Le guide du MO5

André Deledicq

Photo de couverture : Pascal SEVRAN
Maquette et illustrations : Alain DUFOURCQ

TO TEK International
CEDIC
NATHAN

Ce volume porte la référence
ISBN 2-7124-0522-6

CEDIC, 32, boulevard Saint-Germain, 75005 - PARIS

Sommaire

Avant-Propos

Le MO5 peut être mis entre toutes les mains.
Ce livre aussi.
Que ce soient des mains de débutants, d'amateurs un peu débrouillés ou de programmeurs avancés.
Ce livre propose à chacun d'entre eux des entrées différentes, leur permettant d'utiliser le MO5 à leur niveau et de progresser rapidement dans la connaissance de ses merveilleux rouages.
Cependant, avant toute autre chose, n'oubliez pas qu'un bon lecteur doit consulter attentivement le sommaire et se promener un peu dans le livre...
• Les *leçons* vous guident pas à pas vers la maîtrise de votre MO5.
• Les *fiches de référence* vous renseignent, sur tous les détails relatifs à un mot ou un thème particulier (une, deux ou trois étoiles indiquent le niveau supposé du lecteur : débutant, débrouillé ou averti). Il y a 2Ø fiches une étoile, 2Ø fiches deux étoiles, 1Ø fiches trois étoiles.

■ **Pour les débutants,** c'est-à-dire ceux qui vont apprendre à programmer avec le MO5, la méthode est simple : laissez vous guider dans l'ordre des pages !
Étudiez les leçons l'une après l'autre, frappez les programmes proposés, faites les exercices...
Les 4 premières leçons vous habituent au maniement du clavier, à ce qui peut se passer sur l'écran, et aux premières idées sur la programmation.
Les 1Ø leçons suivantes vous permettent de parcourir l'éventail des possibilités de votre MO5.
Petit à petit, vous pourrez vous risquer à consulter les fiches de référence ; vous y trouverez les compléments et détails relatifs aux points du langage BASIC que vous souhaiterez approfondir.

■ **Pour les amateurs un peu éclairés,** c'est-à-dire ceux qui ont déjà commencé à programmer sur une autre machine, la méthode de lecture est plus souple.

Nous vous conseillons de feuilleter d'abord les 4 premières leçons et leur résumé, ou bien, selon votre humeur, de parcourir les fiches de référence "clavier, écran, curseur" et "enregistrer, exécuter, lister". Vous y prendrez contact avec les particularités de votre MO5.
Vous pourrez ensuite suivre les leçons 5 à 14, qui vous permettront d'acquérir une bonne maîtrise de la programmation de votre MO5.
Et, surtout, lisez les **conseils de programmation** à la fin des leçons.
Consultez le plus souvent possible les fiches de référence ; elles vous donneront toutes précisions sur les mots et les finesses du BASIC que vous devriez maintenant connaître.

■ **Pour les programmeurs déjà avertis,** ce livre doit être conçu plutôt comme un dictionnaire ou une encyclopédie.
Vous pouvez évidemment survoler les leçons : en particulier jetez un petit coup d'œil sur les leçons 1,2 (le clavier, l'écran et le curseur), 5 (graphiques), 1Ø (musique) et les deux "résumés et conseils de programmation".
Votre outil principal de travail est évidemment l'ensemble des fiches de référence. Elles sont classées par thèmes ; vous pourrez ainsi très vite vous faire une idée des possibilités offertes, par exemple par les instructions graphiques ou bien le traitement alphabétique.
Vous lirez avec profit les fiches "clavier, écran, curseur", et (si vous êtes super-initiés) les fiches "types et noms de variables", "occupation des mémoires" et les annexes 3 et 7.

■ Cependant, pour les intimes de ce livre et du MO5, l'entrée principale sera rapidement la dernière page ; elle montre tous les chemins : c'est l'INDEX.

FAITES CONNAISSANCE AVEC VOTRE MICRO-ORDINATEUR

Votre micro-ordinateur MO5 est spécialement conçu pour l'usage individuel ou familial et vous permettra :
— d'exécuter des programmes dans de multiples domaines :

- jeux
- éducation assistée
- gestion familiale, etc.

— de programmer en langage BASIC grâce à son interpréteur BASIC résident
— de programmer, selon la cartouche introduite dans l'ordinateur, en langage LOGO, ASSEMBLEUR, etc.
— de faire de la composition musicale sur 5 octaves
— de réaliser des dessins ou graphiques utilisant 16 couleurs différentes
— d'incruster des images créées avec votre MO5 dans une image vidéo externe provenant :

- d'une émission de télévision
- d'un magnétoscope
- d'une caméra vidéo

L'installation de votre micro-ordinateur doit obéir à quelques règles élémentaires :

a) certains composants situés à l'intérieur de l'ordinateur chauffent et nécessitent d'être ventilés ; en conséquence, évitez de placer votre appareil près d'une source de chaleur et d'obstruer les ouvertures d'aération,

b) le raccordement entre votre micro-ordinateur et votre téléviseur ou moniteur s'effectue, soit en utilisant la prise péri-télévision (fig. 3), soit par l'intermédiaire d'un codeur-modulateur dans le cas où votre récepteur est dépourvu de cette prise (fig. 4).

Pour vous permettre de vous familiariser avec votre micro-ordinateur, voici une description extérieure de l'appareil.

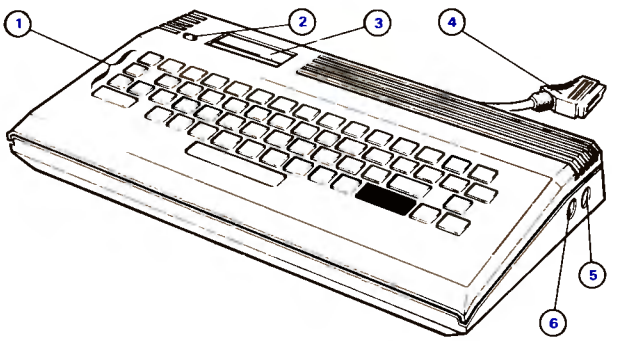


Fig. 1

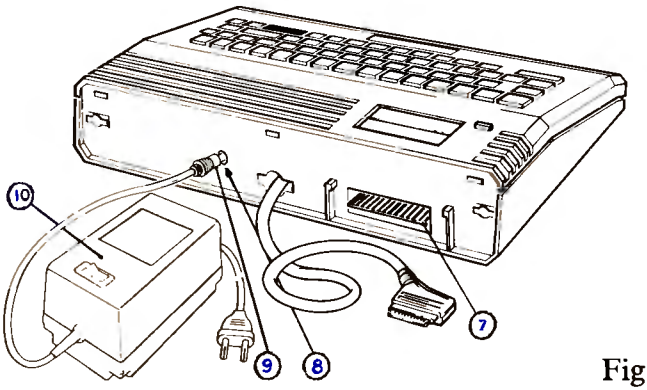
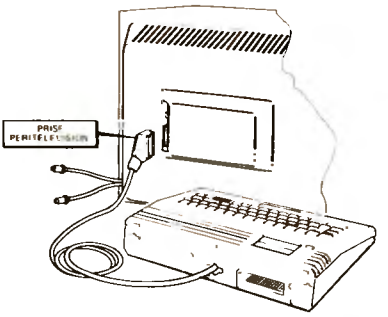


Fig. 2

1. **Clavier.**
2. **Touche « INITIAL PROG » : permet d'interrompre l'exécution d'un programme pour la reprendre à son début.**
3. **Trappe destinée à recevoir les cartouches de programmes.**
4. **Prise Péritélévision : permet le raccordement au récepteur de télévision directement ou par l'intermédiaire d'un codeur modulateur.**
5. **Prise DIN : permet le raccordement d'un lecteur-enregistreur de programmes adapté spécialement au MO5.**
6. **Prise DIN : permet le raccordement du crayon optique.**
7. **Connecteur permettant le raccordement d'une extension.**
8. **Prime alimentation extérieure.**
9. **Fiche alimentation.**
10. **Adaptateur secteur.**



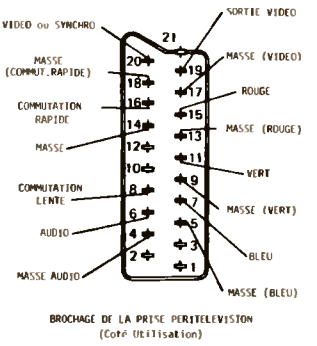


Fig. 3

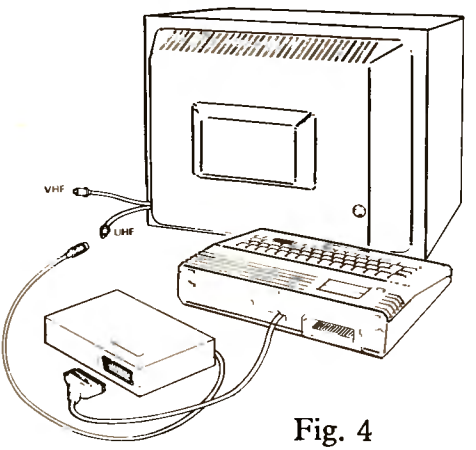


Fig. 4

Mise en service

La mise sous tension de votre installation doit être réalisée dans l'ordre suivant :
a) le récepteur de télévision ou le moniteur
b) les différentes extensions (par exemple les unités de disquettes)
c) l'ordinateur.
A la suite de ces opérations, vous devez régler la luminosité et le contraste de votre téléviseur ou moniteur afin d'obtenir une image correcte. (notez que le branchement de la prise péritélévision interrompt sur la plupart des téléviseurs les réglages de couleur et parfois du contraste).
Dans le cas de l'utilisation d'un codeur-modulateur, vous devez sélectionner sur votre récepteur le bouton de programme spécialisé, s'il existe, (consultez le guide d'utilisation de votre téléviseur), et le régler sur le canal UHF n° 36.

