

Cinquième partie

Le moniteur

1. Généralités

Les moniteurs des TO8, TO9 et TO9+ renferment un ensemble de programmes appelés routines, écrits en langage machine. Le rôle de chacune d'elles est de gérer une fonction de la machine. Ainsi nous trouverons une routine chargée de visualiser des caractères sur l'écran, une autre fera la scrutation du clavier pour éventuellement détecter une touche appuyée, une troisième permettra d'utiliser les manettes de jeux, etc.

Les programmes intégrés par exemple dans le TO9, tels le BASIC ou PARAGRAPHE, utilisent ces routines pour effectuer leurs différents traitements; un utilisateur muni de la cartouche ASSEMBLEUR peut en faire autant.

La démarche, très générale, de l'utilisation d'une routine s'opère en trois temps:

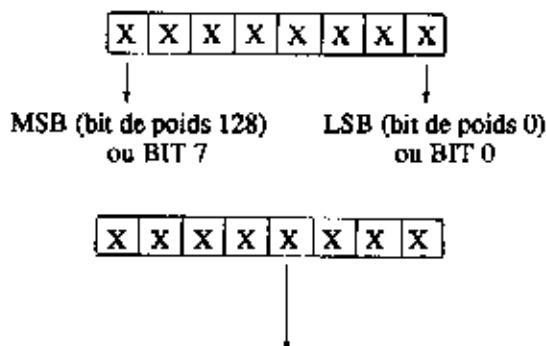
- 1) Vous chargez avec des valeurs ayant une signification précise certains registres du 6809 E, et éventuellement certains registres en RAM. Cette opération s'appelle l'initialisation des paramètres d'entrées.
- 2) Vous appelez la routine du moniteur qui effectue l'opération désirée.
- 3) Vous récupérez, s'il y a lieu, le résultat de l'opération dans certains registres du 6809 E, et/ou dans des registres en RAM. Ce sont les paramètres de retour.

Vous verrez que, dans certains cas, il n'y a pas de paramètres de retour. Quand ils existent, ils peuvent se trouver soit dans des registres du microprocesseur, soit dans des registres en RAM, soit dans les deux.

Dans le cadre de cette étude logicielle, des valeurs sont exprimées en décimal ou en hexadécimal. Pour faire la différence, nous avons employé la même convention d'écriture que la cartouche ASSEMBLEUR: si aucun signe ne précède le chiffre, la valeur est en décimal. Par contre, si le chiffre est précédé du symbole "\$", le nombre est écrit en hexadécimal.

Exemple: 12 = \$0C

La majeure partie des codes ou valeurs employés sont exprimés sur un octet (8 bits) ou un double octet (16 bits). Nous utiliserons en conséquence les termes suivants:



Les appellations "4° LSB" ou "BIT 3" désignent toutes les deux le bit repéré ci-dessus.

\$14

Digit de poids fort ———— Digit de poids faible

\$40 13

Octet de poids fort ———— Octet de poids faible

Certains registres en RAM recevront des valeurs cadrées sur un octet, d'autres sur deux octets. Dans le second cas, une consigne indiquant qu'il faut par exemple mettre la valeur \$13F dans le registre TOTO (\$6030-\$6031) signifie que:

- 1) La valeur \$01 doit être implantée à l'adresse \$6030
- 2) La valeur \$3F doit être implantée à l'adresse \$6031.

L'accès normalisé à une routine du moniteur se fait en utilisant l'instruction "JSR" ou "JMP", suivie de l'adresse du point d'entrée (ces adresses sont au nombre de 24). Souvent des exemples sont donnés sous cette forme:

```

      .
      .
PUTC  EQU    $E803
      LDB    #$07
      JSR    PUTC
      .
      .

```

Les points signifient que des instructions peuvent être avant ou après, il ne s'agit alors que d'une bribe de programme. Si vous voulez "essayer" l'exemple, il convient de rajouter simplement un arrêt (par exemple SWI), et les directives d'assemblage ORG et END.

Exemple: reprise et complément du programme précédent:

```

      ORG    $A000
PUTC  EQU    $E803
      LDB    #$07
      JSR    PUTC
      SWI
      END

```

Les sous-programmes du moniteur sauvegardent les registres du 6809 E. Au retour, tous les registres sont remis dans l'état antérieur de l'appel, sauf le registre code condition du 6809 E (ou registre d'état RE) et, bien sûr, les registres contenant des paramètres de retour (le plus souvent B, X et Y).

Il vous est fortement conseillé d'appeler les routines du moniteur en utilisant l'accès normalisé. En effet, ces routines ne peuvent fonctionner correctement qu'avec le pointeur de pile et certains registres du 6809 E positionnés dans un état bien précis.

2. Gestion alphanumérique de l'écran

Générateur de caractères alphanumériques

Les TO8, TO9 et TO9+ possèdent trois générateurs de caractères conciliant souplesse, rapidité et facilité d'utilisation. Le point commun de ces générateurs est, bien sûr, la manière d'afficher un caractère dans la fenêtre de travail. La méthode est la suivante: chaque caractère est défini par une matrice de 8×8 points, soit 64 points mémorisés par 8 octets de mémoire consécutifs. La codification d'un octet répond à la norme ci-dessous:

0 = le point n'est pas allumé

1 = le point est allumé.

Par exemple, le tracé du caractère A se définira par la matrice suivante:

MSB	LSB
↓	↓
0 0 0 0 0 0 0 0	8eme octet
0 0 0 1 1 0 0 0	7eme octet
0 0 1 0 0 1 0 0	6eme octet
0 1 0 0 0 0 1 0	5eme octet
0 1 1 1 1 1 1 0	4eme octet
0 1 0 0 0 0 1 0	3eme octet
0 1 0 0 0 0 1 0	2eme octet
0 0 0 0 0 0 0 0	1er octet

Sa codification dans le générateur de caractères étant dans l'ordre les octets \$00, \$42, \$42, \$7E, \$42, \$24, \$18, \$00.

Alphabet standard G0

Le générateur de caractères G0 est celui implicitement utilisé par les TO8, TO9 et TO9+. Il est constitué d'une suite de caractères affichables, correspondants au standard ASCII. L'adresse de ce générateur est contenue en RAM dans le registre appelé PTGENE et situé en \$60CF-\$60D0.

De fait, vous avez la possibilité de créer votre propre générateur de caractères selon le principe énoncé ci-dessus, et de l'implanter en RAM. Vous demanderez ensuite au microprocesseur de travailler avec votre police de caractères, en implantant l'adresse de ce nouveau générateur dans le registre PTGENE. Les seules contraintes à respecter sont de débiter ce générateur par le caractère correspondant au code \$20, le second sera appelé par le code \$21, etc. D'autre part, ne pas oublier qu'un caractère est constitué de 8 octets consécutifs, et enfin que le premier octet correspond au bas du caractère, le 8ème au haut du caractère.

Alphabet G2

Cet alphabet suit et complète le générateur G0. Il est composé de 22 éléments permettant l'affichage des caractères accentués (aigus, graves et circonflexes), ainsi que des configurations spéciales comme le tréma et la cédille.

Caractères utilisateurs

Indépendamment des alphabets G0 et G2 pouvant être redéfinis, les T08, T09 et T09+ nous offrent la possibilité de travailler avec des caractères dits "utilisateurs" qui, seuls ou associés, peuvent aussi bien permettre d'afficher du texte que des dessins.

Ce générateur est en BASIC initialisé par la fonction DEFGR\$ et sollicité par PRINTGR\$. Le principe de création et de mémorisation d'un caractère est identique à celui décrit précédemment. L'adresse du début de ce générateur de caractères utilisateur doit être implanté dans le registre USERAF (\$602D-\$602E). Le premier caractère (constitué des 8 premiers octets) correspondra automatiquement au code \$80, le second au code \$81 et ainsi de suite jusqu'à \$FF. La taille maximum de ce générateur est donc de 128 caractères.

Affichage des caractères alphanumériques

La routine PUTC permet d'afficher, à la position courante du curseur d'écran, un caractère contenu dans l'un des différents générateurs de caractères décrits précédemment. Le code du caractère désiré devra être implanté dans l'accumulateur B du CPU juste avant l'appel à la routine. Les codes compris entre \$20 et \$7F appelleront un caractère inclus dans l'alphabet pointé par PTGENE (\$60CF-\$60D0). Théoriquement, il s'agit donc des générateurs G0 et G2, mais nous avons vu qu'il était également possible que ces codes appellent un caractère d'une police redéfinie.

Les codes compris entre \$80 et \$FF correspondent aux caractères "utilisateur". Le générateur sollicité sera celui dont l'adresse de début est implantée dans le registre USERAF (\$602D-\$602E).

Le tableau ci-dessous dresse la liste des codes hexadécimaux et des caractères correspondants pour les alphabets G0 et G2.

Code	Caract.	Code	Caract.	Code	Caract.
20	blanc	40	@	60	—
21	!	41	A	61	a
22	"	42	B	62	b
23	#	43	C	63	c
24	\$	44	D	64	d
25	%	45	E	65	e
26	&	46	F	66	f
27	'	47	G	67	g
28	(48	H	68	h
29)	49	I	69	i
2A	*	4A	J	6A	j
2B	+	4B	K	6B	k
2C	,	4C	L	6C	l
2D	-	4D	M	6D	m
2E	.	4E	N	6E	n
2F	/	4F	O	6F	o
30	0	50	P	70	p
31	1	51	Q	71	q
32	2	52	R	72	r
33	3	53	S	73	s
34	4	54	T	74	t
35	5	55	U	75	u
36	6	56	V	76	v
37	7	57	W	77	w
38	8	58	X	78	x
39	9	59	Y	79	y
3A	:	5A	Z	7A	z
3B	;	5B	[7B	{
3C	<	5C	\	7C	}
3D	=	5D]	7D	}
3E	>	5E	^	7E	—
3F	?	5F	-	7F	■

Nom : FUTC
 Adresse du point d'entrée : \$E803
 Paramètre d'entrée : Accumulateur B du 6809E
 Paramètre de retour : Néant
 Effet : Cette routine affiche à l'écran le caractère correspondant au code envoyé. Les valeurs sont comprises entre \$20 et \$7F.

Exemple:

```
* AFFICHAGE DU GÉNÉRATEUR DE CARACTÈRE
* GO

TITLE GENCARAC
ORG $A000
PUTC EQU $E80B
LDB #$20 20=1 CARACT
SUIT JSR PUTC AFFICH CARACT
INCB
CMPB #$80 7F=DERNIER CARACT
BNE SUIT
SWI
BND
```

Positionnement des caractères

Hormis les valeurs comprises entre S20 et SFF correspondant à des codes de caractères alphanumériques affichables à l'écran, la routine PUTC interprète également les valeurs \$07 à \$1F. Elles déterminent des modes de traitement particulier de PUTC: création d'un bip sonore, séquence d'échappement... En particulier, le code US (\$1F) déclenche une séquence de positionnement du curseur ou de définition de la fenêtre de travail.

Programmation du curseur

Les coordonnées du curseur sont déterminées par la ligne (L) et la colonne (C). Alors que L est toujours compris entre 0 et 24, l'intervalle de C est fonction du mode de visualisation:

En mode 40 colonnes $1 \leq C \leq 40$ décimal
En mode 80 colonnes $1 < C < 80$ décimal

Une séquence de positionnement du curseur s'opère en trois appels de PUTC:

- 1e appel consiste à envoyer le code US (\$1F)
- 2e appel envoie le numéro de ligne
- 3e appel envoie le numéro de colonne.

Les numéros de ligne et de colonne envoyés à PUTC se calculent selon les formules suivantes:

ligne = L + \$40 L est exprimé en base 16
colonne = C + \$40 C est exprimé en base 16.

Exemple: On désire positionner le curseur sur la ligne 12 et la colonne 17:
12 en décimal correspond à \$0C en hexadécimal
17 en décimal correspond à \$11 en hexadécimal

ligne = \$0C + \$40 = \$4C
colonne = \$11 + \$40 = \$51

La séquence s'écrira donc:

PUTC	EQU	\$E803	
	LDB	#S1F	code US envoyé
	JSR	PUTC	1e appel
	LDB	#S4C	numéro ligne
	JSR	PUTC	2e appel
	LDB	#S51	numéro colonne
	JSR	PUTC	3e appel

Nom	PUTC
Adresse du point d'entrée	\$E803
Paramètre d'entrée	Accumulateur B du 6809E
Paramètre de retour	Néant
Effet	Pour des valeurs comprises entre \$07 et \$1F les effets sont les suivants:

Code hexa	Nom	Effet
\$07	BEL	Création d'un bip sonore
\$08	BS	Déplacement curseur d'une position à gauche, ou recopie à gauche du caractère courant si l'adresse \$6043 contient la valeur \$FF
\$09	HT	Déplacement curseur d'une position à droite, ou recopie à droite du caractère courant si l'adresse \$6043 contient la valeur \$FF
\$0A	LF	Descente d'une ligne

\$0B	VT	Remonte d'une ligne
\$0C	FF	Effacement de la fenêtre
\$0D	CR	Retour en début de ligne courante
\$0E	SO	Passage en mode télétype
\$0F	SI	Retour en mode normal
\$11	DC1	Allumage du curseur
\$12	DC2	Répétition du dernier caractère ASCII affiché
\$14	DC4	Extinction du curseur
\$16	ACC	Séquence caractère du G2
\$18	CAN	Effacement d'une ligne à partir de la position du curseur
\$1B	ESC	Séquence d'échappement
\$1E	RS	Retour du curseur dans le coin supérieur gauche de la fenêtre
\$1F	US	Séquence de positionnement curseur ou de définition de fenêtre

Détermination de la fenêtre de travail

La fenêtre de travail sera définie par le repérage de la première ligne ou "haut de page" et de la dernière ligne ou "bas de page". L'un comme l'autre nécessitent trois appels à la routine PUTC:

- 1e appel: envoi du code US (\$1F) Accu B
- 2e appel: envoi du nombre N1 par l'accu B
- 3e appel: envoi du nombre N2 par l'accu B.

