CLEFS POUR MO5

Gilles Blanchard

Editions du P.S.I.

SOMMAIRE

	Page
PRESENTATION	5
BASIC STANDARD	11
Fonctions numériques	12
Fonctions alphanumériques	13
Fonctions de transfert numérique-alphanumérique	14
Fonctions de transfert alphanumérique-numérique	15
Instructions d'affectation	16
Instructions de branchement	19
Instructions de condition	21
Instructions graphiques	22
Instructions d'affichage et de sortie	24
Instructions d'édition	27
Commandes du clavier	29
Entrées-sorties standard	31
Stockage des variables en mémoire	32
Codes des instructions	33
BASIC D.O.S.	35
Nouvelles fonctions du D.O.S.	36
Nouvelles instructions du D.O.S.	38
Nouvelles entrées-sorties avec le D.O.S.	44

UTILITAIRES 47

Choix dans un MENU 48
Boucle d'attente/modif aléas 49
Saisie d'écran restreinte 50
Mouvement par manette de jeu 51
Conversions T07-M05 52
Nombre de deux chiffres sur un caractère 54

MESSAGES D'ERREUR 55

... et conseils pour les faire disparaître 55

JEU D'INSTRUCTIONS DU 6809 59

Registres du 6809 60
Instructions 61
Modes d'adressage 62
Jeu d'instructions du 6809 65

ADRESSES 77

Organisation générale de la mémoire 77
Le MONITEUR 78
Principales routines du MONITEUR 80
Registres du MONITEUR 90
Description de certains registres 93
PIA système 94
PIA communication 96
PIA manettes de jeu 97
Crayon optique 100

CONNECTEURS 101

Prise PERITEL 101
Connecteur d'extension 102
Connecteur de crayon optique 103
Connecteur de Lecteur-Enregistreur 104
Prise CANON de contrôleur de communication 105
Connecteurs DIN de manettes de jeu 106
Brochage des principaux circuits intégrés 107

ORGANES CENTRAUX 109

Clavier 109
Ecran 111
Disquettes 113

COMMENT ? 117

Supprimer le bruitage du clavier 117
Changer le temps de latence du clavier 117
Définir une variable de position du curseur 117
Définir une variable de haut de fenêtre d'écran 118
Définir une variable de bas de fenêtre d'écran 118
Définir une variable avec caractéristiques d'affichage 119
Interdire l'utilisation de touches 120
Changer la forme des caractères 121
Sauvegarder une page d'écran 122
Sauvegarder une bande d'écran graphique 122
Créer un caractère graphique 123
Créer une variable graphique multicases 124
Reconnaître à l'écran un caractère graphique 125
Inhiber le clavier 126
Changer l'en-tête d'une disquette 126

Recopier le D.O.S. sans BACKUP 127
Programmer un auto-start sous D.O.S. 127
Utiliser le graphisme de l'imprimante 128

ANNEXES

ANNEXE I - CODES ESCAPE 131

ANNEXE II - CODES ESPACE 135

ANNEXE III - CARACTERES 137

Code ASCII 137
Caractères accentués 138
Codes "accents" retournés par SCREEN(X,Y) 138

ANNEXE IV - TABLEAUX DES PRINCIPAUX CODES 139

Codes couleur 139
Codes de note de musique 140
Codes d'octave 140
Codes de durée de note 140

ANNEXE V - CARACTERISTIQUES TECHNIQUES 141

Présentation 141
Alimentation extérieure 141
Caractéristiques techniques 141

ANNEXE VI - 143

Conversion Décimal-Hexadécimal 143
Conversion Hexadécimal-Binaire 144
Schéma d'écran "cases" 145
Schéma d'écran "points" 146

Les commandes du Basic sont divisées en deux grands groupes :

- les FONCTIONS ;
- les INSTRUCTIONS.

Les **FONCTIONS** retournent un résultat numérique ou alphanumérique. Nous distinguerons :

- les Fonctions numériques ;
- les Fonctions alphanumériques ;
- les Fonctions de transfert numérique-alphanumérique ;
- les Fonctions de transfert alphanumérique-numérique.

Les **INSTRUCTIONS** effectuent des tâches. Nous distinguerons :

- les Instructions d'affectation (d'espace mémoire ou de variable) ;
- les Instructions de branchement ;
- les Instructions de condition ;
- les Instructions graphiques ;
- les Instructions d'affichage.
- les Instructions d'édition

FONCTIONS NUMERIQUES

Fonction	*Retour*	*Condition*
ABS(x)	La valeur absolue de x.	
COS(x)	Le cosinus de x.	x en radians.
CSRLIN	La ligne du curseur.	
EXP(x)	L'exponentielle de base 'e' de x.	x < 88
FIX(x)	La valeur entière la plus proche de x.	
FRE(x)	L'espace-mémoire libre.	
INT(x)	La partie entière de x.	
LOG(x)	Le logarithme népérien de x.	x positif
PEEK(x)	La valeur de l'octet d'adresse x.	-131Ø71 =< x =< 65535
POS	La colonne du curseur.	
POINT(x,y)	Le code couleur du point (x,y).	Ø =< x =< 319 Ø =< y =< 199
PTRIG	Ø si le crayon optique non appuyé. 1 si le crayon optique est appuyé.	
RND	Un nombre compris entre Ø et 1.	
SCREEN(x,y)	Le code du caractère colonne x, ligne y.	Ø =< x =< 39 Ø = y =< 24
SGN(x)	-1 si x < Ø Ø si x = Ø 1 si x > Ø.	
SIN(x)	Le sinus de l'angle x.	x en radians
SQR(x)	La racine carrée de x.	x >= Ø
STICK(x)	Le code position de la manette de jeux numéro x.	Ø=<x=<1
STRIG(x)	Ø si bouton manette x non appuyé. -1 si bouton manette x appuyé.	Ø=<x=<1
TAN(x)	La tangente de l'angle x.	x en radians.
VARPTR(x)	L'adresse de la variable x.	

FONCTIONS ALPHANUMERIQUES

Fonction	*Retour*	*Condition*
LEFT$(A$,x)	Les x caractères à gauche dans A$.	Ø =< x =< 255
MID$(A$,x,y)	Les y caractères dans A$ à partir du x-ième.	Ø =< x =< y Ø =< y =< 255
RIGHT$(A$,x)	Les x caractères à droite dans A$.	Ø =< x =< 255

FONCTIONS DE TRANSFERT NUMERIQUE-ALPHANUMERIQUE

Fonction	*Retour*	*Condition*
CHR$(x)	Le caractère de code x.	Ø =< x =< 255
GR$(x)	Le caractère graphique de numéro x.	Ø =< x < N ; N est défini par CLEAR,,N
STR$(x)	L'écriture du nombre x en chaîne de caractères.	
VARPTR(A)	L'adresse de stockage de A.	

FONCTIONS DE TRANSFERT ALPHANUMERIQUE-NUMERIQUE

Fonction	*Retour*	*Condition*
ASC(A$)	Le code du premier caractère de A$.	A$ non vide
FRE(A$)	L'espace alphanumérique libre.	
INSTR(A$,B$)	Le numéro du premier caractère de B$ si contenu dans A$. Ø si absent.	
VARPTR(A$)	L'adresse du bloc de définition de la variable chaîne de caractères A$.	
VAL(A$)	La valeur du premier nombre écrit dans la chaîne de caractères A$.	

CLEAR Affecte de la place en mémoire pour les chaines de caractères ou pour le reste (nombres, programmes).

Syntaxe : CLEAR nbr1,nbr2,nbr3
nbr1 : le nombre d'octets réservés pour les chaînes.
nbr2 : l'indication du haut de la mémoire Basic.
nbr3 : le nombre de caractères alphanuméririques réservés.

DATA Liste des données lues par READ.

Syntaxe : DATA nbr1,nbr2,...,'chaîne', 'chaîne2'...

DEFGR$(x)= Affecte au caractère graphique le numéro x les valeurs qui suivent.

Syntaxe : DEFGR$(nbrØ)=nb1,nb2,nb3,nb4,nb5,nb6, nb7,nb8
nbrØ : numéro du caractère graphique.
nb1-nb8 : codes des segments de huit points définissant les huit lignes d'un caractère.

Conditions
Ø =< nb! =< 255
Ø =< nbr < N (N défini par CLEAR,,N).

DIM Affecte de la place en mémoire pour les variables tableaux spécifiés après DIM.

ERROR x Affecte immédiatement le code x à une erreur simulée.

FOR X=... TO... Affectations successives à X d'une suite de valeurs dont on spécifie la première et la dernière. Le changement pour la valeur suivante s'effectue à la rencontre de NEXT.

Syntaxe : FOR var=nbr1 TO nbr2
var : variable affectée par la boucle.
nbr1 : premier nombre de la scrutation.
nbr2 : dernier nombre de la scrutation.

FOR X=..TO..STEP... Identique au précédent, mais les valeurs sont affectées avec un 'pas' spécifié après STEP.

Syntaxe : FOR var=nbr1 TO nbr2 STEP nbr3
Idem FOR...TO..., mais nbr3 est le 'pas' (le saut entre deux valeurs prises).

INPEN x,y Affecte à x l'abscisse et à y l'ordonnée du point vu par le crayon optique.

Syntaxe : INPEN var1,var2
var1 : variable contenant l'abscisse du point lu.
var2 : variable contenant l'ordonnée du point lu.

Remarque : si la lecture est mauvaise, var1=var2 =-1 !

INPUT Affecte à la variable numérique ou alphanumérique, spécifiée après INPUT, la valeur tapée au clavier pendant le déroulement du programme. Pas de virgule.

Syntaxe : INPUT var1(,var2...)
ou : INPUT"chaîne de caractères"; var1(,var2,...)
var1(var2...) est la valeur (sont les valeurs) attribuée au clavier par l'utilisateur du programme.

INPUT# Lecture d'un enregistrement dans un fichier séquentiel.

Syntaxe : INPUT=nbr,var
nbr : numéro du canal ouvert en lecture séquentielle.
var : variable (numérique ou alphanumérique) à laquelle l'enregistrement lu est affecté.

INPUT$(x) Attente de la frappe de x touches, sans affichage.

Syntaxe : INPUT$(nbr)
nbr : nombre de caractères attribués, sans affichage.

INPUTPEN x,y Identique à INPEN, mais la lecture n'a lieu que si le crayon optique est appuyé.

LINE INPUT Comme INPUT, sauf qu'on ne peut saisir qu'une chaîne de caractères. Tous les caractères avant <ENTREE> sont saisis, y compris les virgules.

LINE INPUT# Comme INPUT#, sauf qu'on ne peut lire qu'une chaîne de caractères. Tous les caractères sont lus sur le périphérique jusqu'à un CR (Retour Chariot).

POKE x,y Affecte à l'octet d'adresse x la valeur de y.

Syntaxe : POKE nbr1,nbr2
nbr1 : adresse de l'octet où on inscrit nbr2.

READ Ordre de lecture des données mises en DATA.

Syntaxe : READ A,B,...A$,B$,...

= Symbole de base de l'affectation numérique et alphanumérique. Le membre de droite est affecté à la variable de gauche.

EXEC x Branchement à l'octet d'adresse x.

Syntaxe : EXEC nbr
nbr : adresse de l'octet de la routine en langage machine où l'on se branche.

GOSUB Branchement direct à un sous-programme.

Syntaxe : GOSUB ln
ln : numéro de la première ligne du sous-programme Basic où l'on se branche (pas de GOSUB calculé).

GOTO Branchement direct à une ligne.

Syntaxe : GOTO ln
ln : numéro de ligne Basic où l'on souhaite se brancher.

ON x GOSUB Branchement à des sous-programmes en fonction de la valeur du nombre x.

Syntaxe : ON nbr GOSUB ln1,ln2,...
nbr : variable de choix. Si nbr=0, l'instruction est ignorée. Si nbr=1, il y a GOSUB à la ligne ln1. Si nbr2, il y a GOSUB à la ligne ln2,...

ON x GOTO Branchement à des lignes en fonction de la valeur du nombre x.

Syntaxe : ON nbr GOTO ln1,ln2,...
nbr : variable de choix. Si nbr=0, l'instruction est ignorée. Si nbr=1, il y a GOTO à la ligne ln1. Si nbr=2, il y a GOTO à la ligne ln2,...

RESUME Retour à la ligne d'occurrence d'une erreur, celle-ci étant résorbée.

Syntaxe : RESUME
L'erreur est, si possible, résorbée et l'exécution de l'instruction où l'erreur est apparue est reprise. Si l'erreur n'a pas été résorbée, on risque de se mettre dans une boucle sans fin.

RESUME nl Retour à la ligne nl après résorption d'erreur.

Syntaxe : RESUME nl
Comme RESUME, mais le retour se fait à la ligne Basic nl.

RESUME NEXT Retour à la ligne suivant celle où une erreur est survenue.

Syntaxe : RESUME NEXT

RETURN Retour à l'instruction suivant le GOSUB appelant.

Syntaxe : RETURN
L'exécution se poursuit après le GOSUB qui a appelé le sous-programme qui se termine.

IF...THEN... Si la condition est réalisée, la suite après THEN est exécutée.

Syntaxe : IF 'cond' THEN 'instr'
Si 'cond' est vraie, l'instruction 'instr' s'exécute, sinon le programme se poursuit à la ligne suivante.

Conseil : ne pas mettre de séparateur d'instruction après 'instr'.

IF...THEN...ELSE Si condition réalisée, on exécute la suite après THEN, sinon après ELSE.

Syntaxe : IF 'cond' THEN 'instr1' ELSE 'instr2'
Si 'cond' est vraie, 'instr1' est exécutée et non 'instr2'. Si 'cond' est fausse, 'instr2' est exécutée et non 'instr1'.

ON ERROR GOTO Branchement au numéro de ligne spécifié en cas d'occurrence d'erreur.

Syntaxe : ON ERROR GOTO ln
ln : numéro de ligne Basic où il y a branchement automatique en cas d'occurrence d'erreur. Il n'y a pas de diagnostic d'erreur du type Error XX in XXXX.

BOX Trace un rectangle.

Syntaxe : BOX(nbr1,nbr2)-(nbr3,nbr4),nbr5
nbr1,nbr2 sont les coordonnées d'un sommet du rectangle.
nbr3,nbr4 sont les coordonnées du sommet opposé.
nbr5 est le code de couleur du tracé (optionnel).

Syntaxe : BOX -(nbr1,nbr2),nbr3
Le dernier point graphique affiché est le premier sommet.
nbr1,nbr2 sont les coordonnées du sommet opposé.
nbr3 est le code de couleur du tracé (optionnel).

BOXF Trace un rectangle plein.

Syntaxe : BOXF(nbr1,nbr2)-(nbr3,nbr4),nbr5
nbr1,nbr2 sont les coordonnées d'un sommet du rectangle.
nbr3,nbr4 sont les coordonnées du sommet opposé.
nbr5 est le code de couleur du rectangle plein (optionnel).

Syntaxe : BOXF -(nbr1,nbr2),nbr3
Le dernier point graphique affiché est le premier sommet.
nbr1,nbr2 sont les coordonnées du sommet opposé.
nbr3 est le code couleur du rectangle plein (optionnel).

LINE Trace un segment entre deux points.

Syntaxe : LINE(nbr1,nbr2)-(nbr3,nbr4),nbr5
nbr1,nbr2 sont les coordonnées d'une extrémité du segment.
nbr3,nbr4 sont les coordonnées de l'autre extrémité.
nbr5 est le code de couleur du tracé (optionnel).

Syntaxe : LINE -(nbr1,nbr2),nbr3
Le dernier point graphique affiché est l'extrémité du segment.
nbr1,nbr2 sont les coordonnées de l'autre extrémité.
nbr3 est le code de couleur du tracé (optionnel).

PSET Affiche un point à l'écran

Syntaxe : PSET(nbr1,nbr2),nbr3
Affiche le point de coordonnées nbr1,nbr2 dans la couleur de code nbr3.

ATTRB Fixe la taille des caractères.

Syntaxe : ATTRB nbr1,nbr2
Si nbr1=1, les caractères sont en double largeur.
Si nbr2=1, les caractères sont en double hauteur.

BOX Trace un rectangle de caractères dont deux sommets opposés sont spécifiés.

Syntaxe : BOX(nbr1,nbr2)-(nbr3,nbr4)car$,nbr5
car$ est le caractère dessinant le rectangle.
nbr1,nbr2 sont les coordonnées d'un sommet du rectangle.
nbr3,nbr4 sont les coordonnées du sommet opposé.
nbr5 est le code de couleur des caractères (optionnel).

Syntaxe : BOX -(nbr1,nbr2)car$,nbr3
car$ est le caractère dessinant le rectangle. L'emplacement du curseur est le premier sommet.
nbr1,nbr2 sont les coordonnées du sommet opposé.
nbr3 est le code de couleur des caractères (optionnel).

BOXF Comme BOX, mais le rectangle est plein.

COLOR Fixe la couleur des caractères et du fond sur lequel ils sont inscrits. Peut aussi inverser les couleurs.

Syntaxe : COLOR nbr1,nbr2,nbr3
nbr1 détermine la couleur des caractères.
nbr2 détermine la couleur du cadre où ils s'inscrivent.
Si nbr3=Ø, les couleurs entre cadre et caractères sont permutées.
Idem si nbr3=1.

CONSOLE Fixe le haut et le bas de la page défilable d'écran.

Syntaxe : CONSOLE nbr1,nbr2,nbr3,nbr4
nbr1 détermine le haut de la fenêtre d'écran.
nbr2 détermine le bas de la fenêtre d'écran.
Si nbr3=1, les couleurs de la fenêtre sont figées.
nbr3=Ø supprime le mode précédent.

Si nbr4=2, il n'y a plus de défilement d'écran. Arrivé en bas de page, le curseur remonte en haut sans effacer.
Si nbr4=1, il y a défilement lent.
nbr4=Ø remet le défilement en mode normal.

LINE Trace une ligne de caractères dont les deux extrémités sont spécifiées.

Syntaxe : LINE(nbr1,nbr2)-(nbr3,nbr4)car$,nbr5
car$ est le caractère formant la ligne.
nbr1,nbr2 sont les coordonnées d'une extrémité du segment.
nbr3,nbr4 sont les coordonnées de l'autre extrémité.
nbr5 est le code de couleur du tracé (optionnel).

Syntaxe : LINE -(nbr1,nbr2)car$,nbr3
car $ est le caractère formant la ligne. La position du curseur est la première extrémité du segment.
nbr1,nbr2 sont les coordonnées de l'autre extrémité.
nbr3 est le code de couleur du tracé (optionnel).

LOCATE Positionne le curseur à l'écran.

Syntaxe : LOCATE nbr1,nbr2,nbr3
nbr1,nbr2 sont les coordonnées où l'on souhaite mettre le curseur.
Si nbr3=Ø, le curseur disparaît.
Si nbr3=1, le curseur clignote.

PRINT Affiche à la position courante du curseur les données ou variables qui suivent.

Syntaxe : PRINT ..(chaînes ou variables ou constantes)

PRINT# Envoi d'une chaîne de caractères ou d'un nombre sur un périphérique.

Syntaxe : PRINT=nbr,***,***,***
nbr est le numéro du canal ouvert en sortie.
Les étoiles *** représentent soit des chaînes de caractères, soit des variables alphanumériques, soit des nombres, soit des variables numériques.

PRINT USING Affiche à la position courante du curseur les données ou variables qui suivent le 'champ'. Le 'champ' donne le format d'écriture.

Syntaxe : PRINT USING 'champ formaté' nbr1, nbr2...
Le champ formaté peut contenir des mots ou des descripteurs de champ.
Descripteurs de champ : ###.# définit un champ numérique avec un chiffre après la virgule et trois avant.
+ devant le champ numérique affecte systématiquement le signe à l'écriture du nombre (voir page 52 la simulation de PRINT USING alphanumérique).

PRINT# USING Sortie formatée sur périphérique.
Comme PRINT=, associé à un USING comme ci-dessus pour écrire sur le périphérique à l'aide d'un champ numérique.

PSET Affiche un caractère aux coordonnées spécifiées.

Syntaxe : PSET(nbr1,nbr2)car$,nbr3
Affiche en coordonnées nbr1,nbr2 le caractère car$ dans la couleur nbr3.

DELETE Supprime les numéros de ligne spécifiés.

Syntaxe
DELETE nl1-nl2 : suppression des lignes de nl1 à nl2.
DELETE -nl : suppression des lignes avant nl.
DELETE nl- : suppression des lignes après nl.

DOS Appelle le Disk Operating System. Toute la mémoire est réinitialisée.

END Fin de l'exécution d'un programme. Les valeurs des variables sont conservées, les fichiers fermés.

LIST Liste un programme, en totalité ou en partie, sur le périphérique assigné par le descripteur de fichier. Si celui-ci n'est pas indiqué, c'est le Lecteur-Enregistreur. Le listage se fait en ASCII.

Syntaxe
LIST : liste tout le programme.
LIST nl : liste la ligne nl.
LIST -nl : liste toutes les lignes avant nl.
LIST nl- : liste toutes les lignes après nl.
LIST nl1-nl2 : liste les lignes entre nl1 et nl2.

LOAD Charge un programme à partir du Lecteur-Enregistreur.

Syntaxe
LOAD : charge le premier programme trouvé.
LOAD"" : idem.
LOAD"nom" : charge le programme "nom" si trouvé.
LOAD....,R : comme ci-dessus, mais le programme commence.

LOADM Charge une zone-mémoire avec le fichier binaire spécifié dans le descripteur de fichier.

Syntaxe
LOADM : charge le premier fichier trouvé.
LOADM"nom" : charge le fichier "nom" si trouvé.
LOADM....,nb : idem, avec décalage de nb pour le chargement en mémoire.

MERGE Incorpore à un programme un autre programme sur cassette qui doit être codé en ASCII.

Syntaxe
Comme LOAD, sans option ,R

SAVE Envoie un programme sur le Lecteur-Enregistreur.

Syntaxe
SAVE"nom" : sauve le programme sous "nom".
SAVE"nom",A : idem, mais avec codage ASCII.
SAVE"nom",P : idem, avec protection logicielle.

SAVEM Sauve une partie de la mémoire sur le Lecteur-Enregistreur, sous forme de fichier binaire.

Syntaxe
SAVEM"nom",ad1,ad2,nbr : sauve la portion mémoire entre ad1 et ad2 sous forme de fichier binaire qui sera chargé avec décalage de nbr.