Le message suivant doit apparaître sur l'écran :
MO5 BASIC 1.0
(C) Microsoft 1984
Votre MO5 est prêt à recevoir vos instructions énoncées en Basic.

Pour utiliser une cartouche

La mise en place ou le retrait d'une cartouche de programme doit toujours s'effectuer le MO5 éteint et l'étiquette face au clavier. Pour la mise en place, enfoncez la cartouche dans le sens de la flèche jusqu'à encliquetage (fig. 5). Attention cette cartouche va remplacer le BASIC résidant dans votre MO5. Il vous faudra donc suivre les indications qui vous sont données avec cette cartouche.

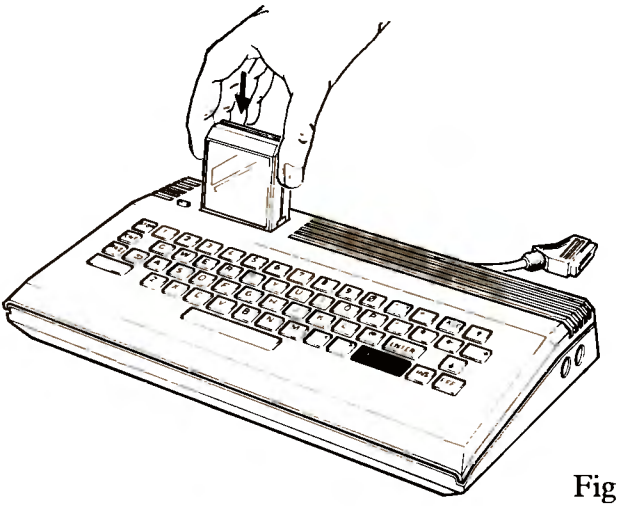
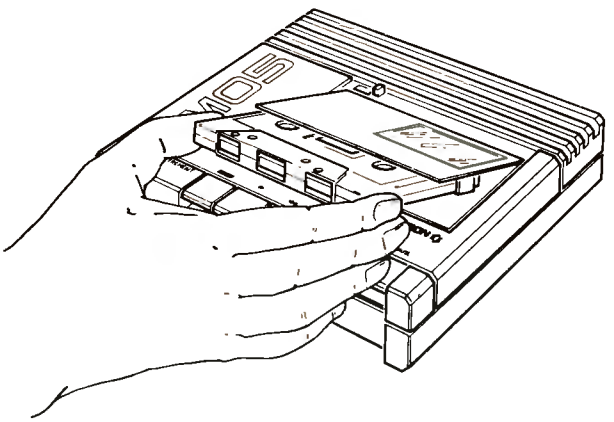


Fig. 5

Pour utiliser une cassette

L'ordinateur étant éteint :
— raccordez le Lecteur Enregistreur de Programmes (LEP) au MO5
— branchez la prise d'alimentation secteur du LEP
— placez la cassette dans le LEP
— allumez le MO5
— appuyez sur la touche lecture
— frappez R U N " ENTREE
— attendez que le programme sur cassette entre dans la mémoire du MO5 et suivez les instructions données avec la cassette. (Nous vous conseillons de lire la leçon 12)

— vous noterez que le LEP est un magnétophone spécifique qui vous permet, sur certaines cassettes, d'entendre de la musique et des commentaires pendant le transfert du programme enregistré vers la mémoire du MO5.



Pour régler le crayon optique

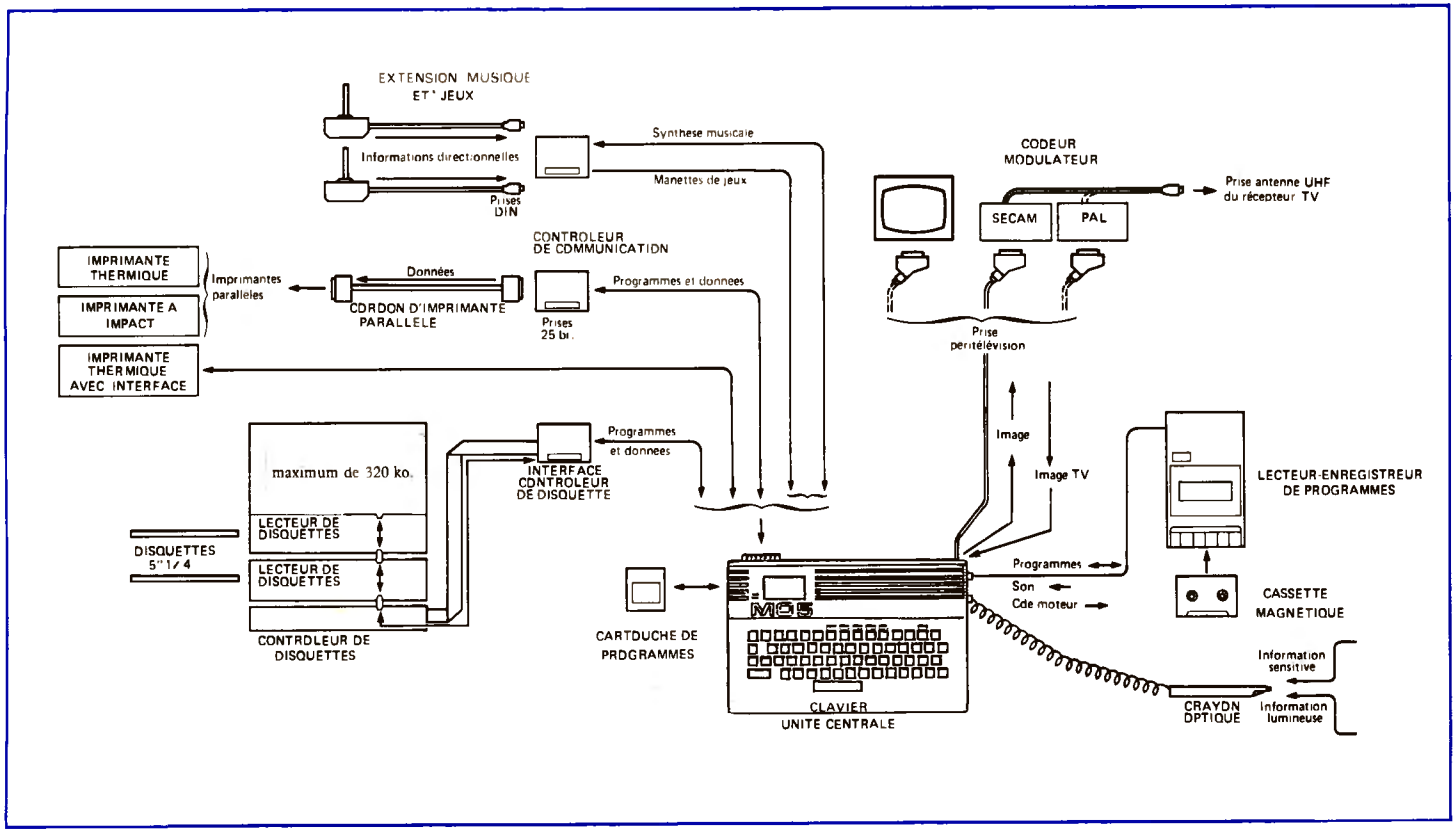
Le crayon optique réalise deux fonctions indépendantes :

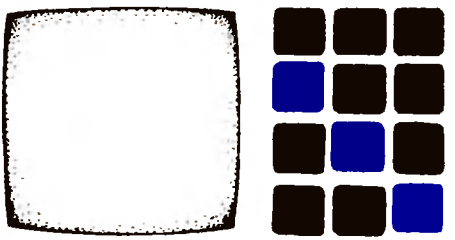
1) La validation qui est assurée par un interrupteur logé dans la pointe du crayon. Le contact se ferme lorsqu'on appuie le crayon sur une surface plane telle que l'écran du téléviseur.
2) La localisation qui indique la position relative du crayon sur l'écran. Ceci est réalisé grâce à un élément photo-sensible. Le fonctionnement reste correct tant que le crayon n'est pas éloigné de plus de 20 cm de l'écran. Un mauvais réglage de la luminosité du téléviseur peut entraîner une saisie difficile des informations. D'autre part, celles-ci ne sont pas prises en compte lorsque la zone pointée est noire ou rouge.

UNE MIRE DE RÉGLAGE PERMET A L'ORDINATEUR DE CALCULER UN PARAMÈTRE D'AUTO-CORRECTION AFIN DE COMPENSER LES ÉVENTUELLES DISPERSIONS DU BALAYAGE HORIZONTAL INTRODUITES PAR LES TÉLÉVISEURS.

Il est impératif d'avoir une couleur de fond de texte différente du noir ou du rouge.

En fonctionnement sous BASIC, frappez au clavier le mot TUNE, puis appuyez le crayon optique sur l'écran pour faire apparaître la mire de réglage constituée par une barre verticale clignotante.
A l'aide des touches du curseur ⇨ et ⇦ , faire coïncider la barre avec la position du crayon optique et validez le réglage avec ENTREE .
Dans le cas d'utilisation de cartouches de programme MO5, les instructions permettant d'afficher la mire de réglage du crayon optique sont indiquées dans la notice d'accompagnement de la cartouche.
Sous contrôle d'un programme, il est possible de lire les coordonnées du point visé sur l'écran et d'autre part de savoir si le contact est fermé ou non. Le crayon optique peut donc être très utile dans des applications interactives telles que : conception, dessin, jeux et enseignement assisté.
NOTA : Il est nécessaire de refaire le réglage du crayon optique lorsque l'alimentation de l'ordinateur a été coupée.