Haut de page

La ligne représentant le haut de page est repérée par un nombre compris entre 00 et 24 (décimal), donc s'écrivant par deux digits. La programmation du haut de page se fera en décomposant ces deux digits pour former N1 et N2 selon la formule suivante:

- N1 = digit de poids fort + \$20
- N2 = digit de poids faible + \$20

Ainsi N1 et N2 seront toujours compris entre \$20 et \$29. Prenons un exemple: On désire que le haut de la fenêtre soit la ligne 01:

- N1 = 0 + \$20 = \$20
- N2 = 1 + \$20 = \$21

La séquence s'écrira:

```
PUTC EQU SE803
      LDB #$1F  code US
      JSR PUTC  1e appel
      LDB #$20  N1
      JSR PUTC  2e appel
      LDB #$21  N2
      JSR PUTC  3e appel
```

Bas de page

La ligne représentant le bas de page est un nombre décimal compris entre 00 et 24, donc constitué de deux digits. La programmation du bas de page se fera en décomposant ces deux digits pour former N1 et N2 selon la formule ci-dessous:

$N1 = \text{digit de poids fort} + \10

$N2 = \text{digit de poids faible} + \10

Ainsi N1 et N2 seront compris entre \$10 et \$19.

Exemple: on désire que le bas de la fenêtre soit la ligne 24:

le digit de poids fort est 2

le digit de poids faible est 4.

Donc,

$N1 = 2 + \$10 = \12

$N2 = 4 + \$10 = \14

La séquence s'écrira:

```
PUTC EQU SE803
      LDB #$1F
      JSR PUTC
      LDB #$12
      JSR PUTC
      LDB #$14
      JSR PUTC
```

Retour du curseur dans le coin gauche

Pour positionner le curseur dans le coin supérieur gauche de la zone définie comme étant la fenêtre de travail, vous utiliserez le code \$1E, envoyé à PUTC de la manière suivante :

```

:
:
PUTC EQU    $E803
      LDB    #$1E
      JSR    PUTC
:
:
```

Descente d'une ligne

Il suffit d'envoyer le code \$0A à la routine PUTC pour provoquer la descente du curseur d'une ligne. La colonne reste inchangée.

```

:
:
PUTC EQU    $E803
      LDB    #$0A
      JSR    PUTC
:
:
```

Remontée d'une ligne

L'opération inverse à celle décrite plus haut est réalisée en envoyant le code \$0B à la routine PUTC.

```

:
:
PUTC EQU    $E803
      LDB    #$0B
      JSR    PUTC
:
:
```

Retour au début de ligne

Le code \$0D implanté dans l'accu B provoque, lors de l'appel de PUTC, un positionnement du curseur sur la colonne 0 de la ligne courante.

Effacements divers

Effacement de la fenêtre

Cette opération est réalisée en envoyant le code \$0C à la routine PUTC.

```

      .
      .
PUTC  EQU    $E803
      LDB    #$0C
      JSR    PUTC
      .
      .

```

Tout ce qui est dans la zone définie comme étant la fenêtre de travail sera alors entièrement effacé.

Extinction et allumage du curseur

Le code \$14 envoyé à la routine PUTC aura pour effet d'effacer le curseur de l'écran.

```

      .
      .
PUTC  EQU    $E803
      LDB    #$14
      JSR    PUTC
      .
      .

```

Pour rallumer ce curseur, vous utiliserez le code \$11.

Effacement d'une ligne

Le code \$18 envoyé à la routine PUTC provoque l'effacement d'une ligne d'écran. La ligne effacée sera la ligne courante du curseur, de même l'effacement se fera à partir de la colonne courante du curseur.

```

      .
      .
PUTC  EQU    $E803
      LDB    #$18
      JSR    PUTC
      .
      .

```

Affichages particuliers

Caractères accentués, alphabet G2

Le code ACC (\$16) appliqué à la routine PUTC permet l'affichage d'une minuscule accentuée ou d'un caractère de l'alphabet G2. Dans le cas d'une minuscule accentuée, la procédure demande trois appels à PUTC :

- 1e appel : envoi du code ACC (\$16)
- 2e appel : envoi du code d'accent
- 3e appel : envoi du code de la minuscule

Les accents disponibles sont les suivants:

Code	Accent ou signe correspondant
\$41	Aigu
\$42	Grave
\$43	Circumflexe
\$48	Tréma
\$4B	Cédille

Nous rappelons que les minuscules sont accessibles par les codes compris entre \$61 (a) et \$7A (z), voir à ce sujet l'affichage des caractères alphanumériques, page 163.

Exemple: Pour afficher un ç nous écrirons:

```
PUTC EQU $E803
LDB #S16 Code ACC
JSR PUTC 1e appel
LDB #S4B code cédille
JSR PUTC 2e appel
LDB #S63 code de c
JSR PUTC
```

D'une manière générale, si le dernier code envoyé ne correspond pas à une minuscule mais à un autre caractère (majuscule ou autre), ce dernier sera néanmoins affiché, et l'accent supprimé automatiquement.

Pour l'affichage d'un caractère de l'alphabet G2, la procédure s'écrit en deux appels:

- 1e appel: envoi du code ACC (\$16)
- 2e appel: envoi du code caractère

Dans ce cas, le code caractère est à choisir parmi les suivants:

Code	Caractère affiché
\$23	livre sterling
\$24	dollar
\$26	dièse
\$27	paragraphe
\$2C	flèche à gauche
\$2D	flèche en haut
\$2E	flèche à droite
\$2F	flèche en bas
\$30	degré
\$31	plus ou moins
\$38	division entière
\$3C	un quart
\$3D	un demi
\$3E	trois quarts
\$6A	o dans e majuscule
\$7A	o dans e minuscule
\$7B	sz allemand

Le programme suivant affiche tous ces caractères:

```
* AFFICHAGE DES CARACTERES ALPHABET G2
* PAR CODE ACC ROUTINE PUTC
```

```

                TITLE   GENG2
                ORG      $A000
PUTC            EQU     $18803
                LDA      #$23
ENCOR          LDB      #$16
                JSR       PUTC
                TFR       A, B
                JSR       PUTC
                INCA
                CMPA     #$7C
                BNE       ENCOR
                SWI
                END
```

Caractères Télétel

Les caractères semi-graphiques aux normes Télétel sont également disponibles dans les TO8, TO9 et TO9+. Vous accèderez à ce générateur spécial de la manière suivante:

1e appel à PUTC, envoi du code \$0E (sélection mode Télétel)

2e appel à PUTC, envoi du code caractère

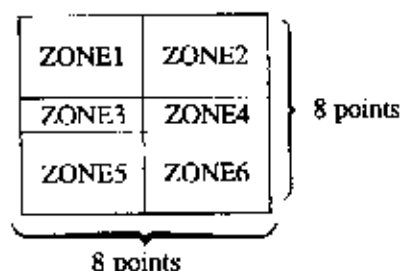
Le code caractère est compris entre les valeurs \$20 à \$7F, soit 64 caractères au total. Le programme suivant permet de les afficher à l'écran.

* AFFICHAGE DES CARACTERES TELETEL
* ET RETOUR AU MODE NORMAL

```

                TITLE  TELETEL
                ORG     $A000
PUTC            EQU     $1803
                LDP      #$0E      MODE TELETEL
                JSR      PUTC
                LDB      #$20
SUITE          JSR      PUTC
                INCB
                CMPB     #$80
                BNE      SUITE
                LDB      #$0F      MODE NORMAL
                JSR      PUTC
                SWI
                END
```

Notez que l'émission du code \$0F permet de revenir au mode normal et, par conséquent, d'accéder à l'alphabet G0 par les codes compris entre \$20 et \$7F. Le principe de base de l'affichage Télétel est de ne plus considérer un caractère par une matrice de 8 x 8 points (comme l'alphabet G0, G2, utilisateur), mais par un ensemble de 6 zones réparties de la manière suivante:



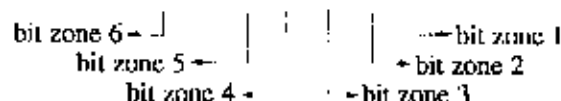
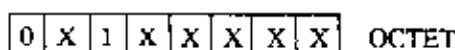
Le caractère semi-graphique est fabriqué en allumant une ou plusieurs zones. Nous comprenons ainsi que le nombre de combinaisons possibles nous limite à $2^6 = 64$ représentations différentes.

Chaque zone est contrôlée par un seul bit mémorisé:

si le bit = 1, la zone est allumée, elle prendra la couleur de forme

si le bit = 0, la zone est éteinte, elle prendra la couleur de fond.

Ainsi un caractère Télétel est mémorisé par un seul octet de mémoire:



Il est bon de noter que le MSB et le 6e LSB doivent toujours respectivement rester à 0 et 1. L'ensemble des caractères pré-définis sont rassemblés dans le tableau suivant:

Caractères semi-graphiques Télétel

Code décimal								
	32	40	48	56	64	72	80	88
	96	104	112	120	128	136	144	152
	160	168	176	184	192	200	208	216
	224	232	240	248	256	264	272	280
	288	296	304	312	320	328	336	344
	352	360	368	376	384	392	400	408
	416	424	432	440	448	456	464	472
	480	488	496	504	512	520	528	536
	544	552	560	568	576	584	592	600

Séquences d'échappement

Les séquences d'échappement permettent de réaliser plusieurs tâches différentes, comme par exemple la modification des attributs de couleurs, le passage en mode d'incrustation vidéo ou le changement de la taille des caractères... Vous devrez utiliser deux méthodes d'appels différents à la routine PUTC (\$E803) selon le travail désiré. Nous devons en effet distinguer deux modes de traitement possibles:

- le mode courant
- le mode plein écran.

Pour bien comprendre la différence, prenons un exemple: vous désirez écrire des caractères en rouge sur l'écran. Si cette couleur n'est utilisée que ponctuellement pour afficher un texte ou une portion de texte, cotrayant des caractères déjà affichés à l'écran (ou à venir) dans une couleur différente, vous utiliserez le mode courant. Par contre, si l'ensemble de tout ce qui est (ou sera) affiché à l'écran doit être écrit en rouge, le mode plein écran sera utilisé. En langage BASIC, la différence est faite par le choix des instructions COLOR ou SCREEN.

Pour appeler une séquence d'échappement en mode courant, la procédure s'écrit en deux appels à la routine PUTC:

- 1e appel : envoi du code d'échappement (\$1B)
- 2e appel : envoi du code réalisant la fonction désirée.

Exemple:

```
      .  
      .  
PUTC  EQU    $E803  
      LDB    #$1B    code d'échappement  
      JSR    PUTC  
      LDB    #$XX    XX = code fonction  
      JSR    PUTC  
      .  
      .
```

Par contre, en mode plein écran, la procédure nécessite 4 appels à la routine PUTC:

- 1e appel: envoi du code d'échappement \$1B
- 2e appel: envoi du code \$23
- 3e appel: envoi du code \$20
- 4e appel: envoi du code réalisant la fonction désirée.

Exemple:

```
PUTC EQU $E803
      LDB #$1B
      JSR PUTC
      LDB #$23
      JSR PUTC
      LDB #$20
      JSR PUTC
      LDB #$XX   XX = code fonction
      JSR PUTC
```

Programmation des couleurs

La séquence d'échappement permet de modifier les attributs de couleurs (forme, fond, tour). La valeur du code appelé XX dans les exemples précédents est définie de la manière suivante:

le **digit de poids fort** précise le destinataire du changement de couleur, il peut donc prendre trois valeurs possibles,

- 4 pour la forme
- 5 pour le fond
- 6 pour le tour.

Le **digit de poids faible** indique la couleur, selon la codification suivante:

- 0 pour le Noir
- 1 Rouge
- 2 Vert
- 3 Jaune
- 4 Bleu
- 5 Magenta
- 6 Cyan
- 7 Blanc

Pour reprendre l'exemple cité plus haut, nous savons maintenant que pour écrire en rouge sur l'écran le code sera égal à:

CODE = \$ 4 1

couleur de forme ———→ couleur rouge

Ainsi le programme suivant passera tous les caractères de la fenêtre de travail en rouge (mode plein écran).

```

      .
      .
PUTC EQU    $E803
      LDB    #$1B    code d'échappement
      JSR    PUTC
      LDB    #$23    code intermédiaire
      JSR    PUTC
      LDB    #$20    code intermédiaire
      JSR    PUTC
      LDB    #$41    code couleur
      JSR    PUTC
      .
      .

```

Alors que le programme suivant écrira le message "OK" en rouge et vert sans modifier le reste de l'écran (mode courant).

```

* CE PROG AFFICHE OK EN ROUGE ET VERT
* ECHAPPEMENT DE TYPE COURANT

```

```

      TITLE  OK
      ORG    $A000
PUTC EQU    $E803
      LDB    #$1B    ECHAPPEMENT
      JSR    PUTC
      LDB    #$41    FORME-ROUGE
      JSR    PUTC
      LDB    #$4F    AFFICHE O
      JSR    PUTC
      LDB    #$1B    ECHAPPEMENT
      JSR    PUTC
      LDB    #$42    FORME-VERT
      JSR    PUTC
      LDB    #$4E    AFFICHE K
      JSR    PUTC
      SWI
      END

```

En conclusion, la valeur du code peut être comprise entre

```

$40 à $47 pour la forme
$50 à $57 pour le tour
$60 à $67 pour le tour

```

Mais, comme vous l'avez déjà certainement remarqué (petits fûtés), nous ne travaillons dans ce cas que sur 8 couleurs. La table de valeurs a donc été étendue de \$70 à \$87 afin d'accéder aux 8 couleurs pastel et, dans ce cas, les allocations sont les suivantes :

\$70 à \$77 pour la forme

\$78 à \$7F pour le fond

\$80 à \$87 pour le tour

avec

0 pour le Gris

1 Rose

2 Vert clair

3 Sable

4 Bleu clair

5 Parme

6 Bleu ciel

7 Orange

Programmation des modes d'affichage

Comme nous l'avons détaillé dans les études matérielles, les TO8, TO9 et TO9+ possèdent de nombreux modes d'affichage différents, permettant de faire ponctuellement des compromis entre couleurs, définition graphique et rapidité de visualisation. L'accès de ces modes nécessite 2 appels à la routine PUTC:

1e appel: envoi du code d'échappement (\$1B)

2e appel: envoi du code de mode.