STOP Arrêt provisoire de l'exécution d'un programme. L'indication Break in nl. indique le numéro de la ligne de l'instruction STOP.
On peut imprimer, modifier les variables en mode direct.
La reprise du programme peut se faire soit par CONT, soit par un GOTO direct.

TROFF Fin du mode TRON.

TRON Provoque l'affichage du numéro de chaque ligne de programme effectuée. Cet affichage se fait entre crochets.

<CNT> <C> : interruption du programme en cours.

<CNT> <W> : soudure de ligne. La ligne suivant le curseur se rattache à celle du curseur.

<CNT> <X> : suppression de tous les caractères après le curseur dans sa ligne.

<CNT> <G> : émission d'un BIP dans le haut-parleur de la T.V.

← : déplacement du curseur vers la gauche : 08 Hexa.

→ : déplacement du curseur vers la droite : 09 Hexa.

↑ : déplacement du curseur vers le haut : 0B Hexa.

↓ : déplacement du curseur vers le bas : 0A Hexa.

⏎ : retour du curseur en haut de l'écran : 1E Hexa.

BASIC : cette touche appuyée, le mot BASIC, écrit sur la touche alors enfoncée, s'inscrit.

INS : pousse les caractères de la ligne courante après le curseur, d'une case vers la droite : 1C Hexa.

EFF : pousse les caractères de la ligne courante après le curseur, d'une case vers la gauche. Le caractère sous le curseur disparaît alors : 1D Hexa.

CNT : cette touche appuyée enlève 64 au code ASCII de la touche qui est alors enfoncée.

STOP : interruption du Basic jusqu'à l'enfoncement d'une quelconque autre touche : 02 Hexa.

ACC : envoie le code 22 Décimal. Cet envoi détermine le clavier à attendre un caractère 'accent' puis une minuscule basse.

INITIAL PROG : *touche blanche fine sour la trappe cartouche.* Provoque un redémarrage "à chaud" avec réinitialisation des pointeurs de table de caractères et de décodage clavier. Le programme en mémoire est conservé.

<é> : enfoncée après <ACC>, imprime directement un é.

<è> : enfoncée après <ACC>, imprime directement un è.

<ù> : enfoncée après <ACC>, imprime directement un ù.

<ç> : enfoncée après <ACC>, imprime directement un ç.

<à> : enfoncée après <ACC>, imprime directement un à.

<?> : seule instruction du Basic qui soit abrégée. Elle remplace PRINT.

Remarque : pour les habitués de TO7 et TO7/70, il n'est pas possible de remplacer par un point le numéro de la dernière ligne de programme modifiée ou de la ligne où il y a eu une erreur.

```
LIST .
LIST .-
```

et autres syntaxes ressortiront un diagnostic d'erreur 02.

Le MO5 peut communiquer avec ses périphériques par l'intermédiaire de canaux. Ceux-ci s'ouvrent à l'aide de l'ordre OPEN suivi d'un descripteur de fichier.

Un descripteur de fichier est une chaîne de caractères. Il comprend d'abord le mnémonique du périphérique choisi, en quatre lettres suivies de deux points :

CASS: pour cassette ;
KYBD: pour clavier ;
LPRT: pour imprimante parallèle ;
SCRN: pour l'écran.

Pour l'imprimante et pour l'écran, suit en option la longueur de ligne entre parenthèses.

Exemples

LPRT:(80) pour imprimer sur 80 colonnes ;
SCRN:(32) pour un écran sur 32 colonnes.

Suit alors le nom du fichier. Il comprend de un à huit caractères suivis en option d'un point et de un à trois autres caractères.

Exemples

```
CASS:PROG.BAS
KYBD:FICHIER.DAT
LPRT:(80)PRINTER.1
SCRN:AZERTYUI.OP
```

Remarque : pour le Lecteur-Enregistreur utilisé sans les disquettes, le mnémonique du périphérique est optionnel.

L'exemple précédent CASS:PROG.BAS est donc équivalent à :

```
PROG.BAS
```

STOCKAGE DES VARIABLES EN MEMOIRE

Les variables du MO5 sont de trois types :

- entières ;
- simple précision ;
- alphanumériques.

Stockage des variables entières

Elles sont contenues en mémoire sous la forme d'une suite de deux octets. Sous Basic, la fonction VARPTR (X%) va retourner l'adresse du premier octet de la variable X%. Sa valeur sera donc comprise entre -32767 et +32767.

Pour retrouver la valeur à partir des deux octets de codage, faire :

```
A=256*PEEK(VARPTR(X%))+PEEK(VARPTR(X%)+1
):A=A+65536*(A>32768)
```

ce qui semble un peu compliqué, mais pourtant...

On prend la valeur contenue dans les 16 bits des deux octets si elle est supérieure à 32768 (2 puissance 15), on enlève 65536 (2 puissance 16). Le bit 7 de l'octet de plus fort poids représente donc le signe (1 pour - et Ø pour +).

Stockage des variables simple précision

Elles sont rangées en mémoire sous une forme utilisant quatre octets :

- le premier octet contient l'exposant de 1Ø ;
- les trois octets suivants contiennent les trois bits de la mantisse, du plus fort au plus faible poids.

Le bit 7 du second octet donne le signe .

Stockage des variables alphanumériques

L'instruction VARPTR du Basic pointe le début de la zone de définition de la variable 'chaîne'. Cette zone contient trois octets :

- le premier donne la longueur de la chaîne ;
- les deux autres donnent l'adresse du début de la chaîne.

On ne peut pas définir une chaîne de caractères de plus de 255 octets.

CODES DES INSTRUCTIONS

Les instructions et mots-clés du Basic sont stockés en mémoire grâce à des octets de valeur > 127.

Mot-clé	*Code Hexa*	*Mot-clé*	*Code Hexa*	*Mot-clé*	*Code Hexa*
END	BØ	PRINT	AB	DSKINI	D6
FOR	B1	CONT	AC	DSKO$	D7
NEXT	B2	LIST	AD	KILL	D8
DATA	B3	CLEAR	AE	NAME	D9
DIM	B4	DOS	AF	FIELD	DA
READ	B5		BØ	LSET	DB
	B6	NEW	B1	RSET	DC
GO	B7	SAVE	B2	PUT	DD
RUN	B8	LOAD	B3	GET	DE
IF	B9	MERGE	B4	VERIFY	DF
RESTORE	BA	OPEN	B5	DEVICE	EØ
RETURN	BB	CLOSE	B6	DIR	E1
REM	BC	INPEN	B7	FILES	E2
'	BD	PEN	B8	WRITE	E3
STOP	BE	PLAY	B9	UNLOAD	E4
ELSE	BF	TAB	BA	BACKUP	E5
TRON	9Ø	TO	BB	COPY	E6
TROFF	91	SUB	BC	CIRCLE	E7
DEFSTR	92	FNC	BD	PAINT	E8
DEFINT	93	SPC	BE	DRAW	E9
DEFSNG	94	USING	BF	RENUM	EA
	95	USR	CØ	SWAP	EB
ON	96	ERL	C1	SGN	FF8Ø
TUNE	97	ERR	C2	INT	FF81
ERROR	98	OFF	C3	ABS	FF82
RESUME	99	THEN	C4	FRE	FF83
AUTO	9A	NOT	C5	SQR	FF84
DELETE	9B	STEP	C6	LOG	FF85
LOCATE	9C	+	C7	EXP	FF86
CLS	9D	-	C8	COS	FF87
CONSOLE	9E	*	C9	SIN	FF88
PSET	9F	/	CA	TAN	FF89
MOTOR	AØ	^	CB	PEEK	FF8A
SKIPF	A1	AND	CC	LEN	FF8B
EXEC	A2	OR	CD	STR$	FF8C
BEEP	A3	XOR	CE	VAL	FF8D
COLOR	A4	EQV	CF	ASC	FF8E
LINE	A5	IMP	DØ	CHR$	FF8F
BOX	A6	MOD	D1	EOF	FF9Ø
	A7		D2	CINT	FF91
ATTRB	A8	>	D3	CSNG	FF92
DEF	A9	=	D4	CDBL	FF93
POKE	AA	<	D5	FIX	FF94

CODES DES INSTRUCTIONS

Mot-clé	*Code Hexa*	*Mot-clé*	*Code Hexa*	*Mot-clé*	*Code Hexa*
HEX$	FF95	RND	FF9F	CVS	FFA9
OCT$	FF96	INKEY$	FFAØ		FFAA
STICK	FF97	INPUT	FFA1	MKI$	FFAB
STRIG	FF98	CSRLIN	FFA2	MKS$	FFAC
GR$	FF99	POINT	FFA3		FFAD
LEFT$	FF9A	SCREEN	FFA4	LOC	FFAE
RIGHT$	FF9B	POS	FFA5	LOF	FFAF
MID$	FF9C	PTRIG	FFA6	SPACE$	FFBØ
INSTR	779D	OSKF	FFA7	STRING$	FFB1
VARPTR	FF9E	CVI	FFA8	OSKI$	FFB2

Ces codes sont ceux contenus en mémoire centrale lors de la présence d'un programme, le séparateur d'instruction <:> ayant son code normal 58.

On appelle D.O.S. le Disk Operating System. En français, S.E.D. ou Système d'Exploitation de Disquettes.

Il s'agit d'un ensemble de fonctions et d'instructions supplémentaires ne figurant pas dans le Basic de base.

Le D.O.S. autorise la gestion du ou des lecteurs-enregistreurs de disquettes (nous dirons drives) et ajoute quelques fonctions et instructions nouvelles au Basic.

Le D.O.S. est livré sur une disquette et vient se loger entre les adresses AØØØ et A7BF Hexa du MO5.

CVI Transcription en simple précision d'un nombre compacté grâce à MKI$.

Syntaxe : CVI(A$) avec A$ sur quatre caractères.

CVS Transcription en entier d'un nombre compacté sur deux octets grâce à MKS$.

Syntaxe : CVS(A$)

DEF FN Définition de fonction numérique.

Syntaxe : DEF FN*(X,Y,...)=.....
L'étoile est une lettre donnant un nom à la fonction.

DSKI$ Chargement dans une variable d'un secteur de la disquette.

Syntaxe : DSKI$(nbr1,nbr2,nbr3)
nbr1 : numéro du drive.
nbr2 : numéro de la piste Ø=< nbr1 =< 39.
nbr3 : numéro du secteur 1=< nbr2 =< 16.
La longueur de DSKI$ est de 128 caractères en simple densité et de 255 en double densité.

DSKF Place libre sur la disquette.

Syntaxe : DSKF(nbr)
nbr : numéro de drive.

DSKO$ Enregistrement direct d'un secteur.

Syntaxe : DSKO$ nbr1,nbr2,nbr3,var$
nbr1 : numéro de drive.
nbr2 : numéro de piste Ø=< nbr2 =< 39.
nbr3 : numéro de secteur 1=< nbr3 =< 16.
var$: est la variable contenant les 128 octets à inscrire dans le secteur en simple densité ; 255 en double densité.

EOF End Of File retourne Ø si la fin d'un fichier séquentiel n'est pas atteinte, -1 si elle l'est.

Syntaxe : EOF(nbr)
nbr : numéro du canal de communication du fichier.

HEX$ Retourne le code hexadécimal du nombre spécifié.

Syntaxe : HEX$(nbr)

LOC Indique la LOCalisation du pointeur de fichier. En accès direct, donne le numéro de l'enregistrement suivant le dernier lu par GET ou écrit par PUT ; en accès séquentiel, retourne le nombre d'enregistrements déjà lus ou écrits.

Syntaxe : LOC(nbr)
nbr : numéro du canal de communication du fichier.

LOF Retourne le numéro du dernier enregistrement, aussi bien en accès séquentiel qu'en accès direct.

Syntaxe : LOF(nbr)
nbr : numéro du canal de communication du fichier.

MKI$ Compactage en deux octets d'un nombre entier.

Syntaxe : MKI$(nbr)
nbr : l'entier à compacter.

MKS$ Compactage en quatre octets d'un nombre simple précision.

Syntaxe : MKS$(nbr)
nbr : le nombre simple précision à compacter.

SPACE$ Retourne une chaîne de caractères 'SPACE' de longueur spécifiée. 'SPACE' est le caractère de code 32.

Syntaxe : SPACE$(nbr)
nbr : longueur de la chaîne ESPACE.

STRING$ Retourne une chaîne composée d'un certain nombre de fois le même caractère.

Syntaxe : STRING$ (nbr1,nbr2)
nbr1 : longueur de la chaîne.
nbr2 : le code ASCII du caractère.

SWAP Permute deux variables de même type.

Syntaxe : SWAP nbr1,nbr2
SWAP var1$,var2$.

AUTO Création automatique des numéros de ligne d'un programme, avec spécification du 'pas' et de la ligne de départ.

Syntaxe : AUTO nbr1,nbr2
nbr1 : ligne de départ (10 par défaut)
nbr2 : 'pas' de progression des lignes (10 par défaut)
Si une étoile apparaît <*> après le numéro de ligne, c'est que la ligne existe déjà. ENTREE pour ne pas l'écraser, avant toute frappe.
Pour arrêter, faire <CNT> <C> ou <RAZ>.

BACKUP Recopie physique totale d'une disquette.

Syntaxe : BACKUP nbr1 TO nbr2
nbr1 : numéro du drive source.
nbr2 : numéro du drive destination (nbr1 par défaut).

CIRCLE Tracé d'un cercle.

Syntaxe : CIRCLE (nbr1,nbr2),nbr3,nbr4
nbr1 : abscisse du centre Ø=< nbr1 =< 319
nbr2 : ordonnée du centre Ø=< nbr2 =< 199
nbr3 : rayon Ø=< nbr3
nbr4 : code couleur

Conditions
Ø=< nbr1-nbr3
Ø=< nbr2-nbr3

CLOSE Fermeture d'un canal (ou de tous les canaux) de fichier.

Syntaxe : CLOSE nbr
nbr : numéro du canal à fermer.
Si nbr est omis, tous les canaux sont fermés.

COPY Copie logique d'un fichier.

Syntaxe : COPY"nom1 fichier"TO"nom2 fichier"
nom1 fichier est le descripteur du fichier source.
nom2 fichier est le descripteur du fichier destination.

DEVICE Définit le périphérique pris par défaut.

Syntaxe : DEVICE"descr. de périph."
Le descripteur de périphérique spécifié détermine celui qui sera dorénavant pris par défaut.

DIR Affiche à l'écran le DIRector de la disquette spécifiée.

Syntaxe : DIR "nbr:
nbr : numéro du drive.

DRAW Trace un dessin de points à partir du dernier point tracé.

Syntaxe : DRAW "chaîne"
"chaîne" comprend des descripteurs de progression :
Lnbr pour aller à gauche de nbr points.
Unbr pour aller vers le haut de nbr points.
Rnbr pour aller vers la droite de nbr points.
Dnbr pour aller vers le bas de nbr points.
Cnbr pour fixer la couleur (codes habituels).
B pour ne pas imprimer le déplacement suivant.
X pour exécuter la chaîne de caractères qui suit.
N pour revenir au point de départ.
Anbr pour changer le 'nord' Ø=Nord /1=Est /2=Sud /3=Sud.

DSKINI Formate une disquette et efface toutes les informations qui s'y trouvaient.

Syntaxe : DSKINI nbr
nbr : est le numéro du drive où doit se passer le formatage.

FIELD Descripteur de champ pour un fichier à accès direct.

Syntaxe : FIELD#1,nbr1 AS var1$, nbr2 AS var2$....
nbr1 : est la taille réservée à la variable de buffer var1$.
nbr2 : est la taille réservée à la variable de buffer var2$.
.....

FILES Détermine le nombre de canaux pouvant être ouverts simultanément.

Syntaxe : FILES nbr1,nbr2
nbr1 : nombre de canaux pouvant être ouverts.
nbr2 : longueur maxi du buffer d'un canal direct.
Instruction à exécuter en mode direct, car elle s'accompagne d'un CLEAR automatique.

GET Mise en tableau d'une forme graphique de l'écran.

Syntaxe : GET (nbr1,nbr2)-(nbr3,nbr4),var
nbr1 : abscisse du coin haut gauche du rectangle à saisir.
nbr2 : ordonnée du coin haut gauche du rectangle à saisir.
nbr3 : abscisse du coin bas droit du rectangle à saisir.
nbr4 : ordonnée du coin bas droit du rectangle à saisir.
var : variable tableau devant contenir la forme. Elle doit être dimensionnée au préalable en longueur-largeur.

GET# Lecture d'un enregistrement dans un fichier à accès direct. Cet enregistrement est envoyé dans le buffer du canal spécifié.

Syntaxe : GET#nbr1,nbr2
nbr1 : numéro du canal ouvert en accès direct.
nbr2 : numéro de l'enregistrement.

INPUT# Voir INPUT# dans les instructions d'affectation classiques.

KILL Effacement d'un fichier du directory de la disquette.

Syntaxe : KILL"nom de fichier"
"nom de fichier" est le descripteur du fichier. Avec le D.O.S., le descripteur de fichier comporte un numéro de drive et le nom complet, y compris l'extension.

LINE INPUT# Voir LINE INPUT# dans les instructions d'affectation classiques.

LIST Voir LIST dans les fonctions d'édition classiques.

Remarque : si on ne spécifie pas le périphérique dans le descripteur de fichier, celui-ci est pris par défaut sur le disque courant ou sur le périphérique assigné par DEVICE.

LOAD Voir LOAD dans les fonctions d'édition classiques.

Remarque : si on ne spécifie pas le périphérique dans le descripteur de fichier, celui-ci est pris par défaut sur le disque courant ou sur le périphérique assigné par DEVICE.

LOADM Voir LOADM dans les fonctions d'édition classiques.

Remarque : si on ne spécifie pas le périphérique dans le descripteur de fichier, celui-ci est pris par défaut sur le disque courant ou sur le périphérique assigné par DEVICE.

LSET Affectation d'un contenu alphanumérique à la variable de buffer spécifiée. Calage à gauche.

Syntaxe : LSET var1$=var2$
var1$: variable de buffer définie dans FIELD.
var2$: contenu à affecter à var1$ (peut être une chaîne de caractères).

MERGE Voir MERGE dans les fonctions d'édition classiques.

Remarque : si on ne spécifie pas le périphérique dans le descripteur de fichier, celui-ci est pris par défaut sur le disque courant ou sur le périphérique assigné par DEVICE.

NAME Changement du nom d'un fichier sur la disquette.

Syntaxe : NAME "nom1" AS "nom2"
nom1 : est l'ancien nom du fichier.
nom2 : est le nouveau nom du fichier.

OPEN Ouverture d'un canal de communication avec un fichier.
Voir OPEN dans les instructions d'entrée-sortie classiques pour les options OPEN"I" et "OPEN"O".

Syntaxe : OPEN"R",#nbr1,"nom fichier",nbr2
nbr1 : numéro du canal à ouvrir.
nbr2 : longueur du buffer de communication.
nom fichier est le nom du fichier en accès direct à ouvrir.
Une telle ouverture en accès direct permet l'écriture et la lecture dans le fichier.

Remarque : OPEN"R" est équivalent à OPEN"D".

PAINT Peinture dans une couleur spécifiée à l'intérieur d'un contour FERME de MEME couleur.

Syntaxe : PAINT(nbr1,nbr2),nbr3
nbr1 : abscisse du point de départ du coloriage.
nbr2 : ordonnée du point de départ du coloriage.
nbr3 : couleur du coloriage.

PRINT# Voir PRINT# dans les instructions classiques d'écriture dans un fichier à accès séquentiel.

PRINT# USING Voir PRINT# USING dans les instructions classiques d'écriture dans un fichier à accès séquentiel.

PUT Affichage du tableau d'une forme graphique de l'écran.

Syntaxe : PUT (nbr1,nbr2)-(nbr3,nbr4),var
nbr1 : abscisse du coin haut gauche du rectangle à afficher.
nbr2 : ordonnée du coin haut gauche du rectangle à afficher.
nbr3 : abscisse du coin bas droit du rectangle à afficher.
nbr4 : ordonnée du coin bas droit du rectangle à afficher.
var : variable tableau devant contenir la forme. Elle doit être saisie au préalable par GET.

PUT# Ecriture d'un enregistrement dans un fichier à accès direct.

Syntaxe : PUT#nbr1,nbr2

nbr1 : numéro du canal ouvert en accès direct.
nbr2 : numéro de l'enregistrement à écrire.
Le contenu du buffer nbr1 est inscrit sur la disquette, dans l'enregistrement numéro nbr2.
Si l'on n'a pas rempli les variables de buffer avec LSET ou RSET, elles conservent les dernières valeurs lues ou écrites.

RENUM Renumérotation automatique des lignes de programme.

Syntaxe : RENUM nbr1,nbr2
nbr1 : première ligne de la nouvelle numérotation (10 par défaut).
nbr2 : première ligne à renuméroter dans l'ancienne numérotation (la première ligne par défaut).

RSET Affectation d'un contenu alphanumérique à la variable de buffer spécifiée. Calage à droite.

Syntaxe : RSET var1$=var2$
var1$: variable de buffer définie dans FIELD.
var2$: contenu à affecter à var1$ (peut être une chaîne de caractères).

SAVE Voir SAVE dans les fonctions d'édition classiques.

Remarque : si l'on ne spécifie pas le périphérique dans le descripteur de fichier, celui-ci est pris par défaut sur le disque courant ou sur le périphérique assigné par DEVICE.

SAVEM Voir SAVEM dans les fonctions d'édition classiques.

Remarque : si l'on ne spécifie pas le périphérique dans le descripteur de fichier, celui-ci est pris par défaut sur le disque courant ou sur le périphérique assigné par DEVICE.

SEARCH Nouvelle instruction du MO5 DOS qui n'existait pas sur TO7 DOS. Cherche toutes les lignes d'un programme Basic où apparaît un groupe de lettres.

Syntaxe : SEARCH "chaîne"
Toutes les lignes du programme où figure "chaîne" apparaissent à l'écran. Que "chaîne" soit composé de variables, de mots du Basic ou de commentaires.

UNLOAD Clôture de tous les canaux ouverts, écriture des buffers en attente.

Syntaxe : UNLOAD
Permet d'enlever une disquette du drive sans courir de danger au point de vue de la mise à jour du directory.

