3

La RAM-ÉCRAN : applications pratiques

A la périphérie de l'unité centrale, étudions plus particulièrement l'écran et sa gestion, de manière pratique. Nous passerons ensuite à la réalisation de programmes en assembleur, considérant que l'essentiel est maintenant acquis.

Ce qui suit n'est valable qu'en mode 40 colonnes, sur TO9. Pour les autres modes d'affichage, reportez-vous au chapitre P9, en fin d'ouvrage.

L'écran se compose de 320 points horizontaux pour une ligne et de 200 points verticaux pour une colonne, donc de 320*200 soit 64000 points ou « pixels » en tout. A raison d'un bit par point, il faut 64000/8 (8000) octets pour le gérer. Ces octets vont, sur TO7(70) et sur TO9, de &H4000* à &H5F40* exclu. Pour « allumer » un point d'un octet, à l'écran, il suffit « d'allumer » le bit correspondant.

Les 320 points horizontaux d'une ligne, regroupés par huit, correspondent à 40 segments de 8 points en mode-caractère, soit 40 colonnes. De même, n'existe-t-il que 200/8, soit 25 lignes à l'écran, pour l'affichage des caractères.

*Sur MO5 : de &H0000 à &H1F40 exclu.

Exemple :

128	64	32	16	8	4	2	1
0	0	0	0	1	0	0	0

octet &H5000 (TO7(70) et TO9) ou &H1000 (MO5) : un point à l'écran

Faites, en basic :

Sur TO7(70) ou TO9 :

```
POKE &H5000,8

POKE &H5028,170
```

Sur MO5 :

```
POKE &H1000,8

POKE &H1028,170
```

Vous avez en &H5000 (ou &H1000) : un point « allumé », et au-dessous en &H5028 (ou &H1028) : un pointillé.

Comme ceci :

valeur décimale	128	64	32	16	8	4	2	1	
&H5000	0	0	0	0	1	0	0	0	8
&H5028	1	0	1	0	1	0	1	0	

128 + 32 + 8 + 2 = 170

En effet la différence entre &H5000 et &H5028, soit &H28, équivaut à 40 en décimal, ce qui fait passer à l'octet immédiatement en dessous (40 octets par ligne).

Nous travaillons ici dans la forme, plus tard nous travaillerons dans la couleur.

Faites SCREEN 4, 0, 0, puis POKE &H5000, 8 et POKE &H5028, 170 (ou &H1000 et &H1028 sur MO5). Les bits « allumés » sont de la couleur-forme donnée par SCREEN 4, 0, 0, c'est-à-dire bleue (4), et les bits « éteints » sont de la couleur de fond, noire (0), donnée par la même instruction.

Pour afficher un caractère à l'écran, il faut une matrice de 8 octets superposés, soit 64 bits. Il est donc possible d'écrire (64000/64) 1 000 lettres sur la totalité de l'écran (40 colonnes * 25 lignes).

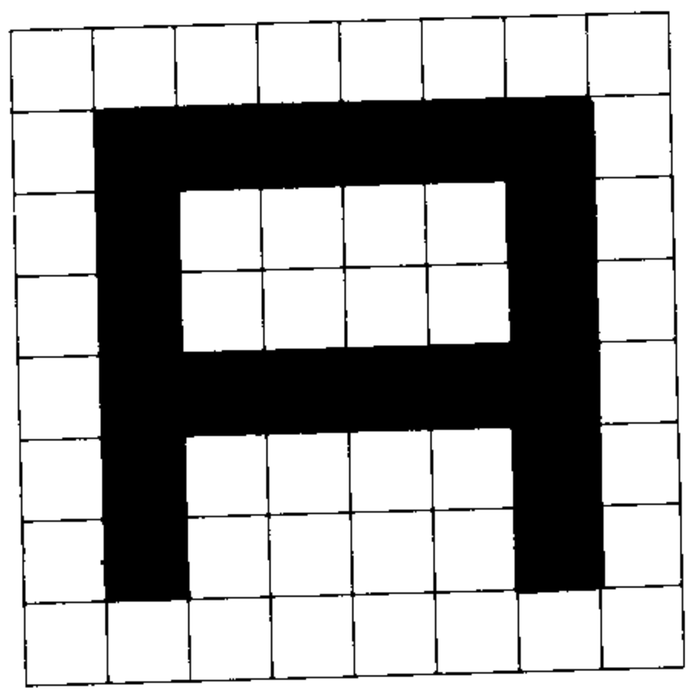
1. Graphisme de la lettre A :

En appuyant sur la touche A du clavier, vous donnez à votre micro-ordinateur un code en ASCII. Ce code lui permet d'aller chercher en mémoire morte, dans un générateur de caractères, la lettre A pour l'afficher.