Le code de mode désiré sera l'un des suivants:

Code	Mode sélectionné
\$48	Page 1
\$49	Page 2
\$4A	Superposition écriture page 2
\$4B	Superposition écriture page 1
\$59	Bit-map 4 couleurs
\$5A	40 colonnes
\$5B	80 colonnes
\$5E	Bitmap 16 couleurs
\$6D	Incrustation
\$6C	Incrustation (off)
\$88	Triple superposition sélection page 1
\$89	Triple superposition sélection page 2
\$8A	Triple superposition sélection page 3
\$8B	Triple superposition sélection page 4

Exemple: pour travailler en mode 80 colonnes, nous utiliserons la procédure suivante:

```
PUTC EQU $E803
LDB  #$1B  code d'échappement
JSR  PUTC
LDB  #$5B  mode 80 colonnes
JSR  PUTC
```

Si l'unité centrale est équipée de l'interface d'incrustation (en option), vous pourrez alors la commuter dans ce mode spécial de visualisation, en envoyant le code \$6D. L'incrustation permet de mélanger une image vidéo issue du téléviseur et l'image synthétique créée par le micro-ordinateur. La coexistence de ces deux images simultanées à l'écran est également fonction de la programmation du circuit PALETTE, détaillée page 212.

Nous vous conseillons de vous reporter à la partie de ce livre traitant du fonctionnement des TO8, TO9 et TO9+ si vous désirez de plus amples détails sur les particularités des modes Page, Bit-map, etc.

Dimensions des caractères

Les caractères alphanumériques peuvent être affichés à l'écran selon diverses dimensions repertoriées ci-dessous.

La déclaration se réalise par deux appels à la routine PUTC:

1e appel : envoi du code d'échappement \$1B (ACCU B)

2e appel : envoi du code de dimension (ACCU B)

Code	Dimensions des caractères
\$4C	Taille normale
\$4D	Double hauteur, largeur normale
\$4E	Double largeur, hauteur normale
\$4F	Double taille

Exemple: pour travailler en double taille, nous écrivons le programme suivant:

```
ORG    $A000
PUTC   EQU    $E803
LDB    #$1B   code d'échappement
JSR    PUTC   premier appel
LDB    #$4F   code double taille
JSR    PUTC   second appel
SWI
END
```

Traitements divers

Certains traitements particuliers répertoriés ci-dessous sont également définis sous le contrôle de séquence d'échappement:

Code Traitement correspondant

\$58	Masquage
\$5F	Démasquage
\$5C	Inversion de la vidéo
\$6A	Scroll normal
\$6E	Scroll doux
\$6B	Mode page (pas de scroll)
\$68	Ecriture d'un caractère sans modifier la couleur
\$69	Ecriture d'un caractère dans la couleur courante

La procédure s'établit en deux appels de PUTC:

1e appel : envoi du code d'échappement \$1B

2e appel : envoi du code représentant le traitement désiré.

Exemple: pour avoir un listing d'éditeur défilant doucement à l'écran, nous pouvons écrire le programme suivant:

```
ORG    $A000
PUTC   EQU    $E803
LDB    #$1B
JSR    PUTC
LDB    #$6E
JSR    PUTC
SWI
END
```

Vous pourrez constater ce scroll doux en faisant des dumpings sous monitor. Notons que certains codes, comme par exemple \$5C pour l'inversion vidéo, peuvent être appelés en mode plein écran selon la procédure décrite en début de

ce chapitre. Le masquage consiste à écrire des caractères en couleur noire sur fond noir, jusqu'à ce que l'attribut de démasquage soit sollicité. En utilisant ce dernier selon le mode plein écran, vous dévoilerez d'un seul coup tous les caractères écrits en mode masqué.

Affichage alphanumérique par la routine PLOT

La routine PLOT, utilisée plus fréquemment pour afficher des points graphiques (voir page 183), peut également être sollicitée pour l'affichage de caractères alphanumériques. Dans ce cas, le code ASCII du caractère à afficher est implanté dans le registre CHDRAW (\$6041), la couleur est précisée dans le registre COLOUR (\$603B), et ses coordonnées sont exprimées dans les registres X et Y du 6809 E (représentant respectivement l'abscisse et l'ordonnée).

Les coordonnées dans X sont comprises entre 1 et 40 décimal, pour le mode 40 colonnes, ou entre 1 et 80 pour le mode 80 colonnes. Les valeurs de Y sont comprises entre 0 et 24 décimal pour les deux modes.

Les attributs de couleurs sont identiques à ceux exprimés page 184. L'octet envoyé à COLOUR est décomposable en deux digits, le digit de poids faible code la couleur de fond, le digit de poids fort code la couleur de forme. En retour de la routine PLOT, les valeurs implantées dans les registres X et Y sont automatiquement récopiées dans les registres PLOTX et PLOTY.

Nom	: PLOT
Adresse du point d'entrée	: \$E80F
Paramètre d'entrée	: registres X et Y du 6809 E registre CHDRAW \$6041 registre COLOUR \$603B
Paramètre de retour	: registre PLOTX \$603D-\$603E registre PLOTY \$603F-\$6040
Effet	: affiche à l'écran un caractère exprimé en ASCII dans CHDRAW, aux coordonnées passées par X et Y, et dans la couleur définie par COLOUR
Exemple	: le programme suivant affiche le caractère "K" au milieu de l'écran, en noir sur fond blanc.

* AFFICHAGE D'UN CARACTERE PAR LA
* LA ROUTINE PLOT

TITLE	PLOTCHAR	
ORG	\$A000	
PLOT	EQU	\$E80F
CHDRAW	EQU	\$6041
COLOUR	EQU	\$603B
LDA	#54B	CODE ASCII DE 'K'
STA	CHDRAW	
LDA	#140	COULEUR NOIR/BLANC
STA	COLOUR	
LDX	#0014	COLONNE 20
LDY	#000C	LIGNE 12
JSR	PLOT	
SWI		
END		

Notes: On peut accéder directement à l'écriture du caractère répété dans CHDRAW en faisant un appel à l'adresse \$E833 (CHPL). L'accès au mode Bit-map 16 et Triple superposition sont interdits dans ce mode de PLOT.

3. Gestion graphique de l'écran

Mémorisation en RAM forme et couleur

Comme nous l'avons expliqué dans l'étude matérielle du TO9, la RAM écran est en fait constituée d'une RAM A et d'une RAM B dont le fonctionnement ainsi que les octets mémorisés sont fonction du type de visualisation utilisé (mode 40 ou 80 colonnes, bit-map, etc.).

Néanmoins, dans le mode commun au TO7/70 (40 colonnes, 16 couleurs, 320 x 200 points), la RAM A correspond à la mémoire forme et la RAM B à la mémoire couleur.

Ces deux RAM sont décodées aux mêmes adresses comprises entre \$4000 et \$5FFF et la sélection de l'une d'entre elles est effectuée par le bit forme issu du PIA interne au 6846 (\$E7C3).

Pour les TO8 et TO9+, les notions sont les mêmes bien que physiquement rien ne soit comparable. La différence fondamentale est liée à l'utilisation du gate mode page qui intègre, grosso modo, les fonctions des gates de gestion machine et gate d'affichage dans un même boîtier.

D'autre part, au niveau de l'organisation mémoire nous passons d'une structure de commutation de banques par commutation de boîtiers RAM (TO9), à une commutation de banques par commutation de pages dans un même boîtier RAM (TO8 et TO9+).

Que deviennent dans ces conditions le bit de forme à l'adresse \$E7C3 et les notions de RAM écran forme et couleur ? Disons tout simplement que le constructeur a eu la bonne idée d'émuler ces mêmes principes dans les TO8 et TO9+. Même si du côté matériel tout est différent, du côté logiciel tout est compatible. Nous pourrions donc continuer à "poker" aveuglément l'adresse \$E7C3 bien que le bit de forme n'existe plus ni physiquement ni à cette adresse ! Miraculeux, non ?

Commutation couleur

Pour travailler dans la mémoire couleur, il suffit de forcer à 0 le bit 0 de l'adresse \$E7C3 (forme). Il est conseillé d'utiliser la technique de programmation par "masques" pour ne modifier que la valeur de ce bit.

Commutation forme

Inversement, pour travailler dans la mémoire forme, il faut forcer à 1 le bit 0 de l'adresse SE7C3 (à l'exclusion d'autres bits).

Allumage ou extinction d'un point graphique

La routine PLOT présentée ci-dessous, permet de changer la couleur d'un point graphique dont les coordonnées sont exprimées dans le registre X (abscisses) et le registre Y (ordonnées). Si l'ordonnée est toujours comprise entre 0 et 199, l'intervalle des abscisses dépend du mode d'affichage choisi:

Valeurs limites pour X		Mode graphique:	
Décimal	Hexadécimal		
0-319	\$0-\$13F	TO7	16 couleurs
0-319	\$0-\$13F	Page 1	2 couleurs
0-319	\$0-\$13F	Page 2	2 couleurs
0-319	\$0-\$13F	Bit-map	4 couleurs
0-319	\$0-\$13F	Superposition	
0-639	\$0-\$27F	80 colonnes	2 couleurs
0-159	\$0-\$9F	Bit-map	16 couleurs

La couleur du point sera précisée dans le registre FORME (\$6038), les valeurs comprises entre -16 et +15 seront interprétées de la manière suivante:

Couleur	Code forme	Code fond
Noir	0	-1
Rouge	1	-2
Vert	2	-3
Jaune	3	4
Bleu	4	-5
Magenta	5	-6
Cyan	6	-7
Blanc	7	-8
Gris	8	-9
Rose	9	-10
Vert clair	10	-11
Sable	11	-12
Bleu clair	12	-13
Parme	13	-14
Bleu ciel	14	-15
Orange	15	-16

Nom	: PLOT
Adresse du point d'entrée	: \$E80F
Paramètres d'entrée	: Registre X et Y du \$809 E Registre FORME \$6038 Registre CHDRAW \$6041 Registre STATUS \$6019
Paramètres de retour	: Registre PLOTX \$603D-\$603E Registre PLOTY \$603F-\$6040
Effet	: Affiche un point graphique dont les coordonnées sont précisées dans X et Y du \$809 E, et la couleur exprimée dans le registre FORME.
Exemple	

* AFFICHAGE D'UN POINT GRAPHIQUE

```

TITLE POINT
ORG $A000
PLOT EQU $E80F
CHDRAW EQU $6041
STATUS EQU $6019
FORME EQU $6038
LDA STATUS
ANDA #$EF BIT4 FORCER A 0
STA STATUS
CLR CHDRAW PLOT EN GEST GRAPH
LDA #$01 COUL ROUGE
STA FORME
LDX #$013F DERNIERE COLONNE
LDY #$0064 LIGNE=100
JSR PLOT
SWI
END

```

En conséquence, il est bon de noter que les valeurs positives appellent des couleurs de forme et inversement, les codes négatifs sélectionnent des couleurs de fond. Un code de couleur fond se déduit d'un code de forme en ajoutant 1 et en prenant l'opposé.

Précisons quelques remarques importantes relatives aux modes utilisés. En mode TO7, si le code de la couleur est négatif, le point sera écrit en fond, cela implique que le bit correspondant dans l'octet de mémoire forme soit mis à 0. Inversement, si le code couleur est positif, le point sera écrit en forme, le bit correspondant dans l'octet de mémoire forme sera mis à 1.

Si le bit 4 du registre STATUS (\$6019) est à 1, seul le bit en mémoire forme sera modifié, la couleur ne sera pas changée. En mode Page1, Page2, Superposition et Triple superposition, un code positif écrit dans la couleur de la page sélectionnée, un code négatif écrit dans la couleur fond.

Pour le mode bit-map 4 couleurs, seuls les 4 premiers codes positifs sont utilisables (de 0 à 3) et travaillent dans la mémoire forme, le fond n'a aucune signification.

Pareillement, le mode bit-map 16 couleurs n'autorise que les 16 codes positifs (0 à 15) en forme, le fond n'ayant aucun sens. Le mode 80 colonnes ne comprend à la fois qu'un code positif écrit en forme et un code négatif écrit en fond.

Enfin, dans tous les modes choisis, le registre CHDRAW (\$6041) doit être mis à 0 pour avoir une gestion graphique de l'écran. Si CHDRAW est différent de 0, la routine PLOT sera utilisée en mode alphanumérique, décrit page 181.

En retour de la routine PLOT, les coordonnées du dernier point affiché seront automatiquement recopiées des registres X et Y (6809 E) dans les registres PLOTX (\$603D-\$603E) et PLOTY (\$603F-\$6040).

Tracé d'un segment de droite

La routine DRAW trace un segment de droite entre deux points déterminés que nous appellerons origine et destination. Les coordonnées du point d'origine (abscisse et ordonnée) sont définies respectivement par le contenu des registres PLOTX (\$603D-\$603E) et PLOTY (\$603F-\$6040). Les coordonnées du point destination (abscisse et ordonnée) sont définies respectivement par le contenu des registres X et Y du 6809 E.

Les valeurs limites de ces coordonnées sont identiques à celles exprimées précédemment (page 183). La couleur du segment tracé est définie par le registre FORME selon les mêmes conventions que ci-dessus. Le contenu du registre CHDRAW doit être nul, sinon la routine DRAW travaillera en mode caractère (voir page 188).

En retour de la routine DRAW, les coordonnées du dernier point tracé sont recopiées automatiquement dans les registres PLOTX et PLOTY. Cette particularité simplifie énormément l'écriture de programmes pour dessiner des figures géométriques tels que rectangles, carrés, triangles, losanges... Car après le tracé du premier segment, seuls les coordonnées du point destination sont à préciser pour tracer les suivants.

