

```
LDB #$41 (code ASCII de « A » → B)
SWI #$02 (appel)
RTS      (retour)
```

■ Affichage d'un point à l'écran

Cette routine affiche un point de la couleur qu'on lui fournit sur l'écran. Les paramètres d'entrée sont les coordonnées du point : dans le registre X, on met l'abscisse (de 0 à 319) et dans Y l'ordonnée (de 0 à 199). On doit aussi mettre la variable système d'adresse \$2036 à 00 et celle d'adresse 2029 de la couleur dont on veut le point. Le code d'indexation est \$10.

L'affichage d'un point rouge en 96,96 se fait donc comme suit :

```
CLR $2036 (effacement de la variable en 2036)
LDA #$01 (code rouge)
STA $2029 (variable couleur du point)
LDX #$0060 (abscisse)
LDY #$0060 (ordonnée)
SWI #$10 (appel)
RTS      (retour)
```

Cette même routine peut aussi servir à afficher un caractère à une position donnée et non plus à la position courante du curseur comme auparavant. Pour ce a, on doit mettre le code ASCII du caractère à afficher, dans la variable située en \$2036. Attention cependant car cette fois-ci, X ne peut plus varier que de 1 à 40 (au lieu de 0 à 319 pour l'affichage d'un point). De même, Y doit être compris entre 0 et 24.

L'affichage d'un « A » à la position 16,16 se fait donc comme suit :

```
LDA #$41 (code ASCII du A)
STA $2036 (variable système)
LDX #$0010 (abscisse)
LDY #$0010 (ordonnée)
SWI #$10 (appel)
RTS      (retour)
```

■ Tracé d'une ligne de points

Cette routine permet le tracé d'une ligne entre la position du dernier point imprimé sur l'écran (par la présente routine ou par celle « Affichage d'un point » décrite précédemment) et un point que l'on fournit et ce de la couleur que l'on fournit. Les paramètres d'entrée sont : la position du point d'arrivée par son abscisse placée dans X (variant de 0 à 319) et par son ordonnée placée dans Y (variant de 0 à 199). On doit éventuellement placer le point de départ par la routine « Affichage d'un point » car sinon, le trait se trace depuis le dernier point imprimé. D'autre part, la variable située en \$2036 (indiquant s'il s'agit d'un caractère ou non) doit en toute logique contenir la valeur 0. La variable située en \$2029 doit contenir la couleur du trait.

Le code d'indexation de cette routine est 0E. Ainsi, si l'on veut tracer un trait du point (0,0) au point (96,96) en rouge, on fait comme suit :

```
CLR $2036 (effacement de la variable système)
IDA #801 (code du rouge)
STA $2029 (variable système couleur)
LDX #8000
LDY #8000 (point de départ)
SWI #810 (affichage d'un point)
(indexation affichage d'un point)
LDX #80060 (point d'arrivée)
LDX #80060
SWI #80F (tracé d'un trait)
RTS (retour)
```

Avec la même méthode que celle développée dans le sous-chapitre « Affichage d'un point », on peut, grâce à cette même routine, afficher une ligne de lettres. Pour cela, on doit placer le code ASCII de la lettre désirée dans la variable située en \$2036 et les positions de départ et d'arrivée doivent alors se situer dans l'intervalle [1,40] pour l'abscisse placée dans X et dans l'intervalle [0,24] pour l'ordonnée placée dans Y.

LE GÉNÉRATEUR DE CARACTÈRES

Le MOS possède une table dans laquelle se trouvent tous les caractères affichables à l'écran, c'est-à-dire de code ASCII compris entre 32 et 127. Cette table se trouve en ROM à partir de l'adresse \$FC9E. Vous pouvez la visualiser grâce à la fonction B du moniteur « QJIN » en tapant B C9E. Les caractères apparaissent à l'envers, ce qui signifie que le premier octet d'un caractère dans la table correspond au bas de la lettre correspondante. Ce jeu de caractères n'est pas transféré en RAM lors de l'initialisation du MOS comme c'est parfois le cas chez d'autres ordinateurs. Il n'est donc pas possible de modifier ces caractères.

Cependant, si cela est impossible, il existe une variable système qui définit le début du jeu de caractères : celle-ci se trouve à l'adresse \$2073-\$2074. Après l'initialisation, nous trouvons à ces deux adresses, l'adresse \$FC9F du jeu de caractères en ROM. Tapez par exemple POKF &H2073, &HC0, puis faites LIST si vous avez un programme en RAM. Vous voyez alors apparaître une suite de choses totalement incompréhensibles. Cependant malgré ces caractères plus que bizarres, vos ordres sont interprétés et exécutés correctement. Donc, si vous construisez votre propre jeu de caractères, vous pouvez vous en servir à condition de modifier l'adresse se trouvant en \$2073, \$2074.

À présent, rien n'est plus facile pour vous que de créer un jeu de caractères propre ou bien même de garder les mêmes caractères que ceux proposés mais en les imprimant à l'envers.

LES ROUTINES D'AFFICHAGE

Il peut être utile de posséder une routine permettant d'afficher une suite d'octets (pour afficher un message particulièrement). Ainsi, cette routine existe dans la ROM

Les paramètres d'entrée sont : l'adresse + 1 de la table qui doit se trouver dans le registre X. La table doit également se terminer par un 0. L'adresse de cette routine en ROM est \$CE2C. Ainsi, l'exécution de la routine :

```
LDX #$EA8A
JSR $CE2C
RTS
```

provoque l'affichage du message qui apparaît lors de l'initialisation du MOS, soit :

```
MOS BASIC 1.0
(C) Microsoft 1984
```

Cette routine tient compte des caractères de contrôle, ce qui vous permet donc de sauter des lignes, de changer les couleurs, etc.

Autre routine :

■ Affichage du contenu du registre D

Une routine en ROM s'en charge, inutile de vous fatiguer à la recréer. Cette routine se trouve en \$D83F. De plus, l'affichage se fait en décimal, ce qui est beaucoup mieux qu'en hexadécimal (pour compter les points d'un jeu par exemple). Si vous exécutez la routine suivante :

```
LDD #$A000
JSR $D83F
RTS
```

vous voyez apparaître à la position courante du curseur le nombre 40960 (ce qui est bien l'expression en décimal du nombre hexadécimal \$A000).

7

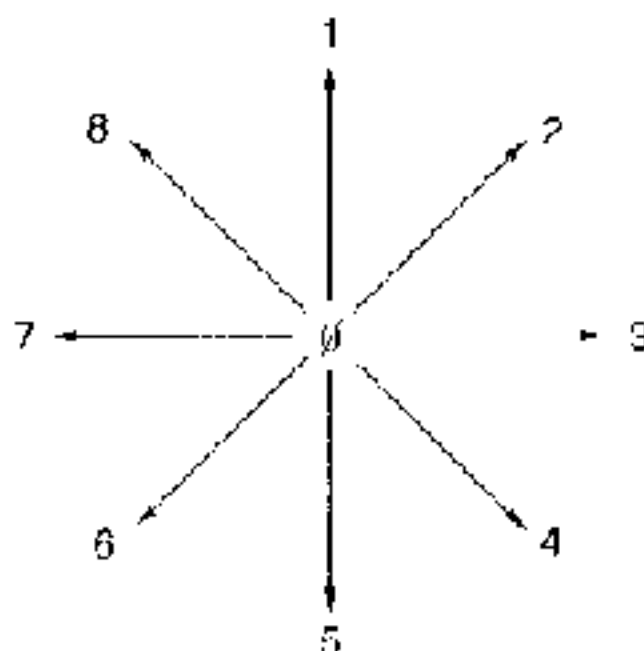
Le joystick

Sur le MO5, on peut brancher deux joysticks. Ces joysticks peuvent être scrutés par l'intermédiaire d'une routine du moniteur accessible par un SWI suivi d'un code d'indexation. Le paramètre d'entrée est le numéro du joystick à scruter qui doit se trouver dans l'accumulateur A. Ce numéro est 00 ou 01. Le code d'indexation de cette routine est S-C. Les paramètres de retour sont : la direction du manche qui se trouve dans l'accumulateur B et l'état du bouton de tir représenté par l'état de la Carry du registre CC.

Codes de retour :

Carry : 1 si le bouton de tir enfoncé
0 sinon

Accumulateur B : un nombre compris entre 0 et 8 selon la convention : 0 pour aucune direction, les autres chiffres étant sur le schéma ci-dessous :



Etat de l'accumulateur B en retour

On peut aussi scruter le joystick en testant directement le P.A. situé en \$A7CC et \$A7CD.

■ \$A7CC : *détermine l'orientation du joystick*

Le quartet de poids faible détermine la poignée droite et le quartet de poids fort, la poignée gauche.

\$A7CC

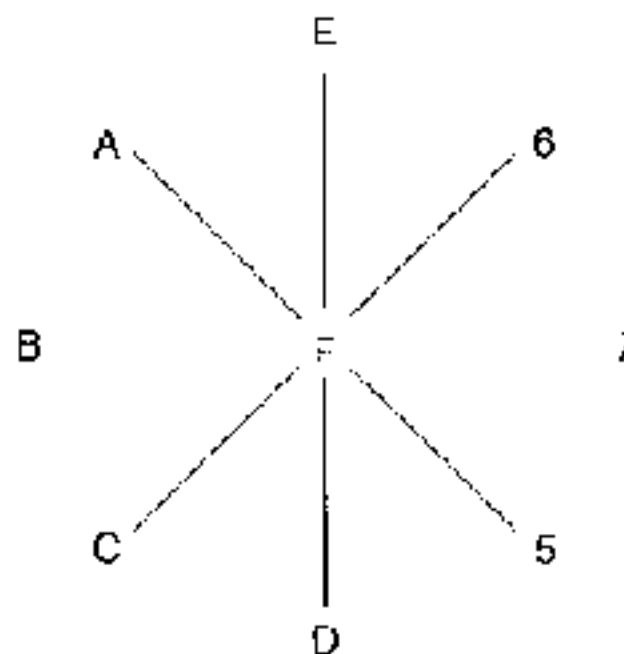
poignée gauche	poignée droite
----------------	----------------

Lorsque le joystick est au repos, le quartet est à \$F. Lorsque le joystick est orienté, les bits concernant la direction sont mis à 0.

bit 3	bit 2	bit 1	bit 0
est	ouest	sud	nord

Exemple : si le joystick est orienté au nord, le bit 0 passe à 0 le quartet aura pour valeur \$F.

