMENT  

12120 REM  

12120 16000 C=AA+DE+C:IFC<0THEND=C+65536  

12120 16010 IFC>65535THENC=C-65536  

12120 16020 C$="000"+HEX$(C):C$=RIGHT$(C$,4):RETURN  

12120 REM  

12120 16998 REM AFFICHAGE DE LA NOTICE  

12120 REM  

12120 17000 CLS:PRINT@64," UTILISATION DU PROGRAMME .."  

12120 PRINT:PRINT"- POUR DONNER UNE NOUVELLE ADRESSE"  

12120 PRINT:PRINT"- SPACE POUR DESASSEMBLER 12 INSTRUCTIONS"  

12120 PRINT:PRINT"- ENTER POUR DESASSEMBLER UNE INSTRUCTION  

E" SUPPLEMENTAIRE  

12120 17040 PRINT:PRINT"- S POUR SORTIR DU PROGRAMME":SCREEN0,1:FORI=1TO10000  

12120 JRN NEXT:RET  

12120 REM  

12120 17990 REM HEXA->DECIMAL ADRESSE  

12120 REM  

12120 18000 AA=B$+LENKA$>:IFR>4THEN18060  

12120 FORT=1TOA:AS=ASC(MID$(A$,A-T+1,1)):IFRS>57ANDRS<65THEN18060  

12120 IFRS<48THEN18060ELSEB$=AC-48  

12120 IFRS>79THEN18060ELSEIFRS>64THENB$=AS-55  

12120 AA=AA+B$16^(T-1)  

12120 NEXT:RETURN  

12120 PRINT"ERREUR":PLAY" T303D":GOTC100  

12120 REM  

12120 18990 REM DECOMPOSITION BINAIRE  

12120 REM  

12120 19000 U=M:FORN=7TO8STEP-1:K(N)=INT(U/(2^N)+.001):U=U-K(N)*(2^N):NEXT  

12120 KM=K(0)+2*K(1)+4*K(2)+8*K(3):RETURN

```

# REUSSISSEZ VOTRE INVESTISSEMENT

avec des ordinateurs simples comme un coup de téléphone

## LISA et Macintosh

vous améliorez vos performances



Concessionnaire agréé



● Confiez nous votre problème on vous présentera des solutions

● Comparaisons entre logiciels Logiciels intégrés Logiciels spécifiques

● Connexions : TELETEL

● Le portable se porte bien : 3.9 kg

**A/C** : son prix est léger sa mémoire est lourde : 128 k

● Gamme complète **APPLE**

● Tarifs spéciaux : enseignants - étudiants écoles - facultés

● MULTIPOTES - Disques durs ...

● Location - Contrat d'assistance

● FORMATION ASSUREE SUR VOS APPLICATIONS

● Service après vente efficace

**ALTI**

67, rue Vendôme 69006 LYON  
Tél. (7) 894 60 56

# Un Reset non destructif pour Canon X 07

Actuellement, lorsque le système se « plante », deux solutions s'offrent à l'utilisateur. La plus commune consiste à actionner, à l'aide d'un objet pointu, le bouton Reset situé sous l'appareil. La seconde est de retirer une pile de son logement pendant quelques secondes. Dans les deux cas, la machine est effectivement « déplante », mais surtout, si on ne les a pas sauvés sur cassette, programmes et données ne sont plus qu'un douloureux souvenir.

Le programme que nous vous proposons a pour but de vous éviter ce type de mésaventures et ce, grâce à une nouvelle touche RESET. Cette dernière est accessible directement par le clavier (donc beaucoup plus facile à actionner que celle située sous l'appareil) et présente surtout l'indéniable avantage de conserver le contenu de la mémoire RAM.

## L'implantation

Il suffit d'exécuter (commande « RUN ») le programme présenté **figure 1**. La routine en langage machine étant totalement relogable (les seuls sauts se font en ROM), l'adresse d'implantation est à votre choix. Ne vous étonnez pas si, en fin d'exécution du programme de la **figure 1**, l'ordinateur s'éteint. Cet état est provoqué par l'utilisation de l'instruction « OFF 1 » qui valide le START\$. Ce dernier a pour rôle, à chaque mise sous tension de l'appareil, d'implanter les octets permettant l'interactivité de la routine situés aux adresses 61 et 62.

La sauvegarde et le chargement de la routine en langage machine peuvent se faire en utilisant le programme de la **figure 1** comme support ou à l'aide des fonctions « S » et « L » du Moniteur-Désassemblleur paru dans notre numéro 42.

## L'utilisation

La mise en action de la routine provoquant le Reset se fait par un appui sur CTRL-Q. Même lorsque les touches d'interruption d'un programme (OFF, ON/BREAK, CTRL-C) sont inopérantes (ce qui arrive

**UTILITAIRE :**  
Un « vrai » Reset  
d'Emmanuel SANDER

Cette routine interactive écrite en langage machine vous permet de « déplanter » votre Canon X 07 sans perdre les programmes et données qui se trouvent en mémoire.

Ordinateur : Canon X 07  
Langage : langage machine  
NSC 800 (compatible Z 80).

```
10000 REM ****
11000 REM ***** TOUCHE RESET ****
12000 REM ***** POUR Canon X07 ****
13000 REM ** (c) EMMANUEL SANDER 1984 **
14000 REM ****
15000 INPUT"ADRESSE D'IMPLANTATION";AD
16000 FORI=0TO37
17000 READA$
18000 POKEAD+I,VAL("&H"+A$)
19000 NEXT
20000 L=ADMOD256
21000 H=AD\256
25000 START$="POKE61,"+STR$(L)+":POKE62,
"+STR$(H)+CHR$(13):OFF1
30000 DATA D9,08,DB,F1
31000 DATA FE,11,C2,9B
32000 DATA C7,08,D9,ED
33000 DATA 57,E2,C3,C3
34000 DATA CD,BD,C0,3A
35000 DATA 2B,00,06,08
36000 DATA B0,32,2B,00
37000 DATA 3E,0C,D3,BB
38000 DATA CD,A2,00,C3
39000 DATA CF,C3
```

Fig. 1. – Programme Basic assurant l'implantation en mémoire de la routine Reset (contenu dans les Data).

fréquemment lors de la programmation en langage machine), un appui sur CTRL-Q donne de nouveau la main à l'utilisateur et ce, sans perdre le contenu de la mémoire.

## Le programme

L'emploi de ports d'entrées/sorties et de routines spécifiques au Canon X 07 rend celui-ci inadaptable sur une autre machine. Il faut également savoir, pour une bonne compréhension, que cette routine est interactive : elle ne doit donc pas se concevoir comme une unité fonctionnelle mais comme inté-

grée au système ; c'est-à-dire dépendante du logiciel de base et de la structure d'entrées/sorties du microprocesseur. Pour le reste, les lecteurs pourront se reporter au listing du programme en langage d'assemblage de la **figure 2** ; sa brièveté exclut tout organigramme ou description générale.

## Remarques

L'imprimante-table traçante qui équipe le Canon X 07 est disponible pour plusieurs micro-ordinateurs. Sa police de caractères ne lui est donc pas spécifique et ne dispose pas du sym-