Nom	DRAW
Adresse du point d'entrée	\$E80C
Paramètres d'entrée	Registre X et Y du 68000 Registre PLOTX \$603D-\$603E Registre PLOTY \$603F-\$6040 Registre CHDRAW \$6041 Registre FORME \$6038 Registre STATUS \$6019 Registre COLOUR \$60E
Paramètre de retour	Registre PLOTX \$603D-\$603E Registre PLOTY \$603F-\$6040
Effet	Trace un segment de droite
Exemple	Le programme ci-dessous utilise quatre fois la routine DRAW pour dessiner un losange.

```
* DESSIN D'UN LOSANGE BLEU PAR LA
* ROUTINE DRAW
```

```

      TITLE      LOSANGE
      ORG        $A000
DRAW   EQU      $E80C
CHDRAW EQU      $6041
STATUS EQU      $6019
FORME  EQU      $6038
PLOTX  EQU      $603D
PLOTY  EQU      $603F

      LDA        STATUS
      ANDA       #1FF      BIT4 FORGE A 0
      STA        STATUS
      CLR        CHDRAW    DRAW EN GEST GRAPH
      LDA        #$04      COUL BLEU
      STA        FORME
      LDX        #$00A0    COLONNE 160
      LDY        #$0040    LIGNE   64
      STX        PLOTX
      STY        PLOTY
      LDX        #$00D0    COLONNE 208
      LDY        #$0070    LIGNE   112
      JSR        DRAW      TRACE SEGMENT 1
      LDX        #$00A0    COLONNE 160
      LDY        #$00A0    LIGNE   160
      JSR        DRAW      TRACE SEGMENT 2

```

```

LDX    #$0070    COLONNE 112
LDY    #$0070    LIGNE  112
JSR    DRAW      TRACE SEGMENT 3
LDX    #$00A0    COLONNE 160
LDY    #$0040    LIGNE  64
JSR    DRAW      TRACE SEGMENT 4
SWI
END

```

Note: Pour modifier les couleurs, le bit 4 de STATUS (\$6019) doit être forcé à 0.

Dessiner avec des caractères

L'intérêt de DRAW est de pouvoir travailler également avec des caractères au lieu de points graphiques. Dans ce cas, le registre CHDRAW doit contenir le code ASCII du caractère qui nous servira à dessiner. La couleur est déterminée par le registre COLOUR, le digit de poids fort définissant la couleur de fond, le digit de poids faible déterminant la couleur de forme.

Les coordonnées passées par PLOTX, PLOTY, X et Y du 6809 E sont comprises entre 0 et 24 décimal pour les ordonnées, les abscisses dépendant du mode de visualisation:

- 1 à 40 pour le mode 40 colonnes
- 1 à 80 pour le mode 80 colonnes.

Ainsi, nous pouvons reprendre le programme précédent dont la structure est la même. En modifiant simplement le contenu de CHDRAW, les valeurs des coordonnées et en initialisant le registre COLOUR, nous traçons le même losange mais avec les lettres "K".

* DESSIN D'UN LOSANGE CONSTITUE DE 'K'

	TITLE	LOSANGK
	ORG	\$A000
DRAW	EQU	\$E80C
CHDRAW	EQU	\$6041
STATUS	EQU	\$6019
PLOTX	EQU	\$603D
PLOTY	EQU	\$603F
COLOUR	EQU	\$603E

LDA	STATUS	
ANDA	#\$EF	BIT4 FORCE A 0
STA	STATUS	
LDA	#\$4B	CODE ASCII DE 'K'
STA	CHDRAW	DRAW EN GEST CARAC
LDA	#\$40	COUL NOIR/BLANC
STA	COLOUR	
LDX	#\$0014	COLONNE 20
LDY	#\$0008	LIGNE 08
STX	PLOTX	
STY	PLOTY	
LDX	#\$001A	COLONNE 26
LDY	#\$000E	LIGNE 14
JSR	DRAW	TRACE SEGMENT 1
LDX	#\$0014	COLONNE 20
LDY	#\$0014	LIGNE 20
JSR	DRAW	TRACE SEGMENT 2
LDX	#\$000E	COLONNE 14
LDY	#\$000E	LIGNE 14
JSR	DRAW	TRACE SEGMENT 3
LDX	#\$0014	COLONNE 20
LDY	#\$0008	LIGNE 08
JSR	DRAW	TRACE SEGMENT 4
SWI		
END		

4. Lecture de l'écran

Lecture d'un point graphique

Le principe général est de viser un point à l'écran qui sera repéré par ses coordonnées (registres X et Y du 6809 E), afin de lire sa couleur (retour dans l'accu B). L'utilisation des valeurs et l'interprétation des résultats de la routine GETP réalisant cette tâche sont fonction du mode d'affichage sélectionné. Afin de mieux vous repérer, nous avons répertoriés dans le tableau ci-dessous les divers cas possibles:

Mode d'affichage	Coordonnées du point Ordonnée Y Abscisse X		Valeur lue dans B
40 colonnes (TO7)	0 à 199	0 à 319	-16 à -1 si coul. fond 0 à +15 si coul. forme
Bit map 4 couleurs	0 à 199	0 à 319	0 à 3
Page 1	0 à 199	0 à 319	0 si point en forme -1 si point en fond
Page 2			
Superposition			
Bit-map 16 couleurs	0 à 199	0 à 159	0 à +15
80 colonnes	0 à 199	0 à 639	0 si point en forme -1 si point en fond

La numérotation des couleurs renvoyées par GETP est identique à la codification énoncée page 184.

Nom	: GETP
Adresse du point d'entrée	: \$E82F
Paramètre d'entrée	: Registre X et Y du 6809 E
Paramètre de retour	: Accumulateur B du 6809 E
Effet	: Retourne dans B la couleur du point visé par X et Y
Exemple	: En supposant que vous travaillez avec la cartouche ASSEMBLEUR TOTÉK, sous editor nous avons un bandeau rouge réservé aux messages d'erreurs ou de confirmation.

Le programme ci-dessous va lire un point correspondant à ce bandeau. En retour nous aurons la valeur \$FE dans l'accumulateur B du 6809 E (\$FE = - 02 couleur de fond rouge).

GETP	ORG	\$A000	
	EQU	\$E821	
	LDX	#\$013E	colonne 318
	LDY	#\$00C7	ligne 199
	JSR	GETP	
	SWI		
	END		

Lecture d'un caractère

La routine GETS peut être utilisée dans les modes TO7, 80 colonnes, Page 1, Page 2, Superposition. Vous ne devez pas l'appeler dans les modes Bit-map 4 couleurs, Bit-map 16 couleurs et Triple superposition. GETS retourne dans l'accumulateur B le code ASCII du caractère dont les coordonnées sont précisées dans le registre X (abscisse) et l'accumulateur A (ordonnée).

$0 \leq A \leq 24$ et $1 \leq X \leq 80$ pour le mode 80 colonnes

$0 \leq A \leq 24$ et $1 \leq X \leq 40$ pour le mode 40 colonnes

Si, à l'endroit visé le caractère n'est pas connu, la routine GETS renverra la valeur 0 dans l'accumulateur B. Sinon trois cas peuvent se produire, nous vous proposons de les étudier.

Caractère normal

Le caractère appartient à l'alphabet G0, GETS retourne dans l'accumulateur B le code ASCII correspondant.

Minuscule accentuée ou c cédille

Le caractère visé est une minuscule accentuée, dans ce cas GETS retourne dans B le code de l'accentuation ACC soit \$16, et il est nécessaire de faire deux autres appels à GETS:

2e appel : B retourne le code de l'accent

3e appel : B retourne le code ASCII de la minuscule.

Caractère de l'alphabet G2

Si le caractère visé appartient à l'alphabet G2, le code ACC (\$16) sera retourné dans B. Il faudra alors faire un 2e appel à GETS pour lire le code du caractère:

2e appel : B retourne le code du caractère.

Nom : GETS
Adresse du point d'entrée : \$E824
Paramètre d'entrée : Accu A et registre X du 6809-E
Paramètre de retour : Accu B du 6809-E
Effet : retourne le code ASCII du caractère défini par ces coordonnées d'écran (A = ordonnée, X = abscisse)
Exemple : Si vous travaillez avec la cartouche ASSEMBLEUR TOTEX, sous monter un bandeau jaune sur le haut de la fenêtre de travail sert à tabuler l'affichage du contenu des registres du microprocesseur 6809-E.

Le programme ci-dessous lit le caractère P de ce bandeau et retourne en conséquence le code \$50 dans B.

```

      ORG    $A000
GETS  EQU    $E824
      CLRA
      LDX    #0002      1e ligne
      ISR    GETS        2e colonne
      SWI
      END
  
```

5. Gestion du clavier

Lecture rapide du clavier

La routine KTST fait un balayage rapide des touches du clavier pour détecter si l'une d'entre elles est appuyée. Le résultat de ce test est consigné dans le BIT C du registre d'état du microprocesseur:

- si C = 0 aucune touche n'est appuyée
- si C = 1 une touche est enfoncée.

Aucune reconnaissance de la touche appuyée n'étant effectuée par KTST, l'exécution de cette routine est très rapide.

Nom : KTST
Adresse du point d'entrée : \$E809
Paramètre d'entrée : Néant
Paramètre de retour : BIT C du registre d'état du 6809 E
Effet : Positionne C à 1 si une touche est appuyée.
Exemple :

* LECTURE RAPIDE DU CLAVIER.

* LA PRESSION D'UNE TOUCHE ENTRAÎNE SWI

	TITLE	TESTCLAV
	ORG	\$A000
KTST	EQU	\$E809
	ANDCC	##FE FORCE C A 0
	LDX	##FFFF
ENCOR	LEAX	-1,X BOUCL INHIB CLAV
	BNE	ENCOR
SUITE	JSR	KTST
	BCC	SUITE BRANCH SI C=0
	SWI	
	END	

Décodage du clavier

La routine GETC est plus complète que la précédente (KTST). En effet, elle teste le clavier pour détecter une touche éventuellement appuyée, mais elle identifie également cette touche et retourne son code ASCII dans l'accumulateur B du microprocesseur. Etant donné le nombre de touches constituant le clavier et leurs fonctions parfois doubles, plusieurs cas peuvent se présenter:

- si la touche est celle d'un caractère simple de l'alphabet GO, son code ASCII sera alors retourné dans l'accu B au premier appel de GETC;
- si aucune touche n'est enfoncée ou s'il s'agit de SHIFT ou CNT appuyée seule, ou encore si plusieurs touches parmi SHIFT ou CNT ont été enfoncées simultanément, la valeur 0 sera retournée dans l'accu B;
- la touche SHIFT sélectionne le caractère se trouvant en haut de la touche, le code ASCII du caractère ainsi sélectionné sera envoyé dans B;
- la touche CNT force à 0 le bit 6 du code ASCII du caractère appuyé simultanément;
- s'il s'agit d'un caractère accentué et quelle que soit la procédure de saisie, la lecture se fera en trois appels de GETC:
 - 1e appel retourne dans B le code ACC (S16)
 - 2e appel retourne dans B le code de l'accent ou de la cédille
 - 3e appel retourne dans B le code de la minuscule.

Comme nous l'avons étudié dans l'analyse matérielle, la chaîne de traitement du clavier envoie une interruption IRQ au microprocesseur à chaque fois qu'une touche est appuyée. Lors de la réception de ce caractère, le microprocesseur le range dans un buffer circulaire dont l'adresse est située dans le registre BUFCLV (\$6079-\$607A).

Notez que ce registre étant en RAM, l'utilisateur peut resituer ce buffer à l'endroit de son choix. Au démarrage, ce buffer est positionné dans la page 0 du moniteur et possède une capacité de 5 caractères. Quand il est plein, les caractères suivants sont ignorés! Précisément, le registre SIZCLV (\$607B) permet de déclarer la taille de ce buffer; on peut ainsi augmenter (ou diminuer) sa capacité. Mais vous noterez ces valeurs limites:

- SIZCLV étant un registre de 8 bits, la valeur maximum (SFF) correspond à un buffer de 255 caractères au plus.
- la taille minimum à préserver est de 3 caractères pour garantir la lecture des caractères accentués.

Les T08, T09 et T09+ disposent de 10 touches de fonctions. Les codes envoyés par ces touches sont compris entre \$90 et \$99 (inclus). Les pavés numériques peuvent être reconfigurés de façon à envoyer des codes différents de ceux inhérents aux touches. Dans ce cas, les 10 chiffres de 0 à 9 envoient les codes de \$9A à \$A3, le point envoie le code \$A4 et la touche ENT le code \$A5.

Nom	: GETC
Adresse du point d'entrée	: \$E806
Paramètre d'entrée	: Registre BUFCLV \$6079-\$607A Registre SIZCLV \$607B
Paramètre de retour	: Bit C du RE-6809 E ACCUB du 6809 E
Effet	: Réalise une lecture du clavier. Si une touche est appuyée: - le bit C du RE passe à 1 - l'accu B retourne le code ASCII

Exemple

```
* LECTURE DU CLAVIER, LA PRESSION D'UNE
* TOUCHE PROVOQUE SWI, LIRE CODE ASCII
* DANS L'ACCUB PAR COMMANDE R
```

```

      TITLE  CLAVIER1
      ORG    $A000
GETC  EQU    $E806
ENCORE JSR    GETC
      BCC    ENCORE
      SWI
      END
```

Dans ce second exemple, nous avons chaîné GETC et PUTC. En effet, cette opération se réalise facilement car le paramètre de sortie de GETC correspond au paramètre d'entrée de PUTC (Accu B, code ASCII). Ainsi le programme ci-dessous affiche à l'écran le caractère tapé au clavier.

* LECTURE DU CLAVIER ET AFFICHAGE DU
 * CARACTERE CORRESPONDANT A L'ECRAN
 * UTILISATION DE GETC ET PUTC

```

                TITLE  CLAVIER2
                ORG     $A000
GETC    EQU     $E806
PUTC    EQU     $E803
ENCORE  JSR     GETC
        BCC     ENCORE    test bit C=1
        JSR     PUTC
        BRA     ENCORE
        END
  
```

Programmation du clavier

Le dialogue entre l'unité centrale et le clavier est bi-directionnel. Nous avons en effet la possibilité d'envoyer divers codes au clavier pour le programmer. C'est une autre utilisation de la routine GETC, employée en association du registre STATUS (\$6019). Pour envoyer un code au clavier, il faut que le bit 1 de STATUS soit à 1. En retour de GETC, le bit 1 de STATUS est automatiquement remis à 0.