EXTENSION MUSIQUE
ET JEUX
Informations directionnelles
Prises
DIN
Synthese musicale
Manettes de jeux
CONTROLEUR
DE COMMUNICATION
CODEUR
MODULATEUR
Prise antenne UHF
du récepteur TV
SECAM
PAL
Prise
péritélévision
IMPRIMANTE
THERMIQUE
IMPRIMANTE A
IMPACT
IMPRIMANTE
THERMIQUE
AVEC INTERFACE
Imprimantes
parallèles
Données
CORDON D'IMPRIMANTE
PARALLELE
Programmes et données
Prises
25 bi.
maximum de 320 ko.
LECTEUR DE
DISQUETTES
LECTEUR DE
DISQUETTES
CONTROLEUR DE
DISQUETTES
DISQUETTES
5"1/4
Programmes
et données
INTERFACE
CONTROLEUR
DE DISQUETTE
Image
Image TV
LECTEUR-ENREGISTREUR
DE PROGRAMMES
Programmes
Son
Cde moteur
CASSETTE
MAGNETIQUE
CARTOUCHE DE
PROGRAMMES
M05
CLAVIER
UNITE CENTRALE
Information
sensitive
CRAYON
OPTIQUE
Information
lumineuse

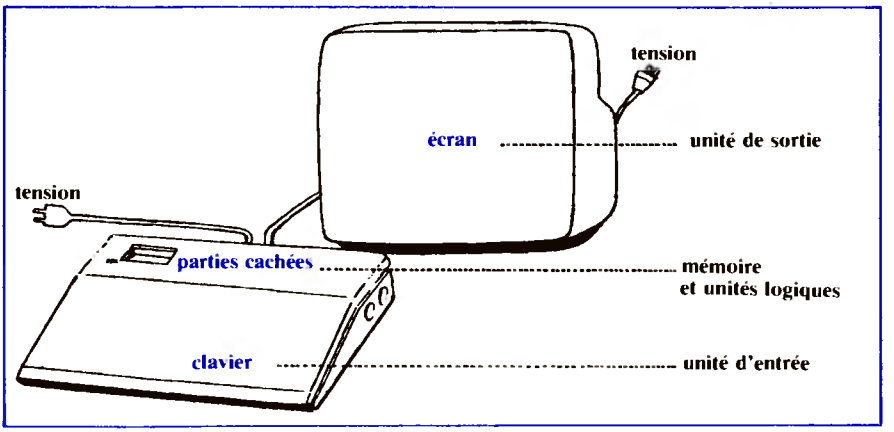


1

Le clavier et l'écran

■ Les trois parties

Votre ordinateur se compose de 3 éléments : le clavier, l'écran et toutes les parties qui vous sont cachées. Ce sont évidemment ces parties cachées qui font tout le travail important !
Le clavier et l'écran vous permettent simplement de communiquer avec les parties cachées : vous "entrez" ce que vous voulez (et pouvez) par les touches du clavier... et vous observez ce qui "sort" sur l'écran (lettres, chiffres, dessins...).

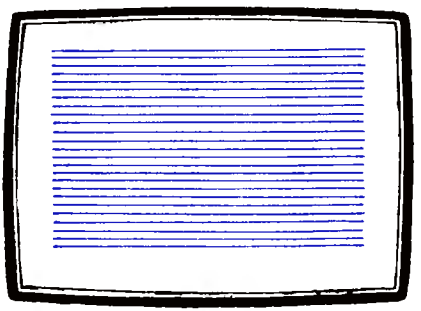


■ Vous "commandez" votre ordinateur en frappant sur le clavier

En fait, ce que vous lui commandez, c'est de modifier telle ou telle de ses parties ; ou bien vous modifiez l'écran (et vous voyez immédiatement le résultat de ce changement), ou bien vous changez l'état des parties cachées (et il vaut mieux alors savoir ce que vous faites).
Dans tous les cas vous ne risquez pas d'abîmer votre ordinateur en frappant sur n'importe quoi ; et si d'aventure, comme le singe dactylographe réécrivant la bible, vous frappiez une phrase ayant un sens pour la machine et rendant ses parties cachées inutilisables, alors vous pourriez toujours couper le courant puis le rebrancher.

■ Le curseur d'écriture

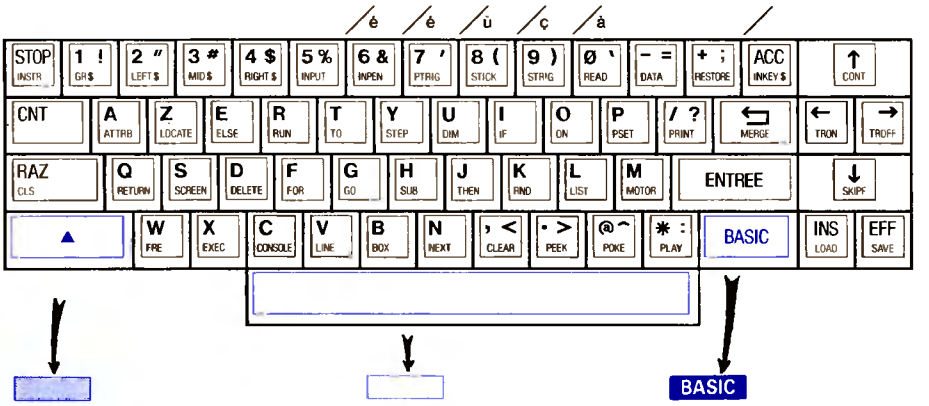
Ainsi s'appelle le petit trait que vous voyez sur votre écran ; il indique l'endroit où va s'écrire le prochain "caractère" frappé au clavier (un "caractère" c'est un chiffre, une lettre ou tout autre signe susceptible d'apparaître au dessus de ce petit trait, comme par exemple #, + ou ∧).
Frappez sur le clavier : vous verrez alors les caractères frappés s'écrire et le curseur se déplacer.
Remarquez que vous ne pouvez pas écrire dans les coins et sur le tour de l'écran : la partie utile de l'écran est limitée à un rectangle de 40 caractères de longueur sur 25 lignes de hauteur.



■ Le clavier

Apprenez la position sur le clavier des différentes touches :
• Les *touches normales d'écriture* de caractères c'est-à-dire :
les dix chiffres
les vingt-six lettres
les signes − + / * , . @
la barre d'espacement, qui fait écrire un caractère "blanc".
Notez que : en laissant votre doigt appuyé sur une touche, vous répétez l'écriture du caractère correspondant.
• Les touches qui font *déplacer le curseur :*
d'une position vers le haut ↑
vers la gauche ← → vers la droite
↓
vers le bas
↖ tout en haut à gauche.

• La touche RAZ qui efface l'écran complètement (*r*emise *à z*éro).



Dans la suite du livre, ces trois touches seront ainsi notées.

• Sachez aussi que :
— les touches INS et EFF servent à corriger ce qui est déjà écrit sur l'écran (voir les fiches de référence ou essayez vous-même),
— la touche ACC sert à écrire des voyelles accentuées (voir la fiche "minuscules et accents"),

— les touches STOP et CNT vous seront expliquées à la leçon 4,
— la touche ENTREE vous sera expliquée à la leçon 2.

• Enfin :

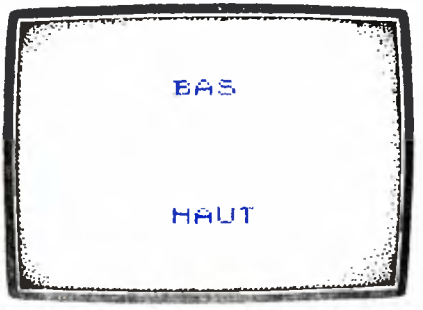
En appuyant sur la touche , située en bas à gauche, puis (en gardant cette touche enfoncée) en appuyant sur une autre touche, on commande l'écriture du signe dessiné en haut à droite de cette touche.

BASIC En appuyant sur la touche BASIC, située en bas à droite, puis (en gardant cette touche enfoncée) en appuyant sur une autre touche, on commande l'écriture du mot écrit dans la partie basse de cette touche.

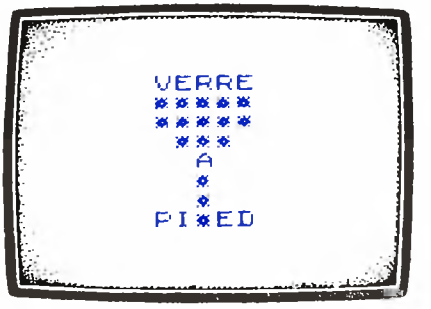
appuyer sur	produit l'écriture de
3 # MID$	3
" + 3 # MID$	" " #
" 3 # MID$ BASIC	" " MID$

Exercices

1. Écrire sur votre écran comme sur celui-ci :

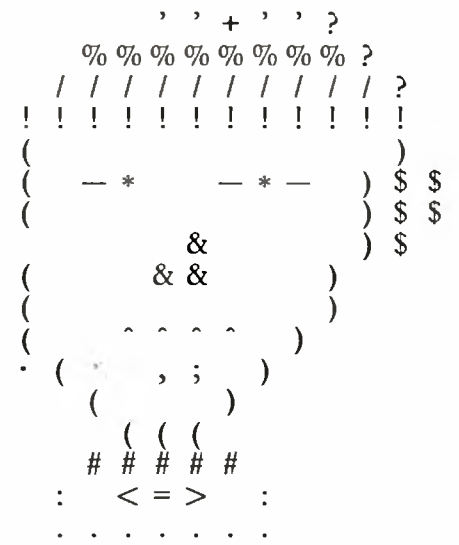


2. Décorer votre écran avec un "dessin alphabétique" analogue à celui-ci :



Dessinez ainsi, une maison, un homme, un arbre, une carte de la France.

3. Recopiez ainsi, sur votre écran le dessin suivant ou quelque autre de votre invention qui utilise des caractères plus ou moins curieux. (Vous ne devez pas utiliser la touche ENTREE).