```
1F00 EXX
1F01 EX af,af'
1F02 IN a,(F1)
1F04 CP 11
1F06 JP NZ,C79B
1F09 EX af,af'
1F0A EXX
1F0B LD a, i
1F0D JP PO,C3C3
1F10 CALL C0BD
1F13 LD a,(002B)
1F16 LD b,08
1F18 OR b
1F19 LD (002B),a
1F1C LD a,0C
1F1E OUT (BB),a
1F20 CALL 00A2
1F23 JP C3CF
```

Fig. 2. – Listing de la routine en mnémomique Z 80.

bole « ¥ » qui est remplacé par le signe « \ » dans le listing du programme de la **figure 1**.

La méthode que nous vous présentons pour conserver le contenu de la mémoire même en cas d'incident est de conception uniquement logicielle. Si elle fonctionne parfaitement dans l'essentiel des cas de perte de contrôle en langage machine, c'est-à-dire la boucle sans fin, il peut arriver que l'utilisation de certaines instructions concernant les entrées/sorties ou les interruptions la mette en défaut. Dans ce dernier cas (heureusement rare), il faut alors vous résoudre à recourir au RESET situé sous l'appareil.

Si vous voulez désactiver le programme, la démarche à suivre est l'emploi de l'instruction : OFF 2. ■

# « Dump & Poke » pour Thomson TO 7

Cet éditeur de code machine (langage 6809) adapté au TO 7 permet, par des commandes simples au clavier, de lister en codes hexadécimaux le contenu de la mémoire à une adresse donnée, de la modifier (si elle se trouve en mémoire vive), de la sauvegarder ou de la recharger sur un magnétophone, et enfin de transférer le contenu d'une zone mémoire vers une autre.

C'est un symbole rouge « > », au début de chaque ligne de l'écran, qui vous rappelle que vous êtes sous contrôle du système « Dump & Poke ». Vos commandes frappées au clavier apparaissent en caractères blancs sur l'écran. Si vous sortez du système (EXIT par la commande « X ») pour revenir au Basic, les caractères passent en vert.

Une ligne normale sous système « Dump & Poke » comporte, après le sigle rouge « > », l'adresse hexadécimale de la première instruction listée, puis les huit octets suivants avec un espace de séparation et, enfin, le décodage ASCII des octets de la ligne s'ils correspondent à un caractère imprévisible. Chaque champ est de couleurs différentes pour mieux les différencier.

## Les initialisations du programme

Suivons ligne par ligne le cheminement des instructions Basic en soulignant quelques particularités du système : tout d'abord, l'initialisation.

• **Ligne 60 :** C'est la définition des types de variables utilisées : « I » et « Y », variables intermédiaires seront entières ; leur valeur sera codée par deux octets seulement vers la fin du programme utilisateur. Positives ou négatives, elles seront limitées à  $\pm 32767$ .

La variable « R » contient la commande frappée au clavier ; « chaîne de caractères », elle sera codée avec l'adresse (sur deux octets) où est stockée son contenu, tout à fait à la fin de la mémoire disponible.

• **Ligne 70 :** On y définit la couleur (caractère vert, sur

fond et pourtour noirs). Le POKE &H60D1 permet de refaire une initialisation du TO 7, suivie de la commande « 1 » sans effacer le programme en mémoire, en cas d'erreur grave. Le POKE &HE7C3 met à « 0 » le bit 3 du PIA et force ainsi le mode Majuscule.

• **Ligne 80 :** Cette instruction complexe repère la position du curseur et imprime en première ligne le titre du système « Dump Poke » en bleu sur fond jaune, avant de revenir à sa position primitive. La ligne du titre, associée au sigle rouge « > » avertit l'utilisateur qu'il est sous contrôle du système Dump Poke ; elle définit aussi la fenêtre d'utilisation (1,24).

• **Ligne 90 :** Cette ligne définit les CHR\$ fréquemment utilisés dans l'édition de chaque ligne listée. Ainsi :

F\$ = CHR\$(22)+« . » et dessine une flèche « → »;  
X\$ = CHR\$(24) efface le reste de la ligne ;  
G\$ = CHR\$(34) imprime le guillemet ;

U\$ = CHR\$(11) fait monter le curseur d'une ligne ;  
E\$ = CHR\$(27) (Escape) permet de modifier les couleurs du point, du fond ou du pourtour de l'écran selon le caractère qui le suivra.

• **Ligne 100 :** Le retour à la ligne 100 affiche le sigle rouge « > » en attente de l'entrée d'une commande au clavier. Pour exécuter une commande « Basic », il faut donc quitter d'abord le programme (EXIT par la frappe : « X » + [Entrée]) : la ligne 1 du titre s'efface alors.

• **Ligne 110 :** La ligne 110 demande à l'utilisateur la commande simplifiée à exécuter. Le TO 7, travaillant avec un éditeur pleine page, il est facile de déplacer le curseur sur n'im-

**UTILITAIRE :**  
**Un éditeur de code machine 6809 de Philippe NEAU**  
**Le microprocesseur 6809 est sans doute le « 8 bits » le plus performant du moment. Son intervention tardive l'a hélas un peu relégué à l'arrière plan, diminuant de ce fait le nombre d'utilitaires disponibles. « Dump & Poke », proposé ici, vient à propos pallier une partie de cette lacune.**  
**Ordinateur : Thomson TO 7**  
**Langage : Basic**

porte quel point à modifier éventuellement. L'instruction K\$ = INKEY\$ évite les rebonds du clavier.

• **Lignes 120 et 130 :** C'est la prise en compte de la commande clavier. Si elle est nulle, il y a retour en ligne 100 ; sinon, les valeurs modifiées selon la position du curseur sont mises dans la variable chaîne « R » pour être interprétées et exécutées.

## Le traitement des commandes

• **Ligne 210 :** Selon le premier caractère de la commande stocké en « R », le programme s'oriente vers différentes routines : M (comme mémoire) entraîne le listage (ligne 100) ; S (comme Save) est équivalent à « SAVEM » (ligne 2000) alors que L (comme Load) provoque un « LOADM » (ligne 3000) ; X (comme Exit) permet la sortie du programme et le retour au Basic (ligne 8500).

• **Ligne 220 :** Si le premier caractère de la variable R est « C » (comme Continue), le programme revient à la ligne précédente qu'il valide et continue aux adresses suivantes jusqu'à la prochaine interruption.

• **Ligne 250 :** Si les trois premiers caractères de la variable « R » sont la chaîne « TFR », le programme est orienté vers une routine de transfert d'instructions dans une autre zone mémoire (routine : ligne 4000).

## L'insertion de codes en mémoire

Cette routine, après avoir testé que le contenu de la variable « R » ne correspondait à au-

cune des commandes précédentes, dirige le programme vers une sous-routine (ligne 8000) pour transformer les adresses ou octets en valeurs en base 10.

• **Ligne 510 :** Les quatre premiers caractères de « R » sont pris comme « adresse ». Si l'un des digits n'est pas « Hexadécimal » (0 à F), l'indicateur F nous renvoie en ligne 100 en attente d'une nouvelle commande non erronée.

• **Ligne 600-700 :** Les 8 octets suivants (0 à 7) sont vérifiés (tous hexadécimaux, par sous-routine ligne 8000), les adresses sont incrémentées. Si aucune erreur n'est détectée, le contenu de la chaîne de caractères « R » est stocké à l'adresse désirée.

## La commande LIST

• **Ligne 1000 :** Cette routine lance le sous-programme de la ligne 1500 aussi longtemps qu'aucune interruption (n'importe quelle touche frappée au clavier) ne vient arrêter la boucle de répétition.

Une ligne affichée comprend :

- le sigle rouge « > » provoqué par un Escape, suivi de « A> » ;
- l'adresse hexadécimale sur 4 digits : A\$ de couleur verte (Escape + « B ») ;
- les 8 octets listés (instruction ou code ASCII) en jaune ;
- enfin, le décodage alphanumérique des 8 octets imprimés en violet.

• **Lignes 1510-1520 :** La variable A\$ prend la valeur hexadécimale de l'adresse ADD. Si la longueur de la chaîne A\$ obtenue est inférieure à 4, des « 0 » sont ajoutés à gauche jusqu'à ce que le format soit correct pour l'affichage.

• **Ligne 1530 :** Les chaînes de caractères B\$ et C\$ sont définies en couleur et en contenu avec des espaces intercalaires entre chaque octet. La couleur est définie pour chaque champ par « Escape » + « caractère alphanumérique ». Ainsi, pour F\$ = CHR\$(27), F\$ + « A » donne la couleur rouge pour les points, F\$ + « Q » donne la couleur rouge pour le fond d'écran.

• **Ligne 1600 à 1700 :** La variable A est chargée du contenu de l'adresse (ADD + I). La valeur hexadécimale de « A », éventuellement complétée par un zéro à gauche, est stockée dans la variable chaîne H\$. Puis l'instruction MID\$(ligne 1620) permet de remplacer chaque terme de B\$ par les nouvelles valeurs H\$, terme à terme. L'instruction (1670) fera le même office pour les caractères impréhensibles de C\$. Les codes de « Contrôle » (CHR\$ de 1 à 31) et les codes supérieurs à CHR\$(128) sont traduits par un point. Cependant, les codes CHR\$(193 à 218), qui correspondent aux majuscules A à Z (avec le bit 7 forcé à « 1 »), sont traduits par un caractère minuscule (a à...z) : cette astuce permet de visualiser les codes d'impression des fonctions Basic (Mémo 7 Basic,

adresse : &H0092 à &H026A) ou les messages d'erreur (tableau de &H1727 à &H176F).

• **Ligne 1800 :** Cette instruction définit l'impression de la ligne par caténation des chaînes : « >, A\$, B\$, C\$ » avec les couleurs correspondantes à chaque champ et effacement des caractères anciens. L'adresse est incrémentée de 8 pour le traitement de la ligne suivante.

### Les fonctions « SAVEM » et « LOADM »

• **Lignes 2010 à 2060 :** Le programme demande le titre à donner au fichier cassette, l'adresse du début et de la fin de la zone de mémoire à sauvegarder, et enfin l'adresse d'exécution. Puis il reproduit : SAVEM « titre », &H adresse... selon le format défini pour le TO 7. Il suffit d'enclencher la touche « Enregistrement » de l'enregistreur et d'appuyer sur « Entrée », pour sauver le programme sur cassette.

• **Lignes 3010 à 3070 :** Le programme procède de la même façon pour lire un programme en langage machine, enregistré sur cassette, et lancer l'exécution.

### La routine de transfert : TFR

• **Ligne 4000, etc. :** Ce programme permet de transférer une zone mémoire comprise entre l'adresse hexadécimale contenue dans A\$ et celle contenue dans B\$, dans une autre zone de mémoires vives en C\$. Les valeurs hexadécimales des trois adresses sont d'abord transformées en décimale, puis contrôlées (B\$ < A\$). Si l'adresse C\$ (où se fait le transfert) est incluse dans la zone comprise entre A\$ et B\$, l'opération a lieu en commençant par la fin (ligne 4120).

Avec cette instruction, on peut ainsi insérer des octets ou en supprimer dans un programme quand on travaille en code Machine.

### Les sous-programmes

• **Ligne 8010 :** Le drapeau (Flag F) est mis à zéro. Si la longueur de la chaîne A\$ n'est pas comprise entre 2 et 4, le flag est positionné à -1 et le programme principal retournera en ligne 100, en attente d'une valeur sans erreur.

• **Ligne 8020 :** Dans toutes les demandes d'entrée (adresse ou donnée), la réponse est toujours

exprimée en valeur hexadécimale. Cette instruction transforme les chaînes A\$ (adresse) et B\$ (données) en valeur décimale pour une utilisation ultérieure dans le programme principal.

• **Ligne 8500 :** C'est l'Exit, obtenu en tapant au clavier : « X », puis « Entrée ». Le retour sous contrôle direct du Basic se traduit par la suppression de la première ligne d'affichage réservée au titre (PSET (0,0) X\$) et par l'impression des commandes en vert et non plus en blanc. Puis le curseur retourne à la ligne où il était auparavant.

### Conclusion

Ce petit programme permet, à l'aide des tables de code Machine du 6809, de créer assez rapidement de petites routines utilisatrices (recherche de chaînes de caractères, bruitage laser...) ou des fonctions qui nous manquent (DRAW, CIRCLE) et même un désassemblleur 6809. L'auteur du programme a ainsi composé une version améliorée, associée à un désassemblleur 6809 (en langage machine) d'environ 1,5 K-octet qui lui permet de mieux comprendre les instructions du Basic Microsoft, utilisées par le TO 7. ■