Nom	: GETC
Adresse du point d'entrée	: \$E806
Paramètre d'entrée	: Registre STATUS \$6019 Accu B du 6809 E
Paramètre de retour	: Néant
Effet	: en fonction du code implanté dans B. Les effets sont les suivants:
	Code dans B Effet
	\$F8 RESET soft du clavier: - CAPSLOCK on - keypad selecté pour les chiffres - périphériques pouvant parler (TO9)
	\$F9 CAPS LOCK on
	\$FA CAPS LOCK off
	\$FB Sélection codes spéciaux pour le clavier

\$FC
\$FD
\$FE

Sélection chiffres pour le clavier
Périphériques autorisés à émettre
Périphériques interdits d'émettre
(TO9)

Exemple :

```
* GE PROG POSITIONNE LE CAPS LOCK OFF  
* (VOYANT ROUGE ETEIND) ,
```

```
                TITLE  CLAVIERS  
                ORG     $A000  
GETC            EQU     $E806  
STATUS         EQU     $6019  
                LDB     STATUS  
                ORB     #$02  
                STB     STATUS  
                LDB     #$FA  
                JSR     GETC  
                SWI  
                END
```

Périphériques du clavier

Une prise 9 broches située sur le flanc arrière du clavier du TO9 permet la connexion de divers périphériques, par exemple la souris. Le bit 6 du registre CONFIG (\$6074) indique la présence du périphérique clavier:

bit 6 = 0 pas de périphérique connecté
bit 6 = 1 périphérique connecté.

Le bit 7 du même registre (CONFIG) indique si la souris remplace ou non le light pen.

bit 7 = 0 light pen en fonctionnement
bit 7 = 1 souris remplace le light pen.

Dans le second cas, les appels à LPIN et GETL (voir page 200) sont déviés sur les routines PEIN et GEPE.

Test des boutons du périphérique

La routine PEIN teste l'état des boutons du périphérique connecté au clavier. La réponse est retournée dans le registre d'état du microprocesseur:

bit C = 1 bouton n°1 enfoncé

bit C = 0 cas contraire

bit Z = 1 bouton n°2 enfoncé

bit Z = 0 cas contraire.

A noter que, dans le cas de la souris, le bouton n°2 est le poussoir de droite.

Lecture du périphérique

La routine GEPE lit les coordonnées issues de la position du périphérique et retourne dans Y l'ordonnée (0 à 199 décimal) et dans X l'abscisse correspondante (selon les modes, 0 à 159, 0 à 319 ou 0 à 639). Si la mesure est correcte, le bit C du registre d'état est forcé à 0, dans le cas contraire il est forcé à 1.

Nom : PEIN
Adresse du point d'entrée : \$EC09
Paramètre d'entrée : Néant
Paramètre de retour : Bit C et bit Z du RE 6809 E
Effet : Test des boutons du périphérique

Nom : GEPE
Adresse du point d'entrée : \$EC06
Paramètre d'entrée : Néant
Paramètre de retour : Bit C du RE
Registre X et Y du 6809 E
Effet : lecture de la position du périphérique
Exemple :

* CE PROGRAMME PERMET DE DESSINER
* AVEC LA SOURIS CLIQUER LE BOUTON
* DE GAUCHE ENTRAINE SWI

	TITLE	SOURIS
CONFIG	EQU	\$8074
GEPE	EQU	\$EC06
PLOT	EQU	\$E80F
FORNE	EQU	\$6038
PEIN	EQU	\$EC09

ORG	\$A000	
LDA	CONF LG	
ORA	#SC0	bit 7 et 6 à 1
STA	CONF IG	course reconnue
LDA	#\$01	01=code coule rouge
SEA	FORME	coule forme rouge
ENCOR	JSR	
	GEPE	
	PL0T	
	PEIN	
	BCC	
	ENCOR	test bout gauche
	SWI	
	END	

6. Gestion du light pen

Test du switch light pen

La routine LPIN permet de tester l'état du switch situé sur l'extrémité du light pen. La réponse est retournée dans le bit C (carry) du registre d'état du microprocesseur 6809 E:

Si C = 1, le switch est (ou a été) enfoncé

Si C = 0, le switch n'est pas enfoncé.

Le bit Z du registre d'état est toujours forcé à 0. Un anti rebond de 10 millisecondes est effectué automatiquement. Dans le cas où la souris est connectée, se reporter page 193.

Nom	:	LPIN
Adresse du point d'entrée	:	\$E81B
Paramètre d'entrée	:	Néant
Paramètre de retour	:	Registre d'état du 6809 E
Effet	:	Positionne C à 1 si le switch est enfoncé
Exemple	:	Dans le programme ci-dessous, l'appui sur le switch entraîne un SWI.
	ORG	\$A000
LPIN	BQU	\$E81B
ENCOR	JSR	LPIN
	BCC	ENCOR test C = 1
	SWI	
	END	

Lecture de la zone pointée

La routine GETL présentée ci dessous lit les coordonnées d'un point visé par le light pen et les retourne dans le registre X (abscisse) et Y (ordonnée) du 6809 E.

$0 < Y < 199$ et $0 < X \leq 159$ pour le mode bit-map 16

$0 \leq Y \leq 199$ et $0 \leq X \leq 638$ pour le mode 80 colonnes

$0 < Y < 199$ et $0 < X \leq 319$ pour les autres modes.

En mode 80 colonnes, l'abscisse est obligatoirement paire. La mesure s'effectue sur une trame TV. En cas de problème de lecture (luminosité de l'écran trop faible, light pen trop éloigné de l'écran...), la routine GETL force le bit C du registre d'état du 6809 E à 1. Dans le cas où la souris est connectée, se reporter page 193.

Nom : GETL
 Adresse du point d'entrée : \$E818
 Paramètre d'entrée : Néant
 Paramètre de retour : Registre X et Y du 6809 E
 bit C du RE du 6809 E
 Effet : Lit le point visé par le light pen et retourne
 ses coordonnées dans X et Y.
 Exemple :

* CE PROGRAMME PERMET DE DESSINER
 * SUR L'ECRAN AVEC LE LIGHT PEN EN BLEU
 * SUR FOND BLANC. ON SORT DU PROGRAMME
 * EN APPUYANT SUR LE SWITCH.

```

                TITLE    DESLIGHT
                ORG      $A000
GETL EQU      $E818
PUTC EQU      $E803
PLOT EQU      $E80F
LPIN EQU      $E81B
CHDRAW EQU    $6041
STATUS EQU    $6019
FORME EQU     $6038
                LDA      STATUS
                ANDA     #$EF      BIT4 FORCE A 0
                STA      STATUS
                CLR      CHDRAW    PLOT EN GEST GRAPH
                LDA      #$04      COUL FORME BLEU
                STA      FORME
                LDB      #$0C
                JSR      PUTC      EFFACE FENETRE
                LDB      #$1B      SEQUENC ECHAPP
                JSR      PUTC
                LDB      #$23      PLBIN -
                JSR      PUTC
                LDB      #$20      -ECRAN
                JSR      PUTC
                LDB      #$57      FOND BLANC
                JSR      PUTC
ENCORE JSR     GETL      LECT LIGHTPEN
                JSR      PLOT    AFFICH POINT VISE
                JSR      LPIN    LECT DU SWITCH
                BCC      ENCORE
                SWI
                END
  
```

7. Gestion des manettes de jeux

La routine JOYS fait la lecture de la position d'une manette de jeux. L'accumulateur A du 6809 E contiendra le code de la manette à lire (0 ou 1), l'accumulateur B retourne sa position conformément au codage suivant:

Code renvoyé dans B Position correspondante

\$00	Centre
\$01	Nord
\$02	Nord-est
\$03	Est
\$04	Sud-est
\$05	Sud
\$06	Sud-ouest
\$07	Ouest
\$08	Nord-ouest

Le bit de retenue (C) du registre d'état 6809 E est mis à 1 si le bouton ACTION a été enfoncé, et à 0 dans le cas contraire. Il n'y a pas d'anti-rebond prévu pour ces fonctions.

Nom : JOYS
Adresse du point d'entrée : \$E827
Paramètre d'entrée : Accu A du 6809 E
Paramètre de retour : Accu B et BIT C du RE 6809 E
Effet : Effectue une lecture de la manette désignée dans A et retourne sa position dans B.
Exemple : Après avoir lancé le programme ci-dessous, inclinez la manette sur l'une des huit positions et appuyez sur le bouton ACTION.

Le bouton ACTION entraîne le SWI. Par la commande R du monitor (cartouche ASSEMBLEUR TOTEK), vérifiez le code dans B.

```

ORG $A000
JOYS EQU $E827
CLR A
INCOR ISR JOYS
BCC ENCOR
SWI
END
    
```

8. Gestion de l'interface de communication

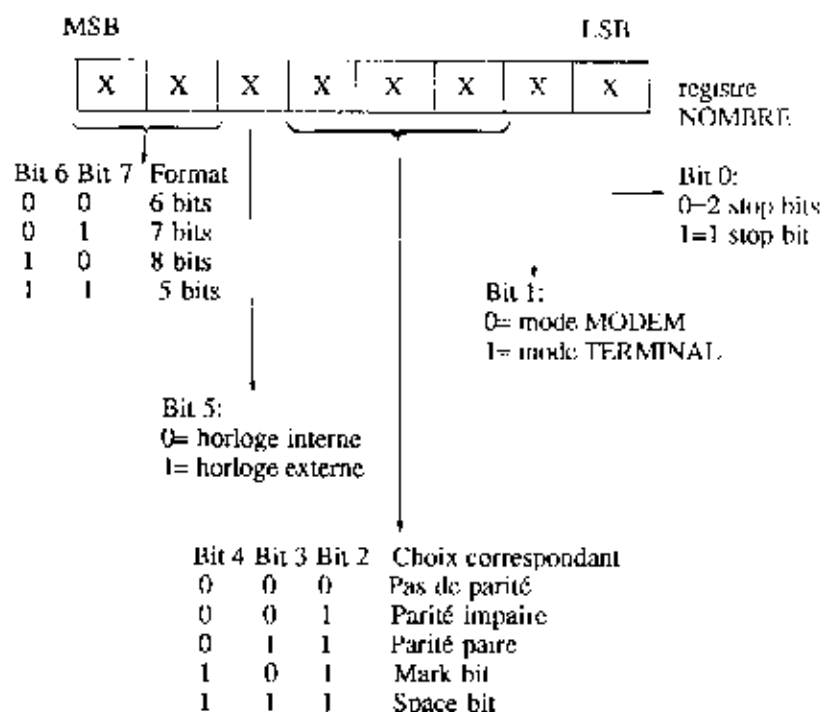
La routine RSCO permet d'envoyer (ou de recevoir) des valeurs transférées en série ou en parallèle, sur l'interface de communication. Les registres à paramétrer sont nombreux, leur contenu précise le mode d'échange choisi, l'état de la communication ou la vitesse de transmission.

Le registre RS.OPC (\$602B) sélectionne le mode de transfert désiré, les codes sont à choisir parmi les suivants:

Opération désirée	Valeur à implanter dans RS.OPC
Ouverture en lecture-écriture RS232	\$01
Lecture d'un caractère en RS232	\$02
Ouverture en écriture seule	\$04
Ecriture d'un caractère en série	\$09
Ecriture d'un caractère en série	\$0C
Fermeture en série	\$11
Fermeture en série	\$14
Ouverture en écriture parallèle	\$40
Ecriture d'un caractère en parallèle	\$08
Fermeture en parallèle	\$10
Copie graphique d'écran	\$20

En cas d'écriture, l'accu B doit contenir l'octet à envoyer; en cas de lecture l'accu B contient le caractère reçu. Si seule la ligne série est ouverte, une fermeture en parallèle sera considérée comme une fermeture série. Pour la copie graphique de l'écran, le registre GRCODE (\$6047) doit contenir le code de mise en mode graphique spécifique à l'imprimante utilisée. GRCODE contient la valeur 7 par défaut. La copie graphique s'exécute selon le mode 40 ou 80 colonnes.

Pour les transmissions série, vous préciserez le format des données échangées, en programmant le registre NOMBRE (\$6046) selon les options suivantes:



Enfin, vous choisirez la vitesse de transmission en programmant le registre BAUDS (\$6044-\$6045).

Vitesse désirée: Valeur à implanter dans BAUDS

50 bauds	\$0001
75 bauds	\$0002
110 bauds	\$0003
135 bauds	\$0004
150 bauds	\$0005
300 bauds	\$0006
600 bauds	\$000A
1 200 bauds	\$0008
1 800 bauds	\$0009
2 400 bauds	\$000A
3 600 bauds	\$000B
4 800 bauds	\$000C

7 200 bauds	\$000D
9 600 bauds	\$000E
19 200 bauds	\$000F

Cependant, pour la compatibilité avec le TO7, ce paramètre peut être pris (pour certaines vitesses) dans la table BDTAB située à l'adresse \$E836. Le premier paramètre (sur 2 octets) représente la vitesse pour 110 bauds, les suivants pour 300, 600, 1 200, 2 400 et 4 800 bauds.

Le registre RS.STA en retour de la routine RSCO nous indique l'état de la communication. L'interprétation de son contenu est conforme au tableau ci-dessous:

Contenu de RS.STA	Interprétation de l'état de la communication
\$01	Ouvert en lecture-écriture RS232
\$04	Ouvert en écriture seule RS232
\$10	Fermé
\$40	Ouvert en écriture parallèle
\$80	Périphérique non prêt

Le bit de retenue (C) du registre d'état 6809 E est forcé à 0 si tout s'est passé normalement, et à 1 dans le cas contraire.