VERIFYON Toutes les entrées-sorties sur disquette sont vérifiées.

Syntaxe : VERIFYON

VERIFYOFF Fin de la procédure VERIFYON.

Syntaxe : VERIFYOFF

WRITE# Voir WRITE# dans les instructions classiques.

L'accès direct est la principale amélioration apportée à la gestion des fichiers par le D.O.S.

Jusqu'à présent, on disposait de fichiers séquentiels sur cassette. Maintenant, on va pouvoir accéder à des enregistrements dans un ordre quelconque à l'intérieur d'un fichier sur disquette.

Pour cela : ouvrir un canal de communication en accès direct :

OPEN"R",#nbr1,"nom de fichier",nbr2 (ou OPEN"D".....).

On ouvre alors un buffer (tampon) entre la mémoire du MO5 et la disquette. Un buffer est une réserve de caractères (une sorte de SAS de transit).

nbr1 est le numéro du canal, nbr2 la longueur du buffer.

L'option"R" est pour RANDOM ACCES : accès aléatoire ou direct.

Définir un champ, c'est-à-dire que l'on découpe le buffer en un certain nombre de variables alphanumériques dont on spécifie, pour chacune, la longueur :

FIELD#nbr,nbr1 AS var1$,.....

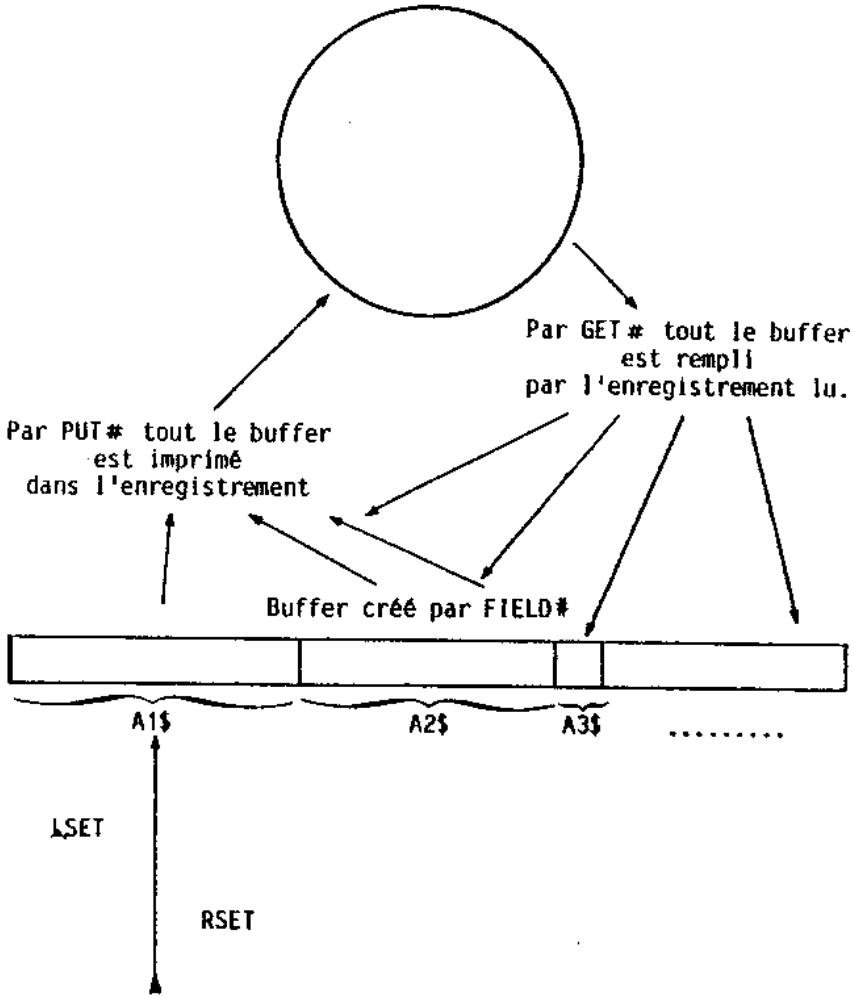
Si l'on souhaite écrire dans le fichier :

- remplir les variables de buffer avec LSET ou RSET.;
- écrire dans le fichier avec PUT#.

Si l'on souhaite lire :

- lire un enregistrement avec GET# ;
- lire les variables de buffer.

ATTENTION : ne jamais remplir les variables de buffer avec = car elles perdraient leur qualité de variable de buffer.



Une variable de buffer est remplie à l'aide de LSET ou de RSET.

Les sous-programmes qui suivent sont utilitaires. Ils remplissent un certain nombre de tâches courantes. Tous les noms de variables qui y interviennent commencent par X ou Y. Ceci pour éviter des conflits avec les vôtres. N'utilisez donc pas de noms de variables commençant par X ou Y.

Les numéros de ligne n'entrent pas en conflit. Vous pouvez donc sauver ces routines, codées en ASCII sur une cassette, et les rappeler au besoin avec MERGE.

... DANS UN MENU

On appelle MENU une liste de fonctions prévues par un programme. Il faut pouvoir choisir une fonction parmi d'autres.

```
10000 CLS:LOCATE 0,0:PRINT XH$(0):XH=0
10010 FOR XH=1 TO XI:LOCATE 0,1+XH:PRINT
CHR$(64+XH);" : ";XH$(XH):NEXT XH:XH=0
10020 LOCATE 0,20:PRINT"Z : Sortie de ce
programme";
10030 LOCATE 20,23:PRINT"Votre choix :
";CHR$(8);
10040 XH$=INPUT$(1):IFXH$="Z"THEN RETURN
10050 IF XH$<"A"OR XH$>CHR$(64+XI)THEN10
030
10060 XH=ASC(XH$)-64:RETURN
```

Appel de ce sous-programme

Il faut définir à l'avance les variables :

XI : nombre des options.
XH$(0) : titre du MENU
XH$(1) : titre du premier choix.
= = = = = = = = = = = = = = =
XH$(XI) : titre du dernier choix.

A l'issue de ce sous-programme, il ressortira :

XH = 0 si on a choisi la sortie de ce programme, sinon :
XH = numéro de l'option choisie.

Exemple d'utilisation

```
100 XH$(0)="MENU DES RECETTES CULINAIRES
"
110 XI=4
120 XH$(1)="Liste des recettes"
130 XH$(2)="Consultation d'une recette"
140 XH$(3)="Saisie d'une nouvelle recett
e"
150 XH$(4)="Modification d'une recette"
160 GOSUB 10000
170 IF XH=0 THEN CLS:END 'remplacer END
par RETURN si ce MENU est appelé par un
autre
180 ON XH GOSUB 1000,2000,3000,4000
190 GOTO 100
```

Remarques : ce sous-programme permet à un menu d'en appeler un autre. 17 options possibles, test de validité du choix.

Avec le D.O.S., on peut très bien mettre les chaînes de caractères XH$(0) ..... XH$(4) dans un fichier à accès direct, ce qui diminue d'autant la place du programme en mémoire vive.

BOUCLE D'ATTENTE AVEC MODIFICATION DES ALEAS

La fonction RND ressort un nombre compris entre 0 et 1. Mais, pour le même programme, la suite des RND est toujours la même. Pour obtenir un vrai tirage aléatoire, il faut donc un artifice : une boucle qui se fera très vite pendant laquelle s'effectueront des lectures de RND que l'on devra terminer par l'enfoncement d'une touche. Le nombre de RND lus ne sera jamais le même.

```
10100 LOCATE 0,23:PRINT"Touche <RAZ> pou
r continuer";
10110 XA=RND:IF INKEY$<>CHR$(12) THEN 10
110
10120 RETURN
```

Appel de ce sous-programme : aucune variable à prédéfinir.

A l'issue du sous-programme : aucune variable affectée.

SAISIE D'ECRAN RESTREINTE

On peut vouloir saisir une suite de caractères, sans accepter n'importe quel caractère du clavier. Il faut un test qui vérifie chaque enfoncement de touche :

```
10200 XH$=""
10210 XJ$=INPUT$(1):IF XJ$=CHR$(13) THEN
RETURN
10220 IF XJ$<>CHR$(8) THEN 10250
10230 IF XH$<>"" THEN XH$=LEFT$(XH$,LEN(
XH$)-1)
10240 XJ$=XJ$+" "+XJ$:GOTO 10270
10250 IF INSTR(X1$,XJ$)=0 THEN PLAY"DOMI
DO":GOTO 10210
10260 XH$=XH$+XJ$
10270 PRINT XJ$;
10280 GOTO 10210
```

Appel du sous-programme

Définir à l'avance la variable :

X1$: liste des caractères acceptables.

En sortie de sous-programme :

XH$: suite des caractères saisis.

Remarques : on termine la saisie par la touche <ENTREE>. La flèche à gauche efface le dernier caractère saisi.

Exemple d'utilisation

On veut saisir des opérations numériques avec des enfants. On autorisera les caractères suivants : 0123456789+-:xX,.

L'appel de la routine de saisie sera donc :

```
100 X1$="0123456789+-:Xx,.":LOCATE 5,12:
GOSUB 10200
```

si l'on souhaite saisir à partir de la case (5,12).

MOUVEMENT D'UN CARACTERE A L'ECRAN PAR MANETTE DE JEU

```
10300 XC(2)=1:XC(3)=1:XC(4)=1:XC(6)=-1:X
C(7)=-1:XC(8)=-1
10310 YC(1)=-1:YC(2)=-1:YC(4)=1:YC(5)=1:
YC(6)=1:YC(8)=-1:RETURN
10400 XB=XA:YB=YA:XX=STICK(XY):IF XX=0 T
HEN RETURN
10410 XA=XA+XC(XX):YA=YA+YC(XX)
10420 IF XA<0 THEN XA=0 ELSE IF XA>39 TH
EN XA=39
10430 IF YA<0 THEN YA=0 ELSE IF YA>23 TH
EN YA=23
10440 LOCATE XB,YB:PRINT" ";:LOCATE XA,Y
A:PRINT"X";:RETURN
```

Appel du sous-programme

Il faut d'abord avoir initialisé la routine en effectuant, dès le début, un GOSUB 10300.

Ensuite, à chaque fois qu'on souhaitera bouger le caractère (ici, un X), définir la manette de jeux (0 ou 1) à affecter à la variable XY, puis :

GOSUB 10400.

Les variables **XA, YA, XB, YB** sont réservées à ce sous-programme.

**Transformation pour le MO5
de l'instruction PEN du TO7**

Les instructions ON PEN GOTO et ON PEN GOSUB existent sur le TO7 et le TO7/7Ø, mais pas sur le MO5. Il est simple de les simuler si on veut adapter sur MO5 un programme provenant d'un TO7. Supposons avoir :

```
3Ø PEN1;(15,32)-(128,63)
4Ø PEN2;(15,64)-(128,95)
5Ø PEN3;(15,96)-(128,127)
```

et en un endroit du programme :

```
23Ø ON PEN GOTO 1ØØØ,12ØØ,14ØØ
24Ø GOTO 23Ø
```

Cette partie de programme ne peut pas fonctionner sur le MO5. Remplacer par :

```
3Ø FORI=1TO3:READX1(I),X2(I),X3(L),X4(I)
:NEXTI
4Ø DATA 15,32,128,63,15,64,128,95,15,96,
128,127
```

Et dans le corps du programme :

```
23Ø INPUTPENX,Y:IF X<Ø THEN 23Ø ELSE XO=
Ø
24Ø FOR I=1 TO 3:XO=XO-I*(X>X1(I) AND X<
X3(I) AND Y>X2(I) AND Y<X4(I)):NEXT I:ON
XO GOTO 1ØØØ,12ØØ,14ØØ
```

**Transformation pour MO5
de PRINT USING alpha du TO7**

Le problème revient à donner une taille déterminée à une chaîne de caractères, soit en rajoutant des blancs au bout, soit en la tronquant.

L'instruction PRINT USING"% %";A$ revient à écrire A$ par la gauche dans une zone fixe de 1Ø caractères. Il suffit de remplacer par :

```
PRINT LEFT$(A$+"  ",1Ø)
```

Si dans un programme à transposer figure un PRINT USING mixte, numérique et alphanumérique :

```
PRINT USING"%  % est âgé de # #ans":XM$;XE
```

remplacer par :

```
PRINT LEFT$(XM$+"  ",1Ø);USING" est âgé de# #ans";XE
```


**Transformation pour le MO5
de l'instruction MID$ du TO7/7Ø**

Le MO5 connaît la fonction MID$ mais pas l'instruction d'affectation MID$.

Remplacer :

```
MID$(XA$,XI,XJ)=XB$
```

par :

```
XA$=LEFT$(XA$,XI-1)+LEFT$(XB$,XJ)+RIGHT$
(XA$,LEN(XA$)-XI-XJ-1+(LEN(XB$)<XJ)*(LEN
(XB$)-XJ))
```

ATTENTION aux fautes de retranscription !

NOMBRES DE DEUX CHIFFRES ECRITS SUR UN SEUL CARACTERE

Dans certains graphiques, il est utile d'indiquer des nombres de manière à utiliser le moins de place possible. Le sous-programme suivant permet d'écrire des nombres à deux chiffres sur un seul caractère.

```
10500 DIM XB%(7),XC%(9,7)
10510 FORXI%=0TO9:FORXJ%=0TO7:READXC%(XI
%,XJ%):NEXT:NEXT:RETURN
10520 DATA 0,14,10,10,10,10,10,14,0,2,6,
10,2,2,2,2,0,14,10,2,4,8,8,14
10530 DATA 0,14,10,2,4,2,10,14,0,2,4,4,8
,14,2,2,0,14,8,8,14,2,2,14
10540 DATA 0,14,10,8,14,10,10,14,0,14,2,
2,4,4,8,8,0,14,10,10,14,10,10,14
10550 DATA 0,14,10,10,14,2,2,14
10600 XD%=XN%\10:XU%=XN%MOD10
10610 FOR XJ%=0TO7:XB%(XJ%)=16*XC%(XD%,X
J%)+XC%(XU%,XJ%):NEXTXJ%
10620 DEFGR$(0)=XB%(0),XB%(1),XB%(2),XB%
(3),XB%(4),XB%(5),XB%(6),XB%(7)
10630 XX$=GR$(0):RETURN
```

Initialisation de la routine

Faire au début du programme un GOSUB 10500.

Appel de ce sous-programme

Définir la variable XN% du nombre à compacter en un seul caractère :

```
XN% < 100
```

Appeler GOSUB 16000.

Retour du sous-programme

La variable XX$ est le nombre une fois compacté.

... et conseils pour les faire disparaître.

Les erreurs sont détectées par le MO5 et leur code est affiché :

- soit en mode direct : Error 2 par exemple ;
- soit en mode programme : Error 2 in 100 par exemple.

Numéro	*Signification*	*Correction éventuelle*
1	Rencontre d'un NEXT sans l'appel FOR qui lui correspond.	
2	Erreur de syntaxe.	Corriger la syntaxe fausse
3	Rencontre d'un RETURN sans le GOSUB qui lui correspond.	Vous avez peut-être oublié le END de fin.
4	Rencontre d'un READ sans le DATA qui lui correspond.	Compléter les DATA.
5	Appel illégal de fonction.	Revoir les conditions d'appel.
6	Dépassement du nombre autorisé.	Revoir le calcul du nombre en cause.
7	Dépassement de la capacité de la mémoire.	Compacter le programme ou vérifier la valeur dans CLEAR.
8	Numéro d'une ligne inexistante.	Changer le numéro d'appel de ligne.
9	Indice de tableau trop grand.	Revoir le calcul de l'indice, le dimensionnement.
10	Tentative illégale de redimensionner.	Ne pas redimensionner. Choisir dès le départ la taille.

Numéro	*Signification*	*Correction éventuelle*
11	Division par zéro.	Tester la nullité du diviseur.
12	Instruction illégale en mode direct.	Ne peut être utilisée qu'en mode programme.
13	Confusion de type de variable.	Ne pas confondre numérique et alphanumérique.
14	Dépassement de la zone-mémoire réservée aux chaînes de caractères.	Voir l'ordre CLEAR.
15	Traitement de chaîne impossible car trop complexe.	Simplifier en plusieurs étapes par exemple.
16	Traitement de calcul impossible car trop complexe.	Simplifier en plusieurs étapes par exemple.
17	Impossible de poursuivre.	Redémarrer le programme.
18	Fonction non prédéfinie.	
19	Instruction RESUME non effectuée après une erreur.	Prévoir une routine de gestion des erreurs comportant RESUME.
2Ø	Rencontre d'un RESUME sans erreur.	Interdire la routine d'erreurs s'il n'y en a pas.
21	Erreur non définie.	
22	Opérande absente.	Faire figurer l'opérande.
23	Rencontre d'un FOR sans le NEXT qui lui correspond.	Mettre le NEXT à un emplacement pertinent.
5Ø	Numéro de fichier illégal.	Utiliser un numéro légal.
51	Mode d'accès au fichier illégal.	Ne pas confondre les fichiers entre eux (binaires-alphanumériques).
52	Tentative d'ouverture d'un fichier déjà ouvert.	Utiliser CLOSE pour fermer ou tester l'ouverture.
53	Erreur de transfert entre le MO5 et un périphérique.	Non résorbable.
54	Tentative de lire dans un fichier déjà lu en entier.	Utiliser EOF pour tester la fin d'un fichier.

Numéro	*Signification*	*Correction éventuelle*
55	Mauvais nom de fichier.	Revoir les syntaxes possibles d'un nom de fichier.
56	Entrée directe dans le fichier.	Provient souvent d'une erreur de fichier (programme/données).
57	Fichier non ouvert.	Ouvrir !
58	Données mal écrites dans un fichier.	Non résorbable.
59	Périphérique déjà occupé.	Non résorbable. Recommencer après l'avoir libéré.
6Ø	Périphérique absent ou hors d'usage.	Brancher ou changer.
61	Programme protégé.	Pas de LIST, SAVE, d'ajout ou de modif de ligne.

LES REGISTRES DU 6809

Le 6809 possède neuf registres internes, quatre sur 8 bits, six sur 16 bits.

8 bits 8 bits	Accumulateur A Accumulateur B Double accumulateur D (16 bits)
16 bits	Registre d'index X.
16 bits	Registre d'index Y.
16 bits	Pointeur de pile système S (System)
16 bits	Pointeur de pile utilisateur U (User)
16 bits	Compteur-programme PC (Program Counter)
8 bits	Registre de page DP (Direct Page)
8 bits	Registre d'état CC (Code Condition)

Description des registres

Accumulateurs A et B : ce sont deux registres 8 bits. Ils ont la particularité de pouvoir être associés en un registre 16 bits appelé D. Plusieurs instructions du 6809 permettent de travailler directement sur D.

D est composé avec A pour ses 8 bits de plus fort poids, avec B pour ses 8 bits de plus faible poids.

Registres d'index X et Y : ce sont des registres d'adresse. Les codes d'instructions concernant X sont souvent plus courts et leur déroulement plus rapide.

Pointeurs de pile S et U : ce sont deux registres 16 bits. S est utilisé par le système pour gérer son fonctionnement interne. U peut être utilisé par le programmeur. Il est ignoré du processeur. Préférer PULU et PSHU à PULS et PSHS pour les sauvegardes et restaurations, avant et après une routine.

Compteur programme PC : il indique, de manière permanente, la prochaine adresse mémoire devant être lue. Il peut donc être

considéré comme registre d'index, à ceci près que son contenu ne cesse de varier avec le déroulement du programme.

Registre de page directe : l'adressage du 6809 permet de raccourcir les cycles-machine et l'écriture des programmes, si on le restreint à une page-mémoire (256 octets). Ce registre pointe la page-mémoire en cours.

Registre d'état : c'est un registre de 8 bits où les 8 bits ont une signification propre :

b0 : retenue ;
b1 : dépassement ;
b2 : zéro ;
b3 : négatif ;
b4 : masque d'IRQ ;
b5 : demi-retenue ;
b6 : masque de FIRQ ;
b7 : drapeau de sauvegarde.

Le classement des instructions du 6809 se fait en quatre groupes :

- instructions sur A, B et contenus-mémoires 8 bits ;
- instructions sur D et contenus-mémoires 16 bits ;
- instructions sur les registres purement 16 bits X, Y, S, U ;
- instructions particulières.

Les instructions peuvent s'écrire sur un nombre d'octets allant de 1 à 5 !

Adressage étendu

On précise en totalité la mémoire contenant l'adresse effective et aucun signe particulier ne suit le mnémonique dans le langage assembleur standard.

ADDD $A200 signifie : ajouter à D le contenu de la mémoire d'adresse A200.

Ne pas confondre avec :

```
ADDD #$A200
```

qui ajoute directement à D la valeur A200 hexa.

Adressage étendu indirect

La mémoire indiquée n'est pas celle où l'on cherche à ranger l'adresse effective. Par contre, la mémoire indiquée contient, sur deux octets, l'adresse de la mémoire où figure l'adresse effective.

Ce procédé permet de créer des tables d'adresse. L'intérêt est de ne changer que le contenu de la table d'adresse et non le programme lui-même quand une adresse change. La notation assembleur standard indique l'adresse indirecte entre crochets.

ADDD ($A200) Si A200-A201 contient B0A6, cette instruction est identique à :

```
ADDB $B0A6
```

et si B0A6 contient F4 hexa, elle est identique à :

```
ADDB #$F4.
```

On peut utiliser ce mode pour permettre par exemple de modifier simplement un programme MO5 en TO7, car les adresses des registres sont différentes, ainsi que celles de la mémoire d'écran ...

Adressage direct

Cet adressage fait intervenir la page de base pointée par le registre Direct-Page DP. Avec un assembleur standard, on écrira :

ADDB $A6 par exemple. Si le registre de page n'a pas été changé depuis l'initialisation, il contient 0. Cette instruction sera donc semblable à :

```
ADDB $00A6.
```


Pour changer le registre de page, il n'y a pas d'instruction directe de chargement de DP. Il faut donc charger A ou B, puis transférer le contenu dans DP. Une bonne précaution consiste à empiler d'abord le registre intermédiaire pour le restaurer ensuite.

```
PSHU A      : sauvegarder le registre intermédiaire.
LDA #$A0    : charger avec la page souhaitée.
TRF A,DP    : transférer dans le registre de page.
PULU A      : restaurer le registre intermédiaire.
```

On prendra comme registre intermédiaire un registre 8 bits.