Si le joystick est orienté au sud-ouest, les bits 1 et 2 passent à 0, le quartet aura pour valeur \$9.



Exemples : — poignée droite au nord, poignée gauche à l'est : \$7E.

— poignée droite au sud-est, poignée gauche au nord-est : \$65.

■ \$A7CD *détermine l'état du bouton de tir*

Le bit 7 correspond à la poignée droite, le bit 6 à la poignée gauche.

Lorsqu'aucun bouton n'est appuyé \$A7CD a pour valeur \$C0.

Si le bouton droit est appuyé, le bit 7 passe à 0. si le bouton gauche est appuyé, le bit 6 passe à 0.

Bouton appuyé : \$A7CD

Aucun : \$C0

Droit uniquement : \$80

Gauche uniquement : \$40

Droit et gauche : \$00

8

Le clavier

SCRUTATION CLAVIER RAPIDE

Cette routine moniteur est appelée par un SWI indexé par la valeur \$0C. Elle permet une exploration rapide du clavier.

- Le drapeau zéro est positionné si aucune touche n'est pressée, sinon il est nul.
- Le code matriciel de la touche est donné dans B (voir *tableau de correspondance*).
- A contient une valeur indiquant la pression sur BASIC, CNT ou SHIFT.

BASIC → A = 1

CNT → A = 2

SHIFT → A = 4

Exemple d'appel : APPEL SWI #\$0C
 BEQ APPEL
 ⋮
 RTS

Cette routine a l'avantage d'effectuer un scanning très rapide, mais cependant on n'obtient pas le code ASCII de la touche pressée. Voyons maintenant comment le faire.

Tableau de codage matriciel

<i>Poids fort</i> \ <i>Poids faible</i>	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0		STOP	ACC	ENT	RAZ	C	↑	W			A	*	Q	V		
1	←	X	2	—	Z	/	S	B	↓	SP	3	0	E	P	D	M
2	→	(w)	4	9	R	O	F	L	↵	.	5	8	T	I	G	K
3	INS	.	6	7	Y	U	H	J	EFF	N						

Exemples de codage : Z a pour code \$14

EFF a pour code \$38

SCRUTATION CLAVIER AVEC RETOUR DU CODE ASCII

Cette routine moniteur est appelée par l'intermédiaire d'un SWI indexé par \$0A.

— Le drapeau zéro est positionné si aucune touche n'est pressée, sinon il est nul.

— Le registre B contient le code ASCII de la touche pressée.

- Si la touche SHIFT est pressée en même temps qu'une autre touche on obtient le code shifté.
- Si la touche CNT est pressée, on obtient le code ASCII avec le bit 6 à zéro.
- Si la touche BASIC est pressée, on obtient un code spécial (voir tableau de correspondance).
- Si seules les touches BASIC, CNT et SHIFT sont pressées, B est à zéro.
- Dans le cas d'une touche accentuée, il faut faire trois appels : le premier détecte le code \$16 (accent), le second, le code de l'accent, le troisième le code de la lettre.

Les codes des accents sont :

Grave : \$41
 Aigu : \$42
 Circonflexe : \$43
 Tréma : \$48
 Cédille : \$4B

Tableau de codage de la touche BASIC

<div> <div>Poids fort</div> <div>Poids faible</div> </div>	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
B			STOP	ACC		ENT	RAZ	C	↑	W	1	+	A	*	Q	V
9	←	X	2	-	Z	/	S	B	↓	SP	3	0	E	P	D	M
A	→	@	4	9	R	O	F	L	↵	.	5	8	T	I	G	K
B	INS	,	6	7	Y	U	H	J	EFF	N						

Exemples de codage : BASIC Z a pour code \$94
 BASIC EFF a pour code \$B8

UTILISATION DE L'ENTREE LIGNE BASIC

Cette routine du BASIC permet d'entrer au clavier une ligne entière. On bénéficie alors de toutes les fonctions de l'éditeur du MOS : une page, insertion, suppression, déplacement du curseur avec les flèches, touches de contrôle...

L'appel doit être fait par la séquence suivante :

```
ENL JSR $CCCF
BCS ENL
```

Lors d'une validation par RETURN, la ligne sur laquelle pointe le curseur sera stockée dans un buffer situé en \$2452. La fin de la ligne est alors signalée par un zéro.

Pour lire ce buffer, il existe une routine nommée CHRGET située en \$21AC. Cette routine incrémente un pointeur de texte TXPTR, lit le caractère pointé en sautant les espaces blancs et signale la présence de chiffres (dans ce cas, la retenue est mise à 1, sinon elle est à 0). Le caractère lu revient dans l'accumulateur A.

Le pointeur TXPTR est situé en \$21B3 et doit être mis à jour après une entrée ligne, avec la valeur \$2451.

Diverses routines permettent de gérer l'entrée ligne :

\$21AC : incrémentation du pointeur de lecture caractère
\$21B2 : lecture du caractère sans incrémentation
\$21CA : décrémentation du pointeur sans lecture.

Exemple d'utilisation :

```

ENL   JSR $ECCF
      RGS FIN
      LDX #2451
      STX /$B3
LECT  JSR /$AC
      1STA
      BEQ FIN
      ; traitement caractère
      ; par caractère
      BRA LECT
FIN   RTS

```

MODIFICATION DU DELAI DE REPETITION

Une variable système détermine le délai entre l'appui sur une touche et la première répétition. Cette variable se trouve en \$2076. Plus la valeur stockée sera faible, plus le délai de répétition sera rapide.

MODIFICATION DES TABLES DU CLAVIER

Chaque pointeur de table est constitué de trois octets. Les deux premiers octets sont l'adresse de la table, le troisième est un indicateur. Vous devez mettre cet indicateur à 1 lorsque vous changez l'adresse, et à 0 lorsque vous remettez l'adresse d'origine.

■ *Table de décodage du clavier \$206D-206F*

En modifiant la table, vous pouvez changer la signification des touches du clavier (exemple : clavier QWERTY).

■ *Table du jeu de caractères \$2070-2072*

En modifiant cette table vous pouvez redéfinir le jeu de caractères entiers (exemple : caractères gothiques).

■ *Table du jeu de caractères supplémentaires \$2073-\$2075*

Cette table correspond aux caractères que vous pouvez définir par la commande BASIC GR\$. Le caractère défini par GR\$(0) a pour code ASCII \$80, GR\$(1) a pour code \$81...

9

Le magnétophone

STOCKAGE DU NOM DU PROGRAMME

Le nom du programme est situé dans un buffer ayant pour adresse \$23FA. Ce buffer a une longueur de douze octets :

- 1 octet pour la longueur du nom,
- 8 octets pour le nom,
- 3 octets pour l'extension (BAS, BIN, DAT, ASM. .)

23FA	23FB								2402		
long.	nom							extension			

Avant toute tentative de sauvegarde ou de chargement en langage machine, il faut mettre le buffer à jour. Pour cela, vous devez mettre la longueur du nom en \$23FA, le nom en \$23FB (s'il fait moins de huit caractères, vous devez compléter avec des espaces) et enfin l'extension en \$2402.

Une routine en ROM effectue ce travail, elle est implantée en \$E076. Faites EXECUTE \$E076 «ESSAI-BAS», vous obtiendrez dans la mémoire :

23FA	23FB								2402		
.	E	S	S	A	I				B	A	S
05	45	53	53	41	49	20	20	20	42	41	53

Cette routine lit directement le nom dans l'entrée ligne à condition que l'on ait positionné le pointeur TXPTR sur le guillemet de début (voir à ce sujet le chapitre « Clavier »).

\$E076 prend par défaut l'extension BAS si vous n'avez pas précisé celle-ci. Si vous désirez qu'une autre extension soit prise par défaut, il vous faut faire :

```
LDU #$ adresse de l'extension
JSR $E079
...
RTS
```

Trois extensions sont déjà codées dans la ROM :

```
BAS en $E460
DAT en $E463
DIN en $E466
```

Dans le cas d'un chargement, si vous désirez le premier programme, il vous faut mettre 0 en \$23FA, et huit espaces pour le nom.

SAUVEGARDE D'UN PROGRAMME

■ Fichier binaire

Voici tout d'abord une précision concernant la syntaxe de la commande SAVEM. Vous devez spécifier l'adresse de début, de fin et impérativement d'exécution (alors que d'après le manuel, cette adresse n'est qu'optionnelle...).

Pour effectuer cette commande vous devez empiler l'adresse de retour de votre sous-programme, l'adresse de début, de fin et d'exécution, et enfin faire un JMP \$E167.

Exemple : réaliser SAVEM « ESSAI.BIN », &H3000, &H3100, &H3000

Après avoir stocké le nom dans le BUFFER, faites

```
LDU #RETOUR
LDY #$3000 début
LDX #$3100 fin
LDD #$3000 exécution
PSHS U,Y,X,B,A
JMP $E167
RETOUR ...
RTS
```

Comme on peut le voir le fait de passer les trois adresses de sauvegarde par la pile oblige de simuler un JSR en stockant en pile l'adresse de retour et en faisant un JMP.