Nom	: RSCO
Adresse du point d'entrée	: \$E812
Paramètres d'entrée	: Accu B du 6809 E (en cas d'écriture) Registre RS.OPC \$602B Registre BAUDS \$6044-\$6045 Registre NOMBRE \$6046 Registre GRCODE \$6047 (TO9)
Paramètres de retour	: Accu B du 6809 E (en cas de lecture) BIT C du RE 6809 E Registre RS.STA \$602C
Effet	: Permet la lecture ou l'écriture en série RS232, ou l'écriture en mode parallèle.

9. Gestion du lecteur-enregistreur de cassettes

La routine K7CO permet de gérer les échanges de données entre l'unité centrale et le lecteur-enregistreur de cassettes. A cet effet, le registre K7.OPC (\$6029) permet de spécifier, par le choix de son contenu, l'opération désirée:

Opération désirée	Valeur à implanter dans K7.OPC
Ouverture en lecture	\$01
Lecture d'un caractère	\$02
Ouverture en écriture	\$04
Ecriture d'un caractère	\$08
Fermeture	\$10

Dans le cas d'une écriture sur la cassette, l'accum B doit contenir l'octet à envoyer avant l'appel de la routine. Dans le cas d'une lecture, l'accum B reçoit le caractère. En retour de la routine K7CO, le registre K7.STA contiendra le code de l'opération réalisée:

Contenu de K7.STA	Interprétation
\$01	Ouvert en lecture
\$04	Ouvert en écriture
\$10	Fermé
\$80	Périphérique non prêt

En cas d'erreur ou si, par exemple, le lecteur n'est pas connecté à l'unité centrale, la routine K7CO positionne automatiquement le BIT C du registre d'état microprocesseur à 1.

Nom	: K7CO
Adresse du point d'entrée	: \$E815
Paramètre d'entrée	: Accum B du 6809 E (en cas d'écriture) Registre K7.OPC \$6029
Paramètre de retour	: Accum B du 6809 E (en cas de lecture) BIT C du RE 6809E Registre K7.STA \$602A
Effet	: Permet d'écrire ou de lire une donnée sur le périphérique cassette.

10. Contrôleur de disquettes

Les disquettes 3,5 pouces utilisées par les TO8, TO9 et TO9+ sont divisées en 80 pistes de 16 secteurs chacune. Au gré de l'utilisateur, elles peuvent être formatées en simple ou double densité ce qui modifiera le format des secteurs:

simple densité = 128 octets par secteur
double densité = 256 octets par secteur.

En conséquence, le contrôleur de disquettes intégré aux unités centrales peut indifféremment travailler dans les deux modes. Nous pouvons gérer les échanges de données entre le microprocesseur et le lecteur de disquettes selon deux principes:

- au niveau physique, en utilisant le point d'entrée du moniteur
 - au niveau logique, en manipulant des fichiers au format BASIC Microsoft.
- Nous vous proposons d'étudier ces deux possibilités.

Gestion physique

Les routines DKFORM et DKCO permettent respectivement de formater une disquette et de lire ou écrire des données. Le nombre de registres constituant les paramètres d'entrées ne doivent effrayer personne; globalement leurs rôles sont les suivants:

DK.OPC contient le code de l'opération demandée
DK.DRV repérage du numéro de lecteur concerné
DK.SEC repérage du numéro de secteur concerné
DK.TRK repérage du numéro de piste concerné
DK.BUF repérage de la zone tampon d'écriture-lecture

Donc, par la programmation du registre DK.OPC vous définirez la tâche à réaliser. Les codes seront choisis parmi les suivants:

Travail à réaliser	Code à implanter dans DK.OPC
Formatage (sans vérification)	\$00
Initialisation du contrôleur	\$01
Lecture d'un secteur	\$02
Positionnement en simple densité	\$04
Ecriture d'un secteur	\$08
Positionnement en double densité	\$10

Recherche de la piste 0	\$20
Recherche de piste (reperée par DK.TRK)	\$40
Vérification en écriture	\$80

Remarques:

- Si l'initialisation s'est déroulée sans problèmes, le bit de retenue (ou Carry C) du registre d'état du microprocesseur est mis à 0 et le type de contrôleur "D" est retourné dans le registre DK.STA. Dans le cas contraire, le bit de retenue est forcé à 1 et le code d'erreur \$40 est mis dans le registre DK.STA.
- Pour la vérification en écriture, il est nécessaire de faire un "OU" logique entre le code \$80 et le code de l'opération à vérifier.

En fonction du code implanté dans DK.OPC, certains registres ci-dessous devront être initialisés:

- Registre DK.DRV (\$6049). Ce registre doit contenir le numéro de lecteur concerné (soit une valeur comprise entre 0 et 4).

Précisons à ce sujet que le lecteur intégré au T09 ne fonctionne que sur une seule face ainsi que le lecteur stand alone référencé DD09-350. Le lecteur intégré au T09+ fonctionne sur les deux faces. Le lecteur stand alone référencé DD90-352 raccordé au T08 par le connecteur DIN 14 broches fonctionne également sur les deux faces. Pour le registre DK.DRV, les numéros 0 et 1 correspondent aux deux faces du lecteur interne, les numéros 2 et 3 aux deux faces du disque externe (donc le 1 et le 3 sont inutilisables sur le T09). Le numéro 4 est le "RAM disk" ou "disque virtuel" physiquement représenté par l'extension mémoire 64 K pour le T09 ou de RAM résidente pour les T08 et T09+. Pour le T09, cette interface sera gérée comme une unité de disque double densité, ayant 16 pistes (de 16 à 32) de 16 secteurs contenant 256 octets.

- Registre DK.TRK (\$604A-\$604B). Ce registre doit contenir le numéro de piste où l'on désire travailler; la valeur est comprise entre 0 et 79.
- Registre DK.SEC (\$604C). Ce registre doit contenir le numéro de secteur concerné par la lecture ou l'écriture; la valeur est comprise entre 1 et 16.
- Registre DK.NUM (\$604D). Ce registre contient l'entrelacement des secteurs logiques sur la disquette lors du formatage. Cette technique permet d'accélérer les temps d'accès au disque. Sur le T09, l'entrelacement utilisé par le BASIC est de 7. Une lecture rapide des secteurs peut être optimale avec un entrelacement de 2.
- Registre DK.BUF (\$604F-\$6050). Ce registre contient l'adresse d'origine d'une zone tampon en RAM, d'une capacité de 128 octets en simple densité ou 256 octets en double densité. Ce buffer contiendra soit les données à écrire sur la disquette, soit les données lues sur la disquette.

En retour des routines DKFORM et DKCO, le registre DK.STA peut être interrogé afin de savoir si l'opération s'est bien déroulée. Dans la négative, DK.STA contiendra un code précisant l'erreur détectée (et bit C = 1). Les interprétations sont les suivantes:

Code lu dans DK.STA	Interprétation
\$01	Disquette protégée.
\$02	Problèmes de timing ou données perdues.
\$04	Erreur de secteur, identificateur incorrect. Secteur ne pouvant être lu ou erreur sur le checksum, cependant la piste peut être correcte.
\$08	Erreur sur les données, l'identificateur de secteur est correct, mais les données ne peuvent être lues, ou le checksum est incorrect.
\$10	Lecteur non prêt; le moteur n'est pas en route ou le lecteur spécifié est inexistant.
\$20	Erreur sur vérification. La zone tampon en mémoire et la zone correspondante écrite sur la disquette ne sont pas identiques.

Nom	: DKFORM
Adresse du point d'entrée	: SE007
Nom	: DKCO
Adresse du point d'entrée	: \$E82A
Paramètre d'entrée	: Registre DK.OPC \$6048 Registre DK.DRV \$6049 Registre DK.SEC \$604C Registre DK.TRK \$604A-\$604B Registre DK.BUF \$604F-\$6050
Paramètre de retour	: BIT C du RE 6809E Registre DK.STA \$604E
Effet	: Permet de formater une disquette et d'accomplir des actions de lecture-écriture.

Format BASIC Microsoft

Les disquettes sont structurées selon le format BASIC Microsoft. Cette règle de base nous assure l'interactivité de fichiers créés sous le contrôle de logiciels différents. Ce format impose que la piste 20 soit une zone réservée et organisée de la manière suivante:

Secteur 1 :	Nom de la disquette sur les 8 premiers octets
Secteur 2 :	Table d'allocation des fichiers (FAT)
Secteur 3 à 16 :	Catalogue

Table d'allocation des fichiers

Les fichiers sont organisés en bloc de 1 K en simple densité ou 2 K en double densité. Dans tous les cas nous aurons deux blocs par piste. Les blocs sont numérotés à partir de 0. Chaque octet de la table d'allocation des fichiers, à partir de l'octet 1, représente un bloc physique.

Organisation de la "FAT":

Octet 0 :	0
Octet 1 :	Bloc 0, piste 0, secteurs 1 à 8
Octet 2 :	Bloc 1, piste 0, secteurs 9 à 16
Octet 3 :	Bloc 2, piste 1, secteurs 1 à 8
Octet 4 :	Bloc 3, piste 1, secteurs 9 à 16
<hr/>	
Octet 2j-1 :	Bloc 2j-2, piste j-1, secteurs 1 à 8
Octet 2j :	Bloc 2j-1, piste j-1, secteurs 9 à 16
<hr/>	
Octet 160 :	Bloc 159, piste 79, secteurs 9 à 16

En simple densité, la FAT est limitée à 127 blocs. Un octet de la "FAT" représentant un bloc physique peut avoir comme valeur:

- \$FF, qui signifie bloc non alloué
- \$FB, qui signifie bloc réservé
- Tout nombre de 0 à \$BF signifie bloc alloué. Dans ce cas, le nombre représente le numéro du bloc logique suivant du même fichier.
- Tout nombre de \$C1 à \$C8 signifie dernier bloc d'un fichier. Les quatre bits de poids faible indiquent le nombre de secteurs utilisés dans ce dernier bloc.

Le catalogue

Le rôle du catalogue est de donner la liste de tous les fichiers enregistrés sur la disquette. Il occupe à ces fins 14 secteurs. Chaque fichier est répertorié sur 32 octets. Il y a 4 fichiers répertoriés par secteur en simple densité et 8 fichiers par secteur en double densité. En conséquence, le catalogue peut donc répertorier au maximum 56 fichiers en simple densité et 112 en double densité.

Chaque fichier dans le catalogue est mémorisé de la manière suivante:

Octets 00 à 07	: Nom du fichier, cadré à gauche et complété par des blancs
Octets 08 à 0A	: Suffixe du fichier (.BAS, .BIN, etc.), cadré à gauche et complété par des blancs.
Octet 0B	: Type de fichier: 0 pour un programme BASIC, ASCII ou binaire; 1 pour des données BASIC en ASCII 2 pour un programme en langage machine 3 pour un fichier assembleur édité en ASCII
Octet 0C	: Sémaphore: \$FF pour de l'ASCII \$00 pour du binaire
Octet 0D	: Numéro du premier bloc logique du fichier
Octets 0E à 0F	: Nombre d'octets utilisés dans le dernier secteur du fichier
Octets 10 à 17	: Commentaire associé au fichier
Octets 18 à 1F	: Réservés.

Le premier octet de chaque entrée dans le catalogue indique son état:

\$00	: Entrée non allouée, pas de fichier répertorié pour cette entrée
\$20 à \$7F	: Code ASCII du premier caractère du nom de fichier, donc entrée allouée
\$FF	: Fin logique du catalogue

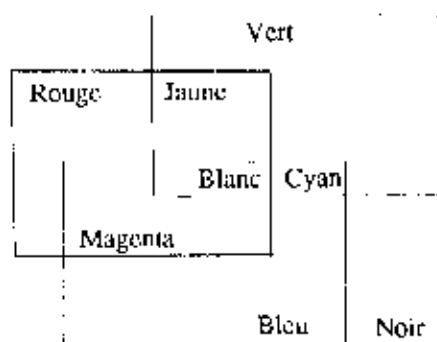
Lors de la création du catalogue, ces octets sont tous mis à \$FF. Chaque fois qu'un fichier est créé, la fin logique du catalogue est déplacée dans le premier octet de l'entrée suivante, jusqu'à concurrence de la capacité maximum (catalogue plein). La destruction d'un fichier entraîne la mise à zéro du premier octet de son entrée (l'entrée devient non allouée). Dans ce cas, tout fichier créé ultérieurement se verra attribuer en priorité cette entrée.

11. Programmation de la Palette

La génération d'une couleur est obtenue en dosant judicieusement les teintes fondamentales utilisées en télévision:

- le Rouge
- le Vert
- le Bleu.

Par association (ou dissociation) de ces trois composantes primaires, nous obtenons quatre autres couleurs:



Le noir constituant une 8e couleur est en fait obtenu par l'absence des trois fondamentales.

En jouant sur des intensités plus ou moins importantes de ces trois teintes fondamentales, nous pouvons créer des nuances intermédiaires et obtenir ainsi, par exemple, un jaune plus ou moins foncé.

A chaque couleurs correspond un registre interne du circuit Palette. Nous disposons sur les TO8, TO9 et TO94 de 16 couleurs utilisables simultanément; en conséquence, Palette possède 16 registres internes différents. Le premier s'appelle 0, son contenu permet de définir la teinte visualisée lors de l'appel de l'attribut de couleur 0, et ainsi de suite jusqu'au registre 15.

Chaque registre est cadré sur 16 bits; le principe de codage est le suivant:

MSB LSB
 XXXM BBBB VVVV RRRR

Les trois bits de poids fort sont indifférents. Le bit M (masque) détermine si la couleur correspondante au registre sera transparente ou non lors d'une incrustation vidéo.

Si M = 0, couleur non transparente.

Si M = 1, cas contraire.

Les trois mots de quatre bits BBBB, VVVV, RRRR permettent respectivement de définir l'intensité de bleu, de vert et de rouge. Il faut, bien sûr, partir du principe que mettre la valeur 0000 dans BBBB aura pour effet ne n'avoir pas de bleu dans la teinte. Par contre la valeur 1111 dans BBBB correspondra à un bleu saturé (maximum d'intensité).