```

10 '*****+
11 '- DUMP & POKE -
12 '*****+
20 '+ PAR PHILIPPE NEAU +
30 '*****+
50 '
60 DEFINT I,Y:DEFSTR R
70 COLOR 2,0:SCREEN,,0:POKE &H60D1,&H26:POKE &HE7C3,PEEK(&HE7C3) AND NOTS
80 Y=CSRLIM:LOCATE 0,0:COLOR 4,3:ATTRB 1:PRINT "CHR$(8)"***** DUMP POKE +
***":ATTRB 0:PRINT "":CONSOLE 1,24:LOCATE 0,Y
90 F$=CHR$(22)+"":X$=CHR$(24):G$=CHR$(34):U$=CHR$(11):E$=CHR$(27)
100 COLOR 1,0:PRINT">>"";CHR$(13);:COLOR7
110 LINEINPUT R:K$=INKEY$
120 IF LEFT$(R,1)=">" THEN R=MID$(R,2)
130 IF R="" THEN PRINT U$::PLAY"600":GOTO 100
190 '
200 '- DISPATCHING M,S,L,X,C,TFR
210 ON INSTR("MSLX",LEFT$(R,1)) GOTO 1000,2000,3000,8500
220 IF R="C" THEN PRINT U$::GOTO 1100
250 IF LEFT$(R,3)="TFR" THEN 4000
290 '
500 '- POKE 1 LIGNE
510 AS=LEFT$(R,4):GOSUB 8000:IF F THEN 100
520 ADD=A
600 FOR I=0 TO 7:AS=MID$(R,6+I♦3,2):GOSUB 8000:IF NOTF THEN POKEADD+I,A
650 NEXT

```

```

700 PRINT US$:GOSUB 1500:PLAY"L1DOSI":GOTO 100
990 '
1000 '- LIST
1010 A$=MID$(R,2):GOSUB 8000:IF F THEN 100 ELSE ADD=A
1100 GOSUB 1500:IF INKEY$="" THEN 1100 ELSE 100
1490 '
1500 '-S/P LIST 1 LIGNE
1510 A$=HEX$(ADD)
1520 IF LEN(A$)<4 THEN A$="0"+A$:GOTO 1520
1530 B$=E$+"C 00 11 22 33 44 55 66 77":C$=E$+"F 01234567"
1600 FOR I=0 TO 7
1610 A=PEEK(ADD+I):H$=HEX$(A):IF A<16 THEN H$="0"+H$
1620 MID$(B$,4+I*3,2)=H$
1650 IF A<128 THEN IF A>31 THEN 1670 ELSE A=32-14*(A>0):GOTO 1670
1660 IF A<193 OR A>218 THEN IF A>255 THEN A=46 ELSE A=127 ELSE A=A-96
1670 MID$(C$,4+I,1)=CHR$(A)
1700 NEXT
1800 PRINT E$"A>"E$"B"A$B$C$X$"
1810 ADD=ADD+8:RETURN
1990 '
2000 '- SAVEM",,
2010 PRINT US$"SAVEM"
2020 LINEINPUT"TITRE ?";A$:IF A$="" THEN 100
2030 INPUT"ADR DEB (HEXA) ?";B$:IF B$="" THEN 100
2040 INPUT"ADR FIN (HEXA) ?";C$:IF C$="" THEN 100
2050 INPUT"ADR EXEC(HEXA) ?";D$:IF D$="" THEN 100
2060 Y=CSRLIN:PRINT:PRINT"GOTO10::PRINT"SAVEM"G$A$G$",&H"BS",&H"CS",&H"DX$":LOCATE 0,Y:GOTO 8500
2990 '
3000 '- LOADM",,R
3010 PRINT US$"LOADM"
3020 LINEINPUT"TITRE ?";A$
3030 PRINT"OFFSET (HEXA) ? 0":LOCATE 14,CSRLIN:INPUT B$:IF B$="" THEN 3060
3040 C$="":IF LEFT$(B$,1)="-" THEN C$="-":B$=MID$(B$,2)
3050 B$=","+C$+"&H"+B$
3060 PRINT"AUTO RUN ? N":LOCATE 9,CSRLIN:INPUT C$:IF C$="" OR LEFT$(C$,1)="N" THEN C$="" ELSE IF LEFT$(C$,1)="O" THEN C$=",R": IF B$="" THEN B$="," ELSE ELSE PRINT US$:GOTO 3060
3070 Y=CSRLIN:PRINT:PRINT"GOTO10::PRINT"LOADM"G$A$G$B$C$X$":LOCATE 0,Y:GOTO 8500
3990 '
4000 '- TFR @DEB,@FIN,@DEB2
4010 INPUT"ADR DEB ORG (HEXA) ?";A$:IF A$="" THEN 100
4020 INPUT"ADR FIN ORG (HEXA) ?";B$:IF B$="" THEN 100
4030 INPUT"ADR DEB TFR (HEXA) ?";C$:IF C$="" THEN 100
4050 A=VAL("&H"+A$):B=VAL("&H"+B$):C=VAL("&H"+C$)
4100 IF B<A THEN PLAY"L6FA":GOTO 100
4110 IF A>C THEN FOR I=0 TO B-A:POKE C+I,PEEK(A+I):NEXT:GOTO 4200
4120 FOR I=B-A TO 0 STEP-1:POKE C+I,PEEK(A+I):NEXT
4200 PLAY"LDOREMI":GOTO 100
7990 '
8000 '-S/P &H ,SI ERREUR F=-1
8010 L=LEN(A$):F=0:IF L<2 OR L>4 THEN 8100
8020 A=VAL("&H"+A$):B$=HEX$(A)
8030 FOR II=1 TO L-LEN(B$):B$="0"+B$:NEXT
8050 IF A$=B$ THEN RETURN
8100 F=-1:PLAY"L6FA":RETURN
8490 '
8500 COLOR 2,0:CONSOLE 0:Y=CSRLIN:PSET(0,0)X$:LOCATE 0,Y:END:GOTO 10

```

# Un jeu d'arcades pour le Canon X 07

Dans ce grand classique, vous survolez une ville. Mais voilà, votre altitude diminue inexorablement : pour atterrir sans percuter un immeuble, une seule solution : détruire cette ville, au moyen d'un canon placé devant votre avion, ainsi que de bombes.

Ce jeu est entièrement écrit en assembleur Z 80, ce qui lui donne une certaine rapidité, mais qui rend également sa transposition sur un autre micro-ordinateur assez délicate.

Au début du jeu, vous êtes situé(e) en haut à gauche de l'écran, et pouvez voir sous votre avion la ville qu'il vous faudra raser. Ce décor, afin de donner une certaine diversité au jeu, est chaque fois dessiné de manière aléatoire.

Vous avancez automatiquement vers la droite, et à peine avez-vous atteint le bord droit de l'écran que vous réapparez à gauche, mais un cran plus bas ! Vous continuez ainsi jusqu'à ce que vous ayez atteint le bas de l'écran ou bien que vous soyiez entré en collision avec un gratte-ciel, auquel cas la partie se finirait par un affichage de votre score ainsi que du record.

Pour détruire cette ville hostile, vous disposez d'un canon et de bombes. Le premier vous permet de tirer droit devant vous, tandis que les secondes sont destinées à être larguées dans votre sillage. Cependant, afin de rendre le jeu plus difficile, sachez que vous ne pouvez larguer une seconde bombe que lorsque la première aura explosé. Il en va de même pour les obus, qui ne peuvent être tirés que l'un après l'autre. Choisissez donc judicieusement le moment du tir !

Au début du jeu, vous pour-

rez choisir vos touches d'action. Ce faisant, si vous commettez une erreur, vous pourrez recommencer en pressant « CTRL Y ». Sachez enfin que le retour au Basic se fait lors de l'appui sur la touche « OFF ».

Après avoir entré ce programme, vous pouvez le sauver puis le charger en utilisant respectivement les options « S » et « L » du désassemblateur d'E. Sander, paru dans le n° 42 du mois de mai.

Afin de simplifier l'entrée initiale du programme, nous vous proposons **figure 1** un petit chargeur écrit en Basic qui permet la saisie contrôlée des codes hexadécimaux fournis **figure 2**. Rappelez-vous que ces contrôles se font par la somme des valeurs entrées, ce qui ne permet pas de détecter deux erreurs ou les inversions de caractères qui s'annuleraient. Toutefois ce système fiable ayant déjà fait ses preuves, nous y restons attachés.

Pour les fanatiques du langage machine, nous proposons **figure 3** le listing des mnémoniques, leur permettant ainsi des adaptations de leur cru. Pour leur faciliter la tâche, la **figure 4** présente la structure interne du programme. ■