Attention

■ Ne confondez pas les touches :

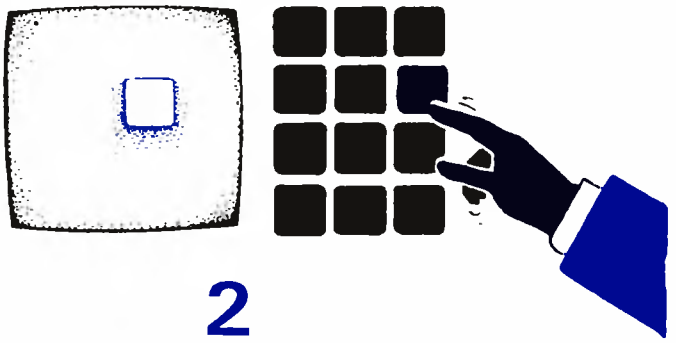
Ø (zéro)

et O (la lettre "o")

Fiches de référence pouvant être consultées :

* Clavier, page 133
* Ligne d'écriture, page 139
* Minuscules et accents, page 140

Page blanche



2

Commandes d'affichages

La précédente leçon vous a peut-être laissé sur un sentiment d'insatisfaction. Car enfin vous ne disposez pas simplement d'une machine à écrire, même s'il s'agit d'écrire sur votre écran de télévision !
Soyez rassuré, les choses sérieuses vont maintenant commencer :

■ La touche ENTREE

Avec la touche marquée ENTREE, vous allez pouvoir donner des ordres.
En effet :
Lorsque vous frappez sur cette touche, *la machine interprète tout ce que vous avez écrit précédemment sur la ligne et exécute l'ordre correspondant* (puis le curseur passe au début de la ligne suivante après avoir écrit "OK").

Par exemple si vous frappez successivement :
? 1 7 * 3 ENTREE

la machine interprète qu'il faut écrire le résultat de la multiplication de 17 par 3. Attention pour frapper le ?, il faut aussi frapper sur ⇧ en même temps. Elle écrit alors :

```
 51
OK
```

■ Passez (bien) votre commande

Cependant vous ne pouvez pas écrire n'importe quoi avant de frapper la touche ENTREE. Vous ne pouvez écrire que des "commandes" susceptibles d'être bien interprétées, bien comprises par la machine.
Pour l'essentiel, ces commandes appartiennent à une famille de langages connus sous le nom de BASIC.
Il vous faut donc apprendre comment s'expriment ces commandes et en quoi consiste leur exécution par la machine. Faute de quoi vous êtes bon pour le dialogue de sourds.

Par exemple si vous frappez :

1 7 * 3 ENTREE

la machine comprend quelque chose que vous n'apprendrez qu'à la leçon suivante et vous répond simplement par un laconique placement du curseur au début de la ligne suivante.
La situation est tout à fait semblable à celle qui consiste à commander un "bortsch" dans un restaurant vietnamien; vous vous exposez au mieux à un silence poli et, au pire, à une injure prononcée dans la langue maternelle du serveur.
Essayez :

B O R 1 S C H ENTREE

La machine répond :

```
Error 2
OK
```

(Voir les codes d'erreurs en annexe 8)

■ Vos premiers mots en BASIC - MO5

La suite de ces leçons va vous apprendre la plupart des mots et des phrases BASIC vous permettant d'écrire vos programmes. Vous pourrez aussi consulter très vite les fiches de référence d'abord réservées au MO-iste confirmé que vous n'allez pas tarder à devenir. (Ces fiches donnent *tous* les détails sur chaque commande).

Voici, pour commencer, quelques *commandes d'affichage* permettant de jouer avec les couleurs :

C L S ENTREE

... et l'écran s'efface complètement (tout comme si vous aviez appuyé sur la touche RAZ).
CLS est l'abréviation de "*cl*ear *s*creen" ("effacer l'écran").

C O L O R 1 ENTREE

... et vous avez l'impression que rien ne s'est passé ! MAIS, à partir de maintenant la machine écrit en rouge.

C O L O R 1 , 2 ENTREE

... et à partir de maintenant, chaque caractère écrit le sera en rouge sur fond vert.
Vous disposez ainsi de 16 couleurs, tant pour l'"encre"d'écriture (la "forme") que pour le "papier" sur lequel vous écrivez (le "fond"). Chaque couleur correspond à un nombre de Ø à 15. Consultez la fiche "COLOR" pour connaître cette correspondance.

S C R E E N 1 , 2 , 3 ENTREE

... et les *caractères* sont alors écrits en rouge (couleur 1) sur un fond vert (couleur 2), le pourtour de votre page d'écriture étant alors colorié en jaune (couleur 3).

Attention : SCREEN A, B, C modifie les couleurs de l'écran au moment de son exécution et A devient la couleur d'écriture, B la couleur de tout le fond et C la couleur du pourtour.
COLOR A, B ne modifie aucune couleur déjà affichée, chaque caractère écrit *ensuite* le sera en couleur A sur fond de couleur B.

Remarque : A la mise sous tension de la machine et du téléviseur, les couleurs normalement choisies sont : bleu (4) pour l'écriture, bleu ciel (6) pour le fond et le pourtour (le nom "scientifique" du bleu ciel est *cyan*).

■ PRINT

Le mot PRINT suivi d'un nombre ou d'un calcul entraîne l'écriture de ce nombre ou du nombre résultat de ce calcul. Plusieurs nombres ou calculs peuvent être demandés à la fois, en les séparant par des virgules.

```
PRINT 8*7,8+7,8-7
```

... et la machine répond par écrit :

```
 56            15            1
OK
```

Pour des raisons de rapidité d'écriture, la suite de cinq lettres : PRINT peut être remplacée par un simple point d'interrogation ?.

Exercices

1. Expérimentez l'effet des lignes de commande suivantes :

```
CLS ENTREE
COLOR 1 ENTREE
'ROUGE ENTREE
COLOR 2 ENTREE
'VERT ENTREE
COLOR 1+2 ENTREE
'JAUNE ENTREE
COLOR 4 ENTREE
'BLEU ENTREE
COLOR 4+1 ENTREE
'VIOLET ENTREE
```

(à partir de maintenant nous écrivons les lignes de commande sans particulariser chaque caractère par un cadre ; c'est tout de même plus lisible !).

Remarque : Toute ligne commençant par une apostrophe ' n'est pas "interprétée" par la machine.

2. Pour en voir de toutes les couleurs...
...essayez, selon votre humeur, des SCREEN X, Y, Z de toutes sortes.
Par exemple :

Après `SCREEN 0,7,0` ENTREE
écrivez `'CONDOLEANCES !` ENTREE
Après `SCREEN 4,6,6` ENTREE
écrivez `'LE POURTOUR NE SE VOIT PAS !` ENTREE
Après `SCREEN 4,1,2` ENTREE
écrivez `'A VOS LUNETTES !` ENTREE
Après `SCREEN 4,4,6` ENTREE
écrivez `'C'EST SYMPATHIQUE !` ENTREE

Mais après être revenu de votre surprise, dépêchez-vous d'y voir plus clair en frappant successivement sur :

ENTREE `COLOR 4,6` ENTREE

(Réfléchissez vous allez comprendre)

3. Sachant que vous disposez de 16 possibilités pour l'écriture, de 16 possibilités pour le fond et de 16 possibilités pour le pourtour, de combien de possibilités différentes disposez-vous au total ?
Pour avoir la réponse frappez :

`? 16 * 16 * 16` ENTREE

Vous voyez que vos dons artistiques vont avoir de quoi s'exprimer !

4. Effectuez les opérations commandées ci-dessous :

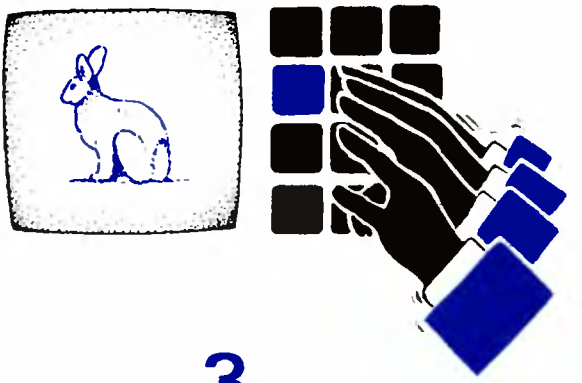
```
? 7*13*11
? 142857*2, 142857*3, 142857*4
? 142857*5, 142857*6, 142857*7
? 12*9+3, 123*9+4, 1234*9+5, 12345*9+6
```

Attention

■ Dans l'écriture de "nombres-à-virgule", la partie entière et la partie décimale doivent être séparées par un *point* (et non par une virgule) !
Ainsi : 3.14 est le nombre 314/1ØØ.
3,14 est un couple de nombres (le nombre "trois" et le nombre "quatorze").
Comparer l'effet des deux instructions ? 3.14 et ? 3,14.
■ Le signe de division est le signe /
• Le signe ■ n'est pas un signe d'opération.
• Le signe de la multiplication est le signe * ; il ne faut pas l'oublier ! Par exemple "2 fois A" s'écrit 2*A et non 2 A qui provoquera une erreur.

Fiches de référence pouvant être consultées :

* COLOR, SCREEN, page 135
* +, −, *, /, @ , MOD, page 172
** CLS, CONSOLE, page 142
* ATTRB, page 138



3

Programmer

Est-il bien raisonnable de commander une chose à l'ordinateur qui l'exécute, puis de commander une autre chose, puis d'en commander une autre... et ainsi de suite ? Tout cela n'est pas très rapide et vous avez sans doute vu fonctionner des ordinateurs bien plus agréablement ; il se passait alors des tas de choses sans que personne n'intervienne jamais sur le clavier.

■ Préparez votre menu à l'avance

La rapidité d'action de la machine est en effet intéressante à condition de lui avoir *d'abord* communiqué une (longue) liste de commandes. Exactement comme vous pouvez le faire au restaurant en commandant un menu complet avant le repas : une fois la liste établie vous n'avez plus qu'à dire : "Vous pouvez servir !" et chacun des plats se succède alors sans que vous ayez à intervenir.