Adressage indexé

Une adresse dite de base est contenue dans un registre (X,Y,S,U). Il suffit d'indiquer alors le décalage entre l'adresse contenue dans le registre de base et l'adresse à atteindre effectivement.

ADDB 12,Y : ajoute à B le contenu de l'adresse obtenue en ajoutant 12 au contenu de Y.

Adressage indexé par accumulateur

Comme le précédent, mais le décalage est donné par le contenu de l'accumulateur spécifié.

ADDB A,Y : ajoute à B le contenu de l'adresse obtenue en ajoutant le contenu de A à celui de Y.

Adressage indexé avec incrémentation ou décrémentation

Un index est spécifié, comme à l'adressage indexé et à l'adressage indexé par accumulateur, mais de plus on fait se mouvoir l'index de 1 ou 2 vers le haut ou vers le bas. Ceci est très utile pour la lecture de tableaux.

ADDB 5,-Y : ajoute à B le contenu de l'adresse obtenue en ajoutant 5 à l'adresse contenue dans Y après un décalage de Y de 1 vers le bas de la mémoire.

ADDB 5,Y- : idem, mais le décalage de Y se fait après l'exécution.

ADDB 5,+Y : idem, mais le décalage se fait vers le haut avant l'exécution.

Remarque : On peut utiliser ++ ou -- pour un décalage de deux octets.

Adressage inhérent

Il s'agit d'un mode d'adressage interne au processeur, donc extrêmement rapide et concis. Il concerne un ou des registres dont le nom figure en op. code et non dans le champ opérande. Ce sont, par exemple :

CLRA pour CLEAR A ;
INCB pour INCrémenter B ;
etc.

Adressage par registre

Il s'agit encore d'un adressage interne au processeur, mais avec spécification en opérande du ou des registres concernés. Par exemple :

EXG X,Y pour EXchanGe X et Y.

Le même mnémonique EXG pouvant servir pour n'importe quelle autre association de registres 16 bits.

De même pour PSHS ou PULU, etc.

Adressage immédiat

Il s'agit ici de mettre en oeuvre une donnée 8 ou 16 bits dont on déclare la valeur en clair. En assembleur standard, le symbole # sépare l'op-code de l'opérande.

Par exemple :

ADDA #$A7 pour ajouter à A la valeur A7 hexa

ou

LDX #$12AØ pour charger X avec la valeur 12AØ hexa.

etc.

Notation : M et Mémoire concernent des mémoires sur 8 bits.
MM et MMémoire concernent des mots de 16 bits.

Sens des mnémoniques

Instructions de branchement

Mnémoniques *Court*	*Long*	*Condition*	
BCC	LBCC	C=Ø	
BCS	LBCS	C=1	Le branchement court se fait dans la page-mémoire pointée par le registre de page DP.
BEQ	LBEQ	Z=Ø	
BGE	LBGE	N+V=Ø	
BGT	LBGT	Z+(N+V)=Ø	
BHI	LBHI	Z+C=Ø	Le branchement long se fait dans la totalité de l'espace adressable.
BHS	LBHS	C=Ø	
BLE	LBLE	Z+(N+V)=1	
BLO	LBLO	C=1	
BLS	LBLS	Z+C=1	
BLT	LBLT	N+V=1	
BMI	LBMI	N=1	
BNE	LBNE	Z=1	
BPL	LBPL	N=Ø	
BRA	LBRA		
BRN	LBRN	BRN=NOP,NOP	LBRN=NOP,NOP,NOP,NOP
BSR	LBSR		
BVC	LBVC	V=Ø	
BVS	LBVS	V=1	

Instructions inconditionnelles

ABX Addition non signée entre B et X : <B> + <X> vers X.

ADCA Addition avec CARRY Mémoire et A : <A> + <M> + <C> vers A.

ADCB Addition avec CARRY Mémoire et B : <B> + <M> + <C> vers B.

ADDA Addition sans CARRY Mémoire et A : <A> + <M> vers A.

ADDB Addition sans CARRY Mémoire et B : <B> + <M> vers B.

ADDD Addition sans CARRY MMémoire et D : <D> + <MM> vers D.

ANDA Et logique entre A et Mémoire : <A> ET <M> vers A.

ANDB Et logique entre B et Mémoire : <B> ET <M> vers B.

ANDCC Et logique entre CC et Mémoire : <CC> ET <M> vers CC.

ASL Décalage à gauche avec bit 7 vers C.
ASLA Décalage à gauche avec bit 7 vers C.
ASLB Décalage à gauche avec bit 7 vers C.
ASR Décalage à droite avec bit Ø vers C.
ASRA Décalage à droite avec bit Ø vers C.
ASRB Décalage à droite avec bit Ø vers C.
BITA Test bit à bit de A et Mémoire.
BITB Test bit à bit de B et Mémoire.
CLR Mise à Ø de la Mémoire.
CLRA Mise à Ø de A.
CLRB Mise à Ø de B.
CMPA Comparaison de A et Mémoire/retour CC.
CMPB Comparaison de B et Mémoire/retour CC.
CMPD Comparaison de D et MMémoire.
CMPS Comparaison de S et MMémoire.
CMPU Comparaison de U et MMémoire.
CMPX Comparaison de X et MMémoire.
CMPY Comparaison de Y et MMémoire.
COM Mise à complément de Mémoire.
COMA Mise à complément de A.
COMB Mise à complément de B.
DAA Ajustement décimal de A.
DEC Décrémentation de la Mémoire.
DECA Décrémentation de A.
DECB Décrémentation de B.
EORA OU exclusif entre A et M/retour dans A.
EORB OU exclusif entre B et M/retour dans B.
EXG Permutation entre deux registres de même taille.
INC Incrémentation de Mémoire.
INCA Incrémentation de A.
INCB Incrémentation de B.
JMP Saut sans condition vers une adresse.
JSR Saut sans condition à un sous-programme.

LDA Chargement de A, direct ou venant d'une Mémoire.
LDB Chargement de B, direct ou venant d'une Mémoire.
LDD Chargement de D, direct ou venant d'une MMémoire.
LDS Chargement de S, direct ou venant d'une MMémoire.
LDU Chargement de U, direct ou venant d'une MMémoire.
LDX Chargement de X, direct ou venant d'une MMémoire.
LDY Chargement de Y, direct ou venant d'une MMémoire.
LEAS Chargement de l'adresse effective de S en mode indexé.
LEAU Chargement de l'adresse effective de U en mode indexé.
LEAX Chargement de l'adresse effective de X en mode indexé.
LEAY Chargement de l'adresse effective de Y en mode indexé.
LSL Voir ASL.
LSLA Voir ASLA.
LSLB Voir ASLB.
LSR Décalage logique de Mémoire à droite.
LSRA Décalage logique de A à droite.
LSRB Décalage logique de B à droite.
MUL Multiplication non signée de A et B/retour dans D.
NEG Mise de mémoire à complément à 2.
NEGA Mise de A à complément de 2.
NEGB Mise de B à complément de 2.
NOP Pas d'opération.
ORA OU logique entre A et Mémoire/retour dans A.
ORB OU logique entre B et Mémoire/retour dans B.
ORCC OU logique entre CC et Mémoire/retour dans CC.
PSHS Empilement dans S.
PSHU Empilement dans U.
PULS Dépilement de S.
PULU Dépilement de U.
ROL Permutation circulaire à gauche de la Mémoire.
ROLA Permutation circulaire à gauche de A.
ROLB Permutation circulaire à gauche de B.
ROR Permutation circulaire à droite de la Mémoire.

ROA Permutation circulaire à droite de A.

ROB Permutation circulaire à droite de B.

SBCA Soustraction avec CARRY <A> - <M> - <C> vers A.

SBCB Soustraction avec CARRY <B> - <M> - <C> vers B.

SEX Extension de signe : A mis à FF si bit 7 de B = 1.
A mis à ØØ si bit 7 de B = Ø

STA Stockage de A en Mémoire.

STB Stockage de B en Mémoire.

STD Stockage de D en MMémoire.

STS Stockage de S en MMémoire.

STU Stockage de U en MMémoire.

STX Stockage de X en MMémoire.

STY Stockage de Y en MMémoire.

SUBA Soustraction sans CARRY <A> - <M> vers A.

SUBB Soustraction sans CARRY <B> - <M> vers B.

SUBD Soustraction sans CARRY <D> - <MM> vers D.

SWI Interruption 'soft' n° 1.

SWI2 Interruption 'soft' n° 2.

SWI3 Interruption 'soft' n° 3.

SYNC Synchro d'interruption.

TFR Transfert du premier registre vers le second (même taille).

TST Test de M.

TSTA Test de A.

TSTB Test de B.

Codes machine des mnémoniques

Op représente l'Op-code.
" représente le nombre de cycles-machine.
= représente le nombre d'octets que l'instruction occupe en mémoire.

Instructions de branchement

	Relatif				*Relatif*		
Mném.	Op	"	=	*Mném.*	Op	"	=
BCC	24	3	2	BLS	23	3	2
LBCC	1Ø	5/6	4	LBLS	1Ø	5/6	4
	24				23		
BCS	25	3	2	BLT	2D	3	2
LBCS	1Ø	5/6	4	LBLT	1Ø	5/6	4
	25				2D		
BEQ	27	3	2	BMI	2B	3	2
LBEQ	1Ø	5/6	4	LBMI	1Ø	5/6	4
	27				2B		
BGE	2C	3	2	BNE	26	3	2
LBGE	1Ø	5/6	4	LBNE	1Ø	5/6	4
	2C				26		
BGT	2E	3	2	BPL	2A	3	3
LBGT	1Ø	5/6	4	LBPL	1Ø	5/6	4
	2E				2A		
BHI	22	3	2	BRA	2Ø	3	2
LBHI	1Ø	5/6	4	LBRA	16	5	3
	22			BRN	21	3	2
BHS	24	3	2	LBRN	1Ø	5	4
LBHS	1Ø	5/6	4		21		
	24			BSR	8D	7	2
BLE	2F	3	2	LBSR	17	9	3
LBLE	1Ø	5/6	4	BVC	28	3	2
	2F			LBVC	1Ø	5/6	4
BLO	25	3	2		28		
LBLO	1Ø	5/6	4	BVS	29	3	2
	25			LBVS	1Ø	5/6	4
					29		

Instructions inconditionnelles

Op représente l'Op-code.
~ représente le nombre de cycles-machine.
= représente le nombre d'octets que l'instruction occupe en mémoire.
+ il faut ajouter les octets occupés par l'adresse d'indexation.

Pour les registres : I signifie Non affecté.
V signifie Mis à 1 si vrai.

Mnémonique	Inhérent			Immédiat			Etendu			Direct			Indexé			Registres				
	Op	~	=	Op	~	=	Op	~	=	Op	~	=	Op	~	=	H	N	Z	V	C
ABX	3A	3	1													I	I	I	I	I
ADCA				89	2	2	B9	5	3	99	4	2	A9	4+	2+	V	V	V	V	V
ADCB				C9	2	2	F9	5	3	D9	4	2	E9	4+	2+	V	V	V	V	V
ADDA				8B	2	2	BB	5	3	9B	4	2	AB	4+	2+	V	V	V	V	V
ADDB				CB	2	2	FB	5	3	DB	4	2	EB	4+	2+	V	V	V	V	V
ADDD				C3	4	3	F3	7	3	D3	6	2	E3	6+	2+	I	V	V	V	V
ANDA				84	2	2	B4	5	3	94	4	2	A4	4+	2+	I	V	V		I
ANDB				C4	2	2	F4	5	3	D4	4	2	E4	4+	2+	I	V	V		I
ANDCC				1C	3	2														
ASL							78	7	3	08	6	2	68	6+	2+	U	V	V	V	V
ASLA	48	2	1													U	V	V	V	V
ASLB	58	2	1													U	V	V	V	V
ASR							77	7	3	07	6	2	67	6+	2+		V	V	I	V
ASRA	47	2	1														V	V	I	V
ASRB	57	2	1														V	V	I	V
BITA				85	2	2	B5	5	3	95	4	2	A5	4+	2+	I	V	V		I
BITB				C5	2	2	F5	5	3	D5	4	2	E5	4+	2+	I	V	V		I

Mnémonique	Inhérent			Immédiat			Etendu			Direct			Indexé			Registres				
	Op	~	=	Op	~	=	Op	~	=	Op	~	=	Op	~	=	H	N	Z	V	C
CLR							7F	7	3	0F	6	2	6F	6+	2+	I		V		
CLRA	4F	2	1													I		V		
CLRB	5F	2	1													I		V		
CMPA				81	2	2	B1	5	3	91	4	2	A1	4+	2+		V	V	V	V
CMPB				C1	2	2	F1	5	3	D1	4	2	E1	4+	2+		V	V	V	V
CMPD				10 83	5	4	10 B3	8	4	10 93	7	3	10 A3	7+	3+	I	V	V	V	V
CMPS				11 8C	5	4	11 BC	8	4	11 9C	7	3	11 AC	7+	3+	I	V	V	V	V
CMPU				11 83	5	4	11 B3	8	4	11 93	7	3	11 A3	7+	3+	I	V	V	V	V
CMPX				8C	4	3	BC	7	3	9C	6	2	AC	6+	2+	I	V	V	V	V
CMPY				10 8C	5	4	10 BC	8	4	10 9C	7	3	10 AC	7+	3+	I	V	V	V	V
COM							73	7	3	03	6	2	63	6+	2+		V	V		V
COMA	43	2	1														V	V		V
COMB	53	2	1														V	V		V
DAA	19	2	1													I	V	V		V
DEC							7A	7	3	0A	6	2	6A	6+	2+	I	V	V	V	I
DECA	4A	2	1													I	V	V	V	I
DECB	5A	2	1													I	V	V	V	I
EORA				88	2	2	B8	5	3	98	4	2	A8	4+	2+	I	V	V		I
EORB				C8	2	2	F8	5	3	D8	4	2	E8	4+	2+	I	V	V		I

Mnémo-nique	Inhérent Op	~	=	Immédiat Op	~	=	Etendu Op	~	=	Direct Op	~	=	Indexé Op	~	=	Registres H	N	Z	V	C
EXG	1E --*	8	2													I	I	I	I	I
INC							7C	7	3	ØC	6	2	6C	6+	2+	I	V	V	V	I
INCA	4C	2	1													I	V	V	V	I
INCB	5C	2	1													I	V	V	V	I
JMP							7E	4	3	ØE	3	2	6E	3+	2+	I	I	I	I	I
JSR							BD	8	3	9D	7	2	AD	7+	2+	I	I	I	I	I
LDA				86	2	2	B6	5	3	96	4	2	A6	4+	2+	I	V	V		I
LDB				C6	2	2	F6	5	3	D6	4	2	E6	4+	2+	I	V	V		I
LDD				CC	3	3	FC	6	3	DC	5	2	EC	5+	2+	I	V	V		I
LDS				1Ø	4	4	1Ø	7	4	1Ø	6	3	1Ø	6+	3+	I	V	V		I
				CE			FE			DE			EE							
LDU				CE	3	3	FE	6	3	DE	6	3	EE	6+	3+	I	V	V		I
LDX				8E	3	3	BE	6	3	9E	5	2	AE	5+	2+	I	V	V		I
LDY				1Ø	4	4	1Ø	7	4	1Ø	6	3	1Ø	6+	3+	I	V	V		I
				8E			BE			9E			AE							
LEAS													32	4+	2+	I	I	I	I	I
LEAU													33	4+	2+	I	I	I	I	I
LEAX													3Ø	4+	2+	I	I	V	I	I
LEAY													31	4+	2+	I	I	V	I	I

--* Cette instruction concerne deux registres de même type, 8 bits ou 16 bits. Former le post-op-code avec les codes des deux registres qui interviennent, en conservant l'ordre A=8 B=9 CC=A D=Ø DP=B PC=5 S=4 U=3 X=1 Y=2.

Exemple : 89 pour A,B et Ø2 pour D,Y.

Mnémo-nique	Inhérent Op	~	=	Immédiat Op	~	=	Etendu Op	~	=	Direct Op	~	=	Indexé Op	~	=	Registres H	N	Z	V	C
LSL							78	7	3	Ø8	6	2	68	6+	2+	I	V	V	V	V
LSLA	48	2	1													I	V	V	V	V
LSLB	58	2	1													I	V	V	V	V
LSR							74	7	3	Ø4	6	2	64	6+	2+	I		V	I	V
LSRA	44	2	1													I		V	I	V
LSRB	54	2	1													I		V	I	V
MUL	3D	11	1													I	I	V	I	
NEG							7Ø	7	3	ØØ	6	2	6Ø	6+	2+		V	V	V	V
NEGA	4Ø	2	1														V	V	V	V
NEGB	5Ø	2	1														V	V	V	V
NOP	12	2	1													I	I	I	I	I
ORA				8A	2	2	BA	5	3	9A	4	2	AA	4+	2+	I	V	V		I
ORB				CA	2	2	FA	5	3	DA	4	2	EA	4+	2+	I	V	V		I
ORCC				1A	3	2														
PSHS	34 --*	5+	2													I	I	I	I	I
PSHU	36 --*	5+	2													I	I	I	I	I
PULS	35 --*	5+	2													I	I	I	I	I

--* Le post-op-code s'obtient en composant un octet avec 8 bits, chacun donnant la présence ou l'absence du registre concerné.

Bits :	7	6	5	4	3	2	1	Ø
	PC	U/S	Y	X	DP	B	A	CC

Mnémo-nique	Inhérent Op	~	=	Immédiat Op	~	=	Etendu Op	~	=	Direct Op	~	=	Indexé Op	~	=	Registres H	N	Z	V	C
PULU	37 --*	5+	2													I	I	I	I	I
ROL							79	7	3	09	6	2	69	6+	2+	I	V	V	V	V
ROLA	49	2	1													I	V	V	V	V
ROLB	59	2	1													I	V	V	V	V
ROR							76	7	3	06	6	2	66	6+	2+	I	V	V	I	V
RORA	46	2	1													I	V	V	I	V
RORB	56	2	1													I	V	V	I	V
RTI	3B	6	1	Si E=0																
		15	1	Si E=1																
RTS	39	5	1													I	I	I	I	I
SBCA				82	2	2	B2	5	3	92	4	2	A2	4+	2+		V	V	V	V
SBCB				C2	2	2	F2	5	3	D2	4	2	E2	4+	2+		V	V	V	V
SEX	1D	2	1													I	V	V		I
STA							87	5	3	97	4	2	A7	4+	2+	I	V	V		I
STB							F7	5	3	D7	4	2	E7	4+	2+	I	V	V		I
STD							FD	6	3	DD	5	2	ED	5+	2+	I	V	V		I
STS							10 FF	7	4	10 DF	6	3	10 EF	6+	3+	I	V	V		I

--* Le post-op-code s'obtient en composant un octet avec 8 bits, chacun donnant la présence ou l'absence du registre concerné.

Bits :	7	6	5	4	3	2	1	0
	PC	U/S	Y	X	DP	B	A	CC

Mnémo-nique	Inhérent Op	~	=	Immédiat Op	~	=	Etendu Op	~	=	Direct Op	~	=	Indexé Op	~	=	Registres H	N	Z	V	C
STU							FF	6	3	DF	5	2	EF	5+	2+	I	V	V		I
STX							BF	6	3	9F	5	2	AF	5+	2+	I	V	V		I
STY							10 BF	7	4	10 9F	6	3	10 AF	6+	3+	I	V	V		I
SUBA				80	2	2	80	5	3	90	4	2	A0	4+	2+		V	V	V	V
SUBB				C0	2	2	F0	5	3	D0	4	2	E0	4+	2+		V	V	V	V
SUBD				83	4	3	83	7	3	93	6	2	A3	6+	2+	I	V	V	V	V
SWI	3F	19	1													I	I	I	I	I
SWI2	10 3F	20	2													I	I	I	I	I
SWI3	11 3F	20	2													I	I	I	I	I
SYNC	13	2	1													I	I	I	I	I
TFR	1F --*	7	2													I	I	I	I	I
TST							7D	7	3	0D	6	2	6D	6+	2+	I	V	V		I
TSTA	4D	2	1													I	V	V		I
TSTB	5D	2	1													I	V	V		I

--* Cette instruction concerne deux registres de même type, deux 8 bits ou deux 16 bits. Former le post-op-code avec les codes des deux registres qui interviennent, en conservant l'ordre : A=8 B=9 CC=A D=0 DP=8 PC=5 S=4 U=3 X=1 Y=2
Exemple : 89 pour A,B et 02 pour D,Y.

ORGANISATION GENERALE DE LA MEMOIRE

ADRESSES en HEXA

0000	Mémoire d'écran	1FFF
2000	Registres du moniteur	20FF
2100	Registre de l'application	21FF
2200	Mémoire utilisateur	9FFF
A000	DOS si disquettes	A7BF
A7C0	PIA 6821	A7C3
A7C4	Libre	A7CB
A7CC	PIA 6821 Jeux manettes	A7CF
A7D0	Contrôleur de disquettes	A7DF
A7E0	PIA 6821 interface communication	A7E3
A7E4	Compteurs crayon optique	A7E7
A7E8	Extensions	A7FF
A800	Libre	AFFF
8000	Cartouche ROM	EFFF
F000	Moniteur	FFFF

L'entrée au MONITEUR du MO5 se fait de manière normalisée : par un SWI suivi du code de la fonction MONITEUR que l'on souhaite appeler. Il y a deux codes différents par fonction exécutable. Ils diffèrent par le mode du retour après exécution de la fonction. Le premier correspond à un JSR, donc à l'appel d'une sous-routine, le second correspond à un JMP, donc à un saut à la routine.

Pour JSR, le retour se fait à l'instruction suivant l'appel.

Pour JMP, le retour se fait à l'instruction suivant celle qui a appelé le SWI. Il y a donc sortie des fonctions moniteur.

Le SWI sauvegarde tous les registres du 6809 et les restaure au retour. Seuls, sont changés le registre code condition et les registres contenant les paramètres de retour

Codes des fonctions MONITEUR

Ces codes sont ceux à passer après un SWI. Sont indiqués les codes de branchement direct et les codes de branchement à sous-routine.