```
5000 'CHARGEUR HEXA
10000 A=&H16D8
15000 PRINT HEX$(A)
20000 S=0
25000 FOR X=1 TO 8
30000 LINEINPUT A$
35000 W=VAL("&H"+A$)
40000 POKE A,W
45000 A=A+1
50000 S=S+W
55000 NEXT X
60000 INPUT" SOMME ";S2
65000 IF S=S2 THEN GOTO 20000 ELSE A=A-8
:GOTO 15000
```

Fig. 1. – Listing du chargeur Basic assurant l'entrée contrôlée du programme en codes hexadécimaux.

## JEU : BOMBER d'Alain RITOUX

**Aux commandes d'un chasseur-bombardier, vous survolez une ville. Pourrez-vous la détruire assez rapidement pour atterrir sans problèmes ?**

**Langage : langage machine Z 80  
Ordinateur : Canon X 07**

### LE PROGRAMME

#### Les variables

|             |                                  |
|-------------|----------------------------------|
| \$1600.1601 | Base de nombres aléatoires       |
| \$1602      | KEY\$                            |
| \$1604.1605 | Position du bombardier           |
| \$1606.1607 | Position de la balle             |
| \$1608.1609 | Position de la bombe             |
| \$160A      | Etat de la bombe                 |
| \$160C      | Etat de la balle                 |
| \$160E      | Fire key                         |
| \$1610      | Bomb key                         |
| \$1612.1613 | Score                            |
| \$1614.1615 | High-score                       |
| \$1616.1624 | Nombres aléatoires pour le décor |

#### La structure interne du programme

|             |  |
|-------------|--|
| \$1800.181B | Initialisation du X 07                               |
| \$181C.1828 | Initialisation de la base aléatoire                  |
| \$1829.183F | Création d'un nombre aléatoire                       |
| \$1840.1852 | Impression de messages                               |
| \$1853.185D | Initialisation du test clavier                       |
| \$185E.186A | Test clavier   |
| \$186D.188F | Explosion d'une bombe                                |
| \$1892.18A4 | Bruit du tir   |
| \$18A5.18B6 | Redéfinition des caractères                          |
| \$18B7.1903 | Routine du choix des touches                         |
| \$1906.1939 | Annulation du déclic et de la répétition des touches |
| \$193A.1962 | Décor  |
| \$1963.1990 | Décor (suite)  |
| \$1991.199B | Initialisation des variables                         |
| \$199C.19B2 | Bruit d'explosion d'un immeuble                      |
| \$19B3.19BE | Suite de la redéfinition des caractères spéciaux     |
| \$19BF.19D4 | Screen   |
| \$19D5.19E3 | Calcul du déplacement d'une bombe                    |
| \$19E5.1A08 | Calcul (cas spécial)                                 |
| \$1A09.1A1A | Calcul du déplacement d'une balle                    |
| \$1A1C.1A2F | Calcul (cas spécial)                                 |
| \$1A30.1A74 | Initialisation générale                              |
| \$1A75.1AE4 | Déplacement d'une bombe                              |
| \$1AE7.BB5D | Déplacement d'une balle                              |
| \$1B60.1B84 | Calcul du déplacement du bombardier                  |
| \$1B87.1BB1 | Déplacement du bombardier                            |
| \$1BB4.1C17 | Impression du score et du high-score                 |

Fig. 4. – Descriptif de la structure des logiciels, mettant en évidence les routines et les zones de variables.

|      |    |    |    |    |    |    |    |    |    |      |     |
|------|----|----|----|----|----|----|----|----|----|------|-----|
| 16D8 | 00 | 80 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | :    | 128 |
| 16E0 | 00 | 00 | 48 | 49 | 47 | 48 | 2D | 53 | :  | 416  |     |
| 16E8 | 43 | 4F | 52 | 45 | 20 | 20 | 3A | 20 | :  | 451  |     |
| 16F0 | 46 | 49 | 52 | 45 | 20 | 20 | 3A | 20 | :  | 448  |     |
| 16F8 | 42 | 4F | 4D | 42 | 20 | 20 | 3A | 20 | :  | 442  |     |
| 1700 | 88 | 00 | 00 | 20 | 00 | 00 | 00 | 00 | :  | 168  |     |
| 1708 | 00 | 89 | 00 | 00 | 00 | 00 | 00 | 00 | :  | 137  |     |
| 1710 | 20 | 00 | 83 | FF | FF | FF | FF | FF | :  | 1438 |     |
| 1718 | FF | FF | FF | 82 | 00 | 00 | 00 | 00 | :  | 895  |     |
| 1720 | FF | FF | FF | FF | 84 | 80 | FB | FF | :  | 1286 |     |
| 1728 | 00 | 00 | 00 | 00 | 00 | 85 | 00 | 00 | :  | 133  |     |
| 1730 | 00 | 00 | 80 | FB | FF | 00 | 86 | 80 | :  | 896  |     |
| 1738 | FB | FF | 00 | FF | FF | FF | FF | 8D | :  | 1667 |     |
| 1740 | 80 | FB | FF | 00 | 00 | 00 | 20 | 00 | :  | 666  |     |
| 1748 | 8A | 00 | 00 | 20 | 00 | FF | FF | FF | :  | 935  |     |
| 1750 | FF | 43 | 48 | 4F | 4F | 53 | 45 | 20 | :  | 736  |     |
| 1758 | 59 | 4F | 55 | 52 | 20 | 4B | 45 | 59 | :  | 600  |     |
| 1760 | 2A | 20 | 20 | 20 | 2A | 2A | 20 | 42 | :  | 320  |     |
| 1768 | 4F | 4D | 42 | 45 | 52 | 20 | 2A | 2A | :  | 489  |     |
| 1770 | 20 | 20 | 20 | 2A | 20 | 20 | 20 | 20 | :  | 266  |     |
| 1778 | 20 | 20 | 20 | 7E | 7E | 7E | 7E | 7E | :  | 726  |     |
| 1780 | 7E | 20 | 20 | 20 | 20 | 20 | 20 | 20 | :  | 350  |     |
| 1788 | 43 | 6F | 70 | 79 | 72 | 69 | 67 | 68 | :  | 837  |     |
| 1790 | 74 | 28 | 43 | 29 | 20 | 31 | 39 | 38 | :  | 458  |     |
| 1798 | 34 | 20 | 42 | 79 | 20 | 20 | 20 | 20 | :  | 399  |     |
| 17A0 | 20 | 41 | 2E | 20 | 20 | 52 | 49 | 54 | :  | 446  |     |
| 17A8 | 4F | 55 | 58 | 20 | 20 | 20 | 20 | 20 | :  | 412  |     |
| 17B0 | 46 | 2E | 4D | 41 | 55 | 43 | 48 | 41 | :  | 547  |     |
| 17B8 | 4D | 50 | 20 | 4A | 2E | 4F | 55 | 54 | :  | 557  |     |
| 17C0 | 48 | 49 | 45 | 52 | 45 | 2E | 53 | 41 | :  | 559  |     |
| 17C8 | 4E | 44 | 45 | 52 | 20 | 50 | 2E | 47 | :  | 526  |     |
| 17D0 | 55 | 49 | 4F | 43 | 48 | 4F | 4E | 20 | :  | 565  |     |
| 17D8 | 59 | 2E | 42 | 45 | 4C | 54 | 43 | 48 | :  | 569  |     |
| 17E0 | 45 | 4E | 4B | 4F | 20 | 20 | 4D | 45 | :  | 511  |     |
| 17E8 | 52 | 43 | 49 | 20 | 20 | 44 | 45 | 20 | :  | 455  |     |
| 17F0 | 56 | 4F | 54 | 52 | 45 | 20 | 53 | 4F | :  | 594  |     |
| 17F8 | 55 | 54 | 49 | 45 | 4E | 20 | 20 | 20 | :  | 485  |     |
| 1800 | 21 | DB | C6 | 11 | 00 | 00 | 01 | AE | :  | 642  |     |
| 1808 | 00 | ED | B0 | CD | BD | C0 | AF | 32 | :  | 1224 |     |