Vous pouvez créer un A à l'écran, comme le générateur de caractères :

	128	64	32	16	8	4	2	1	adresses hexa :
octet n° 1	0	0	0	0	0	0	0	0	5000
octet n° 2	0	1	1	1	1	1	1	0	5028
octet n° 3	0	1	0	0	0	0	1	0	5050
octet n° 4	0	1	0	0	0	0	1	0	5078
octet n° 5	0	1	1	1	1	1	1	0	50A0
octet n° 6	0	1	0	0	0	0	1	0	50C8
octet n° 7	0	1	0	0	0	0	1	0	50F0
octet n° 8	0	0	0	0	0	0	0	0	5118

Graphisme :



0
126
66
66
126
66
66
0

Les adresses de cet exemple sont celles pour TO7(70) et TO9.

En fait, dans le générateur de la R.O.M., la lettre A s'écrit de cette façon, mais l'octet numéro 1 est celui du bas et il faut remonter jusqu'à l'octet numéro 8, qui est celui du haut.

Faites sur TO7(70) ou TO9 :

```
10 CLS
20 SCREEN 4,6,6                                    *
30 FOR I=&H5000 TO &H5118 STEP 40
40 READ A
50 POKE I,A
60 NEXT I
70 DATA 0,126,66,66,126,66,66,0
```

* Sur MO5, en ligne 30, mettez : FOR I = &H1000 TO &H1118 STEP 40

Nous reverrons ce programme en langage-machine, après les explications qui vont suivre et qui sont nécessaires à l'étude de cette application.

2. Mise en mémoire-forme ou en mémoire-couleur :

Ce qui suit n'est valable sur TO9 qu'en mode 40 colonnes. Pour les autres modes d'affichage, reportez-vous au chapitre P9, en fin d'ouvrage.

Le TO7-70 dispose de 16 couleurs comme le MO5. La gestion des couleurs du TO9 est identique à celle du TO7-70, la seule différence entre ces deux appareils étant la possibilité, pour le TO9, de sélectionner 16 couleurs simultanées parmi un éventail de 4096 couleurs. Le TO7 ne dispose quant à lui que de 8 couleurs.

Les couleurs sont obtenues par combinaison de 3 couleurs de base : le rouge, le vert et le bleu.

a. Commutation de la mémoire d'écran :

Les 8 000 octets de l'écran peuvent servir « deux fois », d'une part pour la forme des segments, d'autre part pour leur couleur. L'astuce du système est qu'il y a un « commutateur » mémoire-forme <—> mémoire-couleur. C'est le bit 0 de l'adresse &HE7C3 sur TO7(70) et TO9, &HA7C0 sur MO5.

Il suffit de mettre à 0 le bit numéro zéro de cette adresse pour passer en mémoire-couleur, et de remettre à 1 le même bit pour revenir en mémoire-forme (dite aussi mémoire-caractère).

Il est possible de faire : POKE &HE7C3, 0 ou POKE &HE7C3, 1, mais comme cette adresse possède également d'autres fonctions (couleur du tour sur TO7(70) par exemple), vous risquez d'obtenir des effets indésirables (voir tableau numéro 13). Il existe donc une solution consistant à se servir de l'algèbre de Boole (opérations logiques) pour créer ce que l'on appelle un masque, nous allons en reparler.

&HE7C3 est une adresse du système-PIA 6846 sur TO7(70). Il existe aussi par ailleurs un PIA 6821 sur cet appareil.

&HA7C0 est une adresse du système-PIA 6821 sur MO5, où le PIA 6846 n'existe pas.

Sur TO9 existent l'ACIA 6850 gérant le clavier, le PIA 6846, les PIA 6821, etc.. &HE7C3 est une adresse du PIA 6846.

PIA signifie : PERIPHERAL INTERFACE ADAPTER, c'est-à-dire un système communiquant avec l'extérieur (interface) en parallèle, alors

que l'ACIA est une interface travaillant de manière asynchrone et en série (ASYNCHRONOUS COMMUNICATIONS INTERFACE ADAPTER).

Voici la description complète des adresses mentionnées plus haut, pour TO7(70), TO9 et MO5 :

• TO7(70) :

– Description de l'adresse &HE7C3 :

• bit 0 (sortie) : Commutation mémoire-écran « caractère » et mémoire-écran « couleur ».
• bit 1 (entrée) : interrupteur du crayon optique.
• bit 2 (sortie) : TO7-70 : couleur du tour (pastel à 0).
• bit 3 (sortie) : L.E.D. ou diode du clavier (lumière rouge des minuscules).
• bit 4 (sortie) : couleur du tour, rouge (R).
• bit 5 (sortie) : couleur du tour, verte (V).
• bit 6 (sortie) : couleur du tour, bleue (B).
• bit 7 (entrée) : lecture d'une cassette.