Numéro	*JSR*	*JMP*	
00	02	82	Affichage d'un caractère.
01	04	84	Mise en mémoire couleur des points.
02	06	86	Mise en mémoire de caractères.
03	08	88	Bip sonore.
04	0A	8A	Lecture du clavier.
05	0C	8C	Lecture rapide du clavier.
06	0E	8E	Tracé d'une ligne.
07	10	90	Allumage d'un point.
08	12	92	Ecriture d'un point 'caractère'.
09	14	94	Lecture de la couleur d'un point.
10	16	96	Lecture du bouton du crayon optique.
11	18	98	Lecture du crayon optique.
12	1A	9A	Lecture de l'écran.
13	1C	9C	Lecture des manettes de jeu.
14	1E	9E	Emission de musique.
15	20	A0	Lecture/écriture sur la cassette.

Numéro	*JSR*	*JMP*	
16	22	A2	Mise en route et arrêt du moteur LEP.
17	24	A4	Interface de communication.
18	26	A6	Contrôleur de disques.

Les autres codes sont réservés à l'usage interne du moniteur.

Affichage d'un caractère

On peut envoyer à l'écran des codes affichables, comme des caractères ou des codes interprétables, comme les déplacements de curseur.

- Charger l'accumulateur B avec le code du caractère.
- Appeler la routine Ø2 pour JSR et 82 pour JMP.

Voir par ailleurs les codes interprétables (code de fonction Ø-1F) et les séquences ESPACE et ESCAPE pour les positionnements, définition de fenêtre, de couleur de forme, de fond, de tour, de taille caractère, etc.

Mise en mémoire couleur

La page écran est sélectionnée par cette routine en définition de couleur forme/fond.

- Appeler la routine Ø4 pour un JSR ou 84 pour un JMP.

Ce qui sera inscrit dans la mémoire-écran (Ø-1FFF) modifiera la couleur des points du segment changé.

Mise en mémoire caractère

La page-écran est sélectionnée par cette routine en écriture forme/fond.

- Appeler la routine Ø6 pour un JSR ou 86 pour un JMP.

Ce qui sera inscrit dans la mémoire-écran (Ø-1FFF) modifiera la disposition des points forme/fond du segment changé.

Bip sonore

Cette routine sert à émettre un BIP dans le haut-parleur du téléviseur raccordé au MO5.

- Mettre à Ø le bit 3 du registre STATUS en 2Ø19 hexa.
- Appeler la routine Ø8 pour un JSR ou 88 pour un JMP.

Décodage du clavier

Cette routine décode une touche du clavier enfoncée et sa répétition en cas de prolongement de l'appui. Voir en page 110 le fonctionnement de la latence du clavier et le pointeur de table des codes engendrés par les touches. Pour décoder :

- Appeler la routine ØA pour un JSR et 8A pour un JMP.

En retour, l'accumulateur B du 6809 contiendra le code ASCII du caractère. Si la touche BASIC a été enfoncée, B contiendra le code matriciel de la touche enfoncée.

Remarque 1 : les caractères accentués sont écrits sur trois octets. Il faudra donc, pour lire un accent, faire un test d'enfoncement de la touche de code matriciel 83 <ACC>.

Remarque 2 : la touche SHIFT ne provoque pas de retour de routine. Par contre, quand on l'active en même temps qu'une autre touche, le code ressorti est celui du caractère situé en haut de la touche ou de la minuscule de la lettre indiquée.

Remarque 3 : l'enfoncement de la touche <CNT> ne provoque pas de retour de routine. Par contre, si en même temps on enfonce la touche d'un caractère affichable, le code retourné est celui du caractère - 64.

Remarque 4 : la routine ne détecte pas l'enfoncement de la touche <BASIC> ; par contre, en cas d'enfoncement simultané, elle retourne dans B des codes différents. Ils sont donnés dans le tableau ci-dessous.

Code (en hexa) retourné par la lecture de clavier, en cas d'enfoncement simultané de la touche BASIC .

8Ø	9Ø ←	AØ →	BØ INS
81	91 X	A1 @	B1 ,
82 STOP	92 2	A2 4	B2 6
83 ACC	93 -	A3 9	B3 7
84	94 Z	A4 R	B4 Y
85 ENTREE	95 /	A5 O	B5 U
86 RAZ	96 S	A6 F	B6 H
87 C	97 B	A7 L	B7 J
88 ↑	98 ↓	AB ⇆	B8 EFF
89 W	99 SPACE BAR	A9 .	B9 N
8A 1	9A 3	AA 5	
8B +	9B Ø	AB 8	
8C A	9C E	AC T	
8D *	9D P	AD I	
8E Q	9E D	AE G	
8F V	9F M	AF K	

Lecture rapide du clavier

Cette routine lit rapidement si une touche est enfoncée ou non.

- Appeler la routine ØC hexa pour un JSR et 8C hexa pour un JMP.
- Lire le bit Z du registre CC. S'il est à 1, aucun appui. S'il est à Ø, une touche est appuyée.
- Si Z est à Ø, lire l'accumulateur B. Il contient le code de la touche.
- Le bit Ø de A correspond à la touche BASIC.
- Le bit 1 de A correspond à la touche CNT.
- Le bit 2 de A correspond à la touche SHIFT.

Si un de ces trois bits est à 1, la touche correspondante est enfoncée.

Tracé d'un segment de points

Un segment de points peut être affiché de deux manières : en mode graphique ou caractère. Voir l'instruction BASIC : LINE.

En mode graphique

- Passer dans X l'abscisse de l'extrémité du segment à allumer.
- Passer dans Y l'ordonnée de l'extrémité du segment à allumer.
- Le registre FORME (2Ø29 hexa) contient le code de couleur d'affichage.
- Mettre Ø dans le registre CHDRAW (2Ø36 hexa).
- Mettre dans PLOTX (2Ø32-3Ø33 hexa) l'abscisse de l'origine du segment.
- Mettre dans PLOTY (2Ø34-2Ø35 hexa) l'ordonnée de l'origine du segment.

Remarque : ces registres contiennent les coordonnées du dernier point affiché. Si l'on souhaite tracer une ligne à partir du dernier point allumé, il convient donc de ne pas modifier PLOTX et PLOTY.

- Appeler la routine ØE hexa pour un JSR ou 8E hexa pour un JMP.

En mode caractère

- Passer dans X l'abscisse de l'extrémité du segment à afficher.
- Passer dans Y l'ordonnée de l'extrémité du segment à afficher.
- Le registre COLOUR (2Ø2B hexa) contient la couleur de fond et de forme.
- Mettre dans CHDRAW (2Ø36 hexa) le code du caractère à afficher.
- Mettre dans PLOTX (2Ø32-3Ø33 hexa) l'abscisse de l'origine du segment.
- Mettre dans PLOTY (2Ø34-2Ø35 hexa) l'ordonnée de l'origine du segment.

Remarque : ces registres contiennent les coordonnées du dernier caractère affiché. Si l'on souhaite tracer une ligne à partir du dernier caractère affiché graphiquement, il convient donc de ne pas modifier PLOTX et PLOTY.

- Appeler la routine ØE hexa pour un JSR ou 8E hexa pour un JMP.

Allumage d'un point

Un point peut être affiché de deux manières : en mode graphique ou caractère. Voir les instructions BASIC PSET, LINE, BOX.

En mode graphique

- Passer dans X l'abscisse du point à allumer.
- Passer dans Y l'ordonnée du point à allumer.
- Le registre FORME (2Ø29 hexa) contient le code de couleur d'affichage.
- Mettre Ø dans le registre CHDRAW (2Ø36 hexa).
- Appeler la routine 1Ø hexa pour un JSR ou 9Ø hexa pour un JMP.

En mode caractère

- Passer dans X l'abscisse du caractère à afficher.
- Passer dans Y l'ordonnée du caractère à afficher.
- Le registre COLOUR (2Ø2B hexa) contient la couleur de fond et de forme.

- Mettre dans CHDRAW (2Ø36 hexa) le code du caractère à afficher.
- Appeler la routine 1Ø hexa pour un JSR ou 9Ø hexa pour un JMP.

Ecriture d'un point 'caractère'

- Appeler la routine 12 hexa pour un JSR et 92 hexa pour un JMP.

Lecture de la couleur d'un point

Cette routine retourne le code couleur d'un point graphique.

- Passer dans X l'abscisse du point à tester.
- Passer dans Y l'ordonnée du point à tester.
- Appeler la routine 14 hexa pour un JSR ou 94 hexa pour un JMP.

Voir les codes couleur page 139.

Lecture du bouton du crayon optique

Permet de savoir si le bouton caoutchouc du crayon optique a été enfoncé ou non.

- Appeler la routine 16 hexa pour un JSR ou 96 hexa pour un JMP.

On retrouve, dans le bit de retenue de CC 1 si le bouton est enfoncé, Ø s'il ne l'a pas été.

Lecture du crayon optique

Cette routine retourne la position du crayon optique sur l'écran, indépendamment du fait que son bouton soit enfoncé ou non.

- Appeler la routine 18 hexa pour un JSR et 98 hexa pour un JMP.

Lire le bit de retenue de CC. Si l'on y trouve 1, le crayon n'est pas sur l'écran ou celui-ci n'est pas assez lumineux ; si l'on trouve Ø dans CC, le crayon est bien pointé.

Lire alors les registres X et Y dans lesquels on retrouve les coordonnées du point visé.

Lecture de l'écran

Cette routine retourne un code pour une case d'écran que l'on souhaite tester.

- Passer dans X l'abscisse de la case à tester.
- Passer dans Y l'ordonnée de la case à tester.
- Appeler la routine 1A hexa pour un JSR ou 9A hexa pour un JMP.

On retrouve dans B le code du caractère figurant dans la case X,Y. S'il n'y a pas de caractère reconnu (graphique ou modifié par un tracé), on retrouve Ø dans B.

Remarque : voir page 125, une méthode pour arriver à lire les caractères graphiques à l'écran.

Rappel : un caractère accentué est composé de l'empilement de trois caractères : le caractère 22 décimal, l'accent, le caractère de base.

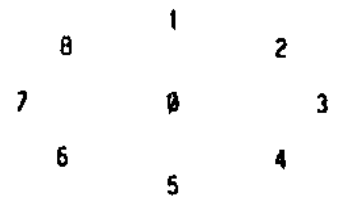
Donc, si on lit 22 décimal dans B après l'appel de la routine, rappeler pour lire le code de l'accent, rappeler enfin pour lire le code de la lettre.

Lecture des manettes de jeu

Les manettes de jeu sont gérées par le MONITEUR à l'aide d'une routine : celle de lecture des manettes.

Pour lire les manettes :

- le numéro de la manette de jeu questionnée doit être passé dans l'accumulateur A ;
- appeler la routine 1C hexa pour un JSR ou 9C hexa pour un JMP ;
- lire dans B la valeur retournée, comme par STICK(du Basic :



- lire le bit de retenue du registre CC pour savoir si le bouton de ladite manette a été enfoncé (1) ou non (Ø).

Génération de musique

Cette routine fait jouer des suites de notes dans le haut-parleur de la TV connectée au MO5.

- Passer dans le registre TEMPO (203A hexa) le tempo souhaité.
- Passer dans le registre DUREE (203C hexa) la durée souhaitée.
- Passer dans le registre TIMBRE (203D hexa) le timbre souhaité.
- Passer dans le registre OCTAVE (203E-203F hexa) l'octave voulue.
- Passer dans B le code de la note devant être jouée.

Voir page 140, les différents codes musicaux relatifs au tempo, à la durée, au timbre, à l'octave, à la note.

- Appeler la routine 1E hexa pour un JSR et 9E hexa pour un JMP.

Lecture et écriture sur cassette

Cette routine est relativement complexe. Elle permet de lire ou d'écrire des données sur bande magnétique.

Pour sélectionner la lecture, mettre dans le registre A un octet non nul ; pour l'écriture, mettre dans A la valeur 0.

Lecture de bande

A n'est donc pas à 0.

- Passer dans Y l'adresse du buffer de lecture.
- Appeler la routine par 20 hexa pour un JSR et A0 pour un JMP.

Au retour, B contiendra le code du type de bloc :

- 00 pour un bloc de début de fichier.
- 01 pour un bloc de données.
- FF pour un bloc de fin de fichier.

Le registre A contiendra au retour le CHECKSUM calculé.

A partir de l'adresse pointée par Y, on retrouvera le bloc de données :

- un octet donnant la longueur du bloc ;
- les octets de donnée (au plus 253) ;
- un octet donnant le CHECKSUM lu.

Soit un total maximum de 255 octets.

Vérifier que les deux CHECKSUM sont égaux.

Ecriture d'un bloc

A est mis à 0. Passer dans B le type de bloc (voir "lecture de bande", page 86).

- Mettre en mémoire, à une adresse qu'on inscrira dans Y, un bloc de données de 255 octets au maximum, structuré comme dans "lecture de bande" page 86).

Le nombre des octets de données est donc égal à la longueur du bloc moins 2.

- Appeler la routine (voir page 86 "lecture de bande").

Mise en route et arrêt du moteur

On peut commander la mise en marche et l'arrêt du moteur du lecteur-enregistreur de programme grâce à cette machine.

Ce qui doit se passer est inscrit dans les deux bits de plus faible poids de A.

xxxxxx00 : arrêt du moteur après 5/10 sec.

xxxxxx01 : mise en route du moteur sans délai.

xxxxxx10 : arrêt du moteur après 5/10 sec.

xxxxxx11 : mise en route du moteur puis délai d'une seconde.

- Appeler la routine 22 hexa pour un JSR et A2 hexa pour un JMP.

On retrouvera au retour, dans le bit de retenue de CC, 0 si le LEP est correctement raccordé, 1 s'il est absent.

Gestion de l'interface de communication

L'interface de communication peut être commandée en mode parallèle, pour quatre fonctions principales.

Ce sont :

- écriture d'un octet en parallèle ;
- copie graphique d'écran ;
- ouverture pour écriture en parallèle ;
- fermeture de la communication.

Chacune de ces fonctions est définie par l'octet inscrit dans le registre PR.OPC (2Ø42 hexa).

1 pour écriture en parallèle.

2 pour copie graphique d'écran.

4 pour l'ouverture en mode parallèle.

16 pour la fermeture de la communication.

- Pour écrire en parallèle, passer dans B l'octet à envoyer, appeler la routine 24 hexa pour un JSR et A4 hexa pour un JMP.
- Pour recopier l'écran, passer dans le registre GRCODE (2Ø77 hexa) le code sélectionnant le mode graphique sur l'imprimante, appeler la routine.
- Pour ouvrir en mode parallèle, appeler simplement la routine.
- Pour fermer la communication, appeler la routine.
- Appeler la routine par 24 hexa pour un JSR et A4 hexa pour un JMP.

En retour de l'appel, on retrouve le bit de retenue du registre CC, à Ø si tout a bien fonctionné et à 1 s'il y a eu un ennui.

Le registre PR.STA retourne le code 8 si le périphérique n'était pas connecté (DTR de la sortie parallèle), ou le code de l'opération effectuée si tout a bien fonctionné.

Entrée-sortie sur disquette

Le registre 2Ø48 hexa doit contenir le code de l'opération souhaitée. Code de la routine : 26 hexa pour un JSR et A6 hexa pour un JMP.

Initialisation du contrôleur
- Mettre Ø1 dans le registre 2Ø48 hexa.
- Appeler la routine.
- Le code d'erreur 4Ø hexa peut être éventuellement retourné dans 2Ø4E hexa.

Lecture d'un secteur
- Mettre Ø2 dans le registre 2Ø48 hexa, dans 2Ø49 hexa le numéro du drive, dans 2Ø4A-2Ø4B le numéro de piste, dans 2Ø4F-2Ø5Ø l'adresse du buffer.
- Appeler la routine.
- Un code d'erreur est éventuellement retourné dans 2Ø4E hexa. Il peut être Ø2 Ø4 Ø8 1Ø 8Ø hexa.

Passage de double en simple densité
- Mettre Ø4 dans le registre 2Ø48 hexa.
- Appeler la routine.

Ecriture dans un secteur
- Mettre Ø8 dans le registre 2Ø48 hexa, dans 2Ø49 hexa le numéro du drive, dans 2Ø4A-2Ø4B le numéro de piste, dans 2Ø4F-2Ø5Ø l'adresse du buffer.
- Appeler la routine.
- Un code d'erreur est éventuellement retourné dans 2Ø4E hexa. Il peut être Ø1 Ø2 Ø4 Ø8 1Ø 2Ø 8Ø hexa.

Pour passer de double en simple densité
- Mettre 16 dans le registre 2Ø48 hexa.
- Appeler la routine.

Pour se positionner sur la piste Ø
- Mettre 32 dans le registre 2Ø48 hexa.
- Appeler la routine.
- Un code d'erreur peut éventuellement être retourné dans 2Ø4E hexa. Il peut être 1Ø ou 8Ø hexa.

Pour se positionner sur une piste
- Mettre 64 dans 2Ø48 hexa.
- Passer dans le registre 2Ø4A-2Ø4B hexa le numéro de piste.
- Appeler la routine.
- Un code d'erreur peut être éventuellement retourné dans 2Ø4E hexa. Il peut être 1Ø ou 8Ø hexa.

Codes d'erreur relatifs à la gestion des disquettes

Ø1 : disquette protégée en écriture.
Ø2 : erreur de piste.
Ø4 : erreur de secteur.
Ø8 : erreur de Checksum ou de lecture physique.
1Ø : lecteur indisponible.
2Ø : erreur de vérification entre le buffer et le secteur lu.
4Ø : contrôleur indisponible.
8Ø : disquette non formatée.

La présence en marge d'une astérisque renvoie à des explications sur le registre concerné, après ce tableau.

Ø1		TERMIN	2ØØØ-2Ø1B	(25) Table des terminateurs de ligne Basic.
Ø2	*	STATUS	2Ø19	(Ø1) Byte divers renseignements **.
Ø3		TABTP	2Ø1A-2Ø1B	(Ø2) Pointeur dans table terminateur ligne.
Ø4		COLN	2Ø1C	(Ø1) Colonne courante.
Ø5		TOPTAB	2Ø1D-2Ø1E	(Ø2) Pointeur sommet table terminateur ligne.
Ø6		BOTTAB	2Ø1F-2Ø2Ø	(Ø2) Pointeur fin de table terminateur ligne.
Ø7		SCRPT	2Ø21-2Ø22	(Ø2) Pointeur courant dans l'écran.
Ø8		STADR	2Ø23-2Ø24	(Ø2) Adresse du premier octet de la fenêtre.
Ø9		ENDDR	2Ø25-2Ø26	(Ø2) Adresse + 1 dernier octet de la fenêtre.
1Ø		BLOCZ	2Ø27-2Ø28	(Ø2) Réservé initialisations.
11		FORMS	2Ø29	(Ø1) Code couleur pour point ou ligne.
12	*	ATRANG	2Ø2A	(Ø1) Byte de gestion d'écran **.
13	*	COLOUR	2Ø2B	(Ø1) Couleur courante **.
14		PAGFLG	2Ø2C	(Ø1) Ø=Pas de scroll/255=Scroll.
15		SCROLS	2Ø2D	(Ø1) Ø=scroll normal/255=scroll lent.
16		CURSFL	2Ø2E	(Ø1) 255=pas de lien de ligne.
17		COPCHR	2Ø2F	(Ø1) 255=recopie si mouvement curseur G/D.
18		EFCMPT	2Ø3Ø	(Ø1) Compteur d'effacements curseur.
19		ITCMPT	2Ø31	(Ø1) Compteur d'interruptions IRQ.
2Ø		PLOTX	2Ø32-2Ø33	(Ø2) Abscisse dernier point ou case allumé(e).
21		PLOTY	2Ø34-2Ø35	(Ø2) Ordonnée dernier point ou case allumé(e).
22		CHDRAW	2Ø36	(Ø1) Ø=tracé graphique sinon code caractère.
23	*	KEY	2Ø37	(Ø1) Code matriciel dernière touche enfoncée.

24		CMPTKB	2Ø38	(Ø1) Compteur répétition clavier.
25		RESERVE	2Ø39	(Ø1)
26	*	TEMPO	2Ø3A	(Ø1) Tempo de génération de musique.
27		RESERVE	2Ø3B	(Ø1)
28	*	DUREE	2Ø3C	(Ø1) Durée de la note.
29	*	TIMBRE	2Ø3D	(Ø1) Type d'attaque de la note.
3Ø	*	OCTAVE	2Ø3E-2Ø3F	(Ø2) Octave de la note.
31		K7DATA	2Ø4Ø	(Ø1) Octet à transmettre au LEP.
32		K7LENG	2Ø41	(Ø1) Longueur du bloc à écrire sur le LEP.
33		PRCPC	2Ø42	(Ø1) Mot de commande pour gestion communication.
34		PR.STA	2Ø43	(Ø1) Etat de la liaison imprimante.
35		TEMP	2Ø44-2Ø45	(Ø2) Registre de transfert de données.
36		SAVEST	2Ø46-2Ø47	(Ø1) Sauvegarde du pointeur de pile.
37		DK.OPC	2Ø48	(Ø1) Mot de commande contrôleur drive.
38		DK.DRV	4Ø49	(Ø1) Numéro du drive
39		DK.TRK	2Ø4A-2Ø4B	(Ø1) Numéro de piste.
4Ø		DK.SEC	2Ø4C	(Ø1) Numéro de secteur.
41		DK.NUM	2Ø4D	(Ø1) Entrelacement des secteurs.
42		DK.STA	2Ø4E	(Ø1) Etat du contrôleur de drive.
43		DK.BUF	2Ø4F-2Ø5Ø	(Ø2) Pointeur buffer comm. disquettes.
44		TRACKØ	2Ø51-2Ø52	(Ø2) Position de la tête du lecteur Ø.
45		TRACK1	2Ø53-2Ø54	(Ø2) Position de la tête du lecteur 1.
46		TRACK2	2Ø55-2Ø56	(Ø2) Position de la tête du lecteur 2.
47		TRACK3	2Ø57-2Ø58	(Ø2) Position de la tête du lecteur 3.
48	*	SEQUCE	2Ø59	(Ø1) Code de la séquence d'écran.
49		US1	2Ø5A	(Ø1) Indicateur Unit Separator.
5Ø		ACCENT	2Ø5B	(Ø1) Indicateur de séquence d'accent.