1. Un petit vin de pays et une carafe d'eau.
2. Foie gras frais maison.

(Ne lésinons pas, nous n'apprenons pas la programmation tous les jours)

3. Confit de canard et pommes à l'ail.

4. Salade.
5. Plateau de fromages.

(Juste un peu, pour la tradition !)

6. Sorbet citron.
7. Café.
8. Fin.

(Non merci, je n'ai plus faim !)

■ Une suite de commandes numérotées

Pour en revenir à votre machine, sachez donc ceci :
Pour communiquer un "programme d'action" à votre ordinateur il vous suffit de numéroter chacune des commandes que vous lui donnez.
Ainsi : toute ligne commençant par un numéro est interprétée par la machine comme faisant partie d'un programme.
Lorsque vous frappez sur la touche ENTREE elle se contente de ranger cette ligne dans une espèce de cahier de texte : elle classe ainsi toutes les lignes numérotées que vous "entrez"... jusqu'à ce que vous lui commandiez l'exécution de toutes ces lignes les unes après les autres.
La commande qui lui intime l'ordre d'exécution est : RUN ENTREE
(le verbe "run" signifie, en anglais : "vas-y, cours !").
Pour être "branché" côté programmation vous devez enfin savoir qu'une commande faisant partie d'un programme s'appelle une *instruction*.
Il est temps maintenant de passer à un exemple "concret".
Écriture d'un premier programme :
— Depuis combien d'heures êtes-vous né ?
Pour répondre à cette angoissante question, vous pouvez frapper un programme qui ressemble à celui que j'ai réalisé pour un ami âgé de 39 ans 8 mois et 23 jours.
Frappez donc successivement :

```
1 ANS=39 ENTREE
2 MOIS=8 ENTREE
3 JOURS=23 ENTREE
4 HEURES=(365*ANS+30*MOIS+JOURS)*24 ENTREE
5 PRINT HEURES ENTREE
6 END ENTREE
```

Soyons juste : le résultat est approché à quelques jours près (certaines années sont bissextiles et les mois n'ont pas tous 30 jours) C'est-à-dire

à quelques centaines d'heures près ! Si vous trouvez cela grave, essayez de réaliser un meilleur programme (nous vous conseillons d'attendre pour cela quelques leçons).

Remarques :

- Chaque fois que vous tapez sur ENTREE, le curseur passe au début de la ligne suivante ; la ligne précédente est alors enregistrée par la machine dans son espèce de "cahier de texte".
- L'instruction END signale à la machine qu'elle doit s'arrêter de travailler.
- Les "mots" ANS, MOIS, JOURS, HEURES sont des noms de *variables*. Souvenez-vous, en mathématiques, à l'école, les noms de variables n'avaient qu'une seule lettre (x, a ou p) ou bien une lettre suivie d'un chiffre (PØ, R1...) ; cela manquait de poésie ou même d'humour. En fabriquant des programmes vous allez pouvoir vous venger en faisant preuve d'imagination !
- N'oubliez pas de distinguer les touches Ø et O !

Attention

Pour entrer en mémoire une ligne de programme, il faut frapper sur la touche ENTREE ; sinon l'ordre est écrit sur l'écran mais il n'aura aucune chance d'être exécuté.

N'utilisez pas les flèches pour déplacer le curseur d'une ligne à l'autre lorsque vous écrivez un programme.

Si jamais vous désobéissez à ce conseil, vous aurez des ennuis qui se traduiront par une erreur de syntaxe qui vous paraîtra mystérieuse.

Un conseil : Si vous n'arrivez pas à détecter l'erreur signalée par :

```
Error 2 in 50
OK
```

... n'hésitez pas à supprimer l'instruction 5Ø de la mémoire en frappant :

```
5Ø ENTREE
```

Puis retaper une nouvelle instruction 5Ø si vous le jugez nécessaire.

■ Exécution du programme

Lorsque vous avez fini d'écrire un programme, il reste affiché à l'écran ; vous pouvez l'effacer de l'écran en appuyant sur la touche : RAZ

N'ayez pas peur ! Il a disparu de l'écran, mais il est encore *en mémoire* quelque part dans la machine.

La preuve c'est que vous pouvez l'exécuter en frappant :

RUN ENTREE

... et la machine affiche alors le résultat attendu :

```
347952
OK
```

■ Lecture et modification du programme

Si vous souhaitez *revoir le programme préalablement mis en mémoire,* vous pouvez le faire écrire sur l'écran grâce à la commande LIST. Frappez :

LIST ENTREE

... et la machine affiche sur l'écran la liste des instructions qu'elle a mémorisées (et cela dans l'ordre des numéros figurant en début de ligne).

Vous pouvez aussi *demander l'écriture d'une seule ligne.*

Par exemple frappez :

LIST 2 ENTREE

... et la machine affichera

```
2 MOIS=8
OK
```

Si vous voulez alors *modifier* cette ligne numéro 2, vous pouvez le faire à l'aide des touches de déplacement du curseur.

Ainsi frappez :

↑ ↑ → → → → → → → → 4 ENTREE

(Ne pas oublier de frapper sur ENTREE , sinon l'instruction n'est pas modifiée.)

Si alors vous frappez :

LIST 2 ENTREE

la machine affichera :

```
2 MOIS=4
OK
```

Pour remplacer une instruction par une autre, il suffit d'en frapper une nouvelle en utilisant le même numéro : comme sur un magnétophone, c'est la dernière chose enregistrée qui reste. Vous pouvez donc refrapper chacune des 3 premières lignes de ce programme pour l'adapter à telle ou telle de vos amies.
Enfin *pour effacer une instruction*, il suffit de frapper son numéro suivi immédiatement de ENTREE (En fait, on remplace une instruction par une instruction vide). L'effacement a lieu dans la mémoire et non sur l'écran !

■ La touche BASIC

Cette touche permet d'écrire plus rapidement vos programmes.
Ainsi au lieu d'écrire :
S C R E E N
vous pouvez frapper simultanément sur
BASIC - SCREEN
Ainsi pour chaque "instruction" écrite sur la partie basse de chaque touche.

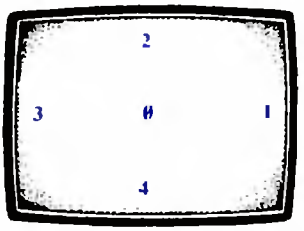
Exercices

1. Écrire et exécuter le programme suivant :

```
1 SCREEN 0,1,0
2 SCREEN 0,2,0
3 SCREEN 0,3,0
4 SCREEN 0,4,0
5 SCREEN 0,5,0
6 SCREEN 0,6,0
7 SCREEN 0,7,0
```

Une fois frappé n'oubliez pas de taper R U N ENTREE pour exécuter ce programme.

2. Modifier le programme précédent de manière qu'à l'exécution le fond reste le même et les caractères écrits changent de couleur.
3. Écrire un programme qui affiche en rouge le nombre 17, en bleu son carré (17 * 17) et en noir son cube (17 * 17 * 17).
4. Écrire un programme qui affiche ceci :



Utilisez pour cela l'instruction :
`LOCATE 20,1`
Cette instruction positionne le curseur sur la position numéro 20 de la ligne numéro 1.
Plus généralement LOCATE X, Y amène le curseur sur la position numéro X de la ligne numéro Y.

Attention

■ Avant de frapper un nouveau programme, il vaut mieux vous débarrasser de celui que vous avez précédemment frappé et qui ne vous sert plus qu'à encombrer la mémoire du MO5.
La commande qui efface les programmes en mémoire est :
N E W ENTREE
Vous pouvez aussi éteindre puis rallumer votre MO5.

Attention

■ *Frappez PUIS exécutez un programme.*
Retenez bien ceci : Pour arriver à "faire marcher" un programme il faut bien distinguer 2 étapes :
— 1re étape : Frappe du programme.
Dans cette étape les lignes sont frappées les unes après les autres ; chaque ligne commence par un chiffre. La machine n'exécute aucun ordre ; elle ne fait que recopier en mémoire chacune des lignes d'instruction.
— 2e étape : Demande d'exécution du programme.
Cette étape débute lorsque vous frappez sur RUN ENTREE . La machine exécute alors la suite des instructions enregistrées lors de la première étape.

■ Les lignes d'instructions écrites sur l'écran ne sont pas forcément les lignes d'instructions qui sont entrées en mémoire. Tout dépend de la position du curseur lorsque vous appuyez sur ENTREE

Pour vérifier ce qui est réellement en mémoire, commandez
LIST ENTREE

■ Pour l'instant n'écrivez qu'une instruction par ligne.
En effet, si vous écrivez la ligne...

```
3 COLOR 1 PRINT 17
```

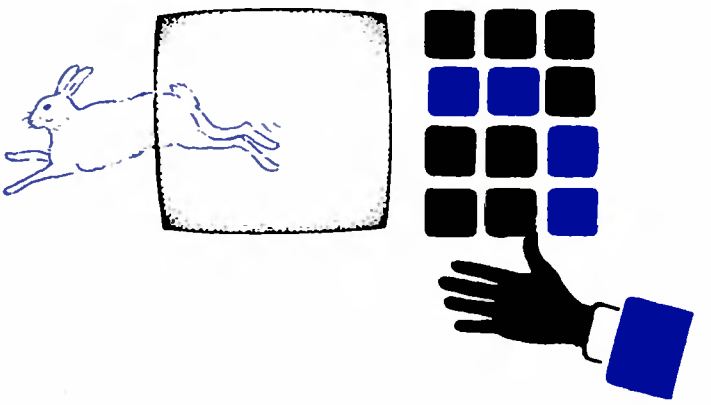
... la machine ne comprendra pas le morceau de phrase "PRINT 17"
Il faut écrire :

```
3 COLOR 1
4 PRINT 17
```

Fiches de référence pouvant être consultées :

** LOCATE, page 145
* LIST, DELETE, NEW, page 155
** Ligne de programme, page 146

Page blanche



4

Faire et refaire

Vous voulez que votre ordinateur travaille beaucoup, mais vous ne pouvez pas passer votre vie à enregistrer des instructions dans votre micro-ordinateur !
Comme il les exécute ensuite très, très vite, son travail est vraiment dérisoire en regard du temps que vous avez passé à écrire au clavier.