Cette adresse de P.I.A permet donc, entre autres choses, la gestion des couleurs du tour de l'écran. Pour avoir un tour rouge, on peut faire POKE &HE7C3, &B00010101.

Sachez que les couleurs « s'additionnent » ainsi :
R + V = Jaune,
V + B = Cyan,
R + B = Magenta,
R + B + V = Blanc,

le noir étant l'absence de couleur.

• TO9 :

– *Description de l'adresse &HE7C3 :*

• bit 0 (sortie) : commutation mémoire-écran « caractère » et mémoire-écran « couleur ».
• bit 1 (entrée) : interrupteur du crayon optique.
• bit 2 (sortie) : sélection banque R.A.M. DISK (cf P9 : tableau BANK R.A.M.).
• bit 3 (sortie) : inutilisable.
• bit 4 (sortie) : sélection slot R.O.M. (cf P9 : BANK R.O.M.).
• bit 5 (sortie) : sélection slot R.O.M. (cf P9 : BANK R.O.M.).

• bit 6 (sortie) : sélection banque R.A.M. DISK (cf P9 : tableau BANK R.A.M.).
• bit 7 (entrée) : lecture d'une cassette.

Ce qui est dit sur le PIA-système 6821, dans le paragraphe sur le MO5 qui suit, est valable pour les TO7(70) et TO9.

• MO5 :

— *Description de l'adresse &HA7C0 :*

• bit 0 (sortie) : Commutation mémoire-écran « caractère » et mémoire-écran « couleur ».
• bit 1 (sortie) : Couleur du tour, rouge (R).
• bit 2 (sortie) : Couleur du tour, verte (V).
• bit 3 (sortie) : Couleur du tour, bleue (B).
• bit 4 (sortie) : Couleur du tour (pastel à 1).
• bit 5 (entrée) : interrupteur du crayon optique.
• bit 6 (sortie) : écriture sur cassette.
• bit 7 (entrée) : lecture d'une cassette.

Le PIA-système 6821 se compose de deux ports : A et B. Les ports, dont nous reparlerons ensuite, servent aux sorties-entrées de l'U.C., vers les périphériques ou inversement (voir tableau n° 13). Ce sont des « voies de passage » des données. Certains de leurs bits sont programmables.

Les couleurs « s'additionnent » comme il est dit plus haut pour le TO7(70).

b. Technique du masque :

• Forcer à 1 le bit numéro zéro de &HE7C3*, sans affecter la valeur des autres bits dont on ignore l'état. Mise donc, en mémoire-caractère :

&HE7C3 = % XXXXXXXX au départ.

''%'' signifie binaire en assembleur (ne peut être utilisé que dans le texte, et non avec la cartouche).

X : bit dont on ignore l'état (1 ou 0).

Choisissons la formule suivante :

```
10 Y = PEEK(&HE7C3) OR 1
20 POKE &HE7C3, Y
```

* Ceci est bien sûr valable sur MO5, où seule l'adresse change (&HA7C0).

Voici ce que l'on appelle la table de vérité de l'opération Booléenne logique OR (tableau numéro 10).

	0	1
0	0	1
1	1	1

0 V 0, donne : 0
0 V 1, donne : 1
1 V 0, donne : 1
1 V 1, donne : 1

Faisons donc l'opération logique avec OR 1 (ou inclusif), bit à bit :

```
&HE7C3 :    x  x  x  x  x  x  x  x
          v 0  0  0  0  0  0  0  1   (masque)
          ------------------------
RESULTAT :  x  x  x  x  x  x  x  1
```

Avec 0, les bits X restent intacts car 0 V 0 = 0 et 1 V 0 = 1.
Seul le bit numéro zéro de &HE7C3 est forcé à 1. L'opérande logique s'appelle un masque.

• Mettre à 0 le bit numéro zéro de &HE7C3, sans connaître l'état de l'ensemble des bits de cette adresse. Mise en mémoire-couleur :

Nous allons choisir l'opérateur logique AND.

Voici la table de vérité de AND :

	0	1
0	0	0
1	0	1

Faisons :

```
10 Y = PEEK(&HE7C3) AND 254
20 POKE &HE7C3, Y
```

Le résultat est : x x x x x x x 0

Sur MO5, nous verrons plus tard que la mise en mémoire-forme, ou en mémoire-couleur, peut être faite plus simplement.