| 1810 | 2B | 00 | DB | F4 | E6 | FE | D3 | F4 | :  | 1445 |     |
| 1818 | CD | 9E | CE | C9 | ED | 5F | 26 | 6F | :  | 1251 |     |
| 1820 | 5E | 23 | 56 | ED | 53 | 00 | 16 | C9 | :  | 758  |     |
| 1828 | 2A | 00 | 16 | 5D | 54 | 29 | 29 | 29 | :  | 364  |     |
| 1830 | 29 | 29 | 29 | A7 | ED | 52 | 16 | 00 | :  | 631  |     |
| 1838 | ED | 5F | 5F | 19 | 22 | 00 | 16 | C9 | :  | 709  |     |
| 1840 | 21 | 60 | 17 | 18 | 03 | 21 | B0 | 17 | :  | 411  |     |
| 1848 | 06 | 4F | 7E | CD | BE | C1 | 23 | 10 | :  | 850  |     |
| 1850 | F9 | 76 | C9 | 21 | 3D | 00 | F3 | 36 | :  | 959  |     |
| 1858 | 5E | 23 | 36 | 18 | FB | C9 | F5 | DB | :  | 1123 |     |
| 1860 | F1 | 32 | 02 | 16 | FE | 04 | CA | C3 | :  | 970  |     |
| 1868 | C3 | F1 | C3 | 99 | C7 | 3E | FF | D3 | :  | 1511 |     |
| 1870 | F4 | AF | D3 | F3 | 0E | 06 | 3E | 80 | :  | 1083 |     |
| 1878 | D3 | F2 | 3E | FF | D3 | F4 | 06 | 40 | :  | 1295 |     |
| 1880 | 10 | FE | AF | D3 | F4 | 3E | 0F | 06 | :  | 983  |     |
| 1888 | FF | 10 | FE | 3D | 20 | F9 | 0D | 20 | :  | 912  |     |
| 1890 | E9 | C9 | 3E | FF | D3 | F4 | AF | D3 | :  | 1592 |     |

|      |    |    |    |    |    |    |    |    |   |      |
|------|----|----|----|----|----|----|----|----|---|------|
| 1898 | F3 | 06 | 30 | 10 | FE | D3 | F2 | 3C | : | 1080 |
| 18A0 | 20 | F7 | D3 | F4 | C9 | 21 | 00 | 17 | : | 991  |
| 18A8 | 06 | 09 | C5 | 01 | 00 | 09 | 3E | 1A | : | 310  |
| 18B0 | CD | 2F | C9 | C1 | 10 | F4 | C9 | 21 | : | 1140 |
| 18B8 | 51 | 17 | 06 | 0F | CD | 9E | CE | 7E | : | 820  |
| 18C0 | CD | BE | C1 | 23 | 10 | F9 | 3E | 0D | : | 963  |
| 18C8 | CD | BE | C1 | 3E | 0A | CD | BE | C1 | : | 1248 |
| 18D0 | 21 | F0 | 16 | 06 | 08 | CD | 4A | 18 | : | 612  |
| 18D8 | DB | F1 | CD | BE | C1 | 32 | 0E | 16 | : | 1134 |
| 18E0 | 3E | 0D | CD | BE | C1 | 3E | 0A | CD | : | 940  |
| 18E8 | BE | C1 | 06 | 08 | CD | 4A | 18 | DB | : | 919  |
| 18F0 | F1 | CD | BE | C1 | 32 | 10 | 16 | 76 | : | 1035 |
| 18F8 | DB | F1 | FE | 19 | C0 | CD | BD | C0 | : | 1517 |
| 1900 | CD | 9E | CE | C3 | B7 | 18 | 01 | 00 | : | 972  |
| 1908 | 00 | 3E | B3 | CD | 2F | C9 | 3E | B8 | : | 940  |
| 1910 | 01 | 00 | 00 | CD | 2F | C9 | ED | 57 | : | 778  |
| 1918 | E8 | E5 | AF | D3 | F4 | 32 | 2B | 00 | : | 1184 |
| 1920 | CD | 9E | CE | CD | BD | C0 | DB | B4 | : | 1554 |
| 1928 | CD | 48 | F5 | 20 | 06 | 06 | 65 | 80 | : | 795  |
| 1930 | C3 | 1B | 19 | E1 | CB | 1F | DC | BE | : | 1116 |
| 1938 | C1 | C9 | 21 | 16 | 16 | E5 | 06 | 04 | : | 710  |
| 1940 | CD | 29 | 18 | D1 | EB | 7B | E6 | 0F | : | 1082 |
| 1948 | 77 | 23 | E5 | 10 | F3 | E1 | 11 | 16 | : | 906  |
| 1950 | 16 | 06 | 09 | 1A | 4F | 81 | 81 | A7 | : | 567  |
| 1958 | 20 | 01 | 3C | E6 | 07 | 23 | 77 | 13 | : | 503  |
| 1960 | 10 | F1 | C9 | 0E | 01 | 69 | 26 | 04 | : | 620  |
| 1968 | 22 | B8 | 00 | 21 | 16 | 16 | 06 | 0E | : | 315  |
| 1970 | 7E | D6 | 02 | 77 | 23 | FA | 7C | 19 | : | 895  |
| 1978 | 3E | 20 | 18 | 09 | 3C | 20 | 04 | 3E | : | 285  |
| 1980 | 82 | 18 | 02 | 3E | 83 | CD | BE | C1 | : | 932  |
| 1988 | 10 | E6 | 0C | 3E | 05 | B9 | 20 | D5 | : | 755  |
| 1990 | C9 | 21 | 00 | 16 | 06 | 30 | 36 | 00 | : | 364  |
| 1998 | 23 | 10 | FB | C9 | 3E | FF | D3 | F4 | : | 1275 |
| 19A0 | AF | D3 | F3 | 3E | 90 | D3 | F2 | 3E | : | 1350 |
| 19A8 | 30 | 06 | FF | 10 | FE | 3D | 20 | F9 | : | 921  |
| 19B0 | D3 | F4 | C9 | 21 | D9 | 16 | 01 | 00 | : | 929  |
| 19B8 | 09 | 3E | 1A | CD | 2F | C9 | C9 | C5 | : | 948  |
| 19C0 | E5 | 45 | 4C | 21 | 00 | 00 | 11 | 14 | : | 444  |
| 19C8 | 00 | 19 | 10 | FD | 09 | 11 | FF | 01 | : | 576  |
| 19D0 | 19 | EB | E1 | C1 | C9 | 2A | 04 | 16 | : | 947  |
| 19D8 | 22 | 08 | 16 | CD | BF | 19 | 1A | E6 | : | 741  |
| 19E0 | 01 | 0F | 3C | 18 | 12 | 2A | 08 | 16 | : | 190  |
| 19E8 | CD | BF | 19 | 1A | E6 | F6 | 22 | B8 | : | 1141 |
| 19F0 | 00 | CD | BE | C1 | 3A | 0A | 16 | 47 | : | 749  |
| 19F8 | AF | 88 | F2 | FE | 19 | 23 | 22 | 08 | : | 909  |
| 1A00 | 16 | EE | 80 | F6 | 01 | 32 | 0A | 16 | : | 717  |
| 1A08 | C9 | 2A | 04 | 16 | 22 | 06 | 16 | CD | : | 536  |
| 1A10 | BF | 19 | 1A | E6 | 01 | 0F | 3C | 32 | : | 598  |
| 1A18 | 0C | 16 | 18 | 0F | 2A | 06 | 16 | CD | : | 348  |
| 1A20 | BF | 19 | 1A | E6 | F6 | 22 | B8 | 00 | : | 936  |
| 1A28 | CD | BE | C1 | 24 | 22 | 06 | 16 | C9 | : | 887  |
| 1A30 | F3 | CD | 00 | 18 | CD | 91 | 19 | CD | : | 1052 |
| 1A38 | 1C | 18 | CD | 29 | 18 | CD | 53 | 18 | : | 634  |
| 1A40 | CD | A5 | 18 | CD | B3 | 19 | CD | 06 | : | 1014 |
| 1A48 | 19 | CD | 40 | 18 | CD | 9E | CE | DB | : | 1106 |
| 1A50 | F1 | FE | 1B | CC | 45 | 18 | CD | B7 | : | 1207 |

Fig. 2. – Liste des codes hexadécimaux du jeu lui-même.

|      |                         |   |      |      |              |      |               |
|------|-------------------------|---|------|------|--------------|------|---------------|
| 1A58 | 18 CD 9E CE CD 3A 19 CD | : | 1086 | 1800 | LD hl,C6DB   | 1876 | LD a,80       |
| 1A60 | 63 19 21 01 01 22 04 16 | : | 219  | 1803 | LD de,0000   | 1878 | OUT (F2),a    |
| 1A68 | 22 B8 00 3E 84 CD BE C1 | : | 1000 | 1806 | LD bc,00AE   | 187A | LD a,FF       |
| 1A70 | AF 32 25 16 76 21 10 16 | : | 473  | 1809 | LDIR         | 187C | OUT (F4),a    |
| 1A78 | 3A 02 16 BE 20 10 AF 32 | : | 545  | 180B | CALL C0BD    | 187E | LD b,40 0     |
| 1A80 | 02 16 3A 0A 16 CB 47 20 | : | 420  | 180E | XOR a        | 1880 | DJNZ 1880 FE  |
| 1A88 | 05 CD D5 19 18 0A 3A 0A | : | 550  | 180F | LD (002B),a  | 1882 | XOR a         |
| 1A90 | 16 CB 47 28 52 CD E5 19 | : | 877  | 1812 | IN a,(F4)    | 1883 | OUT (F4),a    |
| 1A98 | 7D FE 05 20 06 AF 32 0A | : | 657  | 1814 | AND FE       | 1885 | LD a,0F       |
| 1AA0 | 16 18 44 3A 0A 16 CB 27 | : | 446  | 1816 | OUT (F4),a   | 1887 | LD b,FF       |
| 1AA8 | 3E 44 17 47 2A 08 16 CD | : | 501  | 1818 | CALL CE9E    | 1889 | DJNZ 1889 FE  |
| 1AB0 | BF 19 1A B0 E6 8F 22 B8 | : | 1009 | 181B | RET          | 188B | DEC a         |