■ *Fichier Basic*

Pour effectuer une sauvegarde, il suffit de stocker le nom et ensuite de faire un JSR \$E12B.

□ *Fichier protégé*

Après avoir stocké le nom, vous devez faire un JSR \$E0B9.

□ *Fichier ASCII*

Après avoir stocké le nom, vous devez exécuter :

```
LDA #$FF
STA $2179
JSR $E088
```

CHARGEMENT D'UN PROGRAMME

■ *Variables système*

Elles se trouvent en page \$21 :

\$2199-\$219A : définit le type de chargement (RUN, MERGE, LOAD, LOADM, départ automatique...)

\$219B : type de chargement (Basic ou binaire)

\$219C-\$219D : adresse optionnelle de LOADM

■ *Fichier binaire*

L'appel est à faire en \$E2B3 après avoir mis à jour le buffer nom de programme et les variables système chargement.

```
mettre $2199 à 0
      $219A à 0
      $219B à 1
```

□ *Départ automatique*

```
mettre $2199 à 3.
```

Note : alors que le manuel BASIC ne précise rien, il est possible de lancer un programme binaire dès le chargement. Pour ce faire il suffit de taper : LOADM « nom du programme »..R.

■ *Fichier Basic*

L'appel est à faire toujours en \$E2B3 après avoir mis à jour le buffer nom de programme et les variables système chargement :

mettre \$2199 à 0

\$219A à 0

\$219B à 0

I *Départ automatique* ⇒ LOAD « ... », R

mettre \$2199 à 3

II *Départ automatique* ⇒ RUN « »

mettre \$2199 à 2

III *Merge*

mettre \$219A à FF

■ *Tableau récapitulatif*

Variables Commandes	\$2199	\$219A	\$219B
LOADM""	0	0	1
LOADM"" R	3	0	1
LOAD""	0	0	0
LOAD"" R	3	0	0
RUN""	2	0	0
MERGE ""	0	FF	0
MERGE "" R	3	FF	0

Appel commun en SE2B3

Note : d'après le codage effectué en ROM, on observe qu'il existe une commande LOADM « ... », &HXXXX. Celle-ci permet de charger un fichier binaire à une adresse relative à celle du chargement normal. Ainsi LOADM « », &H2000 charge un programme à l'adresse de début : 2000. L'exécution de cette commande en langage machine se fait en stockant l'adresse en \$219C 219D.

UTILISATION DES ROUTINES MONITEUR

■ Télécommande du magnétophone

Comme vous avez pu le remarquer, le magnétophone MO5 est conçu spécialement pour une utilisation informatique. De ce fait, la télécommande n'agit que sur les touches de lecture et d'enregistrement, de plus elle peut être appelée par le moniteur. L'appel doit être fait par SWI ayant pour code d'indexation \$22.

□ MOTOR ON

Le moteur peut être mis en route de deux façons :

- Soit avant une sauvegarde, de ce fait une temporisation d'une seconde doit être effectuée pour obtenir une stabilisation de la bande.

```
Faites : LDA #$03
        SWI #$22
        RTS
```

Soit avant un chargement, dans ce cas il n'y a pas de temporisation.

```
Faites : LDA #$01
        SWI #$22
        RTS
```

□ MOTOR OFF

Le moteur est arrêté après une sauvegarde ou un chargement après une temporisation d'une demi-seconde.

```
Faites : LDA #$02
        SWI #$22
        RTS
```

□ Test de présence du magnétophone

Si après un SWI #\$22, la retenue est mise à 1, c'est que le périphérique est absent. On peut le tester en faisant :

```
LDA #$22
BCS erreur
RTS
```

■ Sauvegarde et chargement d'un bloc

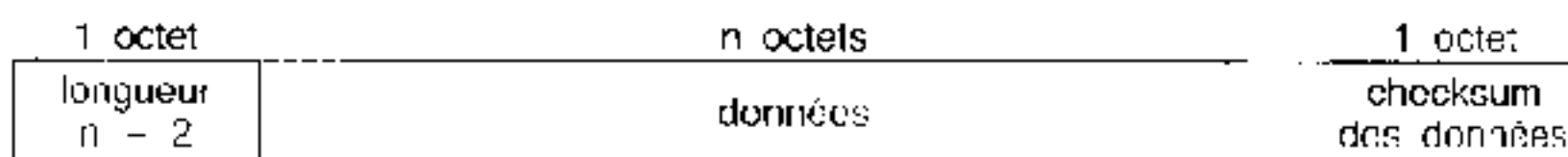
Cette routine permet de transférer un bloc-mémoire stocké dans un buffer vers le magnétophone, ou du magnétophone vers l'ordinateur. L'appel se fait par un SWI indexé par \$20.

Sauvegarde

Les registres d'appel sont A, B, Y.

A doit être mis à 0, il indique la sauvegarde.

Y doit contenir l'adresse du buffer dans lequel se trouvent les données. Ce buffer doit comporter un octet indiquant la longueur du bloc, le bloc par lui-même et enfin un octet de checksum (addition puis complément à deux). La longueur est égale à la longueur réelle du bloc plus deux.



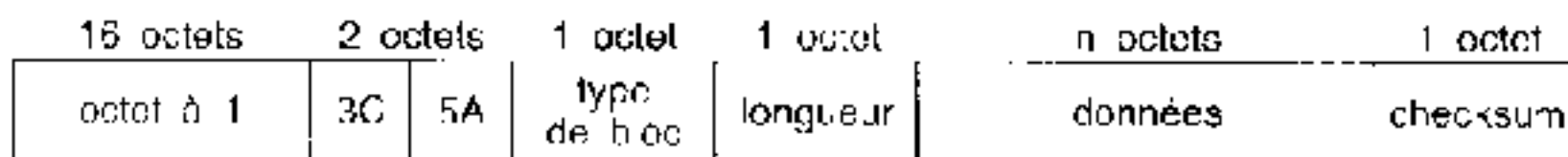
B doit contenir le type de bloc.

\$00 pour un bloc avec signal d'initialisation de fichier

\$01 pour un bloc de données

\$FF pour un bloc avec signal de fin de fichier

Quel que soit le type de bloc à chaque sauvegarde, on trouve sur la bande magnétique les octets suivants :



Le type de bloc ne sert qu'au chargement, pour savoir où l'on en est du programme chargé.

Exemple de sauvegarde : mise en route du moteur

```

.
.
CLRA
LDB # type de bloc
LDY # buffer
SWI #$20
.
.
.
arrêt du moteur

```


□ *Chargement*

Les registres d'appel sont Y et A :

Y doit contenir l'adresse du buffer dans lequel seront stockées les données chargées

A doit être différent de 0 → lecture.

Exemple : mise en route du moteur

```

.
.
.
LDA #$01
LDY # Buffer
SWI #$20
.
.
.
```

Les paramètres de retour sont A et B.

A indique le test de checksum, s'il est à 0, aucune erreur n'a été rencontrée (checksum calculé et checksum lu égaux) sinon il contient le checksum calculé.

B contient le type de bloc.

10

Le lecteur de disquettes

Il est identique à celui du T07 ou T07/70, seul le DOS (Disk Operating System \Rightarrow programme de gestion du disque) est spécifique au MO5.

Le DOS occupe deux emplacements mémoire : de \$2700 à \$4700 et de \$A000 à \$A700.

Il permet, en plus de la gestion du disque, l'accès à de nouvelles fonctions BASIC qui permettent de tirer un meilleur parti du MO5.

Les différences majeures entre le DOS MO5 et le DOS T07 sont l'absence sur MO5 de :

CVD, MKD\$ (calculs en double précision).

Par contre il existe de nouvelles fonctions :

HEX\$: permet la conversion décimal en hexadécimal.

Exemple : ?HEX\$(255)

SEARCH : recherche de chaînes de caractères dans un programme BASIC.

Exemple : SEARCH « RFM » Recherche les Rem du programme et les affiche à l'écran.

AUTO : permet la numérotation automatique lorsque l'on entre des lignes BASIC : on précise la première ligne et l'incrément.

Exemple : AUTO 100,10

DEFFN : permet de définir une fonction numérique.

Exemple : DEFFN A(N) = N + 3

DEFUSR : permet de définir l'adresse de départ d'un sous-programme en langage machine.

Exemple : DEFUSR = &HE8FC

FN : fournit le résultat calculé par une fonction prédéfinie.

Exemple : ?FNA(N)

USR : permet d'appeler un sous-programme en langage machine prédéfini.

Exemple : ?USR(0)

Pour toutes les autres fonctions, veuillez consulter le manuel du « BASIC DOS du IQ7 ».

ADRESSES DES FONCTIONS DOS MO5

Voici comme nous vous l'avions donné pour le BASIC, les adresses des fonctions du DOS MO5.

Code (hexadécimal)	Fonction	Adresse
D6	DSKINI	31FC
D7	DSKOS	44B3
D8	KII I	2A0A
D9	NAME	2A96
DA	FIELD	425B
DB	LSET	4297
DC	RSET	4290
DD	PUT	42DD
DE	GFT	42DE
DF	VERIFY	2A85
E0	DEVICE	2A6D
E1	DIR	2056
E2	FILES	3E7A
E3	WRITE	3F69
E4	UNLOAD	3F34
E5	BACKUP	3FBD
E6	COPY	4112
F7	CIRCLE	33C0
E8	PAINT	3462
E9	DRAW	3B2B
EA	RENUM	38D7
EB	SWAP	3719
EC	SEARCH	3675
FF A7	DSKF	2FF9
FF A8	CVI	4489
FF A9	CVS	448C
FF AA	Néant	Néant
FF AB	MKI\$	44A0

Code (hexadécimal)	Fonction	Adresse
FF AC	MKS\$	44A3
FF AD	Néant	Néant
FF AE	LOC	2F0F
FF AF	LOF	2F57
FF B0	SPACES	3703
FF B1	STRING\$	36E4
FF B2	DSKIS	4514

Note : Les fonctions HEX\$, AUTO, LSR, FN sont codées dans les fonctions BASIC.