Il ne reste plus qu'à bâtir de petits programmes en basic pour vérifier tout ceci.

- *1er exemple* :

« Allumer » un à un, du premier au dernier, les bits de chaque octet de l'ensemble de l'écran et ceci, pour l'instant, dans la couleur courante de forme pour les bits mis à 1, en laissant dans la couleur courante de fond les bits qui sont encore à 0. Pour bien voir l'apparition des bits, les octets seront « allumés » de droite à gauche. En BASIC 128, plus rapide, ajouter en ligne 75 : FOR A = 0 TO 30 : NEXT.

```
10 CLS:LOCATE 0,0,0
20 SCREEN 4,0,0
30 Y=PEEK(&HE7C3)* OR 1
40 POKE &HE7C3 ,Y
50 FOR I=&H4000* TO &H5F3F*
60 FOR J=0 TO 7
70 N=N+2↑J
80 POKE I,N
90 NEXT J
100 N=0
110 NEXT I
120 END
```

Vous pouvez faire un break (CNT-C) avant la fin du programme, car il est long, comme le suivant.

- *2e exemple* :

Quadrillage d'écran.

- *Description* :

« Allumer » les octets de l'écran, en pointillés, du premier au dernier, en changeant la couleur de fond et de forme pour chaque octet.

mémoire-caractère :

Pour faire un pointillé, il suffit de faire : POKE adresse-mémoire

* Sur MO5, &HE7C3 devient &HA7C0 ; &H4000 devient &H0000 ; &H5F3F devient &H1F3F.

d'écran, 153 (avec le bit numéro zéro de &HE7C3 ou de &HA7C0 (MO5) à 1).

153 = 128 + 16 + 8 + 1

1	0	0	1	1	0	0	1

mémoire-couleur :

Chaque octet, en mémoire-couleur (bit numéro zéro de &HE7C3 ou &HA7C0 à 0), peut être manipulé de la façon suivante :

- *TO7(70) et TO9* :

Fd	Fo	B	V	R	B	V	R
X	X						

Fo Fd

Fd = fond (aux extrêmes de l'octet).
Fo = Forme

. bit numéro 7 : à 0 sélectionne les couleurs pastel du fond, et à 1 les couleurs foncées du fond (blanc clair → orange).
. bit numéro 6 : à 0 sélectionne les couleurs pastel de la forme, et à 1 les couleurs foncées de la forme (blanc clair → orange).

Pour le reste, il suffit de lire et de se souvenir des « mélanges » de couleurs décrits précédemment.

Les couleurs pastel n'existent pas sur TO7.

- *MO5* :

P	B	V	R	P	B	V	R
X				X			

Fo Fd

P = bit de pastel (couleurs claires, mais blanc clair → orange). Ce sont les bits de poids fort de chaque quartet.

Fd = fond
Fo = forme

. bit numéro 7 : à 0 sélectionne les couleurs foncées de la forme, et à 1 les couleurs pastel de la forme.
. bit numéro 3 : à 0 sélectionne les couleurs foncées du fond, et à 1 les couleurs pastel du fond.

Pour le reste, il suffit de lire et de se souvenir des « mélanges » de couleurs décrits précédemment.

— *Réalisation du quadrillage d'écran* :

• *TO7(70) et TO9* :

En initialisant A, qui sera la variable « couleur-fond » et « couleur-forme », à 191, nous n'utiliserons depuis A + 1 = 192 que les couleurs vives ou dites foncées de l'appareil ; le graphisme en sera plus joli. Lorsque A sera égal à 256, on le réinitialisera à 192.

Faites NEW, puis :

```
10 CLS
20 SCREEN 4,0,0
30 A=191
40 FOR I=&H4000 TO &H5F3F
50 GOSUB 100' choix des couleurs
60 POKE I,153
70 NEXT I
80 END
100 Y= PEEK(&HE7C3) AND 254
110 POKE &HE7C3,Y
120 A=A+1: IF A=256 THEN A=192
130 POKE I,A
140 Y= PEEK(&HE7C3) OR 1
150 POKE &HE7C3,Y
160 RETURN
```

Le premier octet sera tout noir (192), etc..

1	1	0	0	0	0	0	0

Rien = noir, en mémoire-couleur

Fd Fo Fo Fd

• *MO5* :

En initialisant A, qui sera la variable « couleur-fond » et « couleur-forme », à 127, nous n'utiliserons depuis A + 1 = 128 que les couleurs vives ou dites foncées de la forme ; le graphisme en sera plus joli. Pour le fond, nous aurons tantôt des couleurs claires, tan-

tôt des couleurs foncées. Lorsque A sera égal à 256, on le réinitialisera à 128.