■ GOTO...

En fait, sa rapidité d'exécution n'a de véritable intérêt que s'il exécute de très nombreuses fois une même petite suite d'instructions.
L'instruction lui intimant l'ordre d'exécuter de nouveau une certaine instruction, se présente sous la forme :
"GOTO un numéro d'instruction"
("go to" signifie "aller en").
Après ce "saut", il continue naturellement à exécuter les instructions qui suivent dans l'ordre de leurs numéros.

■ Un programme de clignotant

Voici un programme de clignotant éternel illustrant l'utilisation du "GOTO".

```
1 LOCATE 20,12
2 COLOR 1,2
3 PRINT 0
4 LOCATE 20,12
5 COLOR 2,2
6 PRINT 0
7 GOTO 1
```

le curseur se positionne au milieu de l'écran (position numéro 20 de la ligne numéro 12).
on écrira rouge sur fond vert
écrire le nombre zéro

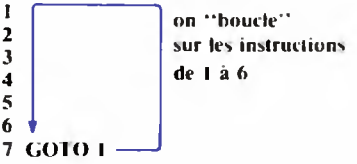
on écrira vert sur fond vert
écrire le nombre zéro
aller refaire l'instruction numéro 1.

Remarque : C'est encore plus joli si vous complétez la première instruction comme ceci :

```
1 LOCATE 20,12 : ATTRB 1,1
```

(Voir la fiche de référence ATTRB).

Vous venez de réaliser votre première *boucle* de programme :



■ STOP

Évidemment le spectacle n'est pas très varié et, de plus, il n'est pas prêt de se terminer.
Vous pouvez l'arrêter en intervenant "manuellement" sur le clavier.
Dès que vous frappez la touche STOP, l'exécution s'arrête.
C'est là qu'une surprise vous attend : dès que vous frappez une autre touche, la machine se remet à marcher et reprend l'exécution indéfinie du programme.
Vous êtes alors désespéré, coincé dans une *boucle sans fin* que vous pouvez juste interrompre de temps en temps pour vous reposer les yeux !

Cependant, il y a un moyen de vous en sortir sans éteindre carrément la machine. Ce moyen est un peu curieux mais c'est ainsi : il suffit d'appuyer simultanément sur les touches CNT et C (en langage parlé : "*cont*rol cé").

CNT + C

... et la machine interrompt définitivement l'exécution du programme et indique même le numéro de l'instruction où elle a été interrompue ; elle affiche, par exemple, le message :

```
Break in 5
OK
```

■ Un programme de compteur

Refaire toujours exactement la même chose, cela n'est pas franchement excitant. Mais si un petit quelque chose n'est plus le même à chaque exécution de la boucle, alors s'ouvre devant nous une formidable variété de possibilités.

Témoin, le petit programme suivant.

Le changement consiste à augmenter de 1 la valeur d'une variable entre deux exécutions de la boucle.

Cette augmentation est commandée par l'instruction A = A + 1.

(le fonctionnement détaillé de cette commande sera expliqué à la leçon 7).

`1 A=0`	A prend la valeur Ø
`2 LOCATE 20,12`	positionnement du curseur (voir remarque du clignotant)
`3 PRINT A`	écriture de la valeur de A
`4 A=A+1`	A augmente sa valeur de 1
`5 GOTO 2`	refaire l'instruction numéro 2.

Pour arrêter l'exécution, vous pouvez frapper la touche STOP quand vous le voulez. D'où l'idée d'un petit jeu de réflexe utilisant ce programme.

■ Un jeu de réflexe

• Enregistrez le programme précédent et frappez :

R U N ENTREE

• Essayez d'arrêter l'exécution du programme sur l'affichage d'un nombre rond (100, 200,...) en frappant sur :

STOP

• En frappant une autre touche, le programme redémarre et vous pouvez continuer à jouer chacun votre tour en essayant de viser de mieux en mieux.

Exercices 

1. Nous vous proposons de modifier quelque peu le programme du compteur (utiliser l'affichage de la liste des instructions — LIST — et le "curseur-pleine-page", c'est-à-dire "la possibilité, pour le curseur de se déplacer et d'écrire sur toute la page").
Pour chaque modification, essayez d'abord de prévoir le résultat de l'exécution ; ensuite déclenchez l'exécution pour vérifier vos prévisions :

a) Remplacez l'instruction numéro 1 par :

```
1 A=2
```

b) Remplacez l'instruction numéro 4 par :

```
4 A=A*2
```

c) Remplacez l'instruction numéro 4 par :

```
4 A=A+A
```

d) Remplacez l'instruction numéro 4 par :

```
4 A=A*A
```

e) Supprimez l'instruction numéro 2 et remplacez l'instruction 5 par :

```
5 GOTO 3
```

2. Pour en voir de *toutes les couleurs*, analysez, frappez et exécutez le programme suivant :

```
1 I=1
2 SCREEN I-1,I+1
3 I=I+1
4 GOTO 2
```

(Évidemment après une quinzaine de changements de décor, il y a quelque chose qui ne va pas ! La leçon 6 vous sortira de ce mauvais pas).

3. Analysez, frappez et exécutez le programme suivant :

```
1 CLS
2 K=0
3 COLOR K
4 PRINT 888888
5 K=K+1
6 GOTO 3
```

4. Puisque vous commencez à comprendre ce qu'est un programme, il est temps de faire un peu d'humour avec le programme-gag suivant :

```
1 GOTO 1
```

Fiche de référence pouvant être consultée :

* RUN, STOP, END, CNT-C, GOTO, page 153

Page blanche

Résumé et conseils aux apprentis programmeurs

■ Frapper un programme et le faire exécuter

• *Pour enregistrer votre programme*, instruction par instruction :
— Toute instruction doit être numérotée en début de ligne.
— Elle est enregistrée en mémoire dès que vous frappez la touche ENTREE.

• *Pour faire exécuter votre programme* par la machine frappez :
RUN ENTREE
— Si vous voulez interrompre l'exécution frappez STOP ou CNT-C

• *Pour afficher la liste des instructions* déjà enregistrées en mémoire frappez :
LIST ENTREE

• *Pour modifier une instruction* :
— Affichez cette instruction en commandant par exemple LIST 7 ENTREE (et l'instruction numéro 7 s'affiche).
— Utilisez le curseur plein-écran pour changer, ajouter ou supprimer des caractères.
— N'oubliez pas de frapper ENTREE pour enregistrer la nouvelle instruction. Vous pouvez faire lister l'instruction pour vérifier.

• *Pour effacer l'instruction numéro 12 frappez :*
1 2 ENTREE

■ Attention à quelques erreurs classiques

• Si vous frappez
1 RUN ENTREE, il ne se passera rien.
La seule chose que fera la machine, est d'enregistrer l'instruction numéro 1 dans sa mémoire.

• Si vous frappez
10+25 ENTREE

N'espérez pas que la machine fasse la somme de 10 et de 25. Puisque la ligne commence par un numéro, elle interprète 10 comme un numéro d'instruction et elle range alors l'instruction qui s'énonce : "+ 25". D'ailleurs elle ne comprendra rien à cette instruction.
! Alarme !
Après avoir frappé 1 0 + 2 5 ENTREE, il y a plus grave.
Si jamais, vous aviez déjà enregistré une instruction numéro 10, elle a été effacée par cette bêtise : "+ 25".

• ATTENTION : *Ce qui est affiché sur l'écran n'est pas forcément en mémoire et inversement.*

On oublie souvent par exemple, d'effacer une instruction dont on a modifié le numéro.
Soit par exemple, l'instruction affichée :
7 COLOR 1,2
Si on désire plutôt donner à cette instruction le numéro 9 (parce qu'elle doit suivre et non précéder l'instruction 8), alors il faut faire successivement ceci :
— Amenez le curseur sous le 7 en début de ligne
7 COLOR 1,2
— Frappez le chiffre 9
9 COLOR 1,2
— Frappez ENTREE
L'instruction 9 COLOR 1,2 est alors enregistrée.
Attention !
L'instruction numéro 7 n'est plus affichée à l'écran mais elle existe toujours en mémoire.

Il faut donc l'effacer en frappant :

7 ENTREE

Il n'y a plus alors d'instruction numéro 7 en mémoire, mais il y a une nouvelle instruction numéro 9.

■ Quelques conseils

• Pour savoir ce que fait un programme ou pour le corriger *mettez-vous à la place de la machine* :
Exécutez fidèlement les instructions les unes après les autres en imitant les actions commandées à la machine.

• N'hésitez pas à *numéroter vos instructions de 10 en 10* (ou de 5 en 5) afin de pouvoir ultérieurement insérer de nouvelles instructions intermédiaires.

• Cette habitude vous aidera aussi à *écrire d'abord l'architecture générale* de votre programme, *puis* (lorsqu'il marche bien) à le *"raffiner"* par des instructions qui améliorent ses performances ou sa présentation. Ne cherchez donc pas tout de suite à réaliser un programme sophistiqué.
Soyez, à la fois, humble et sûr de vous : il n'y a rien de mystérieux ; la machine fait exactement ce que vous lui commandez de faire !

• Utilisez la possibilité de *placer des commentaires* dans vos instructions grâce à l'apostrophe.

```
9 COLOR 1,2    ' rouge sur vert
```

En début de ligne l'apostrophe peut être remplacée par REM (début de *"remarque"*). Par exemple :

```
1 REM JEU DU COMPTEUR
```

... et l'instruction 1 n'est ni interprétée ni exécutée par la machine.