SAUVEGARDE ET CHARGEMENT AU FORMAT BASIC

Les routines de sauvegarde et chargement sur disquettes sont identiques à celles sur cassettes, seule une variable système est à positionner pour sélectionner le périphérique : celle-ci est située en \$218C. Il faut mettre \$2 pour sélectionner le magnétophone et \$00 pour sélectionner le lecteur de disquettes. Veuillez consulter le chapitre « *Magnétophone* » pour les appels aux différentes routines.

FORMAT DISQUE MICROSOFT

Le MOS utilise un formatage dérivé du standard IBM 3740. Il travaille en simple face, simple ou double densité.

Il y a 40 pistes, composées elles-mêmes de 16 secteurs. Chaque secteur comporte 128 octets. On a donc 80 Koctets formatés. Certaines pistes sont réservées à la gestion du disque, les autres sont disponibles pour l'utilisateur.

■ Table d'allocation mémoire

Cette table permet au système de reconnaître la place libre sur le disque.

Le MOS divise les 80 Koctets de la disquette en 80 blocs de 1 Koctets. Ces blocs sont numérotés de 0 à 79. Chaque bloc comporte huit secteurs, il y a donc deux blocs par piste. Voici comment sont codés les blocs :

Bloc	Piste	Secteurs
0	0	1-8
1	0	9-16
2	1	1-8
3	1	9-16
4	2	1-8
5	2	9-16
78	39	1-8
79	39	9-16

La table d'allocation des fichiers (File Allocation Table) répertorie chaque bloc et indique son état. Elle se trouve en piste 20, secteur 2.

Chaque bloc est répertorié par un octet.

L'octet 0 correspond à zéro. L'octet 1 correspond au bloc 0, l'octet 2 au bloc 1, ... et l'octet 80 au bloc 79. L'état de chaque bloc est défini de la façon suivante :

\$FF : bloc libre

\$FE : bloc réservé au système ou inexistant physiquement

— valeur comprise entre [0,\$BF] : bloc occupé. La valeur correspond au numéro du prochain bloc du fichier.

— valeur comprise entre [\$C1-\$C8] : dernier bloc d'un fichier. Le quartet de poids faible indique le nombre de secteurs utilisés dans le dernier bloc.

■ Directory

Il contient le nom des fichiers du disque et se trouve piste 20, secteurs 3 à 16.

On peut coder en simple densité jusqu'à 56 noms, et 112 en double densité, chaque nom faisant 32 octets. Ces 32 octets sont répartis comme suit :

octets \$0 à \$7 : nom du fichier (8 lettres)

octets \$8 à \$A : extension du nom (3 lettres)

octet \$B : type de fichier, 0 programme BASIC
 1 données
 2 fichier binaire
 3 fichier source

octet \$D : drapeau ASCII, \$FF ASCII
 \$00 binaire

octets \$E-\$F : nombre d'octets utilisés dans le dernier secteur du fichier

octets \$10-\$1F : inutilisés

Le premier octet de chaque emplacement réservé au nom détermine l'état de cet emplacement.

\$00 : emplacement libre (dans le cas d'une suppression de fichier, l'emplacement est codé comme emplacement libre, mais sera prioritaire pour la création du prochain fichier).

[\$20-\$7E] : emplacement occupé (la valeur correspond à la première lettre du nom).

\$FF : fin logique du directory.

■ *Bootstrap*

Ce programme de chargement du DOS se situe piste 0, secteur 1. Il ne doit pas excéder 128 octets et doit être complété à deux. Le dernier octet doit être le checksum des 127 premiers octets non complétés + \$55.

UTILISATION DES ROUTINES MONITEUR

■ *Formatage d'une disquette*

L'appel se fait par un SWI indexé par \$2A. Les paramètres sont les variables système \$204F, \$2050, \$2049 et \$204D.

\$204F-\$2050 : doit contenir l'adresse d'un buffer de 256 octets nécessaire au formatage

\$2049 : doit contenir le numéro de l'unité de disquettes

\$204D : doit contenir le facteur d'entrelacement (normalement 4).

Les paramètres de retour sont le bit C de CC qui est à 1 s'il y a une erreur. Dans ce cas, l'octet \$2043 contient le code de l'erreur.

\$01 : disque protégé en écriture.

\$10 : le lecteur de disquettes n'est pas prêt (le moteur tourne ou le loquet est ouvert)

\$40 : contrôleur inopérant.

Exemple : SWI # \$2A
BCS erreur
RIS

■ *Bootstrap*

Il permet le chargement du DOS et du programme d'initialisation sauvé sous le nom «AUTO BAT». L'appel à cette routine se fait par un SWI indexé par \$28.

Les paramètres de retour sont \$2043 et \$2080 :

\$2043 : contient le code de l'erreur si le boot a été interrompu :

- \$00 erreur de checksum,
- pour les autres valeurs voir chapitre suivant.

\$2080 : correspond au drapeau de présence du lecteur :

- \$FF disque présent,
- \$00 disque absent.

Si le disque est absent, ou s'il y a une erreur de checksum, \$2080 est mis à zéro et un appel est fait à la routine de départ à froid (\$F000).

Exemple : SWI # \$28
RIS

■ *Routine RWTS*

Elle permet l'écriture ou la lecture d'un secteur d'une piste. L'appel se fait par SWI indexé par \$26.

Les paramètres d'appel sont les suivants :

\$2049 : doit contenir le numéro du drive [0,3].

\$204B : doit contenir le numéro de la piste [0,39].

\$204C : doit contenir le numéro du secteur [1,16].

\$204F-\$2050 : doit contenir l'adresse d'un buffer de 128 octets.

\$2048 : doit contenir le code opérateur définissant l'opération désirée.

- \$01 : initialisation du contrôleur
- \$02 : lecture d'un secteur
- \$08 : écriture d'un secteur
- \$20 : positionnement de la tête piste 0
- \$40 : positionnement de la tête piste n
- \$80 : vérification de sauvegarde

□ *Paramètres de retour*

Le bit C de CC est à 1 s'il y a une erreur. Dans ce cas, \$2043 contient le code de l'erreur :

\$01 : disque protégé en écriture

\$02 : erreur de piste (elle ne correspond pas à celle attendue)

\$04 : erreur de secteur (illicible)

\$08 : erreur de donnée (illisible)

\$10 : disque non prêt (le moteur tourne ou bien le loquet n'est pas fermé)

\$20 : erreur de vérification (il y a une différence entre le buffer et le secteur sauve)

\$40 : contrôleur inopérant

\$80 : disquette non formatée.

— Initialisation du contrôleur

Cette option permet de tester l'état du contrôleur et renseigne sur la densité du drive. Si le carry est nu, \$2043 contient le type du drive (\$43 : simple densité, \$44 double densité). Dans le cas contraire \$2043 contient \$40 c'est-à-dire contrôleur inopérant.

Exemple :

```
LDA #$01
STA $2048
SWI #$26
BCS erreur
RTS
```

— Positionnement de la tête en piste n

Cette opération permet de positionner la tête sur la piste précisée en \$204B. Si le bit C est à 1, \$2043 peut contenir les erreurs \$10 ou \$80.

Exemple :

```
LDA #$10
STA $204B  > positionnement piste 16
LDA #$10
STA $2048
SWI #$26
BCS erreur
RTS
```

— Positionnement de la tête en piste 0

Permet de positionner la tête en piste 0. Si le bit C est à 1, \$2043 contient les erreurs \$10 ou \$80.

Exemple :

```
LDA #$0
STA $2048
SWI #$26
BCS erreur
RTS
```


— *Lecture d'un secteur*

Lit le secteur spécifié en \$204C de la piste spécifiée en \$204B et le stocke à l'adresse précisée en \$204F-\$2050.

Exemple : LDA #02
 STA \$2048
 LDA #03
 STA \$204C — lecture piste 20 secteur 3
 LDA #20
 STA \$204B
 LDX #06000
 STX \$204F
 SWI #026
 BCS erreur
 RTS

— *Ecriture d'un secteur*

Ecrit le buffer précisé en \$204F-\$2050 à la piste précisée en \$204B, secteur précisé en \$204C.

Exemple : LDA #08
 STA \$2048
 LDA #03
 STA \$204C
 LDA #02 — écriture piste 2 secteur 3
 STA \$204B
 LDX #06000
 STX \$204F
 SWI #026
 BCS erreur
 RTS

Vérification d'une opération

Le code opératoire est égal à \$80 plus le code opératoire de l'opération que l'on veut vérifier. Ainsi pour vérifier une sauvegarde de secteurs, le code opératoire est \$88.

Exemple : LDA #088
 STA \$2048
 SWI #026
 BCS erreur
 RTS

11

L'interface de communication

L'interface de communication Thomson ne fonctionne sur MO5 qu'en mode parallèle, voici comment on peut l'utiliser avec une imprimante aux normes Centronics. L'appel à la routine de sortie parallèle se fait par un SWI indexé par \$24.

- *Les paramètres d'appels sont le registre B et les variables système \$2042 et \$2077*

B contient le code ASCII à envoyer à l'imprimante.

\$2042 représente le type d'opération effectuée :

Ecriture d'un octet : 1
 Copie graphique : 2
 Ouverture du port : 4
 Fermeture du port : 16

\$2077 contient dans le cas d'une copie graphique, l'octet à envoyer à l'imprimante pour la faire passer en mode graphique.