REGISTRES UTILISES PAR LE MONITEUR

51	SS2GET	205C-205D	(01)	Registre de caractères accentués.
52	SWIIPT	205E-205F	(02)	Pointeur de routine d'appel MONITEUR.
53	RESERVE	2060	(01)	
54	TIMEPT	2061-2062	(02)	Pointeur routine utilisateur IRQ.
55		2063	(01)	
56	IRQPT	2064-2065	(02)	Pointeur routine moniteur IRQ.
57	RESERVE	2066	(01)	
58	FIRQPT	2067-2068	(02)	Pointeur routine FIRQ.
59	RESERVE	2069	(01)	
60	SIMUL	206A-206B	(02)	Pointeur table entrée MONITEUR.
61	RESERVE	206C	(01)	
62	CHRPTR	206D-206E	(02)	Pointeur table décodage clavier.
63	RESERVE	206F	(01)	
64	USERAF	2070-2071	(02)	Pointeur table caractères graphiques.
65	RESERVE	2072	(01)	
66	GENPTR	2073-2074	(02)	Pointeur table caractères standards.
67	RESERVE	2075	(01)	
68	LATCLV	2076	(01)	Latence de répétition clavier.
69	GRCODE	2077	(01)	Mot mise imprimante en graphique.
70	DECALG	2078	(01)	Décalage crayon optique.
71	RESERVES	2079-207E	(06)	
72	DEFDST	207F	(01)	Indicateur type de densité drive.
73	DKFLG	2080	(01)	Indicateur présence drive.
74	STKEND	2081-20CC	(76)	Pile système.
75	LPBUFF	20CD-20E4	(24)	Zone-tampon de lecteur crayon optique.
76	RESERVES	20E5-20FD	(25)	
77	TSTRST	20FE-20FF	(01)	Indicateur démarrage chaud ou froid.

DESCRIPTION DE CERTAINS REGISTRES

STATUS	b0	Touche clavier déjà lue.
	b1	Répétition clavier.
	b2	Etat curseur (0 = invisible, 1 = visible).
	b3	Lecture clavier.
	b4	1 : graphiques sans écriture de couleur.
	b5	
	b6	1 : scroll caractère sans couleur.
	b7	0 = majuscule, 1 = minuscule.
ATRANG	b0	0 = simple hauteur, 1 = double hauteur.
	b1	0 = simple largeur, 1 = double largeur.
	b2	
	b3	
	b4	
	b5	
	b6	1 : forme plein écran.
	b7	1 : fond plein écran.

COLOUR Les quatre bits de poids faible : la couleur de fond.
Les quatre bits de poids fort : la couleur de forme.

Le PIA Système est constitué d'un circuit 6821. Celui-ci se programme aussi bien en entrée qu'en sortie. Il possède deux ports appelés A et B.

- Le port A transmet des données dans l'octet d'adresse A7CØ hexa.
- Le port B transmet des données dans l'octet d'adresse A7C1 hexa.

Registre de données du port A
Adresse A7CØ hexa

Voici les différentes utilisations des huit bits de données du port A. Il est indiqué s'ils sont fixés alors que le PIA fonctionne en entrée ou en sortie :

Ø	sortie	commutation mémoire-écran ou caractères.
1	sortie	couleur du tour Rouge.
2	sortie	couleur du tour Vert.
3	sortie	couleur du tour Bleu.
4	sortie	couleur du tour, demi-teinte.
5	entrée	interruption crayon optique.
6	sortie	écriture cassette (broche 5 de la fiche L.E.P.).
7	entrée	lecture cassette (broche 6 de la fiche L.E.P.).

Voir Annexe IV page 139 le codage binaire des 16 couleurs du MO5 à partir des trois couleurs de base.

Registre de contrôle du port A
Adresse A7C2 hexa

CA1	entrée	lecture du crayon optique (broche 3 de la fiche crayon optique).
CA2	sortie	commande du moteur du L.E.P.

Registre de données du port B
Adresse A7C1 hexa

Voici les différentes utilisations des huit bits de données du port B. Il est indiqué s'ils sont fixés alors que le PIA fonctionne en entrée ou en sortie.

Pour les huit bits de données du port B :

Ø	sortie	son
1	sortie	matriçage/dématriçage du clavier.
2	sortie	matriçage/dématriçage du clavier.
3	sortie	matriçage/dématriçage du clavier.
4	sortie	matriçage/dématriçage du clavier.
5	sortie	matriçage/dématriçage du clavier.
6	sortie	matriçage/dématriçage du clavier.
7	entrée	lecture du clavier.

Registre de contrôle du port B
Adresse A7C3

CB1	entrée	interruptions 5Ø Hz.
CB2	sortie	commande d'incrustation vidéo.

INTERFACE DE COMMUNICATION

L'interface de communication comporte un PIA 6821. Celui-ci est programmable en entrée ou en sortie.

Chaque port possède un registre de données et un registre de contrôle.

- Le registre du port A est à l'adresse A7EØ hexa.
- Le registre du port B est à l'adresse A7E1 hexa.

Registre de données du port A
Adresse A7EØ

Non utilisé, le MO5 n'étant pas pourvu en standard Basic d'une gestion de liaison série. Il est cependant utilisable.

Registre de contrôle du port A
Adresse A7E2

CA1	entrée	request to send.

Registre de données du port B
Adresse A7E1

Ø	sortie	émission du bit Ø d'un octet en liaison parallèle.
1	sortie	émission du bit 1 d'un octet en liaison parallèle.
2	sortie	émission du bit 2 d'un octet en liaison parallèle.
3	sortie	émission du bit 3 d'un octet en liaison parallèle.
4	sortie	émission du bit 4 d'un octet en liaison parallèle.
5	sortie	émission du bit 5 d'un octet en liaison parallèle
6	sortie	émission du bit 6 d'un octet en liaison parallèle.
7	sortie	émission du bit 7 d'un octet en liaison parallèle.

Registre de contrôle du port B
Adresse A7E3

CB1	sortie	ACKnowledge.
CB2	sortie	Strobe.

Voir le brochage du connecteur de cette interface page 105 de ce livre.

ADRESSES RELATIVES AU PIA MANETTES DE JEU

Un circuit PIA 6821 est contenu dans le boîtier d'extension manette et jeux.

Ce PIA est initialisé en entrée, mais peut se programmer en sortie par exemple.

Chaque port possède un registre de données.

Le port A transmet ses données dans l'octet d'adresse A7CC hexa.

Le port B transmet ses données dans l'octet d'adresse A7CD hexa.

Registre de données du port A
Adresse A7CC

Il reprend les données d'entrée directes au port A.

Chaque manette de jeu comporte à l'intérieur, pour la direction du manche, quatre contacts : un pour la gauche, un pour la droite, un pour le haut et un pour le bas. Quand le manche est en diagonale, il ferme donc deux contacts. Quand il est activé droit, il ferme un contact.

Composition de l'octet du registre de données du port A :

Ø	entrée	Ø si manche manette Ø vers le haut.
1	entrée	Ø si manche manette Ø vers le bas.
2	entrée	Ø si manche manette Ø vers la gauche.
3	entrée	Ø si manche manette Ø vers la droite.
4	entrée	Ø si manche manette 1 vers le haut.
5	entrée	Ø si manche manette 1 vers le bas.
6	entrée	Ø si manche manette 1 vers la gauche.
7	entrée	Ø si manche manette 1 vers la droite.

Registre de contrôle du port A
Adresse A7CE

CA1	entrée	action manette Ø.

Registre de données du port B
Adresse A7CD, bit par bit

Ø	entrée	convertisseur digital-analogique (son synthétiseur).
1	entrée	convertisseur digital-analogique (son synthétiseur).
2	entrée	convertisseur digital-analogique (son synthétiseur).
3	entrée	convertisseur digital-analogique (son synthétiseur).
4	entrée	convertisseur digital-analogique (son synthétiseur).
5	entrée	convertisseur digital-analogique (son synthétiseur).
6	entrée	Ø si bouton action manette Ø enfoncé.
7	entrée	Ø si bouton action manette 1 enfoncé.

Les six lignes de Ø à 5 du port B sont programmables en sortie pour pouvoir utiliser le synthétiseur de musique des manettes de jeu. Pour cela :

- passer Ø dans A7CF hexa pour commander le registre de direction du port B ;
- passer ØØ111111 binaire (3F hexa) dans A7CD pour commander en sortie les premières lignes et en entrée les deux dernières (pour pouvoir encore utiliser les boutons des manettes de jeu) ;
- passer ØØØØØ1ØØ binaire (Ø4 hexa) dans A7CF pour sélectionner le port B.

Il suffit maintenant d'envoyer dans B, par l'intermédiaire du registre de données de B, des valeurs choisies par vous.

Exemple sous Basic

```
10 POKE &HA7CF,&H00
20 POKE &HA7CD,&H3F
30 POKE &HA7CF,&H04
40 POKE &HA7CD,&H00
50 POKE &HA7CD,&H3F
60 GOTO 40
```

Remarque : seules six sorties correspondent au synthétiseur. Il est donc utile d'envoyer des valeurs supérieures à 3F hexa à la ligne 5Ø.

Registre de contrôle du port B
Adresse A7CF

CB1 entrée action manette 1.

Voir le brochage des prises de manette en page 106 de ce livre.

ADRESSES RELATIVES AU CRAYON OPTIQUE

Le PIA Système gère le fonctionnement du crayon optique. Quand la routine de lecture est activée :

- les ports A et B sont programmés en entrée ;
- la ligne de contrôle CA1 est activée en entrée pour le bouton du crayon optique ;
- la ligne de contrôle CB1 est activée en entrée pour le cycle d'interruptions 50 Hz.

Le bit 5 du registre de données du port A (A7C0 hexa) contient la valeur logique de l'interruption crayon optique.

Les registres A7E4 hexa et A7E5 hexa contiennent des informations relatives au crayon optique.

Cette paire de registres contient un compteur de points TV. On y trouve un mot de 16 bits donnant la position du crayon optique en face de l'écran. On part de 0 à l'extrême gauche de l'écran, à hauteur de la première ligne de la fenêtre standard d'écran. On arrive à 65535 à l'extrême droite de l'écran, à hauteur de la dernière ligne de la fenêtre normale d'écran.

On voit donc que le balayage affecte aussi le cadre extérieur de la fenêtre d'écran.

Le registre 2078 hexa contient un octet de décalage du crayon optique. C'est l'octet mis en place pendant l'exécution d'un TUNE. On peut le modifier volontairement pour changer le point lu par INPUT PEN ou INPEN ou les routines moniteur de lecture du crayon optique.

PRISE PERITEL (SCART)

Synchro image	:	Sortie vidéo
Masse commutation rapide	:	Masse vidéo
Commutation rapide	:	Signal rouge
Masse	:	Masse rouge
----------	:	Signal vert
----------	:	Masse vert
Commutation lente	:	Signal bleu
Signal audio	:	Masse bleu
Masse audio		----------
----------		----------

Le basculement d'un téléviseur en entrée vidéo PERITEL se fait grâce à l'application à la broche commutation lente d'une tension de 12V. Cette tension est fournie par le MO5. Il n'est donc pas utile d'utiliser la position AV de certains téléviseurs.

Cependant, si ledit téléviseur est syntonisé sur un canal TV à fort signal, il peut arriver que l'image tremble. Syntoniser alors sur un canal sans réception.

CONNECTEUR D'EXTENSION

Le MO5 ne comporte qu'un connecteur d'extension sur sa face arrière. Ceci est une limitation importante de ses capacités par rapport au TO7/7Ø. Le brochage est le même pour les deux machines. Il s'agit d'un connecteur de type 'coin de carte' 38 contacts au pas de 1.27 mm.

A : Barette du dessus

19 18 17 16 15 14 13 12 11 1Ø 9 8 7 6 5 4 3 2 1

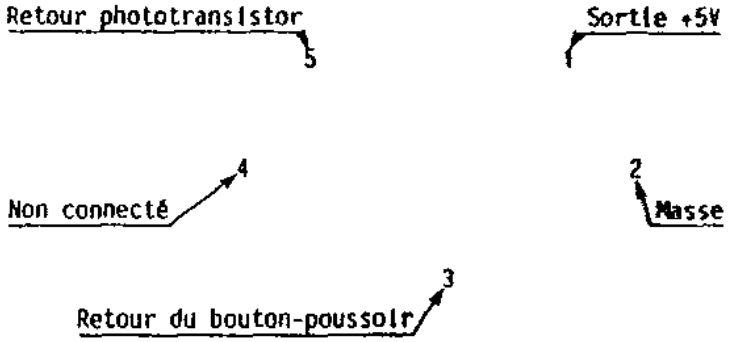
19 18 17 16 15 14 13 12 11 1Ø 9 8 7 6 5 4 3 2 1

B : Barette du dessous

A1	: masse	B1	: + 12v
A2	:	B2	: masse
A3	: E7C	B3	: audio
A4	: CSE	B4	: R/W
A5	: A11	B5	: E
A6	: A1Ø	B6	: D7
A7	: A9	B7	: D6
A8	: A8	B8	: D5
A9	: A7	B9	: D4
A1Ø	: A6	B1Ø	: D3
A11	: A5	B11	: D2
A12	: A4	B12	: D1
A13	: A3	B13	: DØ
A14	: A2	B14	: RST
A15	: A1	B15	: FIRQ
A16	: AØ	B16	: NMI
A17	: CSO	B17	: IRQ
A18	: CSC	B18	: VIDEO
A18	: + 5v	B19	: - 5v

CONNECTEUR DE CRAYON OPTIQUE

Le crayon optique est connecté au MO5 par une prise normalisée dite DIN 5 broches 260°.

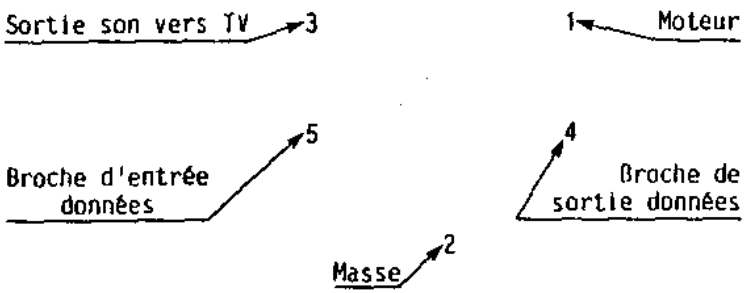


Le retour du bouton-poussoir est un retour de +5V, qui s'effectue quand le bouton caoutchouté à l'extrémité du crayon optique est enfoncé.

Le retour phototransistor est un retour de tension variable suivant l'éclairement dudit phototransistor. La tension croit avec l'éclairement.

LECTEUR-ENREGISTREUR DE PROGRAMMES

Le Lecteur-Enregistreur de Programmes (L.E.P.) est connecté au MO5 par une prise normalisée dite DIN 5 broches 180°.



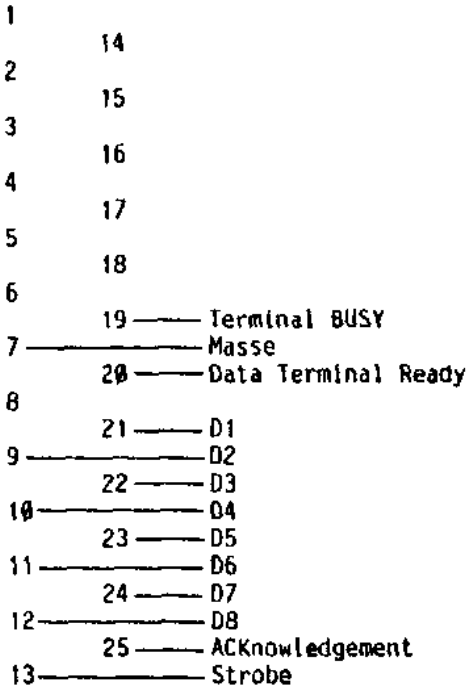
La broche moteur (n° 1) sert à commander la mise en marche ou l'arrêt du L.E.P.

La broche (n° 4) sert au MO5 à reconnaître la présence du magnétophone dédié.

Pour ce faire, il reconnaît la présence d'une tension de + 5V à cette broche.

PRISE CANON DE CONTROLEUR DE COMMUNICATION

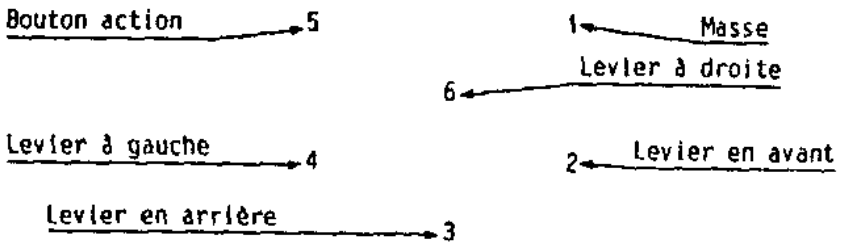
Le contrôleur de communication est programmable en Basic standard en sortie PARALLELE type CENTRONICS. Le brochage en est celui-ci :



DE MANETTES DE JEU

Les manettes de jeu s'enfichent dans l'interface musique et jeu par l'intermédiaire de deux prises au format standard DIN 6 broches.

Le brochage de chacune en est :



Le PIA contrôleur est attaqué au port A directement par les contacts relatifs aux directions (chacun étant cependant régulé à la masse par l'intermédiaire d'une capacité de quatre micro-farad), suivant le tableau ci-dessous :

- A0 Manche en avant, manette 0
- A1 Manche en arrière, manette 0.
- A2 Manche à gauche, manette 0
- A3 Manche à droite, manette 0
- A4 Manche en avant, manette 1.
- A5 Manche en arrière, manette 1.
- A6 Manche à gauche, manette 1.
- A7 Manche à droite, manette 1.

Le port B est attaqué aux ports 7 et 6 par les boutons-poussoirs :

- B6 Poussoir manette 0 (relié au CA1 du PIA).
- B7 Poussoir manette 1 (relié au CA2 du PIA).

Ces deux dernières broches étant reliées à la tension 5V par une résistance de 1 Kohm et régulées à la masse par une capacité de 100 nano-farad.

BROCHAGE DES PRINCIPAUX CIRCUITS INTEGRES

Microprocesseur 6809

BROCHAGE

VSS	MASSE	1	40	HALT
	NMI	2	39	XTAL
	IRQ	3	38	EXTAL
	FIRQ	4	37	RESET
	BS	5	36	MRDY
	BA	6	35	Q
VCC	+5V	7	34	E
	A0	8	33	DMA/BREQ
	A1	9	32	R/W
	A2	10	31	D0
	A3	11	30	D1
	A4	12	29	D2
	A5	13	28	D3
	A6	14	27	D4
	A7	15	26	D5
	A8	16	25	D6
	A9	17	24	D7
	A10	18	23	A15
	A11	19	22	A14
	A12	20	21	A13

Registres du 6809

- A, B Accumulateurs sur 8 bits.
- D = Juxtaposition de A et B.
- DP Registre de page directe.
- CC Registre d'état (Code Condition).
- U Pointeur de pile utilisateur.
- S Pointeur de pile système.
- PC Compteur ordinal.
- X, Y Registres d'index.

BROCHAGE DES PRINCIPAUX CIRCUITS INTEGRES

Circuit PIA 6821

BROCHAGE

VSS	MASSE	1	40	CA1
	PA0	2	39	CA2
	PA1	3	38	IRQA
	PA2	4	37	IRQB
	PA3	5	36	RS0
	PA4	6	35	RS1
	PA5	7	34	RESET
	PA6	8	33	D0
	PA7	9	32	D1
	PD0	10	31	D2
	PB1	11	30	D3
	PB2	12	29	D4
	PB3	13	28	D5
	PB4	14	27	D6
	PB5	15	26	D7
	PB6	16	25	E
	PB7	17	24	CS1
	CB1	18	23	CS2
	CB2	19	22	CS0
	VCC	20	21	R/W

Remarques sur le 6821

- Deux ports A et B de huit lignes programmables en entrée ou sortie.
- Deux lignes de contrôle par port CA1 et CA2 pour A. CB1 et CB2 pour B.
- Deux registres de contrôle CRA et CRB.
- Deux registres de sortie ORA et ORB.
- Deux registres de direction DDRA et DDRB.

On trouve un 6821 comme PIA système, comme PIA manettes et son, comme PIA communication.

Voir les différentes applications aux chapitres correspondants.

LE CLAVIER

Le clavier est en caoutchouc moulé. Les contacts sont du type simple-contact sur circuit imprimé.

Architecture matricielle du clavier : 8 x 8 non encodé.

6	7	Y	U	H	J	EFF	N	7
5	8	T	I	G	K	INS	,	6
4	9	R	O	F	L	⏎	.	5
3	0	E	P	D	M	→	@	4
2	-	Z	/	S	B	↓	ESP	3
1	+	A	*	Q	V	←	X	2
STOP	ACC	CNT	ENTER	RAZ	C	↑	W	1
						BASIC	SHIFT	0
7	6	5	4	3	2	1	0	

Port B

Port A du PIA clavier

Octets du moniteur concernant le clavier

2019 H : le bit 4 contient le code de bruitage du clavier : 0 pour OUI et 1 pour NON.

2037 H : code de la dernière touche enfoncée. Ce n'est pas le code ASCII mais le code matriciel du clavier.
Il s'obtient ainsi : revoir la matrice du clavier.
Si COL est le bit du port A clavier de la touche enfoncée, si RAN est le bit du port B clavier de la touche enfoncée, le code matriciel ressorti est 8x(7-RAN)+COL+1.
Pour trouver le code ASCII d'un caractère enfoncé, appeler la routine moniteur de décodage par le code 0A H ou 8A H. Le code ASCII est retourné dans le registre B du processeur.
On peut aussi lire l'octet d'adresse : contenu de 2037 hexa + contenu de la paire 206D-206E hexa.

2060-206E H : paire contenant le pointeur de la table de décodage du clavier. Cette table peut être remplacée par l'utilisateur dont la nouvelle table en RAM sera pointée par le contenu de 2060-206E.

2076 H : contient la valeur en dixièmes de secondes de la latence entre l'enfoncement d'une touche et sa répétition si on la garde appuyée.

Remarque : le bouton blanc allongé noté initialisation programme est situé sous la trappe de cartouches enfichables. Quand il est appuyé, la tension aux broches RESET et RST des circuits logiques passe de 5V à 0V, ce qui provoque une réinitialisation du système.

Point de vue graphique

L'écran est la représentation vidéo de la zone-mémoire située entre les adresses 0000 H et 1FFF H. Soit 16 Ko.