| 1AB8 | 00 CB 4F 28 27 CB 47 28 | : | 675  | 181C | LD a,r       | 188C | JR NZ,1887 F9 |
| 1AC0 | 23 E5 2A 12 16 23 22 12 | : | 433  | 181E | LD h,6F 0    | 188E | DEC c         |
| 1AC8 | 16 E1 1A 3D CB 87 21 0A | : | 715  | 1820 | LD e,(hl)    | 188F | JR NZ,187A E9 |
| 1AD0 | 16 CB 4E 20 06 B0 F5 CB | : | 965  | 1821 | INC hl       | 1891 | RET           |
| 1AD8 | CE 18 05 F5 AF 32 0A 16 | : | 737  | 1822 | LD d,(hl)    | 1892 | LD a,FF       |
| 1AE0 | CD 9C 19 F1 CD BE C1 21 | : | 1248 | 1823 | LD (1600),de | 1894 | OUT (F4),a    |
| 1AE8 | 0E 16 3A 02 16 BE 06 02 | : | 316  | 1827 | RET          | 1896 | XOR a         |
| 1AF0 | 20 10 AF 32 02 16 3A 0C | : | 367  | 1828 | LD hl,(1600) | 1897 | OUT (F3),a    |
| 1AF8 | 16 CB 47 20 05 CD 09 1A | : | 573  | 182B | LD e,l       | 1899 | LD b,30 0     |
| 1B00 | 18 14 3A 0C 16 CB 47 28 | : | 450  | 182C | LD d,h       | 189B | DJNZ 189B FE  |
| 1B08 | 57 CB 4F 28 06 AF 32 0C | : | 652  | 182D | ADD hl,hl    | 189D | OUT (F2),a    |
| 1B10 | 16 18 4D CD 1C 1A 7C FE | : | 260  | 182E | ADD hl,hl    | 189F | INC a         |
| 1B18 | 14 20 06 AF 32 0C 16 18 | : | 341  | 182F | ADD hl,hl    | 18A0 | JR NZ,1899 F7 |
| 1B20 | 3F 3A 0C 16 CB 27 3E 44 | : | 527  | 1830 | ADD hl,hl    | 18A2 | OUT (F4),a    |
| 1B28 | 17 4F 2A 06 16 CD BF 19 | : | 593  | 1831 | ADD hl,hl    | 18A4 | RET           |
| 1B30 | 1A B1 E6 8F 22 B8 00 CB | : | 997  | 1832 | ADD hl,hl    | 18A5 | LD hl,1700    |
| 1B38 | 4F 28 06 CB 47 28 02 18 | : | 465  | 1833 | AND a        | 18A8 | LD b,09       |
| 1B40 | 07 CD BE C1 10 CD 18 18 | : | 864  | 1834 | SBC hl,de    | 18AA | PUSH bc       |
| 1B48 | E5 2A 12 16 23 22 12 16 | : | 420  | 1836 | LD d,00      | 18AB | LD bc,0900    |
| 1B50 | E1 1A 3D CB 87 CD BE C1 | : | 1238 | 1838 | LD a,r       | 18AE | LD a,1A       |
| 1B58 | CD 9C 19 3E 03 32 0C 16 | : | 535  | 183A | LD e,a       | 18B0 | CALL C92F     |
| 1B60 | CD BD C0 2A 04 16 CD BF | : | 1050 | 183B | ADD hl,de    | 18B3 | POP bc        |
| 1B68 | 19 1A 22 B8 00 E6 FA CD | : | 954  | 183C | LD (1600),hl | 18B4 | DJNZ 18AA F4  |
| 1B70 | BE C1 24 3E 14 BC 20 0F | : | 736  | 183F | RET          | 18B6 | RET           |
| 1B78 | 26 01 3A 25 16 CB 7F 28 | : | 526  | 1840 | LD hl,1760   | 18B7 | LD hl,1751    |
| 1B80 | 01 2C EE 80 32 25 16 3A | : | 578  | 1843 | JR 1848 03   | 18BA | LD b,0F       |
| 1B88 | 25 16 CB 27 3E 42 17 22 | : | 486  | 1845 | LD hl,17B0   | 18BC | CALL CE9E     |
| 1B90 | 04 16 4F 84 85 FE 9C 28 | : | 820  | 1848 | LD b,4F 0    | 18BF | LD a,(hl)     |
| 1B98 | 1B CD BF 19 1A B1 E6 8F | : | 1024 | 184A | LD a,(hl)    | 18C0 | CALL C1BE     |
| 1BA0 | CB 4F 28 07 CB 47 28 03 | : | 646  | 184B | CALL C1BE    | 18C3 | INC hl        |
| 1BA8 | A7 18 0A 22 B8 00 CD BE | : | 814  | 184E | INC hl       | 18C4 | DJNZ 18BF F9  |
| 1BB0 | C1 C3 D2 1B 37 F5 CD 9E | : | 1288 | 184F | DJNZ 184A F9 | 18C6 | LD a,0D       |
| 1BB8 | CE 06 09 21 E7 16 7E CD | : | 838  | 1851 | HALT         | 18C8 | CALL C1BE     |
| 1BC0 | BE C1 23 10 F9 2A 12 16 | : | 765  | 1852 | RET          | 18CB | LD a,0A       |
| 1BC8 | CD 98 BB 76 F1 DA 59 1A | : | 1236 | 1853 | LD hl,003D   | 18CD | CALL C1BE     |
| 1BD0 | 18 0C 3E 30 06 00 10 FE | : | 422  | 1856 | DI           | 18D0 | LD hl,16F0    |
| 1BD8 | 3D 20 F9 C3 75 1A 3E 0D | : | 755  | 1857 | LD (hl),5E ^ | 18D3 | LD b,08       |
| 1BE0 | CD BE C1 3E 0A CD BE C1 | : | 1248 | 1859 | INC hl       | 18D5 | CALL 184A     |
| 1BE8 | 06 0E 21 E2 16 7E CD BE | : | 822  | 185A | LD (hl),18   | 18D8 | IN a,(F1)     |
| 1BF0 | C1 23 10 F9 2A 14 16 ED | : | 814  | 185C | EI           | 18DA | CALL C1BE     |
| 1BF8 | 5B 12 16 A7 ED 52 38 02 | : | 675  | 185D | RET          | 18DD | LD (160E),a   |
| 1C00 | 19 EB EB 22 14 16 CD 98 | : | 928  | 185E | PUSH af      | 18E0 | LD a,0D       |
| 1C08 | BB 76 21 00 00 22 0A 16 | : | 404  | 185F | IN a,(F1)    | 18E2 | CALL C1BE     |
| 1C10 | 22 0C 16 22 12 16 76 C3 | : | 455  | 1861 | LD (1602),a  | 18E5 | LD a,0A       |
| 1C18 | 59 1A 00 00 00 00 00 00 | : | 115  | 1864 | CP 04        | 18E7 | CALL C1BE     |
|      |                         |   |      | 1866 | JP Z,C3C3    | 18EA | LD b,08       |
|      |                         |   |      | 1869 | POP af       | 18EC | CALL 184A     |
|      |                         |   |      | 186A | JP C799      | 18EF | IN a,(F1)     |
|      |                         |   |      | 186D | LD a,FF      | 18F1 | CALL C1BE     |
|      |                         |   |      | 186F | OUT (F4),a   | 18F4 | LD (1610),a   |
|      |                         |   |      | 1871 | XOR a        | 18F7 | HALT          |
|      |                         |   |      | 1872 | OUT (F3),a   | 18F8 | IN a,(F1)     |
|      |                         |   |      | 1874 | LD c,06      | 18FA | CP 19         |

Fig. 2. – Liste des codes hexadécimaux (suite et fin).

Fig. 3. – Le programme écrit en mnémonique permettra aux possesseurs d'un assemebleur d'entrer directement dans le jeu.

|       |                     |      |                    |      |                     |      |                     |
|-------|---------------------|------|--------------------|------|---------------------|------|---------------------|
| 18FC  | RET NZ              | 1974 | INC hl             | 19EB | LD $\alpha$ ,(de)   | 1A82 | LD $\alpha$ ,(160A) |
| 18FD  | CALL C0BD           | 1975 | JP M,197C          | 19EC | AND F6              | 1A85 | BIT 0, $\alpha$     |
| 1900  | CALL CE9E           | 1978 | LD $\alpha$ ,20    | 19EE | LD (00B8),hl        | 1A87 | JR NZ,1A8E 05       |
| 1903  | JP 18B7             | 197A | JR 1985 09         | 19F1 | CALL C1BE           | 1A89 | CALL 19D5           |
| 1906  | LD bc,0000          | 197C | INC $\alpha$       | 19F4 | LD $\alpha$ ,(160A) | 1A8C | JR 1A98 0A          |
| 1909  | LD $\alpha$ ,B3 2   | 197D | JR NZ,1983 04      | 19F7 | LD b, $\alpha$      | 1A8E | LD $\alpha$ ,(160A) |
| 190B  | CALL C92F           | 197F | LD $\alpha$ ,82    | 19F8 | XOR $\alpha$        | 1A91 | BIT 0, $\alpha$     |