- *Les paramètres de retour sont le registre CC et la variable système \$2043*

La **retenue C** est positionnée à 1 s'il y a une erreur dans l'opération effectuée.

\$2043 contient l'état de l'imprimante après un appel à la routine (erreur ou type d'opération effectuée).

Ecriture	: 1	Périphérique occupé	: 8
Copie graphique	: 2	Fermeture	: 16
Ouverture	: 4		

■ *Ouverture du port*

```
LDA # $04
STA $2042
SWI # $24
BCS erreur
RTS
```

■ *Ecriture d'un octet*

```
LDA # $01
STA $2042
LDB #octet
SWI # $24
BCS erreur
RTS
```

■ *Fermeture du port*

```
LDA # $10
STA $2042
SWI # $24
RTS
```

■ *Copie graphique*

Fonctionne sur imprimante PR 90.040 et sur toutes les imprimantes compatibles.

Il faut mettre \$07 en \$2077 (à l'allumage la valeur est prise par défaut)

```
LDA # $02
STA $2042
SWI # $24
BCS erreur
RTS
```

12

Son et musique

BEEP CLAVIER

Plusieurs appels différents permettent d'effectuer ce beep sonore.

- Tout d'abord, on peut l'appeler par l'intermédiaire d'un SWI ayant pour code d'indexation \$08. Il vous suffit donc d'exécuter :

```
SWI # $08 3F 08
RTS      39
ou SWI # $88 3F 88
```

- Vous pouvez aussi faire un appel direct à la routine BASIC du beep. Pour ce faire il suffit d'exécuter :

```
JMP $ E8FA
```

- Vous pouvez enfin appeler le moniteur directement en \$FACB. A cette adresse se trouve une courte routine réalisant une boucle d'appel du PIA 6821.

C'est la durée de cette boucle qui fera varier la longueur du beep.

Examinons cette routine :

FR00-	4F	CLR	
FR01-	E7 B7 C1	STH	\$07C1
FR02-	80	INCR	
FR03-	5F	CLRB	
FR04-	50	INCB	
FR05-	2F F0	BPL	\$FR01
FR06-	81 11	CMPL	#11
FR07-	25 F4	BHE	\$FR00
FR08-	59	RTE	

Nous voyons que c'est le CMPA #11 qui détermine le nombre de boucles effectuées. Il suffit donc, si vous désirez modifier le beep, de transférer cette petite routine dans une zone mémoire libre et de modifier la valeur #11.

Voici un exemple de transfert avec le programme ODIN :

6000-	4F	CLR	
6001-	E7 B7 C1	STH	\$07C1
6002-	80	INCR	
6003-	5F	CLRB	
6004-	50	INCB	
6005-	2F F0	BPL	\$6006
6006-	81 59	CMPL	#59
6007-	25 F4	BHE	\$6001
6008-	59	RTE	

Vous obtenez ainsi un beep ayant une durée quatre fois supérieure à la durée normale.

SYNTHETISEUR MUSICAL

De la même façon qu'en BASIC, on peut faire jouer de la musique sur un MO5.

L'appel à la routine musique se fait par un SWI ayant pour valeur d'indexation \$1E.

La note à jouer doit être mise dans l'accumulateur B, tandis que les paramètres tempo, durée, timbre et octave doivent être stockés dans les variables systèmes réservées.

■ Valeur des notes	Note	Code	Note	Code
		hexadécimal		hexadécimal
	Pause	30	FA#	37
	DO	31	SOL	38
	DO#	32	SOL#	39
	RE	33	LA	3A
	RE+	34	LA#	3B
	MI	35	SI	3C
	FA	36	DO	3D

■ *Tempo \$203A*

Ce paramètre définit le tempo général de l'air que vous voulez jouer.

Ce tempo est compris entre 0 et \$FF : plus la valeur est faible, plus le tempo est rapide. Pour plus de précisions, consultez *Le manuel Basic*, page 84.

■ *Durée \$203C*

Ce paramètre définit la durée relative de la note. Cette durée est comprise entre 1 et \$60. Pour plus de précisions, consultez *Le manuel Basic*, page 83.

■ *Timbre \$203D*

Ce paramètre définit le timbre de la note. Ce timbre est compris entre 0 et \$FF, 0 faisant jouer un son continu et \$FF un son très rapidement amorti. Pour plus de précisions, consultez *Le manuel basic*, page 85.

■ *Octave \$203F*

Ce paramètre définit l'octave de la note. Cette octave est comprise entre 1 et \$10, 1 étant l'octave la plus aiguë, \$10 la plus grave.

■ *Valeur par défaut*

À l'allumage, chacun de ces paramètres possède une valeur propre :

Tempo : \$05

Durée : \$18

Timbre : \$00

Octave : \$02

Si vous ne desirez pas modifier ces paramètres vous n'êtes donc pas obligés de les mettre à jour avant l'exécution d'une note.

Exemple : Exécution d'un Do avec les paramètres d'origine :

```
LDB #$31
```

```
SWI #$9F
```

Execution d'un Do, octave 3, timbre \$10, Durée \$10, tempo \$10 :

```
8603 LDA #$03
```

```
B7203F STA $203F octave
```

```
8610 LDA #$10
```

```
B7203D STA $203D timbre
```

```
B7203C STA $203C durée
```

```
B7203A STA $203A tempo
```

```
C631 LDB #$31
```

```
3F9E SWI #$9F
```

13

Les variables systèmes

Ce court chapitre est un résumé de l'ensemble des variables systèmes que nous avons découvertes à ce jour.

Il vous permet également de retrouver la page qui vous donnera plus de détails sur une variable système donnée.

Adresse	Function	Page
2019	Variable de statut	
201B	Ligne du curseur	76
201C	Colonne du curseur	76
201E	Première ligne de la fenêtre	
2020	Dernière ligne de la fenêtre	
2029	Variable de couleur pour les graphiques	82
202B	Couleurs encre et fond	76
2030	Variable d'affichage	82
	code ASCII pour PLOT et DRAW	
2039 }	Tempo pour la musique	110
203A }		
203B }	Durée pour une note	110
203C }		
203D	Timbre	110
203E }	Octave	110
203F }		
2042	Commande de la ligne d'imprimante	106
2043	Statut de la ligne imprimante	106
2048	Commande pour le drive	103

Adresse	Fonction	Page
2049	Numéro du drive	103
204A } 204B }	Numéro de la piste	103
204C	Numéro du secteur	103
204D	Facteur d'entrelacement	102
204E	Statut du contrôleur	
204F } 2050 }	Pointeur du buffer entrée/sortie du drive	102
205E } 2060 }	Pointeur SWI	
2061 } 2063 }	Pointeur timer d'interruption + drapeau	
2064 } 2066 }	Pointeur IRQ + drapeau	
2067 } 2069 }	Pointeur FIRQ + drapeau	
206A } 206C }	Table des points d'entrée du moniteur + drapeau	
206D } 206F }	Table de décodage du clavier + drapeau	90
2070 } 2072 }	Pointeur du générateur de caractères utilisateur + drapeau	90
2073 } 2075 }	Pointeur du générateur de caractères système + drapeau	90
2076	Délai de répétition clavier	90
207F	Drapeau de simple ou double densité	
2080	Drapeau de présence du drive	103
2113 } 2114 }	Début BASIC	
2115 } 2116 }	Fin BASIC	
2117 } 2118 }	Fin variables	
2119 } 211A }	Fin tableau	

Annexe 1

Tableau de conversion

- Conversions : décimal, hexadecimal, binaire
- Codes ASCII
- Compléments à un, à deux

Valeur de l'octet

Complément à un

Complément à deux

Déc.	ASCII	Héxa	Binaire	Déc.	Héxa.	Binaire	Déc.	Héxa.	Binaire
0		0	00000000	255	FF	11111111	0	0	00000000
1		1	00000001	254	FE	11111110	255	FF	11111111
2		2	00000010	253	FD	11111101	254	FE	11111110
3		3	00000011	252	FC	11111100	253	FD	11111101
4		4	00000100	251	FB	11111011	252	FC	11111100
5		5	00000101	250	FA	11111010	251	FB	11111011
6		6	00000110	249	F9	11111001	250	FA	11111010
7		7	00000111	248	F8	11111000	249	F9	11111001
8		8	00001000	247	F7	11110111	248	F8	11111000
9		9	00001001	246	F6	11110110	247	F7	11110111
10		A	00001010	245	F5	11110101	246	F6	11110110
11		B	00001011	244	F4	11110100	245	F5	11110101
12		C	00001100	243	F3	11110011	244	F4	11110100
13		D	00001101	242	F2	11110010	243	F3	11110011
14		E	00001110	241	F1	11110001	242	F2	11110010
15		F	00001111	240	F0	11110000	241	F1	11110001
16		10	00010000	239	EF	11101111	240	F0	11110000
17		11	00010001	238	EE	11101110	239	EF	11101111
18		12	00010010	237	ED	11101101	238	EE	11101110
19		13	00010011	236	EC	11101100	237	ED	11101101
20		14	00010100	235	EB	11101011	236	EC	11101100
21		15	00010101	234	EA	11101010	235	EB	11101011
22		16	00010110	233	E9	11101001	234	EA	11101010
23		17	00010111	232	E8	11101000	233	E9	11101001
24		18	00011000	231	E7	11100111	232	E8	11101000
25		19	00011001	230	E6	11100110	231	E7	11100111
26		1A	00011010	229	E5	11100101	230	E6	11100110
27		1B	00011011	228	E4	11100100	229	E5	11100101
28		1C	00011100	227	E3	11100011	228	E4	11100100
29		1D	00011101	226	E2	11100010	227	E3	11100011
30		1E	00011110	225	E1	11100001	226	E2	11100010
31		1F	00011111	224	E0	11100000	225	E1	11100001
32		20	00100000	223	DF	11011111	224	E0	11100000
33		21	00100001	222	DE	11011110	223	DF	11011111
34	*	22	00100010	221	DD	11011101	222	DE	11011110