L'écran se compose de 320 x 200 pixels.

Ces pixels sont organisés en 40 x 200 segments de 8 pixels.

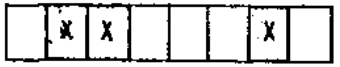
Chaque segment occupe en mémoire :

- 1 octet de 8 bits "forme" ;
- 1 octet de 8 bits "couleur".

Un octet de "forme" définit sur huit pixels quels sont ceux en couleur de forme et ceux en couleur de face.

Exemple

L'octet 0 1 1 0 0 0 1 0 correspond à :



- un octet de "couleur" définit sur ce segment de huit pixels la nature de la couleur de fond et celle de forme ;
- quatre bits de poids faible pour la couleur de fond et quatre bits de poids fort pour la couleur de forme.

Sur ces quatre bits, dans l'ordre de poids croissant :

1 bit pour la présence ou non de rouge 0 = non, 1 = oui.
1 bit pour la présence ou non de vert 0 = non, 1 = oui.
1 bit pour la présence ou non de bleu 0 = non, 1 = oui.
1 bit pour la demi-teinte ou pleine teinte 0 = demi, 1 = pleine.

Une **exception** : orange à la place de blanc demi-teinte.

Exemple

Codage sur couleur de fond magenta et forme jaune pâle :

- le magenta s'obtient en ajoutant rouge et bleu ;
- le jaune s'obtient en ajoutant rouge et vert.

L'octet correspondant aura comme bits de poids fort : 0011.

De droite à gauche :

1 pour la présence de rouge ;
1 pour la présence de vert ;
0 pour l'absence de bleu ;
0 pour demi-teinte.

Le même octet aura comme bits de poids faible : 1101.

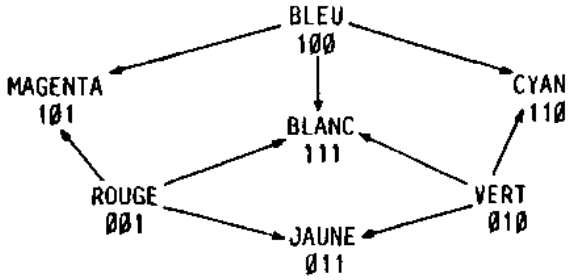
De droite à gauche :

1 pour la présence de rouge ;
0 pour l'absence de vert ;
1 pour la présence de bleu ;
1 pour pleine teinte.

L'octet correspondant sera donc :

00111101 en binaire, soit 3D hexa, soit 61 décimal.

Rappel des associations de couleurs



Point de vue des caractères

Le code des caractères affichables est le code ASCII. **Voir en ANNEXE le tableau de ces codes.**

Le générateur de caractères commence à l'adresse contenue dans le registre GENPTR (2073-2074 H). Vous pouvez modifier ce registre pour changer la forme des caractères affichés en pointant une table créée par vous en RAM.

Le caractère 20 H sera lu sur les huit premiers octets à partir du contenu de GENPTR, le 21 H sur les octets 9 à 16 comptés à partir de GENPTR, etc.

Les caractères de 7 à 1F hexa sont appelés interprétables.

Les caractères ROM et les caractères interprétables sont codés sur 7 bits.

Il existe d'autres caractères affichables codés avec le bit 7 à 1. Ce sont les caractères dits "utilisateurs".

Le pointeur de début de table des caractères utilisateurs se trouve indiqué dans le registre USERAF (2027-2028 H).

Le caractère 80 H sera lu sur les huit premiers octets à partir du contenu de USERAF, le 81 H sur les octets 9 à 16 comptés à partir de USERAF, etc.

Organisation physique

Une disquette est formatée par le drive au moment de son initialisation par DSKINI.

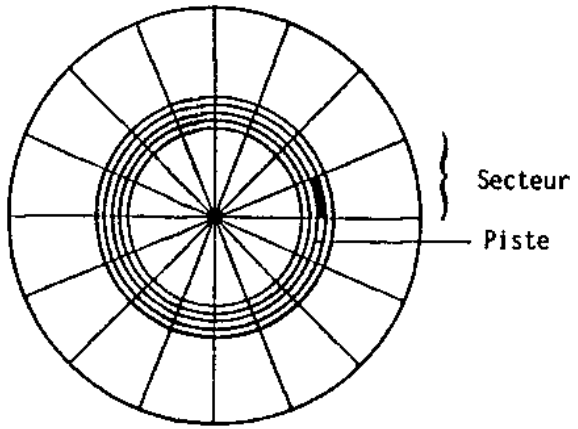
Le drive découpe magnétiquement la disquette en pistes et secteurs.

Il y a 40 pistes comportant chacune 16 secteurs.

Les pistes sont numérotées de 0 à 39.

Les secteurs sont numérotés de 1 à 16.

Chaque secteur de chaque piste comporte 128 octets avec un contrôleur simple densité, 255 avec un contrôleur double densité.



Entrelacement des secteurs : la disquette tournant relativement vite, il n'est pas possible de lire un secteur, de transférer le contenu dans un buffer avant de lire le secteur immédiatement contigu. L'initialisation détermine donc un facteur d'entrelacement, c'est-à-dire le nombre de secteurs sautés entre deux secteurs se suivant logiquement. Cet entrelacement est fixé en standard à 4. Il peut être modifié par DSKINI nbr1,nbr2 où nbr2 est le facteur d'entrelacement qu'on impose.

Organisation logique

Les cinq premières pistes de chaque disquette sont réservées. C'est sur ces pistes que l'on imprime le D.O.S. qui sera lu au moment de son initialisation. Donc, pistes de 0 à 4.

La piste 20 est réservée au directory de la disquette, c'est-à-dire à l'ensemble des informations qui permettent de savoir ce qui se trouve sur elle, et où.

Le directory se trouve sur la piste 20. Il contient la table d'allocation des fichiers et les descripteurs internes de fichiers de la disquette.

La table d'allocation des fichiers se trouve secteur 2 de la piste 20. Les descripteurs de fichiers se trouvent à partir du secteur 3 de la piste 20 et jusqu'à la fin de la piste.

Les fichiers sont composés suivant le format standard MICROSOFT. Ils sont organisés en blocs. Un bloc contient huit secteurs, donc 1024 caractères en simple densité, 2040 en double densité.

Chaque piste de la disquette est divisée en deux blocs. Chaque bloc de la disquette possède un index dans la table d'allocation. Cet index est un octet.

Si l'index d'un bloc contient FE hexa; ce bloc est libre.

Si l'index d'un bloc contient FF hexa, ce bloc est réservé.

Si l'index contient un octet compris entre 00 et 8F hexa, ce bloc est alloué à un fichier et la valeur de l'octet pointe le prochain bloc alloué au fichier.

Il suffit donc de connaître le premier bloc alloué à un fichier pour pouvoir le lire entièrement. On se reporte à chaque lecture à la table d'allocation pour connaître le bloc suivant à lire.

L'indication du premier bloc alloué à un fichier se trouve dans le descripteur interne de fichier.

Un descripteur interne de fichier se compose de 32 caractères. Il y a donc quatre descripteurs par secteur donc 4 x 14 = 56 fichiers possibles par disquette.

Organisation d'un descripteur interne

01-08	Nom principal du fichier, calé à gauche.
09-11	Extension du nom, calée à gauche.
12	Code du type de fichier : 0 pour programme Basic ; 1 pour fichier données Basic ; 2 pour fichier binaire.
13	
14	Code du type de données. 0 pour des données binaires ; 255 pour des données ASCII.
15	
16	Numéro du premier bloc de la table d'allocation attribué au fichier.
17-18	Nombre d'octets occupés par le fichier dans le dernier secteur qui lui est attribué.
19-32	Inutilisés.

Registres du moniteur concernant le contrôleur de drive

2048 hexa	Contient le code de l'opération en cours sur le drive. 01 : initialisation du contrôleur ; 02 : lecture d'un secteur ; * 04 : passage de double densité en simple densité ; 08 : écriture dans un secteur ; * 16 : passage de simple densité en double densité ; 32 : positionnement piste 0 ; 64 : positionnement sur la piste de numéro contenu dans 204A-204B hexa.

* Si le contrôleur est du type double densité.

SUPPRIMER LE BRUITAGE DU CLAVIER

Faire :

```
POKE &H2019,PEEK(&H2019)+8
```

Pour le remettre une fois enlevé :

```
POKE &H2019,PEEK(&H2019)-8
```

CHANGER LE TEMPS DE LATENCE DU CLAVIER

L'octet d'adresse 2076 hexa contient la durée en dixièmes de secondes du temps séparant la saisie du caractère quand on enfonce une touche et le moment où commence la répétition automatique, si l'on conserve le doigt dessus. A l'initialisation, sa valeur est 7.

Si l'on veut mettre deux secondes, faire :

```
POKE &H2076,20
```

Pour 25 secondes :

```
POKE &H2076,250
```

VARIABLE DE POSITION DU CURSEUR

Si vous souhaitez inscrire souvent au même endroit, il faudra, à chaque fois, faire LOCATE X,Y en Basic. Il y a un autre moyen plus simple, celui de définir une variable de position du curseur.

Si X et Y sont connus de la machine, faire :

```
A$=CHR$(31)+CHR$(64+Y)+CHR$(64+X)
```

Pour écrire le mot BONJOUR en position X,Y, faire :

```
PRINT A$;"BONJOUR"
```

Ce A$ pourra servir autant que vous le souhaitez pour mettre, en colonne X, rangée Y, vos commentaires. On peut de même prévoir plusieurs variables de position du curseur, ce qui revient à définir des masques d'écran.

VARIABLE DE HAUT DE FENETRE D'ECRAN

On peut, comme pour la position du curseur, définir une variable dont l'affichage va modifier le haut de fenêtre d'écran avec un chiffre de dizaine et un chiffre d'unité (d u).

Faire :

```
A$=CHR$(31)+(CHR$(32+d)+CHR$(32+u)
```

Exemple

Pour le haut de fenêtre rangée 15 :

```
A$=CHR$(31)+CHR$(33)+CHR$(37)
```

VARIABLE DE BAS DE FENETRE D'ECRAN

On peut, comme pour la position du curseur, définir une variable dont l'affichage va modifier le haut de fenêtre d'écran. Si H est la rangée souhaitée de haut de fenêtre, H s'écrit avec un chiffre de dizaine et un chiffre d'unité (d u).

Faire :

```
A$=CHR$(31)+CHR$(16+d)+CHR$(16+u)
```

Exemple

Pour le haut de fenêtre rangée 19 :

```
A$=CHR$(31)+CHR$(17)+CHR$(25)
```


Supposons que vous souhaitiez afficher à plusieurs reprises, en des endroits différents du programme, le mot "FAUX" en double taille double hauteur, en colonne 10 rangée 5, après avoir effacé l'écran entre la ligne 2 et le 7 ! Rien de moins.

Pour cela, il vous faudrait plusieurs fois utiliser plusieurs instructions Basic. Vous pouvez aussi mettre des codes ESCAPE ou ESPACE dans une variable que vous n'aurez plus qu'à afficher.

Dans ce cas :

```
A$=CHR$(31)+CHR$(16)+CHR$(21)+CHR$(12)+C
HR$(27)+CHR$(100)+CHR$(31)+CHR$(69)+CHR$
(74)+"FAUX"
```

A chaque affichage de A$, toutes les caractéristiques d'affichage seront modifiées pour devenir celles prévues.

Voir Annexes 1 et 2 pour la totalité des codes interprétables par l'affichage.

Ceci est particulièrement intéressant avec le D.O.S. quand on crée un fichier de messages. Le programme peut alors être totalement débarrassé de tous les affichages. On peut alors enregistrer des commentaires ou des pages d'écran alphanumériques dans un fichier à accès direct et appeler simplement le numéro d'enregistrement pour obtenir l'affichage. Le gain de place en mémoire centrale peut être énorme !

INTERDIRE L'UTILISATION DE QUELQUES TOUCHES DU CLAVIER

Il suffit de définir en RAM une nouvelle table de décodage matriciel dans laquelle, au lieu d'indiquer le code ASCII de la touche interdite, on met Ø.

Exemple

On veut permettre la seule utilisation des dix chiffres <Ø-9>.

Il faut composer à partir de l'adresse AØØØ hexa une nouvelle table de décodage, comme à la page suivante.

Table de décodage entre Code matriciel (C.m)
Caractère (CAR)
Code ASCII (C.A)

C.m :	Ø1	Ø2	Ø3	Ø4	Ø5	Ø6	Ø7	Ø8	Ø9	1Ø	11
CAR :	SHIFT	STOP	ACC	CNT	ENT	RAZ	C	↑	W	1	+
C.A :			22		13	12	67	11	87	49	43
C.m :	12	13	14	15	16	17	18	19	2Ø	21	22
CAR :	A	*	Q	V	←	X	2	-	Z	/	S
C.A :	65	42	41	86	8	88	5Ø	45	9Ø	47	83
C.m :	23	24	25	26	27	28	29	3Ø	31	32	33
CAR :	B	↓	SPA	3	Ø	E	P	D	M	→	@
C.A :	66	1Ø	32	33	48	69	8Ø	68	77	Ø9	64
C.m :	34	35	36	37	38	39	4Ø	41	42	43	44
CAR :	4	9	R	O	F	L	↵	.	5	8	T
C.A :	52	57	82	79	7Ø	76		46	53	56	83
C.m :	45	46	47	48	49	5Ø	51	52	53	54	55
CAR :	I	G	K	INS	,	6	7	Y	U	H	J
C.A :	73	71	75			54	55	89	85	72	74
C.m :	56	57									
CAR :	EFF	N									
C.A :		78									

Vous pouvez utiliser ce tableau pour transformer les codes ASCII retournés par l'enfoncement d'une touche.

Recopier en-dessous de chaque caractère son code si vous souhaitez pouvoir l'utiliser ou Ø sinon. Ceci fait, POKEr ces valeurs dans une table à un emplacement de la mémoire centrale choisi par vous (ne pas oublier CLEAR pour la réservation de place non accessible au Basic). Changer enfin le pointeur de début de table de conversion aux adresses 2Ø6D et 2Ø6E hexa.

Exemple

```
10 CLEAR ,&H9000
20 PTAB=256*PEEK(&H206D)+PEEK(&H206E)
30 FOR I=0 TO 59
40 A=PEEK(PTAB+I):POKE &H9000+I,A' Recopie la table en 9000 Hexa
50 NEXT I
60 POKE &H206D,&H90                ' Change adresse pointeur table
70 POKE &H206E,&H00                '    "      "       "      "
80 POKE &H9027,&H00  ' Inhibition du caractère 'L'
90 END
```

Ce petit programme recopie la table normale à partir de 9ØØØ hexa, change le pointeur de table pour que la machine utilise votre table, puis annule le code ASCII renvoyé par l'enfoncement de la touche 'L'. On peut bien sûr inhiber plusieurs touches, mêmes toutes...

CHANGER LA FORME DES CARACTERES ALPHANUMERIQUES

Le générateur de caractères alphanumériques est en ROM, à une adresse donnée par le contenu du registre GENPTR = 2Ø73-2Ø74 hexa du moniteur.

Il suffit donc de créer, à un endroit protégé de la RAM, un autre générateur de caractères, d'en inscrire l'adresse du premier octet dans le registre GENPTR.

Le générateur de caractère affiche à partir du code 32.

Le caractère 56 sera donc le 24e à partir du début de table. Or, un caractère est stocké sur huit octets. Il sera donc stocké à partir du 192e octet suivant le début de table.

Les huit octets composant le caractère sont constitués comme les caractères graphiques.

La page d'écran est située entre les adresses 0000 et 1FFF hexa, mais on peut la considérer de deux points de vue de la forme ou de la couleur.

Pour sauvegarder une page d'écran "forme" : il faut que le registre A7C0 hexa ait son bit 0 à la valeur 1.

Pour sauvegarder une page d'écran "couleur" : il faut que le registre A7C0 hexa ait son bit 0 à la valeur 0.

Il faudra donc faire deux sauvegardes successives de la page d'écran. Une pour la couleur et l'autre pour la forme.

```
1000 X0=PEEK(&HA7C0)-PEEK(&HA7C0)MOD2
1010 POKE &HA7C0,X0+1
1020 SAVEM"PAGE.FRM",0,&H1FFF,0
1030 POKE &HA7C0,X0
1040 SAVEM"PAGE.COU",0,&H1FFF,0
```

Pour recharger ladite page d'écran :

```
1100 X0=PEEK(&HA7C0)-PEEK(&HA7C0)MOD2
1110 POKE &HA7C0,X0+1
1120 LOADM"PAGE.FRM"
1130 POKE &HA7C0,X0
1140 LOADM"PAGE.COU"
```

SAUVEGARDER UNE BANDE D'ECRAN GRAPHIQUE

Comme la page d'écran, mais l'adresse de départ dépend de la première ligne de la bande et d'adresse d'arrivée dépend de la dernière ligne de la bande.

Soit XP la première ligne de la bande et XD la dernière :

0 =< XP =< 199 et 0 =< XD =< 199

Pour sauvegarder la bande d'écran :

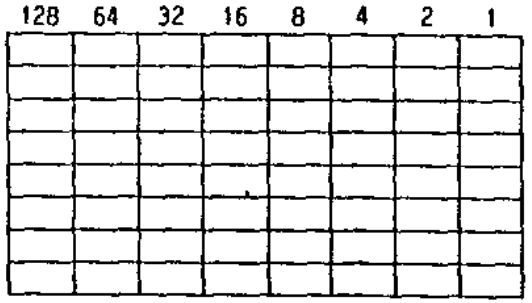
```
1000 X0=PEEK(&HA7C0)-PEEK(&HA7C0)MOD2
1010 POKE &HA7C0,X0+1
1020 SAVEM"PAGE.FRM",40*XP,40*XD,0
1030 POKE &HA7C0,X0
1040 SAVEM"PAGE.COU",40*XP,40*XD,0
```

Pour recharger ladite page d'écran :

```
1100 X0=PEEK(&HA7C0)-PEEK(&HA7C0)MOD2
1110 POKE &HA7C0,X0+1
1120 LOADM"PAGE.FRM"
1130 POKE &HA7C0,X0
1140 LOADM"PAGE.COU"
```


Mettre au début du programme un CLEAR,,nbr (voir cette instruction).

Noircir les cases devant être allumées dans cette grille 8x8 :



Rangée par rangée (il y en a 8) : prendre les nombres en haut des colonnes contenant une case noircie, en faire la somme. Ecrire ce nombre à droite de la rangée. On notera, par exemple, NB1 NB2 NB3 etc. ces huit nombres.

Utiliser l'instruction :

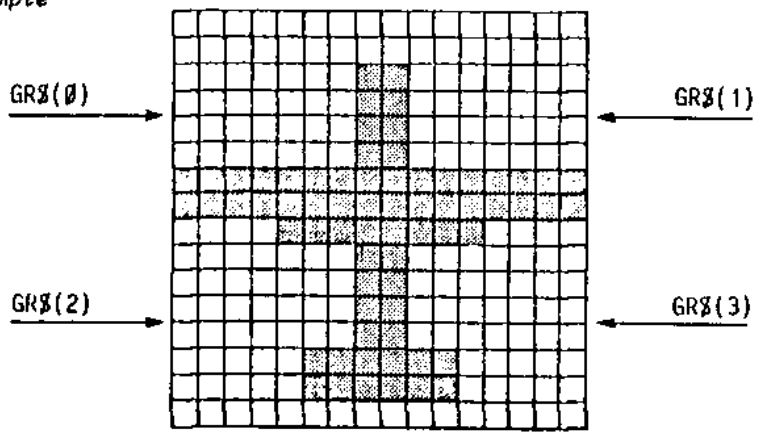
DEFGR$(N)=NB1,NB2,NB3,NB4,NB5,NB6,NB7,NB8

pour définir le caractère numéro N.

ATTENTION : pour définir GR$(0), il faut CLEAR,,1. Pour définir GR$(8), il faut CLEAR,,9.

On peut décider de créer une variable graphique, par exemple un avion. Le dessin tient sur plusieurs cases.

Exemple



Définir d'abord chaque case de l'avion comme un caractère graphique. Il en faudra quatre. De GR$(Ø) à GR$(3).

La variable sera :

```
XAV$=GR$(Ø)+GR$(1)+CHR$(1Ø)+CHR$(8)+CHR$(8)+GR$(2)+CHR$(3)
```

Explication

Le caractère 1Ø fait sauter une case. Le caractère 8 fait revenir le curseur en arrière d'une case.

Remarque : la longueur de cette variable est de sept caractères. ATTENTION donc aux affichages trop près du bord droit de l'écran qui passent à la ligne.

La fonction SCREEN(X,Y) retourne Ø en cas de rencontre avec un caractère graphique. De même, la routine du moniteur de lecture d'écran retourne Ø dans l'accumulateur B.

Il semble donc impossible de reconnaître un caractère graphique et, a fortiori, d'en distinguer plusieurs. C'est très utile dans les jeux vidéo pour reconnaître sur quelle case on arrive après un déplacement. Ami ou ennemi ?

La méthode originale indiquée ici utilise une autre astuce indiquée dans cet ouvrage : le changement des caractères affichables (voir page 117).

L'idée est de charger une nouvelle table de caractères affichables en RAM, de changer le pointeur de table pour la rendre utilisable et d'y inscrire le ou les caractères graphiques à la place de caractères ASCII qu'on n'utilisera pas.

Ils pourront être affichés grâce à CHR$(..) ou à l'aide de PUTCH sous MONITEUR.

Ils seront décodés par SCREEN(X,Y) ou par la lecture d'écran sous MONITEUR.

Exemple

```
10 CLEAR400,&H9CFF
20 FOR I=0 TO 767:B=PEEK(&HFC9E+I):POKE
&H9D00+I,B:NEXT I
30 POKE&H2073,&H9D:POKE&H2074,&H00
40 FOR I=0 TO 7:READ CAR:POKE&H9D00+8*58
+249-I,CAR:NEXT I    '58 est le code  du
caractère à modifier ( ici <:> )
50 DATA 73,42,28,8,8,28,36,102
```

Après exécution de ce programme, vous aurez un caractère 58 qui ne sera plus <:> mais un bonhomme. Ce caractère sera affiché tel quel et reconnu comme le caractère 58 par SCREEN(X,Y).