| 190E  | LD $\alpha$ ,B8 2   | 1981 | JR 1985 02         | 19F9 | ADC $\alpha$ ,b     | 1A93 | JR Z,1AE2 52        |
| 1910  | LD bc,0000          | 1983 | LD $\alpha$ ,83    | 19FA | JP P,19FE           | 1A95 | CALL 19E5           |
| 1913  | CALL C92F           | 1985 | CALL C1BE          | 19FD | INC hl              | 1A98 | LD $\alpha$ ,l      |
| 1916  | LD $\alpha$ ,i      | 1988 | DJNZ 1970 E6       | 19FE | LD (1608),hl        | 1A99 | CP 05               |
| 1918  | RET PE              | 198A | INC c              | 1A01 | XOR 80              | 1A9B | JR NZ,1AA3 06       |
| 1919  | PUSH hl             | 198B | LD $\alpha$ ,05    | 1A03 | OR 01               | 1A9D | XOR $\alpha$        |
| 191A  | XOR $\alpha$        | 198D | CP c               | 1A05 | LD (160A), $\alpha$ | 1A9E | LD (160A), $\alpha$ |
| 191B  | OUT (F4), $\alpha$  | 198E | JR NZ,1965 D5      | 1A08 | RET                 | 1AA1 | JR 1AE2 44          |
| 191D  | LD (002B), $\alpha$ | 1990 | RET                | 1A09 | LD hl,(1604)        | 1AA3 | LD $\alpha$ ,(160A) |
| 1920  | CALL CE9E           | 1991 | LD hl,1600         | 1A0C | LD (1606),hl        | 1AA6 | SLA $\alpha$        |
| 1923  | CALL C0BD           | 1994 | LD b,30 0          | 1A0F | CALL 19BF           | 1AA8 | LD $\alpha$ ,44 D   |
| 1926  | IN $\alpha$ ,(B4)   | 1996 | LD (hl),00         | 1A12 | LD $\alpha$ ,(de)   | 1AAA | RLA                 |
| 1928  | CALL F548           | 1998 | INC hl             | 1A13 | AND 01              | 1AAB | LD b, $\alpha$      |
| 192B  | JR NZ,1933 06       | 1999 | DJNZ 1996 FB       | 1A15 | RRCA                | 1AAC | LD hl,(1608)        |
| 192D  | LD b,65 e           | 199B | RET                | 1A16 | INC $\alpha$        | 1AAF | CALL 19BF           |
| 192F  | ADD $\alpha$ ,b     | 199C | LD $\alpha$ ,FF    | 1A17 | LD (160C), $\alpha$ | 1AB2 | LD $\alpha$ ,(de)   |
| 1930  | JP 191B             | 199E | OUT (F4), $\alpha$ | 1A1A | JR 1A2B 0F          | 1AB3 | OR b                |
| 1933  | POP hl              | 19A0 | XOR $\alpha$       | 1A1C | LD hl,(1606)        | 1AB4 | AND 8F              |
| 1934  | RR $\alpha$         | 19A1 | OUT (F3), $\alpha$ | 1A1F | CALL 19BF           | 1AB6 | LD (00B8),hl        |
| 1936  | CALL C,C1BE         | 19A3 | LD $\alpha$ ,90    | 1A22 | LD $\alpha$ ,(de)   | 1AB9 | BIT 1, $\alpha$     |
| 1939  | RET                 | 19A5 | OUT (F2), $\alpha$ | 1A23 | AND F6              | 1ABB | JR Z,1AE4 27        |
| 193A  | LD hl,1616          | 19A7 | LD $\alpha$ ,30 0  | 1A25 | LD (00B8),hl        | 1ABD | BIT 0, $\alpha$     |
| 193D  | PUSH hl             | 19A9 | LD b,FF            | 1A28 | CALL C1BE           | 1ABF | JR Z,1AE4 23        |
| 193E  | LD b,04             | 19AB | DJNZ 19AB FE       | 1A2B | INC h               | 1AC1 | PUSH hl             |
| 193B  | CALL 1829           | 19AC | DEC $\alpha$ C ,FF | 1A2D | LD (1606),hl        | 1AC2 | LD hl,(1612)        |
| 1943  | POP de              | 19AE | JR NZ,19A9 F9      | 1A2F | RET                 | 1AC5 | INC hl              |
| 1944  | EX de,hl            | 19B0 | OUT (F4), $\alpha$ | 1A30 | DI                  | 1AC6 | LD (1612),hl        |
| 1945  | LD $\alpha$ ,e      | 19B2 | RET                | 1A31 | CALL 1800           | 1AC9 | POP hl              |
| 1946  | AND 07              | 19B3 | LD hl,16D9         | 1A34 | CALL 1991           | 1ACA | LD $\alpha$ ,(de)   |
| 1948  | LD (hl), $\alpha$   | 19B6 | LD bc,0900         | 1A37 | CALL 181C           | 1ACB | DEC $\alpha$        |
| 1949  | INC hl              | 19B9 | LD $\alpha$ ,1A    | 1A3A | CALL 1829           | 1ACC | RES 0, $\alpha$     |
| 194A  | PUSH hl             | 19BB | CALL C92F          | 1A3D | CALL 1853           | 1ACE | LD hl,160A          |
| 194B  | DJNZ 1940 F3        | 19BE | RET                | 1A40 | CALL 18A5           | 1AD1 | BIT 1,(hl)          |
| 194D  | POP hl              | 19BF | PUSH bc            | 1A43 | CALL 19B3           | 1AD3 | JR NZ,1ADB 06       |
| 194E  | LD de,1616          | 19C0 | PUSH hl            | 1A46 | CALL 1906           | 1AD5 | OR b                |
| 1951  | LD b,09             | 19C1 | LD b,l             | 1A49 | CALL 1840           | 1AD6 | PUSH $\alpha$ f     |
| 1953  | LD $\alpha$ ,(de)   | 19C2 | LD c,h             | 1A4C | CALL CE9E           | 1AD7 | SET 1,(hl)          |
| 1954  | LD c, $\alpha$      | 19C3 | LD hl,0000         | 1A4F | IN $\alpha$ ,(F1)   | 1AD9 | JR 1AE0 05          |
| 1955  | ADD $\alpha$ ,c     | 19C6 | LD de,0014         | 1A51 | CP 1B               | 1ADB | PUSH $\alpha$ f     |
| 1956  | ADD $\alpha$ ,c     | 19C9 | ADD hl,de          | 1A53 | CALL Z,1845         | 1ADC | XOR $\alpha$        |
| 1957  | AND $\alpha$        | 19CA | DJNZ 19C9 FD       | 1A56 | CALL 18B7           | 1ADD | LD (160A), $\alpha$ |
| 1958  | JR NZ,195B 01       | 19CC | ADD hl,bc          | 1A59 | CALL CE9E           | 1AE0 | CALL 199C           |
| 195A  | INC $\alpha$        | 19CD | LD de,01FF         | 1A5C | CALL 193A           | 1AE3 | POP $\alpha$ f      |
| 195B  | AND 07              | 19D0 | ADD hl,de          | 1A5F | CALL 1963           | 1AE4 | CALL C1BE           |
| 195D  | INC hl              | 19D1 | EX de,hl           | 1A62 | LD hl,0101          | 1AE7 | LD hl,160E          |
| 195E  | LD (hl), $\alpha$   | 19D2 | POP hl             | 1A65 | LD (1604),hl        | 1AEA | LD $\alpha$ ,(1602) |
| 195F  | INC de              | 19D3 | POP bc             | 1A68 | LD (00B8),hl        | 1AED | CP (hl)             |
| 1960  | DJNZ 1953 F1        | 19D4 | RET                | 1A6B | LD $\alpha$ ,84     | 1AEE | LD b,02             |
| 1962  | RET                 | 19D5 | LD hl,(1604)       | 1A6D | CALL C1BE           | 1AF0 | JR NZ,1B02 10       |
| 1963  | LD c,01             | 19D8 | LD (1608),hl       | 1A70 | XOR $\alpha$        | 1AF2 | XOR $\alpha$        |