Valeur de l'octet				Complément à un			Complément à deux		
Déc.	ASCII	Héxa.	Binaire	Déc.	Héxa.	Binaire	Déc.	Héxa.	Binaire
35	5	23	00100011	220	DE	11011000	221	DF	11011001
36	6	24	00100100	219	DD	11011011	220	DC	11011100
37	7	25	00100101	218	DA	11011010	219	DB	11011011
38	8	26	00100110	217	DB	11011001	218	DB	11011010
39	9	27	00100111	216	DB	11011000	217	DB	11011001
40	.	28	00101000	215	DB	11010111	216	DB	11011000
41	,	29	00101001	214	DB	11010110	215	DB	11010111
42	*	2A	00101010	213	DB	11010101	214	DB	11010110
43	+	2B	00101011	212	DB	11010100	213	DB	11010101
44	-	2C	00101100	211	DB	11010011	212	DB	11010100
45	=	2D	00101101	210	DB	11010010	211	DB	11010011
46	_	2E	00101110	209	DB	11010001	210	DB	11010010
47	/	2F	00101111	208	DB	11010000	209	DB	11010001
48	0	30	00110000	207	DB	11001111	208	DB	11010000
49	1	31	00110001	206	DB	11001110	207	DB	11001111
50	2	32	00110010	205	DB	11001101	206	DB	11001110
51	3	33	00110011	204	DB	11001100	205	DB	11001101
52	4	34	00110100	203	DB	11001011	204	DB	11001100
53	5	35	00110101	202	DB	11001010	203	DB	11001011
54	6	36	00110110	201	DB	11001001	202	DB	11001010
55	7	37	00110111	200	DB	11001000	201	DB	11001001
56	8	38	00111000	199	DB	11000111	200	DB	11001000
57	9	39	00111001	198	DB	11000110	199	DB	11000111
58	:	3A	00111010	197	DB	11000101	198	DB	11000110
59	;	3B	00111011	196	DB	11000100	197	DB	11000101
60	<	3C	00111100	195	DB	11000011	196	DB	11000100
61	=	3D	00111101	194	DB	11000010	195	DB	11000011
62	>	3E	00111110	193	DB	11000001	194	DB	11000010
63	?	3F	00111111	192	DB	11000000	193	DB	11000001
64	@	40	01000000	191	DB	10111111	192	DB	11000000
65	A	41	01000001	190	DB	10111110	191	DB	10111111
66	B	42	01000010	189	DB	10111101	190	DB	10111110
67	C	43	01000011	188	DB	10111100	189	DB	10111101
68	D	44	01000100	187	DB	10111011	188	DB	10111100
69	E	45	01000101	186	DB	10111010	187	DB	10111011
70	F	46	01000110	185	DB	10111001	186	DB	10111010
71	G	47	01000111	184	DB	10111000	185	DB	10111001
72	H	48	01001000	183	DB	10110111	184	DB	10111000
73	I	49	01001001	182	DB	10110110	183	DB	10110111
74	J	4A	01001010	181	DB	10110101	182	DB	10110110
75	K	4B	01001011	180	DB	10110100	181	DB	10110101
76	L	4C	01001100	179	DB	10110011	180	DB	10110100
77	M	4D	01001101	178	DB	10110010	179	DB	10110011
78	N	4E	01001110	177	DB	10110001	178	DB	10110010
79	O	4F	01001111	176	DB	10110000	177	DB	10110001
80	P	50	01010000	175	DB	10101111	176	DB	10110000
81	Q	51	01010001	174	DB	10101110	175	DB	10101111
82	R	52	01010010	173	DB	10101101	174	DB	10101110
83	S	53	01010011	172	DB	10101100	173	DB	10101101
84	T	54	01010100	171	DB	10101011	172	DB	10101100
85	U	55	01010101	170	DB	10101010	171	DB	10101011
86	V	56	01010110	169	DB	10101001	170	DB	10101010
87	W	57	01010111	168	DB	10101000	169	DB	10101001
88	X	58	01011000	167	DB	10100111	168	DB	10101000
89	Y	59	01011001	166	DB	10100110	167	DB	10100111
90	Z	5A	01011010	165	DB	10100101	166	DB	10100110
91	[5B	01011011	164	DB	10100100	165	DB	10100101
92	\	5C	01011100	163	DB	10100011	164	DB	10100100

Valeur de l'octet				Complément à un			Complément à deux		
Dec.	ASC.	Hexa.	Binaire	Dec.	Hexa.	Binaire	Dec.	Hexa.	Binaire
93	j	5D	01011101	162	A2	10100010	163	A3	10100011
94	k	5E	01011110	161	A1	10100001	162	A2	10100010
95	l	5F	01011111	160	A0	10100000	161	A1	10100001
96	m	60	01100000	159	9F	10011111	160	A0	10100000
97	n	61	01100001	158	9E	10011110	159	9F	10011111
98	o	62	01100010	157	9D	10011101	158	9E	10011110
99	p	63	01100011	156	9C	10011100	157	9D	10011101
100	q	64	01100100	155	9B	10011011	156	9C	10011100
101	r	65	01100101	154	9A	10011010	155	9B	10011011
102	s	66	01100110	153	99	10011001	154	9A	10011010
103	t	67	01100111	152	98	10011000	153	99	10011001
104	u	68	01101000	151	97	10010111	152	98	10011000
105	v	69	01101001	150	96	10010110	151	97	10010111
106	w	6A	01101010	149	95	10010101	150	96	10010110
107	x	6B	01101011	148	94	10010100	149	95	10010101
108	y	6C	01101100	147	93	10010011	148	94	10010100
109	z	6D	01101101	146	92	10010010	147	93	10010011
110	[6E	01101110	145	91	10010001	146	92	10010010
111	\	6F	01101111	144	90	10010000	145	91	10010001
112]	70	01110000	143	8F	10001111	144	90	10010000
113	^	71	01110001	142	8E	10001110	143	8F	10001111
114	_	72	01110010	141	8D	10001101	142	8E	10001110
115	`	73	01110011	140	8C	10001100	141	8D	10001101
116	a	74	01110100	139	8B	10001011	140	8C	10001100
117	b	75	01110101	138	8A	10001010	139	8B	10001011
118	c	76	01110110	137	89	10001001	138	8A	10001010
119	d	77	01110111	136	88	10001000	137	89	10001001
120	e	78	01111000	135	87	10000111	136	88	10001000
121	f	79	01111001	134	86	10000110	135	87	10000111
122	g	7A	01111010	133	85	10000101	134	86	10000110
123	h	7B	01111011	132	84	10000100	133	85	10000101
124	i	7C	01111100	131	83	10000011	132	84	10000100
125	j	7D	01111101	130	82	10000010	131	83	10000011
126	k	7E	01111110	129	81	10000001	130	82	10000010
127	l	7F	01111111	128	80	10000000	129	81	10000001
128		80	10000000	127	7F	01111111	128	80	10000000
129		81	10000001	126	7E	01111110	127	7F	01111111
130		82	10000010	125	7D	01111101	126	7E	01111110
131		83	10000011	124	7C	01111100	125	7D	01111101
132		84	10000100	123	7B	01111011	124	7C	01111100
133		85	10000101	122	7A	01111010	123	7B	01111011
134		86	10000110	121	79	01111001	122	7A	01111010
135		87	10000111	120	78	01111000	121	79	01111001
136		88	10001000	119	77	01110111	120	78	01111000
137		89	10001001	118	76	01110110	119	77	01110111
138		8A	10001010	117	75	01110101	118	76	01110110
139		8B	10001011	116	74	01110100	117	75	01110101
140		8C	10001100	115	73	01110011	116	74	01110100
141		8D	10001101	114	72	01110010	115	73	01110011
142		8E	10001110	113	71	01110001	114	72	01110010
143		8F	10001111	112	70	01110000	113	71	01110001
144		90	10010000	111	6F	01101111	112	70	01110000
145		91	10010001	110	6E	01101110	111	6F	01101111
146		92	10010010	109	6D	01101101	110	6E	01101110
147		93	10010011	108	6C	01101100	109	6D	01101101
148		94	10010100	107	6B	01101011	108	6C	01101100
149		95	10010101	106	6A	01101010	107	6B	01101011
150		96	10010110	105	69	01101001	106	6A	01101010