Explication

1Ø : réservation de RAMTOP pour rendre la table non accessible sous Basic.

2Ø : transfert à partir de 9DØØ des caractères standard.

3Ø : transfert du pointeur de table de caractères.

4Ø : lecture d'un caractère graphique et insertion dans la table créée. Début de table + 8* le numéro du caractère dans la table. Le numéro est la différence entre le code ASCII et 32 car la table ne comporte que les caractères affichables de code >32.

...isation du CLAVIER

Il est parfois utile d'interdire l'utilisation du clavier, par exemple quand on a réalisé un jeu fonctionnant exclusivement avec les manettes. Mettre alors dans le programme :

```
POKE&HA7C3,4
```

ATTENTION : ne pas taper cela en mode direct, vous ne pourriez plus taper RUN par exemple !

CHANGER L'EN-TETE D'UNE DISQUETTE

L'en-tête d'une disquette s'inscrit à l'écran au moment de l'appel du D.O.S.

On peut vouloir personnaliser cet en-tête, voire donner un nom, une date de création, etc.

Cet en-tête est écrit sur la disquette, piste 4, secteurs 2 et 3.

Mettre dans B$ le nouveau contenu souhaité de l'en-tête.
B$="....."

Penser, pourquoi pas, à y inclure des codes ESCape de forme, de couleur, des codes de saut de ligne, d'effacement d'écran...

ATTENTION : longueur de B$, tout compris, 50 caractères.

Faire :

```
A$=DSKI$(0,4,2)
A$=LEFT$(A$,85)+LEFT$(B$,43)
DSKO$ 0,4,2,A$
A$=DSKI$(0,4,3)
A$=RIGHT$(B$,7)+RIGHT$(A$,121)
DSKO$ 0,4,3,A$
```

C'est fini. Vous pouvez bien sûr faire un petit programme qui fasse tout cela automatiquement, inscrivant sur la disquette la date qu'il vous demande à l'écran, ainsi que le nom de la disquette et de l'utilisateur d'origine. C'est bien pratique pour s'y retrouver.

RECOPIER LE D.O.S. SANS EFFACER LE CONTENU DE LA DISQUETTE (BACKUP)

Pour inscrire le D.O.S. sur une disquette, la seule solution préconisée est de la recopier en entier.

Mais il peut arriver qu'après une fausse manipulation, le D.O.S. ne soit plus lisible. Nous vous proposons de créer un fichier dans lequel vous inscrirez le D.O.S. pour pouvoir le recopier ensuite en cas de besoin.

Programme de création du fichier

```
10 CLEAR 2200:DIM A$(16)
20 OPEN"R",1,"DOS.MO5":FIELD1,128 AS F$
30 FOR P=0 TO 4:FOR S=1 TO 16:A$(S)=DSKI$
(0,P,S):NEXT S
40 FOR S=1 TO 16:LSET F$=A$(S):PUT1,16*P+
S:NEXT S:NEXT P
```

Programme de lecture de fichier/copie du D.O.S.

```
10 CLEAR11000:DIM A$(4,16)
20 OPEN"R",1,"DOS.MO5":FIELD1,128 AS F$
30 FOR P=0 TO 4:FOR S=1 TO 16:GET1,16*P+S
40 A$(P,S)=F$:NEXT S:NEXT P:UNLOAD
50 CLS:PRINT"Mettre le disque récepteur"
60 PRINT"Et appuyer 2 touches"
70 A$=INPUT$(2)
80 FOR P=0 TO 4:FOR S=1 TO 16
90 DSKO$ 0,P,S,A$(P,S):NEXT:NEXT:UNLOAD
```

Le fichier DOS.MO5 est lu et emmagasiné dans le tableau A$. On demande en 50-60 de mettre le disque auquel on destine le D.O.S. Appuyer alors deux fois la barre d'espacement. Le D.O.S se recopie en 80-90.

AUTODEMARRAGE SUR UN PROGRAMME SOUS D.O.S

Il est possible de demander au MO5 de démarrer automatiquement l'exécution d'un programme dès que le D.O.S. est chargé. Celui-ci se charge avec l'instruction DOS.

Il suffit d'appeler votre programme AUTO.BAT et de le mettre sur la disquette. ATTENTION, l'extension .BAT est obligatoire !

Pour demander au MO5 d'appeler un programme d'un autre nom en auto-start (c'est le nom consacré), rechercher sur la piste 4, secteur 2, le nom AUTO.BAT, grâce à l'instruction de lecture de secteur DSKI$. Changer le nom, remettre le secteur avec DSKO$.

UTILISER LE GRAPHISME DE L'IMPRIMANTE

L'imprimante à impact THOMSON ne fait pas que recopier vos listings ou utiliser les caractères standard. On peut aussi lui faire faire du graphisme.

Voici comment procéder, ce n'est pas si facile !

- Ouvrir un canal de communication "imprimante" OPEN"O",=1,"LPRT:(24Ø)".
- Mettre l'imprimante en mode graphique en lui envoyant le code 8, PRINT=1,CHR$(8).
- Envoyer ensuite les caractères définis graphiquement.

Explication

L'imprimante ne va pas écrire ligne par ligne, mais rangée de sept lignes par rangée de sept lignes.

Chaque segment vertical de sept points sera défini par un nombre. Pour calculer ce nombre : partir du haut du segment et compter de Ø à 6.

Si l'on doit imprimer le point n du segment, compter alors pour lui 4 "puissance" n.

Le compte étant terminé, ajouter 128.

Exemple

Pour définir ce segment vertical de sept points :

Ø	XX	Compter 1 (2 puissance Ø)
1		Compter Ø
2		Compter Ø
3	XX	Compter 8 (2 puissance 3)
4	XX	Compter 16 (2 puissance 4)
5		Compter Ø
6	XX	Compter 64 (2 puissance 6)

Total : 1+8+16+64 auquel il faut rajouter 128, soit = 217.

Il faudra donc envoyer à l'imprimante le code 217 par l'ordre :

```
PRINT=1,CHR$(217);
```

ANNEXES

ANNEXE I

CODES ESCAPE

Les codes suivants correspondent à des ordres Basic. La suite des caractères indiquée à gauche effectue l'ordre Basic spécifié à droite.

Une telle suite de caractères peut être définie comme variable et affichée avec d'autres.

Exemple

```
A$=CHR$(27)+CHR$(115)+"BONJOUR"
```

A chaque fois qu'on affichera A$, ce sera en gros caractères, même sans spécifier ATTRB 1,1.

Suite des codes	*Ordre Basic équivalent*
27 & 32 & 64 & 32 :	SCREEN Ø
27 & 32 & 65 & 32 :	SCREEN 1
27 & 32 & 66 & 32 :	SCREEN 2
27 & 32 & 67 & 32 :	SCREEN 3
27 & 32 & 68 & 32 :	SCREEN 4
27 & 32 & 69 & 32 :	SCREEN 5
27 & 32 & 7Ø & 32 :	SCREEN 6
27 & 32 & 71 & 32 :	SCREEN 7
27 & 32 & 72 & 32 :	SCREEN 8
27 & 32 & 73 & 32 :	SCREEN 9
27 & 32 & 74 & 32 :	SCREEN 1Ø
27 & 32 & 75 & 32 :	SCREEN 11
27 & 32 & 76 & 32 :	SCREEN 12
27 & 32 & 77 & 32 :	SCREEN 13
27 & 32 & 78 & 32 :	SCREEN 14
27 & 32 & 79 & 32 :	SCREEN 15
27 & 32 & 8Ø & 32 :	SCREEN ,Ø
27 & 32 & 81 & 32 :	SCREEN ,1
27 & 32 & 82 & 32 :	SCREEN ,2
27 & 32 & 83 & 32 :	SCREEN ,3
27 & 32 & 84 & 32 :	SCREEN ,4
27 & 32 & 85 & 32 :	SCREEN ,5
27 & 32 & 86 & 32 :	SCREEN ,6
27 & 32 & 87 & 32 :	SCREEN ,7
27 & 32 & 88 & 32 :	SCREEN ,8
27 & 32 & 89 & 32 :	SCREEN ,9
27 & 32 & 9Ø & 32 :	SCREEN ,1Ø

```
27 & 32 & 91 & 32   :   SCREEN ,11
27 & 32 & 92 & 32   :   SCREEN ,12
27 & 32 & 93 & 32   :   SCREEN ,13
27 & 32 & 94 & 32   :   SCREEN ,14
27 & 32 & 95 & 32   :   SCREEN ,15
27 & 32 & 123 & 32  :   SCREEN ,,,0

27 & 64  :. COLOR 0
27 & 65  :  COLOR 1
27 & 66  :  COLOR 2
27 & 67  :  COLOR 3
27 & 68  :  COLOR 4
27 & 69  :  COLOR 5
27 & 70  :  COLOR 6
27 & 71  :  COLOR 7
27 & 72  :  COLOR 8
27 & 73  :  COLOR 9
27 & 74  :  COLOR 10
27 & 75  :  COLOR 11
27 & 76  :  COLOR 12
27 & 77  :  COLOR 13
27 & 78  :  COLOR 14
27 & 79  :  COLOR 15
27 & 80  :  COLOR ,0
27 & 81  :  COLOR ,1
27 & 82  :  COLOR ,2
27 & 83  :  COLOR ,3
27 & 84  :  COLOR ,4
27 & 85  :  COLOR ,5
27 & 86  :  COLOR ,6
27 & 87  :  COLOR ,7
27 & 88  :  COLOR ,8
27 & 89  :  COLOR ,9
27 & 90  :  COLOR ,10
27 & 91  :  COLOR ,11
27 & 92  :  COLOR ,12
27 & 93  :  COLOR ,13
27 & 94  :  COLOR ,14
27 & 95  :  COLOR ,15

27 & 96  :  SCREEN ,,0
27 & 97  :  SCREEN ,,1
27 & 98  :  SCREEN ,,2
27 & 99  :  SCREEN ,,3
27 & 100 :  SCREEN ,,4
27 & 101 :  SCREEN ,,5
27 & 102 :  SCREEN ,,6
27 & 103 :  SCREEN ,,7
```



```
27 & 104 :  SCREEN ,,8
27 & 105 :  SCREEN ,,9
27 & 106 :  SCREEN ,,10
27 & 107 :  SCREEN ,,11
27 & 108 :  SCREEN ,,12
27 & 109 :  SCREEN ,,13
27 & 110 :  SCREEN ,,14
27 & 111 :  SCREEN ,,15

27 & 112 :  ATTRB 0,0
27 & 113 :  ATTRB 0,1
27 & 114 :  ATTRB 1,0
27 & 115 :  ATTRB 1,1

27 & 116 :  CONSOLE ,,0
27 & 117 :  CONSOLE ,,1

27 & 118 :  SCREEN ,,,0  Sortie d'incrustation vidéo
27 & 119 :  SCREEN ,,,1  Mise en mode incrustation

27 & 120 :  CONSOLE ,,,0
27 & 121 :  CONSOLE ,,,1
27 & 122 :  CONSOLE ,,,2
27 & 123 :  INVERSION VIDEO
```

ANNEXE II

CODES ESPACE

Un certain nombre d'attributs vidéo sont commandés par le code 31. Ce sont :

- le positionnement du curseur ;
- la définition de haut de fenêtre d'écran ;
- la définition de bas de fenêtre d'écran.

Positionnement du curseur

Il se fait par l'envoi d'une suite de trois caractères :

'31' & '64+Y' & '64+X'

où X représente le numéro de case et Y représente le numéro de rangée.

Définition de haut de fenêtre d'écran

Si HF est le numéro de la première rangée de la fenêtre, la définition se fait par l'envoi d'une suite de trois caractères :

'31' & '32+d' & '32+u'

où d est le chiffre des dizaines de HF et u est le chiffre des unités de HF.

Définition de bas de fenêtre d'écran

Si BF est le numéro de la dernière rangée de la fenêtre, la définition se fait par l'envoi d'une suite de trois caractères :

'31' & '16+d' & '16+u'

où d est le chiffre des dizaines de BF et u est le chiffre des unités de BF.

ANNEXE III

CARACTERES

Le code ASCII du MO5

On appellera "**affichables**" les caractères de code supérieur à 31.

On appellera "**interprétables**" ceux des codes inférieurs à 32.

00 : NUL	32 : Esp	64 : @	96 : -
01 :	33 : !	65 : A	97 : a
02 : STOP	34 : "	66 : B	98 : b
03 :	35 : #	67 : C	99 : c
04 :	36 : $	68 : D	100 : d
05 :	37 : %	69 : E	101 : e
06 :	38 : &	70 : F	102 : f
07 : BEEP	39 : ,	71 : G	103 : g
08 : Retour arrière curseur	40 : (	72 : H	104 : h
09 : Avance curseur	41 :)	73 : I	105 : i
10 : Saut de ligne	42 : *	74 : J	106 : j
11 : Remontée de ligne	43 : +	75 : K	107 : k
12 : CLS (effacement écran)	44 : '	76 : L	108 : l
13 : ENTREE	45 : -	77 : M	109 : m
14 :	46 : .	78 : N	110 : n
15 :	47 : /	79 : O	111 : o
16 :	48 : 0	80 : P	112 : p
17 : Clignotement curseur	49 : 1	81 : Q	113 : q
18 :	50 : 2	82 : R	114 : r
19 :	51 : 3	83 : S	115 : s
20 : Fin clignotement	52 : 4	84 : T	116 : t
21 :	53 : 5	85 : U	117 : u
22 : ACCents	54 : 6	86 : V	118 : v
23 : Soudure ligne	55 : 7	87 : W	119 : w
24 :	56 : 8	88 : X	120 : x
25 :	57 : 9	89 : Y	121 : y
26 :	58 : :	90 : Z	122 : z
27 :	59 : ;	91 : [	123 : {
28 :	60 : <	92 : \	124 : |
29 :	61 : =	93 :]	125 : }
30 :	62 : >	94 : ↑	126 : ~
31 :	63 : ?	95 : _	127 : ■

Caractères accentués

é : appuyer <ACC> puis <6>
è : appuyer <ACC> puis <7>
ù : appuyer <ACC> puis <8>
ç : appuyer <ACC> puis <9>
à : appuyer <ACC> puis <Ø>
â : appuyer <ACC> puis <touche jaune> puis sans la relâcher <^> puis a
ê : appuyer <ACC> puis <touche jaune> puis sans la relâcher <^> puis e
î : appuyer <ACC> puis <touche jaune> puis sans la relâcher <^> puis i
ô : appuyer <ACC> puis <touche jaune> puis sans la relâcher <^> puis o
û : appuyer <ACC> puis <touche jaune> puis sans la relâcher <^> puis u

Codes des caractères accentués retournés par SCREEN(X,Y)

128 : ç	133 : é	138 : ï	143 : ô
129 : a	134 : ê	139 : i	144 : o
130 : â	135 : ë	140 : î	145 : ù
131 : ä	136 : è	141 : o	146 : u
132 : à	137 : ï	142 : o	147 : u

ANNEXE IV

TABLEAUX DES PRINCIPAUX CODES

Codes couleur

Couleur	*Forme*	*Fond*	*Codage binaire*
Noir	Ø	-1	ØØØØ
Rouge	1	-2	ØØØ1
Vert	2	-3	ØØ1Ø
Jaune	3	-4	ØØ11
Bleu marine	4	-5	Ø1ØØ
Magenta	5	-6	Ø1Ø1
Bleu clair	6	-7	Ø11Ø
Blanc	7	-8	Ø111
Gris	8	-9	1ØØØ
Rouge pâle	9	-1Ø	1ØØ1
Vert pâle	1Ø	-11	1Ø1Ø
Jaune pâle	11	-12	1Ø11
Bleu	12	-13	11ØØ
Magenta pâle	13	-14	11Ø1
Bleu pâle	14	-15	111Ø
Orange	15	-16	1111

Rappel sur le codage binaire : les bits allant du plus faible poids vers le plus fort, on trouve :

- présence de rouge ;
- présence de vert ;
- présence de bleu ;
- présence de demi-teinte.

TABLEAUX DES PRINCIPAUX CODES

Codes des notes

Notes	Codes
Silence	00
DO	01
DO♯	02
RE	03
RE♯	04
MI	05
FA	06
FA♯	07
SOL	08
SOL♯	09
LA	10
LA♯	11
SI	12
SI♯	13

Codes d'octave

Les codes d'octave définis sous Basic par 01, 02, 03, 04, 05 sont dans le registre OCTAVE sous les représentations suivantes :

1	10000
2	01000
3	00100
4	00010
5	00001

Codes de durée

Notes	*Triolet*	*Normale*	*Pointée*
Triple croche	2	3	*
Double croche	4	6	9
Croche	8	12	18
Noire	16	24	36
Blanche	32	48	72
Ronde	64	96	*

ANNEXE V

PRINCIPALES CARACTERISTIQUES

Présentation

Coffret : moulé plastique noir.
Clavier : touches caoutchouc.
Dimensions : 29 cm x 18,3 cm x 5 cm.
Poids :

Alimentation extérieure

Boîtier : moulé plastique noir/bouton rouge lumineux.
Dimensions : 12 cm x 6,8 cm x 5,3 cm.
Tension entrée : 220 volts/50 hertz.
Tension sortie : 12-17 volt/750 mA.

Caractéristiques techniques

Processeur et mémoires

Microprocesseur : 6809.
Horloge : 1 Mhz
Mémoire d'écran : 16 Ko.
Mémoire morte : 4 Ko.
Mémoire vive : 44 Ko. dont 32 Ko utilisateur
Mémoire module : 16 Ko.

Sortie vidéo

Cordon : sortie Péritel normalisée/RVB + synchro + son.
Définition pts : 320 x 200 pixels.
Définition case : 40 x 25 cases caractères.

Clavier : 57 touches AZERTY accentué.
Touche d'écriture rapide des mots Basic.

Musique : Synthèse musicale sur cinq octaves.
Tempo.
Durée.
Attaque.

Lecteur-Enregistreur de Programmes (L.E.P.)

Transmission : 1200 bauds avec télécommande moteur.
Piste sonore émissible par HP télévision.

Extension interface imprimante

Type : parallèle CENTRONICS.

Extension jeux

Type : manette huit directions "tout ou rien" + bouton poussoir. Générateur d'enveloppe musicale pour synthèse sonore pour quatre voix simultanées sur sept octaves.

Extension incrustation d'image

Type : remplacement des points noirs de l'écran par des points venant d'un signal vidéo extérieur. Commande directe sous Basic.

ANNEXE VI

TRANSCRIPTION DECIMAL-HEXADECIMAL

000	00	043	2B	086	56	129	81	172	AC	215	D7
001	01	044	2C	087	57	130	82	173	AD	216	D8
002	02	045	2D	088	58	131	83	174	AE	217	D9
003	03	046	2E	089	59	132	84	175	AF	218	DA
004	04	047	2F	090	5A	133	85	176	B0	219	DB
005	05	048	30	091	5B	134	86	177	B1	220	DC
006	06	049	31	092	5C	135	B7	178	B2	221	DD
007	07	050	32	093	5D	136	88	179	B3	222	DE
008	08	051	33	094	5E	137	89	180	B4	223	DF
009	09	052	34	095	5F	138	8A	181	B5	224	E0
010	0A	053	35	096	60	139	8B	182	B6	225	E1
011	0B	054	36	097	61	140	8C	183	B7	226	E2
012	0C	055	37	098	62	141	8D	184	B8	227	E3
013	0D	056	38	099	63	142	8E	185	B9	228	E4
014	0E	057	39	100	64	143	8F	186	BA	229	E5
015	0F	058	3A	101	65	144	90	187	BB	230	E6
016	10	059	3B	102	66	145	91	188	BC	231	E7
017	11	060	3C	103	67	146	92	189	BD	232	E8
018	12	061	3D	104	68	147	93	190	BE	233	E9
019	13	062	3E	105	69	148	94	191	BF	234	EA
020	14	063	3F	106	6A	149	95	192	C0	235	EB
021	15	064	40	107	6B	150	96	193	C1	236	EC
022	16	065	41	108	6C	151	97	194	C2	237	ED
023	17	066	42	109	6D	152	98	195	C3	238	EE
024	18	067	43	110	6E	153	99	196	C4	239	EF
025	19	068	44	111	6F	154	9A	197	C5	240	F0
026	1A	069	45	112	70	155	9B	198	C6	241	F1
027	1B	070	46	113	71	156	9C	199	C7	242	F2
028	1C	071	47	114	72	157	9D	200	C8	243	F3
029	1D	072	48	115	73	158	9E	201	C9	244	F4
030	1E	073	49	116	74	159	9F	202	CA	245	F5
031	1F	074	4A	117	75	160	A0	203	CB	246	F6
032	20	075	4B	118	76	161	A1	204	CC	247	F7
033	21	076	4C	119	77	162	A2	205	CD	248	F8
034	22	077	4D	120	78	163	A3	206	CE	249	F9
035	23	078	4E	121	79	164	A4	207	CF	250	FA
036	24	079	4F	122	7A	165	A5	208	D0	251	FB
037	25	080	50	123	7B	166	A6	209	D1	252	FC
038	26	081	51	124	7C	167	A7	210	D2	253	FD
039	27	082	52	125	7D	168	A8	211	D3	254	FE
040	28	083	53	126	7E	169	A9	212	D4	255	FF
041	29	084	54	127	7F	170	AA	213	D5		
042	2A	085	55	128	80	171	AB	214	D6		

TRANSCRIPTION HEXADECIMAL-BINAIRE

Cette transcription est très facile à faire à l'aide d'un certain nombre de codes transcrits.

0000	0
0001	1
0010	2
0011	3
0100	4
0101	5
0110	6
0111	7
1000	8
1001	9
1010	A
1011	B
1100	C
1101	D
1110	E
1111	F

Pour transcrire un nombre hexadécimal en binaire, il faut le faire chiffre par chiffre. Par exemple, pour convertir D9 en binaire, convertir d'abord D en 1101 puis 9 en 1001. La conversion de D9 sera donc 11011001.

Le procédé est tout-à-fait similaire pour convertir un binaire en hexadécimal ; on le fait d'abord pour les quatre premiers bits puis ensuite pour les quatre derniers. Par exemple, pour convertir 10001011 en hexadécimal, on trouve dans le tableau de gauche que 1000 est 8 et que 1011 est B. La conversion donnera donc 8B.

PAGE D'ECRAN GRAPHIQUE