■ Voici les instructions et commandes que vous connaissez déjà :

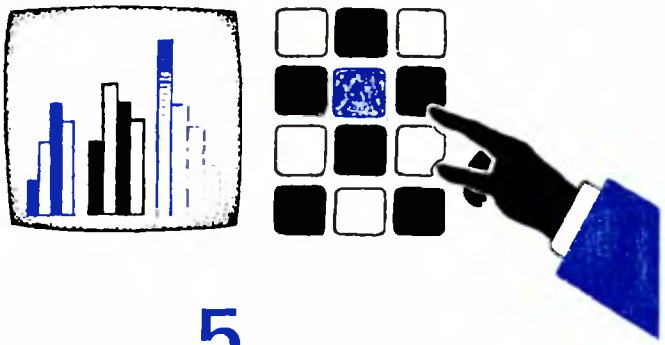
CLS
LOCATE 12,17
COLOR 4
COLOR 5,12
SCREEN 2,11,8
X = X + 1
PRINT 17

PRINT X
GOTO 4
RUN
LIST
LIST 8
STOP
CNT-C

Si vous vous sentez plus à l'aise, n'hésitez pas à consulter les *fiches de référence* pour avoir plus de détails sur ces instructions.

Consultez aussi l'*Index* en dernière page.

N'oubliez pas de frapper ENTREE pour indiquer la fin de chaque commande ou de chaque instruction. C'est la touche qui fait toute la différence entre une machine à écrire et un ordinateur !
C'est une sorte de point d'exécution terminant une phrase et ordonnant d'agir de manière appropriée.



5

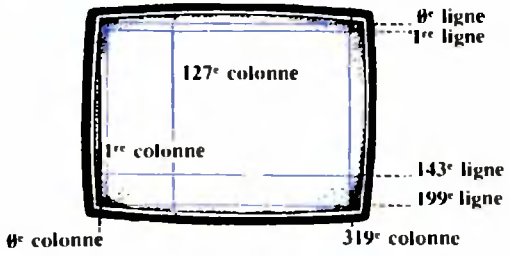
Graphiques

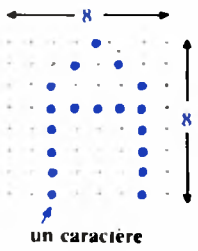
Peut-être avez vous vu chez des amis de magnifiques dessins en couleurs, affichés sur leur écran de télévision. On ne vous a pas encore parlé de la manière d'obtenir de telles œuvres d'art. Vous allez bientôt tout savoir sur le sujet.

■ Une feuille de dessin

Pratiquement, tout se passe comme si votre écran était une feuille de dessin sur laquelle, vous pouvez marquer des points de couleur en 64 ØØØ endroits précis. Précisément, chaque position de caractère est constituée de 8 points en ligne et 8 points en colonne et vous disposez de 25 lignes de 4Ø caractères.
Votre "feuille" est donc constituée de 2ØØ lignes et de 32Ø points.

Ces lignes et ces colonnes de points sont "numérotées" comme indiqué sur le schéma suivant :





■ Points

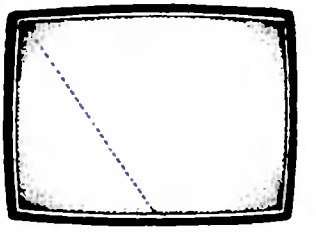
La commande :

```
PSET (127,143) ENTREE
```

a pour effet, d'allumer le point situé sur la 127e colonne et la 143e ligne.
La couleur d'allumage est celle précédemment choisie pour l'écriture.

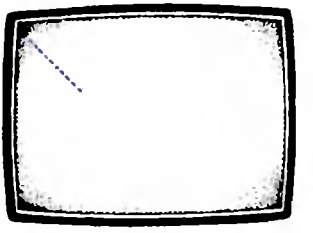
Voici trois programmes utilisant cette instruction et leur effet sur l'écran :

```
1 I=0 : CLS
2 COLOR 1
3 PSET (5*I,5*I)
4 I=I+1
5 GOTO 3
```



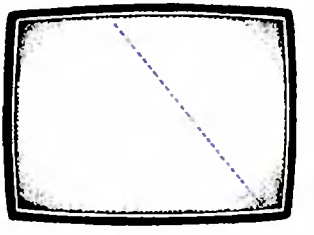
Tracé d'une "diagonale" de points rouges puis de quelques points en bas de l'écran.
Arrêtez l'exécution en frappant CNT-C ; sinon après quelques temps vous obtiendrez un message d'erreur numéro 6.

```
1 I=0
2 CLS
3 COLOR I
4 PSET (I*5,I*5)
5 I=I+1
6 GOTO 3
```



Tracé de 16 points de couleurs différentes puis arrêt sur un message d'erreur numéro 5, écrit en orange (il y a eu essai d'écrire avec la couleur numéro 16 qui n'existe pas).

```
1 I=319
2 COLOR 1
3 PSET(I,I-120)
4 I=I-1
5 GOTO 3
```



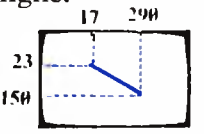
Tracé d'une "diagonale" issue du coin droit en bas et arrêt sur un message d'erreur numéro 5 (il y a eu essai de tracer un point de coordonnées négatives).

■ Lignes

La commande :

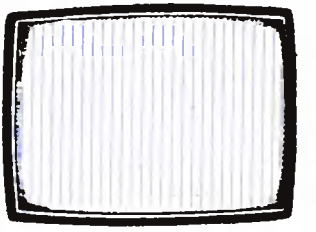
```
LINE (17,23)-(290,150) ENTREE
```

a pour effet de tracer un segment dont une extrémité est le point de la 17e colonne et 23e ligne, et dont l'autre extrémité est le point de la 290e colonne et 150e ligne.



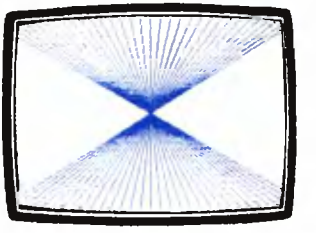
Voici trois programmes utilisant cette instruction et leur effet sur l'écran :

```
1 C=0
2 COLOR 1
3 LINE(C,0)-(C,199)
4 C=C+10
5 GOTO 3
```



Même programme avec :

```
3 LINE (C,0)-(319-C,199)
```



Même programme avec :

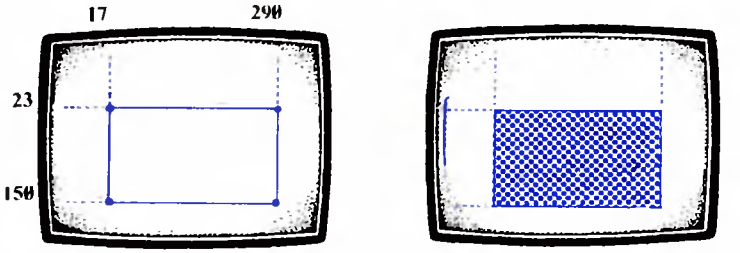
```
3 LINE (C,0)-(0,C)
```

■ Boîtes

La commande :

```
BOX (17,23)-(290,150)
```
ENTREE

a pour effet, de tracer le contour d'un rectangle ayant le point (17,23) et le point (290, 150) comme sommets opposés.



La commande :

```
BOXF(17,23)-(290,150)
```
ENTREE

a presque le même effet mais l'intérieur du rectangle est colorié (de la couleur choisie précédemment pour l'écriture).

Remarque :

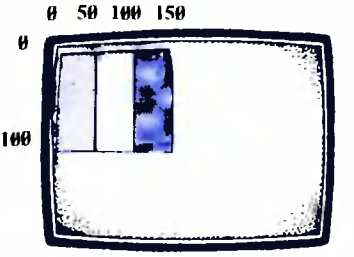
BOX signifie "boîte" et F est la première lettre de full ("pleine")

■ Drapeaux et emboîtements

Voici trois programmes utilisant cette instruction et leur effet sur l'écran :

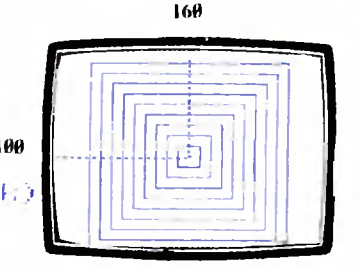
drapeaux

```
1 COLOR 4
2 BOXF(0,0)-(50,100)
3 COLOR 7
4 BOXF(50,100)-(100,0)
5 COLOR 1
6 BOXF(100,0)-(150,100)
```



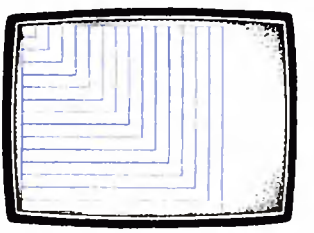
carrés centrés

```
1 B=0
2 COLOR 1
3 BOX(80-B,100-B)-(80+B,100+B)
4 B=B+10
5 GOTO 3
```



essayez le même programme avec :

```
2 COLOR B/16
5 GOTO 2
```

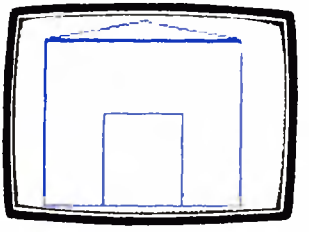


carrés en coin

```
1 B=15 : LOCATE 0,20
2 COLOR B
3 BOXF(0,0)-(16*B,19*B)
4 B=B-1
5 GOTO 2
```

Exercices

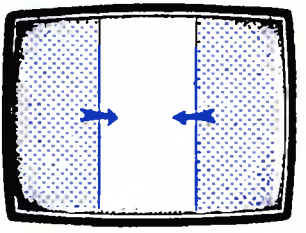
1. Écrire un programme qui dessine une maison :



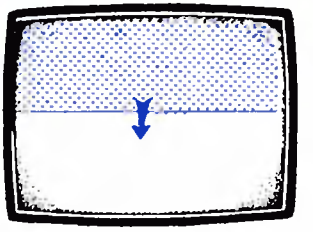
(Vous ne saurez colorier l'intérieur du triangle constituant le toit en rouge qu'après la leçon suivante).