| 1965  | LD l,c              | 19DB | CALL 19BF          | 1A71 | LD (1625), $\alpha$ | 1AF3 | LD (1602), $\alpha$ |
| 1966  | LD h,04             | 19DE | LD $\alpha$ ,(de)  | 1A74 | HALT                | 1AF6 | LD $\alpha$ ,(160C) |
| 1968  | LD (00B8),hl        | 19DF | AND 01             | 1A75 | LD hl,1610          | 1AF9 | BIT 0, $\alpha$     |
| 196B  | LD hl,1616          | 19E1 | RRCA               | 1A78 | LD $\alpha$ ,(1602) | 1AFB | JR NZ,1B02 05       |
| 196E  | LD b,0E             | 19E2 | INC $\alpha$       | 1A7B | CP (hl)             | 1AFD | CALL 1A09           |
| 1970  | LD $\alpha$ ,(hl)   | 19E3 | JR 19F7 12         | 1A7C | JR NZ,1A8E 10       | 1B00 | JR 1B16 14          |
| 1971. | SUB 02              | 19E5 | LD hl,(1608)       | 1A7E | XOR $\alpha$        | 1B02 | LD $\alpha$ ,(160C) |
| 1973  | LD (hl), $\alpha$   | 19E8 | CALL 19BF          | 1A7F | LD (1602), $\alpha$ | 1B05 | BIT 0, $\alpha$     |

Fig. 3. - Listing en assembleur (suite).

|      |                |
|------|----------------|
| 1B07 | JR Z, 1B60 57  |
| 1B09 | BIT 1, a       |
| 1B0B | JR Z, 1B13 06  |
| 1B0D | XOR a          |
| 1B0E | LD (160C), a   |
| 1B11 | JR 1B60 40     |
| 1B13 | CALL 1A1C      |
| 1B16 | LD a, h        |
| 1B17 | CP 15          |
| 1B19 | JR NZ, 1B21 06 |
| 1B1B | XOR a          |
| 1B1C | LD (160C), a   |
| 1B1F | JR 1B60 3F     |
| 1B21 | LD a, (160C)   |
| 1B24 | SLA a          |
| 1B26 | LD a, 44 D     |
| 1B28 | RLA            |
| 1B29 | LD c, a        |
| 1B2A | LD hl, (1606)  |
| 1B2D | CALL 19BF      |
| 1B30 | LD a, (de)     |
| 1B31 | OR c           |
| 1B32 | AND 8F         |
| 1B34 | LD (00B8), hl  |
| 1B37 | BIT 1, a       |
| 1B39 | JR Z, 1B41 06  |
| 1B3B | BIT 0, a       |
| 1B3D | JR Z, 1B41 02  |
| 1B3F | JR 1B48 07     |
| 1B41 | CALL C1BE      |
| 1B44 | DJNZ 1B13 CD   |
| 1B46 | JR 1B60 18     |
| 1B48 | PUSH hl        |
| 1B49 | LD hl, (1612)  |
| 1B4C | INC hl         |
| 1B4D | LD (1612), hl  |
| 1B50 | POP hl         |
| 1B51 | LD a, (de)     |
| 1B52 | DEC a          |
| 1B53 | RES 0, a       |
| 1B55 | CALL C1BE      |
| 1B58 | CALL 199C      |
| 1B5B | LD a, 03       |
| 1B5D | LD (160C), a   |
| 1B60 | CALL C0BD      |
| 1B63 | LD hl, (1604)  |
| 1B66 | CALL 19BF      |
| 1B69 | LD a, (de)     |
| 1B6A | LD (00B8), hl  |
| 1B6D | AND FA         |
| 1B6F | CALL C1BE      |
| 1B72 | INC h          |
| 1B73 | LD a, 14       |
| 1B75 | CP h           |
| 1B76 | JR NZ, 1B87 0F |
| 1B78 | LD h, 01       |
| 1B7A | LD a, (1625)   |
| 1B7D | BIT 2, a       |
| 1B7F | JR Z, 1B82 01  |
| 1B81 | INC I          |
| 1B82 | XOR 80         |
| 1B84 | LD (1625), a   |
| 1B87 | LD a, (1625)   |
| 1B8A | SLA a          |
| 1B8C | LD a, 42 B     |
| 1B8E | RLA            |
| 1B8F | LD (1604), hl  |
| 1B92 | LD c, a        |
| 1B93 | ADD a, h       |
| 1B94 | ADD a, l       |
| 1B95 | CP 9C          |
| 1B97 | JR Z, 1BB4 1B  |
| 1B99 | CALL 19BF      |
| 1B9C | LD a, (de)     |
| 1B9D | OR c           |
| 1B9E | AND 8F         |
| 1BA0 | BIT 1, a       |
| 1BA2 | JR Z, 1BAB 07  |
| 1BA4 | BIT 0, a       |
| 1BA6 | JR Z, 1BAB 03  |
| 1BA8 | AND a          |
| 1BA9 | JR 1BB5 0A     |
| 1BAB | LD (00B8), hl  |
| 1BAE | CALL C1BE      |
| 1BB1 | JP 1BD2        |
| 1BB4 | SCF            |
| 1BB5 | PUSH af        |
| 1BB6 | CALL CE9E      |
| 1BB9 | LD b, 09       |
| 1BBB | LD hl, 16E2    |
| 1BBE | LD a, (hl)     |
| 1BBF | CALL C1BE      |
| 1BC2 | INC hl         |
| 1BC3 | DJNZ 1BBE F9   |
| 1BC5 | LD hl, (1612)  |
| 1BC8 | CALL BB98      |
| 1BCB | HALT           |
| 1BCC | POP af         |
| 1BCD | JP C, 1A59     |
| 1BD0 | JR 1BDE 0C     |
| 1BD2 | LD a, 30 0     |
| 1BD4 | LD b, 00       |
| 1BD6 | DJNZ 1BD6 FE   |
| 1BD8 | DEC a          |
| 1BD9 | JR NZ, 1BD4 F9 |
| 1BDB | JP 1A75        |
| 1BDE | LD a, 0D       |
| 1BE0 | CALL C1BE      |
| 1BE3 | LD a, 0A       |
| 1BE5 | CALL C1BE      |
| 1BE8 | LD b, 0E       |
| 1BEA | LD hl, 16E2    |
| 1BED | LD a, (hl)     |
| 1BEE | CALL C1BE      |
| 1BF1 | INC hl         |
| 1BF2 | DJNZ 1BED F9   |
| 1BF4 | LD hl, (1614)  |
| 1BF7 | LD de, (1612)  |
| 1BFB | AND a          |
| 1BFC | SBC hl, de     |
| 1BFE | JR C, 1C02 02  |
| 1C00 | ADD hl, de     |
| 1C01 | EX de, hl      |
| 1C02 | EX de, hl      |
| 1C03 | LD (1614), hl  |
| 1C06 | CALL BB98      |
| 1C09 | HALT           |
| 1C0A | LD hl, 0000    |
| 1C0D | LD (160A), hl  |
| 1C10 | LD (160C), hl  |
| 1C13 | LD (1612), hl  |
| 1C16 | HALT           |
| 1C17 | JP 1A59        |

Fig. 3. – Listing en assembleur (suite et fin).

# DISQUES POUR TRS MODÈLES 3 & 4

## QUALITÉ

Pour cela, nous avons sélectionné :

- le meilleur contrôleur qui soit. Il vous permet de protéger 4 disques **5 ou 8 pouces**. Ses connexions plaquées or vous assurent une fiabilité à toute épreuve.

- TANDON, les disques les plus fiables et les plus performants, offrant un temps d'accès maximum de **5ms**.

De plus, l'assemblage, le montage et les tests individuels sont assurés par nos équipes compétentes (prévoir 48 heures).

## PIUSSANCE

Ne vous limitez pas à 175 Ko. par disquette.

Pour un faible supplément, équipez-vous de disquettes double face en 40 pistes (384 Ko.), ou en 80 pistes (768 Ko.). Ces unités peuvent être combinées de façon à satisfaire tous vos besoins, même si votre ordinateur est déjà équipé d'un disque constructeur.

## PRIX

configuration de base  
**disque 0 à**

**4.995 TTC**

En démonstration permanente chez

|                          |                      |
|--------------------------|----------------------|
| <b>MICRO-INFLUX</b>      | <b>SIVÉA</b>         |
| 20, rue Laennec          | La Croix du Palais   |
| 78330 FONTENAY-LE-FLEURY | 33081 BORDEAUX Cedex |
| (1) 460 07 53            | (56) 96 28 11        |

**GARANTIE 1 AN p. & m.o.**  
qualité oblige

Importation et Diffusion  
d'Équipement Micro-Informatique  
34 bis, rue Sorbier - 75020 PARIS  
Tél. : (1) 358.44.35



Importateur exclusif  
Recherchons des revendeurs  
sur toute la France

Demandez notre catalogue de produits pour Modèle III