Faites NEW, puis :

```
10 CLS
20 SCREEN 4,0,0
30 A=127
40 FOR I= &H0000 TO &H1F3F
50 GOSUB 100' choix des couleurs
60 POKE I,153
70 NEXT I
80 END
100 Y= PEEK(&HA7C0) AND 254
110 POKE &HA7C0,Y
120 A=A+1: IF A=256 THEN A=128
130 POKE I,A
140 Y= PEEK(&HA7C0) OR 1
150 POKE &HA7C0,Y
160 RETURN
```

Nous verrons plus tard que ces programmes sont nettement plus rapides, quasi instantanés, en langage-machine.

3. Graphisme de A en langage-machine :

Revenons maintenant au graphisme de la lettre A. Le « programme » qui suit ne demande que des connaissances qui vous sont acquises. Il ne s'agit pas réellement d'un programme d'ailleurs, car sa structure linéaire n'est acceptable qu'à titre de démonstration.

a. TO7(70) et TO9 :

```
7D00                        ORG     $7D00
7D00 B6    E7C3             LDA     $E7C3
7D03 8A    01               ORA     #1
7D05 B7    E7C3             STA     $E7C3
7D08 86    00               LDA     #0
7D0A B7    5000             STA     $5000
7D0D 86    7E               LDA     #&126
7D0F B7    5028             STA     $5028
7D12 86    42               LDA     #&66
7D14 B7    5050             STA     $5050
7D17 86    42               LDA     #&66
7D19 B7    5078             STA     $5078
7D1C 86    7E               LDA     #&126
7D1E B7    50A0             STA     $50A0
7D21 86    42               LDA     #&66
7D23 B7    50C8             STA     $50C8
7D26 86    42               LDA     #&66
```

```
7D28 B7    50F0          STA     $50F0
7D2B 86    00            LDA     #0
7D2D B7    5118          STA     $5118
7D30 3F               FIN SWI
           0000          END

00000 Total Errors
```

b. MO5 :

Remplacez $E7C3 par $A7C0, et les adresses de $5000 à $5118 par les adresses de $1000 à $1118.
N'oubliez pas non plus de changer SWI par STOP.

c. Pour tous :

Ceux qui n'auraient pas lu le paragraphe sur les cartouches-assembleur sont invités à le faire pour comprendre les directives ORG et END, ainsi que l'interruption logicielle SWI sur TO7(70) et TO9, ou le break STOP sur MO5.

Pour lancer ce programme, après assemblage (A), il faut aller dans le moniteur et écrire G$7D00, puis faire ENTREE, sur TO7(70) et TO9 comme sur MO5.

FIN est une étiquette, inscrite dans la 1re zone réservée à cet effet.

d. Pour ceux qui n'ont pas de cartouche-assembleur :

- *TO7(70) et TO9* :

```
10 CLS
20 FOR I=&H7D00 TO &H7D00+48
30 READ A$
40 POKE I,VAL("&H"+A$)
50 NEXT I
60 EXEC &H7D00
70 DATA B6,E7,C3,8A,01,B7.E7,C3,86,00,B7,50,00,86,7E,B7,50,28,86,42,B7,50,5
0,86, 42,B7,50,78,86,7E,B7,50,A0,86,42,B7,50,C8,86,42,B7,50,F0,86,00,B7,51,
18,39
```

En basic, on utilise RTS =&H39 au lieu de SWI.

- *MO5* :

Même programme que pour les TO7(70) et TO9, seuls les « DATA » en ligne 70 changent :

```
70 DATA B6,A7,C0,8A,01,B7,A7,C0,86,00,B7,10,00,86,7E,B7,10,28,86,42,B7,10,5
0,86,42,B7,10,78,86,7E,B7,10,A0,86,42,B7,10,C8,86,42,B7,10,F0,86,00,B7,11,1
8,39
```

En basic, on utilise RTS = &H39 au lieu de STOP.

Le nombre de données en DATA demandera par la suite parfois plus d'un numéro de ligne. A vous alors de les répartir sur plusieurs lignes de programmation.

- *Commentaires sur le programme en basic* :

Ce programme ne démarre pas rapidement, car l'implantation du programme-machine est faite en basic.

Enlevez la ligne 60 et faites RUN. Faites ensuite EXEC &H7D00 : voilà la vraie rapidité du langage-machine, mais elle n'a rien de surprenant pour l'instant.

Vous en savez désormais assez sur les notions essentielles, nous pouvons donc passer au langage-machine appliqué. Il est vrai que celui-ci ne vous est déjà plus tout à fait étranger.