Par fondamentale, nous avons donc $2^4 = 16$ combinaisons possibles. Ayant trois teintes fondamentales codées sur le même principe, cela fait:
 $16 \times 16 \times 16 = 4\,096$ teintes différentes.

Ecriture-lecture d'un registre couleur

La routine SETP permet d'effectuer toutes les opérations de lecture et d'écriture des registres couleurs du circuit Palette. Le numéro de registre (ou de couleur) est, dans ce cas, implanté dans l'accumulateur A. La technique de programmation dite "des masques" est automatiquement utilisée par la routine SETP. A cet effet le registre X est un masque ET, et le registre Y un masque OU. Nous rappelons que, par principe, le masque ET permet de forcer des 0.

Exemple:

X X X M	B B B B	V V V V	R R R R	Contenu du registre
1 1 1 1	1 1 1 1	0 0 0 0	1 1 1 0	Contenu de X
<hr/>				
X X X M	B B B B	0 0 0 0	R R R 0	Résultat

Alors que le masque OU permet de forcer des 1,

Exemple:

X X X M	B B B B	V V V V	R R R R	Contenu du registre
+ 0 0 0 0	0 0 0 0	1 1 1 1	0 0 0 1	Contenu de Y
<hr/>				
X X X M	B B B B	1 1 1 1	R R R 1	Résultat

Donc par association de X et de Y, vous pouvez redéfinir chaque bit de chaque registre couleur. En fait, l'opération logique réalisée par SETP est:

$$\text{REGISTRE} = (\text{REGISTRE} \cdot X) + Y$$

Prenons un exemple: on désire changer le contenu du registre 0 et obtenir à la place du noir, un jaune demi teinte. Il faut alors mélanger du rouge demi-teinte avec du vert demi-teinte.

Soit:

BBBB = 0000 Pas de bleu
VVVV = 0111 Moitié de vert
RRRR = 0111 Moitié de rouge

Les opérations binaires sont:

XXXXM	BBBB	VVVV	RRRR	Cont. init. reg. 0
1111	0000	0111	0111	Contenu de X

XXXXM	0000	0VVV	0RRR	Résult. interméd.
-------	------	------	------	-------------------

Puis

XXXXM	0000	0VVV	0RRR	
+ 0000	0000	0111	0111	Contenu de Y

XXXXM	0000	0111	0111	Résultat final
-------	------	------	------	----------------

Donc pour cette teinte, le registre X doit contenir la valeur \$F077, le registre Y la valeur \$0077, et l'accumulateur A la valeur 0 (registre numéro 0), cf. le programme PALETTE ci-contre.

Pour lire le contenu d'un registre sans le modifier, X doit contenir \$FFFF et Y doit contenir \$0000.

Nom	: SETP
Adresse de point d'entrée	: \$E000
Paramètre d'entrée	: Accu A
	: Registre X
	: Registre Y du 68000
Paramètre de retour	: Registre X
Effet	: Permet d'écrire dans le registre Palette désigné par A, la valeur binaire résultant d'un masquage ET et OU des registres X et Y. Cette opération est effectuée sans attente de retour frame. En cas de lecture, la valeur est à lire dans le registre X.
Exemple	:

* REMPLACEMENT DU NOIR PAR UN JAUNE
* DEMI TEINTE.

```

TITLE PALETTE
ORG $A000
STEP EQU $EC00
CLR A COUL=0
LDX #$F077
LDY #$0077
JSR STEP
SVI
END

```

Programmation complète de la palette

Si vous voulez redéfinir le contenu de l'ensemble des 16 registres couleurs du circuit Palette, vous pouvez évidemment refaire la procédure décrite ci-dessus 16 fois de suite !! Mais il y a plus simple. Le registre A doit contenir dans ce cas la valeur \$FF. Le registre X doit pointer une table de 32 octets qui représentent les 16 valeurs à implanter dans les registres. Les deux premiers octets codent la couleur 0, les deux derniers codent la couleur 15 (ceci en mode 40 colonnes).

Vous appelez ensuite une seule fois la routine SETP, et l'ensemble des registres seront reprogrammés par un transfert automatique du contenu de la table dans les registres Palette.

Nom	: SETP
Adresse du point d'entrée	: \$EC00
Paramètre d'entrée	: Accu A (valeur \$FF) Registre X du 6809 E
Paramètre de retour	: Néant
Effet	: SETP prend des valeurs dans une table pointée par X, et les charge automatiquement dans les 16 registres couleurs. Cette opération est effectuée avec attente de retour franc.
Exemple	: Le programme ci-dessous peut se scinder en 2. Le premier, PROG1, implante 32 octets à partir de l'adresse \$B000. Le second, PROG2, réalise la programmation complète de la palette (transfert des octets de la table dans les registres couleurs). Notez que les valeurs correspondent aux 16 possibilités de la fondamentale Rouge. Le résultat final sera un dégradé du rouge-noir au rouge saturé que vous pourrez visualiser après un RESET dans le menu Réglages et Préférences.

Modes	Numéro de couleur															
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
40 col.	Ff	Ff	Ff	Ff	Ff	Ff	Ff	Ff	Ff	Ff	Ff	Ff	Ff	Ff	Ff	Ff
80 col.	f	F	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Page 1	f	-	F	-	-	-	-	-	-	-	-	-	-	-	-	-
Page 2	f	F	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Superpos.	f	F2	F1	-	-	-	-	-	-	-	-	-	-	-	-	-
Bitmap 4	F	F	F	F	-	-	-	-	-	-	-	-	-	-	-	-
Bitmap 16	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F
Triple sup.	f	F1	F2	-	F3	-	-	-	F4	-	-	-	-	-	-	-

Légende: f = couleur de fond F = couleur de forme - = cases inaccessibles

En programmation complète de la palette, les numéros logiques ne correspondent plus aux numéros physiques. Il faut donc accéder à la table de conversion suivante pour modifier les couleurs selon le mode sélectionné.

Modes	Numéro de couleur															
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
40 col.	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
80 col.	8	14	10	11	12	13	9	15	0	1	2	3	4	5	6	7
Page1	8	9	10	11	12	13	14	15	0	1	2	3	4	5	6	7
Page2	8	9	10	11	12	13	14	15	0	1	2	3	4	5	6	7
Superpos.	8	9	10	11	12	13	14	15	0	1	2	3	4	5	6	7
Bitmap 4	8	9	10	11	12	13	14	15	0	1	2	3	4	5	6	7
Bitmap 16	8	9	10	11	12	13	14	15	0	1	2	3	4	5	6	7
Triple sup.	8	9	10	11	12	13	14	15	0	1	2	3	4	5	6	7

Fichier PALETTE.CFG

Le fichier PALETTE.CFG contient les données d'une palette sauvegardée par l'utilitaire "Choisir sa palette de couleurs" issu du menu de garde "Réglages et Préférences ". Il s'agit d'un fichier binaire de 32 octets correspondant aux 16 mots de 16 bits, à transférer dans les 16 registres couleurs du circuit Palette. Le cadrage est, bien entendu, identique à celui décrit au début de ce chapitre.

```

MSB                                     LSB
XXXXM BBBB VVVV RRRR

```

12. Génération de sons

Création d'un bip

La création d'un bip sonore est obtenue en envoyant le code BEI. (S07) à la routine PUTC (\$E803), par l'accumulateur B du 6809 E.

Exemple:

```
ORG    SA000
PUTC   EQU    $E803
LDB    #S07
JSR    PUTC
SWI
END
```

Création musicale

La routine NOTE permet d'écrire des compositions musicales que jouera docilement votre micro-ordinateur. D'une manière générale, chaque note sera caractérisée par:

- son identification
- son octave
- sa durée
- son timbre

et l'ensemble du morceau musical sera conditionné par le temps.

L'identification de la note à jouer sera précisée dans l'accumulateur B du 6809 E. Vous avez 13 notes à votre disposition, de DO à UT, plus le silence. Le tableau ci-dessous donne la correspondance entre les notes désirées et les codes à envoyer.

Note désirée	Code à envoyer dans B
Silence	\$30
DO	\$31
DO#	\$32
RE	\$33
RE#	\$34
MI	\$35
FA	\$36
FA#	\$37
SOL	\$38
SOL#	\$39
LA	\$3A
LA#	\$3B
SI	\$3C
UT	\$3D

L'octave permet de situer la hauteur de la note au sein de la gamme générée par l'unité centrale. Cinq octaves sont disponibles, de l'octave 1 la plus grave à l'octave 5 la plus aiguë. Les codes à implanter dans le registre OCTAVE (\$6036-\$6037) sont à choisir parmi les suivants:

Octave désirée	Code correspondant
1	\$0010
2	\$0008
3	\$0004
4	\$0002
5	\$0001

La durée de la note sera indiquée dans le registre DUREE (\$6033-\$6034). Notez que les codes correspondants suivent une progression arithmétique identique à la durée relative des notes. Ainsi, comme une blanche vaut deux noires, le code de la blanche étant 48, celui de la noire est 24.

Durée de la note	Valeur correspondante:
Ronde	96
Blanche pointée	72
Blanche	48
Noire pointée	36
Noire	24
Croche pointée	18
Croche	12
Double croche pointée	09
Double croche	06
Triple croche pointée	05
Triple croche	03

Dans un triolet:

Noire	16
Croche	08
Double croche	04
Triple croche	02

Le coefficient appelé timbre et chargé dans le registre du même nom (\$6035) peut varier de 0 à 5. Physiquement, il permet de modifier le rapport cyclique du signal créant le son, modifiant ainsi son taux d'harmoniques. La valeur 0 correspond à une note "plate".

Le tempo détermine la vitesse générale d'exécution du morceau. C'est comme le métronome que l'on règle plus ou moins vite et qui change la base ou référence de temps, sans changer la durée relative des notes (une croche sera toujours une croche). La valeur du tempo est chargée dans le registre TEMPO (\$6031-\$6032) et peut varier entre 1 et 255.

Nom	: NOTE
Adresse du point d'entrée	: SE3E
Paramètres d'entrée	: Accu B du 6809 B Registre OCTAVE \$6036-\$6037 Registre DUREE \$6033-\$6034 Registre TIMBRE \$6035 Registre TEMPO \$6031-\$6032
Paramètres de retour	: Néant
Effet	: Joue la note écrite dans B, selon les caractéristiques implantées dans les registres d'entrée.

13. Commutation des mémoires ROM

Le TO9 possède cinq boîtiers ROM accessibles entre les adresses 0000 et 3FFF (1W56, 40, 39, 38 et la cartouche extérieure). Ces boîtiers, hormis la cartouche, contiennent l'ensemble des logiciels intégrés au TO9:

- FICHES ET DOSSIERS
- PARAGRAPHE
- BASIC 128
- BASIC 1.0
- REGLAGES ET PREFERENCES
- EXPLOITATION DE FICHIERS

et L'EXTRAMON.

Cet espace adressable de 16 Ko ne convient pas à tous les logiciels. Certains en effet dépassent cette capacité et sont alors organisés en banques commutables de 16 Ko. La routine COMS permet d'accéder à n'importe quel sous-programme implanté dans n'importe quelle banque de n'importe quel boîtier ROM. Dans ce cas, le registre U du microprocesseur 6809 E doit contenir l'adresse du début du programme à exécuter, et l'accumulateur A contient un octet défini de la manière suivante:

00SS00BB

SS repère le numéro de boîtier et BB le numéro de banque. Le tableau ci-dessous vous aidera dans ces choix.

Programme	SS	BB
Cartouche externe	11	XX
Fiches et dossiers	10	XX
Paragraphe	01	XX
BASIC 128	00	00
Extramôn	00	01
BASIC 1	00	10
Exploitation fichiers	00	11

Chaque ROM possède à l'adresse \$20 son numéro de banque. Ainsi le moniteur peut s'assurer de l'identité de la banque sélectionnée, afin de mieux se repérer dans les boîtiers constitués de plusieurs banques de 16 Ko en parallèle. Dans le cas des T08 et T09+, le tableau suivant doit être utilisé pour la détermination du contenu de A:

Programme	SS	BB
Cartouche externe	11	XX
BASIC 512	00	00
Extramon	00	01
BASIC 1	00	10
Exploitation fichiers	00	11

Nom	COMS
Adresse du point d'entrée	\$EC03
Paramètre d'entrée	Accès A
Paramètre de retour	Régistre U du 6809 E
Effet	NCant Permet d'accéder à n'importe quel programme en ROM

14. Accès à l'extramoniteur

La routine EXTRA permet d'accéder aux routines de l'extramoniteur. En entrée, l'accumulateur B contient le numéro de la routine à exécuter; les registres nécessaires des pages \$6100-\$62FF devant être positionnés selon la routine appelée. En sortie, l'accumulateur B contient un numéro d'erreur, les registres des pages \$6100-\$62FF étant modifiés si nécessaire.

Pour de plus amples détails sur l'extramon, reportez-vous à la sixième partie de cet ouvrage traitant de son utilisation.

Nom	: EXTRA
Adresse du point d'entrée	: SEC0C
Paramètre d'entrée	: Accu B Registres \$6100-\$62FF, selon routine
Paramètre de retour	: Accu B Registres \$6100-\$62FF, selon routine
Effet	: Accès aux routines de l'extramon

15. Gestion des interruptions

Nous avons appris, au travers de l'étude matérielle, que certaines interruptions du 6809 E sont utilisées:

- L'interruption IRQ est déclenchée soit pour le dialogue clavier, soit par le timer du 6846 pour faire clignoter le curseur à l'écran toutes les 100 ms, soit encore pour gérer la souris ou les manettes (TO8 et TO9+).
- L'interruption IRQ est utilisée pour la gestion du light pen.

Par les différents exemples d'utilisation des routines du moniteur donnés dans les chapitres précédents, nous savons que les interruptions logicielles SWI sont utilisées pour arrêter un programme ou "reprenre la main".