Valeur de l'octet				Complément à un			Complément à deux		
Déc.	ASC	Héxa.	Binaire	Déc.	Héxa.	Binaire	Déc.	Héxa.	Binaire
151		97	10010111	104	68	01101000	105	69	01101001
152		98	10011000	103	67	01100111	104	60	01101000
153		99	10011001	102	66	01100110	103	67	01100111
154		9A	10011010	101	65	01100101	102	66	01100110
155		9B	10011011	100	64	01100100	101	65	01100101
156		9C	10011100	99	63	01100011	100	64	01100100
157		9D	10011101	98	62	01100010	99	63	01100011
158		9E	10011110	97	61	01100001	98	62	01100010
159		9F	10011111	96	60	01100000	97	61	01100001
160		AC	10100000	95	5F	01011111	96	60	01100000
161		AD	10100001	94	5E	01011110	95	5F	01011111
162		AE	10100010	93	5D	01011101	94	5E	01011110
163		AF	10100011	92	5C	01011100	93	5D	01011101
164		A0	10100100	91	5B	01011011	92	5C	01011100
165		A1	10100101	90	5A	01011010	91	5B	01011011
166		A2	10100110	89	59	01011001	90	5A	01011010
167		A3	10100111	88	58	01011000	89	59	01011001
168		A4	10101000	87	57	01010111	88	58	01011000
169		A5	10101001	86	56	01010110	87	57	01010111
170		A6	10101010	85	55	01010101	86	56	01010110
171		A7	10101011	84	54	01010100	85	55	01010101
172		A8	10101100	83	53	01010011	84	54	01010100
173		A9	10101101	82	52	01010010	83	53	01010011
174		AA	10101110	81	51	01010001	82	52	01010010
175		AB	10101111	80	50	01010000	81	51	01010001
176		AC	10110000	79	4F	01001111	80	50	01010000
177		AD	10110001	78	4E	01001110	79	4F	01001111
178		AE	10110010	77	4D	01001101	78	4E	01001110
179		AF	10110011	76	4C	01001100	77	4D	01001101
180		B0	10110100	75	4B	01001011	76	4C	01001100
181		B1	10110101	74	4A	01001010	75	4B	01001011
182		B2	10110110	73	49	01001001	74	4A	01001010
183		B3	10110111	72	48	01001000	73	49	01001001
184		B4	10111000	71	47	01000111	72	48	01001000
185		B5	10111001	70	46	01000110	71	47	01000111
186		B6	10111010	69	45	01000101	70	46	01000110
187		B7	10111011	68	44	01000100	69	45	01000101
188		B8	10111100	67	43	01000011	68	44	01000100
189		B9	10111101	66	42	01000010	67	43	01000011
190		BA	10111110	65	41	01000001	66	42	01000010
191		BB	10111111	64	40	01000000	65	41	01000001
192		BC	11000000	63	3F	00111111	64	40	01000000
193		BD	11000001	62	3E	00111110	63	3F	00111111
194		BE	11000010	61	3D	00111101	62	3E	00111110
195		BF	11000011	60	3C	00111100	61	3D	00111101
196		C0	11000100	59	3B	00111011	60	3C	00111100
197		C1	11000101	58	3A	00111010	59	3B	00111011
198		C2	11000110	57	39	00111001	58	3A	00111010
199		C3	11000111	56	38	00111000	57	39	00111001
200		C4	11001000	55	37	00110111	56	38	00111000
201		C5	11001001	54	36	00110110	55	37	00110111
202		C6	11001010	53	35	00110101	54	36	00110110
203		C7	11001011	52	34	00110100	53	35	00110101
204		C8	11001100	51	33	00110011	52	34	00110100
205		C9	11001101	50	32	00110010	51	33	00110011
206		CA	11001110	49	31	00110001	50	32	00110010
207		CB	11001111	48	30	00110000	49	31	00110001
208		CC	11010000	47	2F	00101111	48	30	00110000

Valeur de l'octet				Complément à un			Complément à deux		
Déc.	Asc.	Héxa.	Binaire	Déc.	Héxa.	Binaire	Déc.	Héxa.	Binaire
209		D1	11010001	46	2E	00101110	47	2F	00101111
210		D2	11010010	45	2D	00101101	46	2E	00101110
211		D3	11010011	44	2C	00101100	45	2D	00101101
212		D4	11010100	43	2B	00101011	44	2C	00101100
213		D5	11010101	42	2A	00101010	43	2B	00101011
214		D6	11010110	41	29	00101001	42	2A	00101010
215		D7	11010111	40	28	00101000	41	29	00101001
216		D8	11011000	39	27	00100111	40	28	00101000
217		D9	11011001	38	26	00100110	39	27	00100111
218		DA	11011010	37	25	00100101	38	26	00100110
219		DB	11011011	36	24	00100100	37	25	00100101
220		DC	11011100	35	23	00100011	36	24	00100100
221		DD	11011101	34	22	00100010	35	23	00100011
222		DE	11011110	33	21	00100001	34	22	00100010
223		DF	11011111	32	20	00100000	33	21	00100001
224		E0	11100000	31	1F	00011111	32	20	00100000
225		E1	11100001	30	1E	00011110	31	1F	00011111
226		E2	11100010	29	1D	00011101	30	1E	00011110
227		E3	11100011	28	1C	00011100	29	1D	00011101
228		E4	11100100	27	1B	00011011	28	1C	00011100
229		E5	11100101	26	1A	00011010	27	1B	00011011
230		E6	11100110	25	19	00011001	26	1A	00011010
231		E7	11100111	24	18	00011000	25	19	00011001
232		E8	11101000	23	17	00010111	24	18	00011000
233		E9	11101001	22	16	00010110	23	17	00010111
234		EA	11101010	21	15	00010101	22	16	00010110
235		EB	11101011	20	14	00010100	21	15	00010101
236		EC	11101100	19	13	00010011	20	14	00010100
237		ED	11101101	18	12	00010010	19	13	00010011
238		EE	11101110	17	11	00010001	18	12	00010010
239		EF	11101111	16	10	00010000	17	11	00010001
240		F0	11110000	15	0F	00001111	16	10	00010000
241		F1	11110001	14	0E	00001110	15	0F	00001111
242		F2	11110010	13	0D	00001101	14	0E	00001110
243		F3	11110011	12	0C	00001100	13	0D	00001101
244		F4	11110100	11	0B	00001011	12	0C	00001100
245		F5	11110101	10	0A	00001010	11	0B	00001011
246		F6	11110110	9	09	00001001	10	0A	00001010
247		F7	11110111	8	08	00001000	9	09	00001001
248		F8	11111000	7	07	00000111	8	08	00001000
249		F9	11111001	6	06	00000110	7	07	00000111
250		FA	11111010	5	05	00000101	6	06	00000110
251		FB	11111011	4	04	00000100	5	05	00000101
252		FC	11111100	3	03	00000011	4	04	00000100
253		FD	11111101	2	02	00000010	3	03	00000011
254		FE	11111110	1	01	00000001	2	02	00000010
255		FF	11111111	0	00	00000000	1	01	00000001

Annexe 2

Code des couleurs

0 : Noir	8 : Gris
1 : Rouge	9 : Rouge clair
2 : Vert	10 : Vert clair
3 : Jaune	11 : Jaune clair
4 : Bleu	12 : Bleu clair
5 : Magenta	13 : Magenta clair
6 : Cyan	14 : Cyan clair
7 : Blanc	15 : Orange

Annexe 3

Caractères de contrôle

Code	Signification (touche)	CNT
00	NUL	
01	SOH	A
02	SIX..(STOP)	B
03	ETX	C
04	EOT	D
05	ENO	E
06	ACK	F
07	BE_	G
08	BS..(←)	H
09	HT..(→)	I
10	LF..(↵)	J
11	VT..(^)	K
12	FF (Raz)	L
13	CH..(Entrée)	M
14	SO	N
15	SI	O
16	DLE	P
17	DC1	Q
18	DC2	R
19	DC3	S
20	DC4	T

Code	Signification (touche)	CNT
21	NAK	U
22	SYN (ACC)	V
23	ETB	W
24	CAN	X
25	EM	Y
26	SUR	Z
27	ESC	
28	FS (INS)	
29	GS (EFF)	
30	RS	
31	US	

Annexe 4

Particularités du T07/70

Le T07/70 est semblable au MO5 sur beaucoup de points : il possède le même processeur (6809) et seules son organisation mémoire et ses variables systèmes changent d'adresse par rapport au MO5.

Ce chapitre vous sert (si vous possédez un T07/70) à adapter tout ce que nous avons écrit pour le MO5 sur votre ordinateur.

La première chose qui change est la MAP que nous donnons en page 31 et que nous redonnons ici pour votre T07/70.

CARTE-MÉMOIRE — T07/70

Adresses (en hexadécimal)	Occupées par
0000 → 3FFF	Place réservée aux cartouches de ROM (16K)
4000 → 5F3F	Ecran (8K)
5F3F → 5FFF	Libre
6000 → 60FF	Page 0 du moniteur
6100 → 61FF	Page 0 du BASIC
6200 → DFFF	Mémoire disponible pour l'utilisateur (programmes + variables) = 31,5K

Adresses (en hexadécimal)	Occupées par
F000 → F7BF	Réservé pour le floppy
F7C0 → E7C7	
E7C8 → E7CB	Système 6821 (PIA)
E7CC → E7CF	6821 : extension jeu
E7D0 → E7DF	Réservé pour le contrôleur de floppy
E7E0 → E7E3	6821 : interface parallèle
E7E4 → E7FF	Libre pour extensions
E800 → FFFF	Moniteur

Tout ce qui concerne le langage machine sur le MO5 reste valable pour le T0770 puisque ces deux machines possèdent le même processeur, le 6809. La première partie de ce livre, est donc directement applicable sur un T0770 (excepté pour la MAP que nous venons de voir plus haut).

En revanche, la suite du livre doit être quelque peu modifiée pour l'adaptation sur un T0770.

Les attributs dont nous parlons en page 77 existent sur le T0770, mais leurs codes diffèrent. Nous vous donnons ici le code qu'il faut taper après le CHR\$(27). L'obtention d'un résultat est due à ?CHR\$(27)+CHR\$(XX) avec XX donné dans le tableau suivant :

Quarter fort Quarter faible	FORME	FOND	TOUR	FORME	TOUR
	4	5	6	7	8
0	noir	noir	noir	gris	gris
1	rouge	rouge	rouge	rose	rose
2	vert	vert	vert	vert clair	vert clair
3	aune	jaune	jaune	jaune clair	jaune clair
4	bleu	bleu	bleu	bleu clair	bleu clair
5	magenta	magenta	magenta	magenta clair	magenta clair
6	cyan	cyan	cyan	cyan clair	cyan clair
7	blanc	blanc	blanc	orange	orange
8		masquage	caractères sans couleur	gris	
9			caractères avec couleur	rose	
A			scroll rapide	vert clair	
B			mode page	jaune clair	
C	taille normale	inversion video		bleu clair	
C	double hauteur			magenta clair	
E	double largeur		scroll lent	cyan clair	
F	double taille	demas- quage		orange	

Par exemple, si vous voulez obtenir un scroll lent avec les points sans les couleurs, vous tapez :

```
?CHR$(27); CHR$(&H8E); CHR$(27); CHR$(&H68)
```

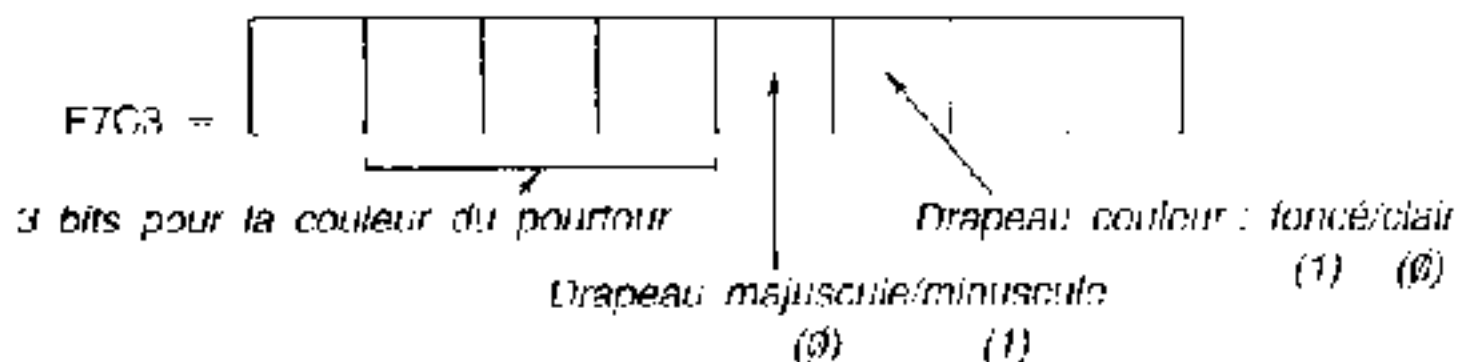
D'autre part, toutes les variables systèmes que nous donnons dans la deuxième partie de ce livre existent sur le T07/70, mais diffèrent par leurs adresses. Ceci est aisément compréhensible puisque la page 0 moniteur se situe en 20000 pour le MO5 et en 60000 pour le T07/70. Parfois même, le système de codage n'est pas le même dans le MO5 et le T07/70. Dans ce cas, nous indiquons la manière dont la variable est codée dans le T07/70. Dans le cas contraire, il vous suffira de vous reporter à la variable correspondante pour le MO5 pour obtenir son principe de codage.

■ Variables systèmes

MO5 TO7/70 Signification

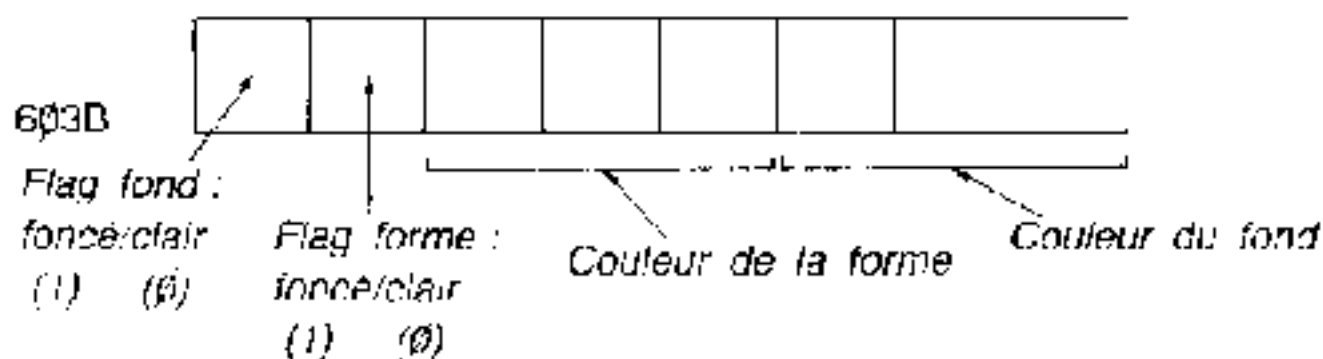
A7C0 (74) E7C3 Codage du pourtour

Le codage diffère sensiblement entre MO5 et TO7/70. Pour le TO7/70, il en est ainsi :



202B (75) 603B Codage couleur forme et fond

Là aussi, le codage diffère :



Ainsi si on veut obtenir de l'encre vert clair sur un fond rouge, il faut opérer ainsi :

Forme : vert	→ 2 = 010	} d'où '001000' = 145
clair	→ 0 = 0	
Fond : rouge	→ 1 = 001	
foncé	→ 1 = 1	

201B (76) 601B Ligne du curseur
(codage identique)201C (76) 6020 Colonne du curseur
(codage identique)

2020 (81-82) 603B Couleur d'un point à afficher

2036 (81-82) 6041 Code ASCII de la lettre à afficher

201E 601D Première ligne de la fenêtre

2020	601F	Dernière ligne de la fenêtre
2019	6019	Registre de status
2039 (110) 203A	6031 6032	Tempo (musique)
203B (110) 203C	6033 6034	Durée (musique)
203E (110) 203F	6036 6037	Octave (musique)
203D (110)	6035	Timbre (musique)
2042	602B	Ligne imprimante
2043	602C	Status ligne imprimante
205E 2060	602F 6030	Pointeur SWI
2061 2062	6027 6028	Pointeur Timer
2064 2065	6021 6022	Pointeur IRQ
2067 2068	6023 6024	Pointeur FIRQ
2070 2071	602D 602E	Pointeur du générateur de caractères utilisateur

■ Fonctions d'affichage

Affichage d'un caractère : comme pour le MO5 sauf que l'adresse de la routine est E803 (<=> SWI 02).

Affichage d'un point graphique : identique au MO5. Adresse de la routine : E80F (<=> SWI #510).

Affichage d'un caractère : identique au MO5. Adresse de la routine : E833 (<=> SWI #510).

Tracé d'un trait : identique au MO5. Adresse de la routine : E80C (<=> SWI #50F).

Repositionnement de fenêtre plein écran : E800.

■ Le clavier

Deux routines comme sur le MO5 : une scrutant rapide et une retournant le code ASCII de la touche pressée :

Scrutation rapide : adresse de la routine : F809. Si la touche est pressée C = 1 (carry) sinon C = 0.

Scrutation lente : adresse de la routine : E806. Le code ASCII de la touche pressée est retourné dans l'accumulateur B. Si aucune touche n'est pressée, alors B = 0. Une routine de scrutation peut donc être :

```
APPEL JSR #E806
      TSTB
      BEQ APPEL
      RTS
```

■ Le crayon optique

Deux routines concernent ce périphérique : l'une sert à tester si le contact de validation est ou non ouvert, l'autre retourne les coordonnées du point visé.

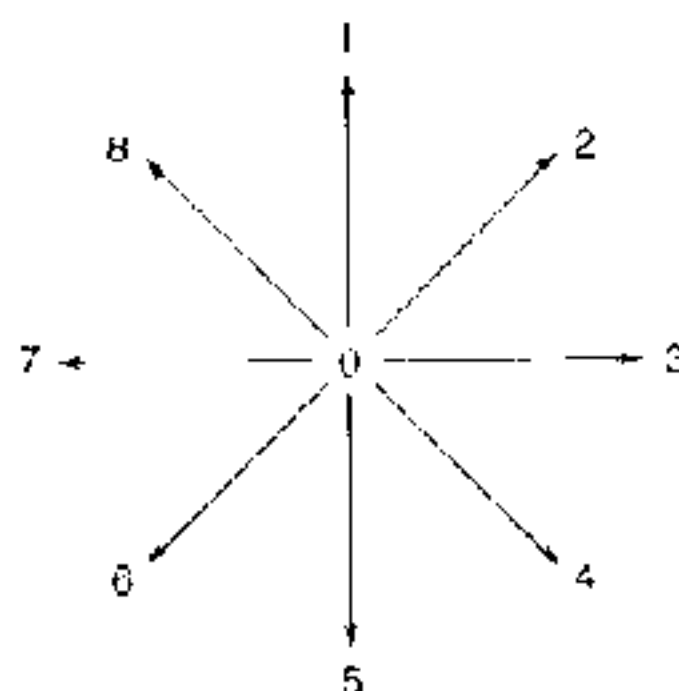
Etat du microcontacteur : adresse de la routine : F81B. Si l'on appuie sur le microcontacteur, alors C = 1 (carry), sinon C = 0.

Coordonnées du point visé : adresse de la routine : E818. Au retour : si C = 1, la lecture n'est pas valable, si C = 0, la lecture est valable et on a alors l'abscisse dans X (de 0 à 319) et l'ordonnée dans Y (de 0 à 199).

■ Le joystick

Adresse de la scrutation joystick : E827. Paramètre d'entrée : le numéro du joystick dans A (0 ou 1). Au retour : valeur du manche dans B et état du bouton dans C (carry) :

C = 0 \Rightarrow bouton lâché.
C = 1 \Rightarrow bouton pressé.



Achevé d'imprimer en janvier 1985
sur les presses de l'imprimerie Laballery et C^e
58500 Clamecy
Dépôt légal : janvier 1985

N^o d'impression : 501017
N^o d'édition : 86595-206-1
ISBN : 2-86595-206-1