Mais ces différentes interruptions sont programmables. A chacune d'elles correspond un registre en RAM contenant l'adresse du programme qui doit la traiter. A la mise sous tension ou après un redémarrage "à chaud", ces registres ont été initialisés avec l'adresse d'un programme du moniteur. En conséquence, vous pouvez dériver ou aiguiller ces interruptions sur des programmes personnels, en ré-initialisant ces registres !

• Aiguillage des IRQ

L'adresse de votre programme de gestion de l'IRQ générée par le timer doit être implantée dans le registre TIMEPT (\$6027-\$6028) et dans le registre IRQPT (\$6021-\$6022).

- Le bit 5 du registre STATUS (\$6019) doit être forcé à 1.

Les registres DP et S doivent être conservés.

- Votre programme doit obligatoirement finir par un JMP KBIN (\$E830) pour valider l'interruption.

Si l'interruption IRQ est générée par une autre source que le timer, l'adresse du programme sera implantée en TIMEPT.

Le programme de la page suivante donne un exemple d'une telle procédure. Le programme de gestion de l'IRQ est écrit à partir de l'adresse \$A000, et le programme de dérivation est écrit en \$7000 (attention, lancez le programme en \$7000 et non en \$A000). Après lancement, la couleur du cadre changera toute les 100 ms sans pour autant "monopoliser" votre machine.

• Aiguillage des FIRO

Vous devez mettre l'adresse de votre programme en FIROPT (\$6023 \$6024).

• Aiguillage des SWI

Pour gérer les SWI, vous devez mettre l'adresse de votre programme en SWI1 (\$602F-\$6030). SWI2 saute directement en \$6800, et SWI3 en \$7000.

Mais attention, certaines routines du moniteur sont interruptibles et vos programmes ne doivent pas modifier leurs paramètres. Un JMP MENU (SE82D) fait revenir à la page d'en-tête.

* PROGRAMME DE GEST. IRQ ECRIT EN \$A000

* PROGRAMME DE DERIVATION EN \$7000

```

                TITLE  GEST2IRQ
                ORG     $A000
PUTC EQU        $E803
RETOUR EQU      $E830
FILE EQU        $E800
                LDB     #$1E
                JSR     PUTC
                LDB     FILE
                CMPB    #$67
                BNE     SUIT
                LDB     #$60
                STE     FILE
SUIT JSR         PUTC
                INC     FILE
                JMP     RETOUR

```

```

                ORG     $7000
STATUS EQU      $6019
TIMEPT EQU      $6027
IRQPT EQU       $6021
                LDA     STATUS
                ORA     #$20
                STA     STATUS
                LDX     #$A000
                STX     TIMEPT
                STX     IRQPT
                LDA     #$60
                STA     FILE
                SWI
                END

```

16. Initialisation

Le jargon communément employé pour expliquer les processus de fonctionnement logiciel d'une machine utilise les termes de "démarrage à chaud" ou "démarrage à froid" pour qualifier un "reset". Pourtant, un reset sera toujours la conséquence d'un passage à 0 de l'entrée RESET d'un microprocesseur. Alors ?

Le démarrage à froid qualifie le reset crée matériellement et automatiquement par le hard de la machine, à la mise sous tension. Le démarrage à chaud est le reset déclenché volontairement par l'utilisateur, à l'aide du poussoir appelé INIT. Un flag en RAM permet au microprocesseur de faire la différence. La distinction entre les deux implique des tâches différentes effectuées par le programme d'initialisation.

• En cas de reset ou démarrage à chaud, ne seront pas modifiés:

- le réglage du crayon optique
- la programmation du circuit Palette
 - le code de mise en mode graphique, spécifique à l'imprimante utilisée
- le contenu du disque RAM.

• En cas de reset ou démarrage à froid, ne seront pas modifiés:

- l'initialisation de la page 0
- le formatage du disque RAM disque s'il existe (TO9)
- la programmation de la palette avec couleurs standards.

• Dans les deux cas, il y a les modifications suivantes:

- les registres d'aiguillage d'interruptions sont initialisés avec les adresses des routines moniteur correspondantes;

- les redirections des routines sont réinitialisées aux valeurs du moniteur, sauf celle du crayon optique;

les registres contenant les adresses des différentes tables (décodage clavier, générateur de caractères standard, générateur de caractères utilisateur) sont réinitialisés de manière à pointer sur les tables standards;

- le clavier est réinitialisé et son buffer de reception est remis à la valeur initiale;

les interruptions sont fermées et le curseur est éteint;

- la première banque RAM est sélectionnée;

- les bits du registre CONFIG (\$6074) sont positionnés;

le MODEM de seconde génération ainsi que toutes les extensions dont le champ d'adressage se trouve entre \$E7F0 et \$E7A5 sont réinitialisés;

- tous les autres registres sont remis aux valeurs standard, le plus souvent à 0.

17. Informations complémentaires

Points d'entrées standard du moniteur

Nom	Point d'entrée	Effet
EXTRA	\$EC0C	Appel de l'extramonitor
PEIN	\$EC09	Lecture des boutons du périphérique clavier
GEPE	\$EC06	Lecture du périphérique clavier
COMS	\$EC03	Appel d'un sous-programme en ROM
SETP	\$EC00	Programmation de la palette
CIPL	\$E833	Écriture d'un point "caractère"
KBIN	\$E830	Sortie du programme d'interruption
MENU	\$E82D	Retour au menu initial
OKCO	\$E82A	Contrôleur de disque
JOYS	\$E827	Lecture des manettes de jeux
GETS	\$E824	Lecture de l'écran
GETP	\$E821	Lecture de la couleur d'un point
NOTE	\$E81E	Génération de musique
LPIN	\$E81D	Lecture du bouton du crayon optique
GETL	\$E818	Lecture du crayon optique
K7CO	\$E815	Lecture-écriture sur la cassette
RSCO	\$E812	Gestion de l'interface de communication
PLOT	\$E80F	Allumage ou extinction d'un point
DRAW	\$E80C	Tracé d'un segment de droite
KTST	\$E809	Lecture rapide du clavier
GETC	\$E806	Lecture du clavier
PUTC	\$E803	Affichage d'un caractère

Registres du moniteur (page 0)

Détails des attributions de mémoire RAM entre les adresses \$6000 et \$601F:

Adresse	Nom	Traitement
*\$6000-\$6015	REDIR	<p>Les routines suivantes du moniteur font une indirection en RAM. Si vous voulez reprendre le contrôle lors d'un appel à l'une de ces routines, il suffit de modifier l'adresse de renvoi dans cette table:</p> <p>\$6000-\$6001: Indirection de GETLP \$6002-\$6003: Indirection de LPIN \$6004-\$6005: Indirection de GETP \$6006-\$6007: Indirection de GACH \$6008-\$6009: Indirection de PUTC \$600A-\$600B: Indirection de GETC \$600C-\$600D: Indirection de DRAW \$600E-\$600F: Indirection de PLOT \$6010-\$6011: Indirection de RSCONT \$6012-\$6013: Indirection de GETP \$6014-\$6015: Indirection de GETS</p>
*\$6016	PLAN	<p>Numéro du plan dans les modes superposition</p> <p>b2 : Numéro de plan en superposition b1-0 : Numéro de plan en triple superposition</p>
*\$6016-\$6017	SAVPAL	Sauvegarde de la palette 14 en mode 80 colonnes
*\$6019	STATUS	<p>Différents sémaphores:</p> <p>b7 : Semi-graphique b6 : Scroll rapide b5 : IRQ timer validée b4 : Graphique sans écriture de couleurs b3 : Forme seule b2 : Curseur visible ou invisible b1 : Transmission par GETC b0 : Traitement des séquences SS2 dans GETC</p>
*\$601A	TABPT	Pointeur dans la table des terminateurs de lignes
*\$601B	RANG	Ligne logique courante
*\$601C	TOPTAB	Pointeur sur le sommet logique de la table des terminateurs de lignes
*\$601D	TOPRAN	Première ligne logique de la fenêtre

*\$601E	BOITAB	Pointeur sur la fin logique de la table des terminateurs de lignes
*\$601F	BOTRAN	Dernière ligne logique de la fenêtre
*\$6020	COLN	Colonne logique courante
*\$6021-\$6022	IRQPT	Pointeur sur la routine moniteur de traitement des interruptions IRQ
*\$6023-\$6024	FIHQPT	Pointeur sur la routine de traitement des interruptions rapides FIHQ
*\$6025-\$6026	COPBUF	Copie de BUFFAT, réservé au système
*\$6027-\$6028	TIMEPT	Pointeur sur la routine utilisateur de traitement des interruptions TIMER
*\$6029	K7OPC	Code opération du L.E.P
*\$602A	K7STA	Code d'état du L.E.P
*\$602B	RSOPC	Mot de commande pour la gestion de la communication
*\$602C	RSSTA	Etat courant de la liaison communication
*\$602D-\$602E	USERAF	Pointeur sur le générateur de caractères utilisateur
*\$602F-\$6030	SWII	Pointeur sur SWI
*\$6031-\$6032	TEMPO	Tempo général pour la musique
*\$6033-\$6034	DUREE	Durée de la note
*\$6035	TIMBRE	Attaque de la note
*\$6036-\$6037	OCTAVE	Octave de la note
*\$6038	FORME	Code de la couleur
*\$6039	ATRANG	Sémaphore pour la gestion d'écran b7 : sémaphore de scroll b6 à b2 : réservé b1 : largeur simple ou double b0 : hauteur simple ou double
*\$603A	ATRSCR	Sémaphore de gestion plein écran b7 : sémaphore de fond plein écran b6 : sémaphore de forme plein écran b5 à b2 : réservé b1 : largeur simple ou double b0 : hauteur simple ou double
*\$603B	COLOUR	Couleur courante
*\$603C	TELETL	Si = \$F, alors mode page
*\$603D-\$603E	PI OTX	Abscisse du dernier point
*\$603F-\$6040	PLOTY	Ordonnée du dernier point
*\$6041	CIDRAW	Code ASCII du caractère servant à dessiner
*\$6042	CURSTL	Sémaphore de mouvement de curseur
*\$6043	COPCHR	Sémaphore de recopie caractère
*\$6044-\$6045	BAUDS	Paramètre de vitesse de la liaison série
*\$6046	NOMBRE	Définition des paramètres de la liaison série
*\$6047	GRCODE	Mise en mode graphique de l'imprimante
*\$6048	DKOPC	Commande du contrôleur de disque
*\$6049	DKDRV	Numéro du disque sélectionné
*\$604A-\$604B	DKTRK	Numéro de piste drive

*\$604C	DKSEC	Numéro de secteur drive
*\$604D	DKNUM	Entrelacement des secteurs au formatage
*\$604E	DKSTA	Etat du contrôleur de disquettes
*\$604F-\$6050	DKBUF	Pointeur de la zone tampon d'I/O disque
*\$6051-\$6052	TRACK0	Position de la tête du lecteur 0
*\$6053-\$6054	TRACK1	Position de la tête du lecteur 1
*\$6055-\$6056	TEMP1	Registre temporaire
*\$6057	TEMP2	Registre temporaire
*\$6058	ROTAT	Flag de rotation du moteur
*\$6059	SEQUCE	Code indiquant dans quelle séquence de gestion d'écran on se trouve
*\$605A-\$605B	SCRPT	Pointeur courant dans l'écran
*\$605C	SAVCOL	Sauvegarde de la couleur courante
*\$605D	ASCII	Code du dernier caractère affiché
*\$605E	READCLV	Pointeur de lecture du buffer clavier
*\$605F	SCRMOD	Flag indiquant le mode d'affichage
*\$6060-\$6061	STADR	Adresse du premier octet de la fenêtre
*\$6062-\$6063	ENDDR	Adresse+1 du dernier octet de la fenêtre
*\$6064	TCSRSAV	Sauvegarde de l'état courant du timer
*\$6065-\$6066	TCTSAV	Sauvegarde du compte courant du Timer
*\$6067	WRITECLV	Pointeur d'écriture dans buffer clavier
*\$6068-\$6069	SAVATR	Sauvegarde des attributs courants d'écran
*\$606A	US1	Sémaphore des séquences "unit separator"
*\$606B	COMPT	Compteur de caractères répétés
*\$606C-\$606D	TEMP	Registre temporaire pour le transfert de données
*\$606E-\$606F	SAVEST	Sauvegarde du pointeur de pile
*\$6070	ACCENT	Sémaphore de séquence d'accent
*\$6071	SS2GET	Minuscule accentuée
*\$6072	SS3GET	Idem
*\$6073	BUZZ	Sémaphore d'extinction du buzzer
*\$6074	CONFIG	Flag de présence de périphériques
*\$6075	EFMPT	Compteur d'effacement du curseur
*\$6076-\$6077	BLOCZ	2 octets à 0 pour les initis
*\$6078	SCROLS	Sémaphore de scroll doux
*\$6079-\$607A	BUFCLV	Adresse du buffer de réception clavier
*\$607B	SIZCLV	Longueur du buffer clavier
*\$607C	ACCES	Validation d'une info périphérique clavier
*\$607D	PERIPH	Echo des 3 LSB retournés par le clavier lors de l'envoi d'une commande
*\$607E	PERIPH1	Assure la chronologie des infos issues du clavier
*\$607F	RUNFLG	Sémaphore indiquant que l'option auto a été choisie
*\$6080	DKFLG	Sémaphore de présence du contrôleur disque
*\$6081-\$6085	IDSAUT	Buffer clavier par défaut
*\$6086	CURFLG	Page dans laquelle le curseur clignote en mode 80 colonnes
*\$6087	TEMP2	Registre temporaire

*\$6088-\$608A	RESETP	Adresse d'initialisation des nouveaux périphériques
*\$608B-\$60CC	STACK	Pile système
*\$60CD-\$60CE	PTCLAV	Pointeur sur la table de décodage du clavier
*\$60CF-\$60D0	PTGENE	Pointeur sur le générateur de caractères standards
*\$60D1	APPLIC	Checksum de l'application en cours
*\$60D2	DECALG	Ajustement du crayon optique
*\$60D3-\$60FD	LPBUFF	Zone tampon de I/O crayon ou souris
*\$60FE-\$60FF	TSTRST	Sémaphore de démarrage à chaud ou à froid