2. Écrire un programme qui "ferme le rideau"

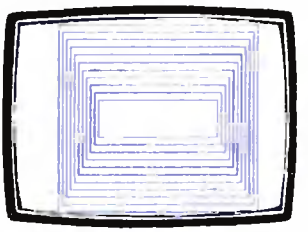
comme ceci :



ou comme cela :



3. Écrire un programme dessinant des boîtes pleines emboîtées.



4. Que dessine le programme suivant ?

```
1 CLS
2 COLOR 0
3 LINE(100,100)-(100,50)
4 LINE -(200,50)
5 LINE -(150,20)
6 LINE -(100,50)
7 LINE -(200,100)
8 LINE -(200,50)
9 LINE -(100,100)
10 LINE -(200,100)
```

Sachez que l'instruction LINE – (X, Y) a pour effet de tracer une ligne depuis le dernier point dessiné jusqu'au point (X, Y).

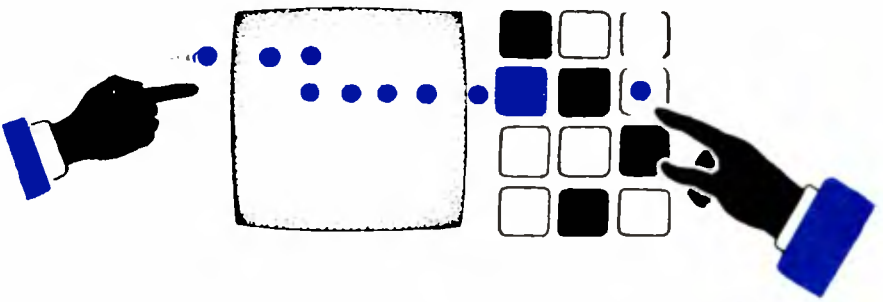
Attention

■ Lorsque vous listez et lorsque vous commandez l'exécution d'un programme, il y a du texte écrit sur l'écran. Si l'exécution du programme provoque le tracé de dessins, il peut y avoir superposition du texte et des dessins et ce n'est pas joli du tout. Il vaut donc mieux commencer tout programme de dessin par l'instruction qui *C*hasse *L*a *S*aleté :
1 CLS (en anglais : *Cl*ear *S*creen)

■ Lorsque vous demandez le tracé sur l'écran blanc d'un segment rouge puis d'un segment bleu coupant le rouge, il y a une petit bavure au voisinage de l'intersection. (Les fiches de référence ** vous en expliqueront les causes).

Fiches de références pouvant être consultées :

* PSET, LINE, BOX, BOXF, page 201
** Fond et écriture, page 203



6
Répétitions et aiguillages

Dans la leçon précédente, quelques programmes donnés en exemple s'arrêtent sur un message d'erreur.
Il doit pourtant bien avoir un moyen de refaire quelque chose un certain nombre de fois, au lieu d'avoir à le refaire soit indéfiniment soit jusqu'à ce qu'un incident vienne interrompre le cours des événements.

■ Répétez 7 fois

Précisément : Pour répéter 7 fois une suite d'instructions (ici numérotées de 11 à 19), on écrit :

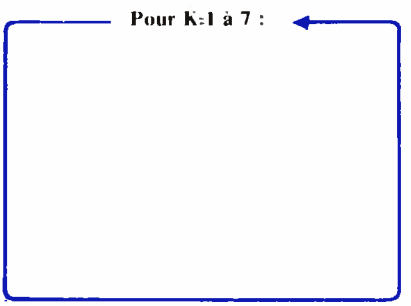
```
10 FOR K = 1 TO 7
11 '
12 '
13 '
14 '
15 '
16 '
17 '
18 '
19 '
20 NEXT K
```

10 *Pour K = 1 à 7 exécuter des instructions :* 11, 12, 13, 14, 15, 16, 17, 18, 19

20 *Revenir en* 10 avec la valeur de K suivante (c'est-à-dire augmentée de 1) ; mais lorsque K vaut 7 + 1, passer à l'instruction suivante.

La traduction en français des deux instructions "FOR..." et "NEXT..." est donnée ci-dessus en couleur.
On peut aussi imaginer que le programme précédent est équivalent à la suite d'instructions :

K = 1
suite des instructions 11 à 19
K = 2
suite des instructions 11 à 19
K = 3
suite des instructions 11 à 19
K = 4
suite des instructions 11 à 19
K = 5
suite des instructions 11 à 19
K = 6
suite des instructions 11 à 19
K = 7
suite des instructions 11 à 19



On dit que l'on exécute une "boucle" pour K variant de 1 à 7. Dans la leçon 4, nous avions déjà rencontré une "boucle" sans fin. Maintenant nous savons boucler un certain nombre de fois précisé par l'instruction FOR...

FOR... indique le début de la boucle ; NEXT indique la fin de la boucle c'est-à-dire l'endroit où l'exécution est renvoyée en début de boucle si la variable n'a pas atteint sa valeur maximum.

■ Les 16 couleurs

Vous pouvez alors reprendre chacun des programmes donnés dans la leçon précédente en remplaçant le brutal 6 GOTO 2 par quelque chose du type :

```
2 FOR C=0 TO 15
:
6 NEXT C
```

La variable C représente la couleur d'écriture. Pour rendre les programmes plus lisibles, on peut d'ailleurs lui donner un nom de 7 lettres au lieu d'une. Comme ceci :

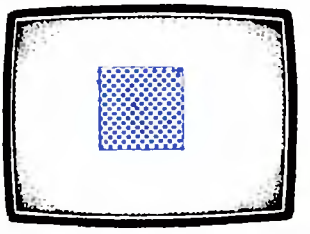
```
2 FOR COULEUR=0 TO 15
:
6 NEXT COULEUR
```

Pour l'exemple, voici deux programmes de démonstrations des 16 couleurs du MO 5.

```
1 CLS
2 FOR COULEUR=0 TO 15
3 BOXF(16*COULEUR,0)-(16*COULEUR+15,199),
COULEUR
4 NEXT COULEUR
5 END
```

(Les 16 bandes colorées)

```
1 CLS
2 FOR COULEUR=0 TO 15
3 BOXF(100,40)-(220,160),COULEUR
4 NEXT COULEUR
5 GOTO2
```



(pulsation d'un carré de couleur)

■ Si... alors... sinon...

Les boucles commandées par GOTO... équivalent à :
RÉPÉTER quelque chose A L'INFINI.
Les boucles commandées par FOR... NEXT équivalent à :
RÉPÉTER quelque chose UN CERTAIN NOMBRE DE FOIS.
Mais il arrive que l'on ne sache pas à l'avance le nombre de répétitions que l'on désire. Par exemple, on peut désirer continuer JUSQU'A CE QU'une certaine condition soit réalisée (ou un certain objectif atteint) ou bien TANT QU'une certaine condition reste vraie (par exemple, on dispose encore de carburant). Ou bien encore, on peut vouloir faire telle ou telle chose SELON la valeur d'une certaine variable.

Le langage BASIC dispose d'une instruction de "branchement conditionnel" pour s'adapter aux différents désirs énoncés ci-dessus.
Cette instruction se présente ainsi :

```
IF X=3 THEN GOTO 7 ELSE PRINT X
```

A l'exécution, la machine calcule si la condition énoncée après IF est vraie ou fausse. (ici "X est-il égal à 3"?).

Si la condition est vraie *alors* elle exécute l'instruction écrite après THEN (ici elle va en 7 suivre son destin).

Si la condition est fausse (ELSE signifie "sinon"), elle exécute l'instruction écrite après ELSE (ici elle écrit la valeur de X), puis elle passe à l'instruction suivante.

■ Un programme de carrés géométriquement emboîtés

Le programme suivant dessine un carré puis, à l'intérieur, un carré de côté égal à 90 % du précédent et ainsi de suite. On arrête de dessiner lorsque le carré devient trop petit (précisément lorsqu'il devient de la taille d'un caractère : son côté vaut alors 8 points).

```
5 CLS
10 COTE=100
15 BOX(0,0)-(COTE,COTE)
20 COTE=0.90*COTE
25 IF COTE<8 THEN GOTO 30 ELSE GOTO 15
30 LOCATE 0,15
35 END
```

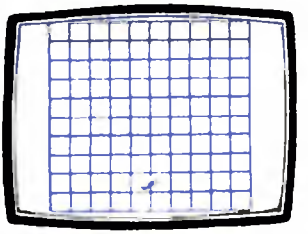
Remarques :

• L'instruction IF... THEN... ELSE... admet de multiples variantes en particulier pour simplifier son écriture. Mais il vaut mieux attendre d'avoir une certaine habitude de la programmation pour les utiliser (voir la fiche de référence).

• Dans l'écriture décimale des nombres, notre virgule française est remplacée par un point anglo-saxon.

Exercices

1. Dessiner un quadrillage régulier sur l'écran.



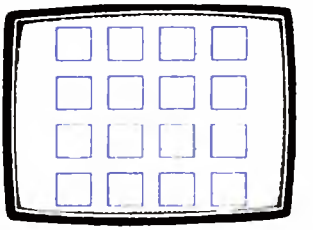
Il vous faudra pour cela utiliser 2 boucles successives du genre :

```
FOR X=0 TO 10
...
NEXT X

FOR Y=0 TO 10
...
NEXT Y
```

2. Analyser et expérimenter le programme suivant :

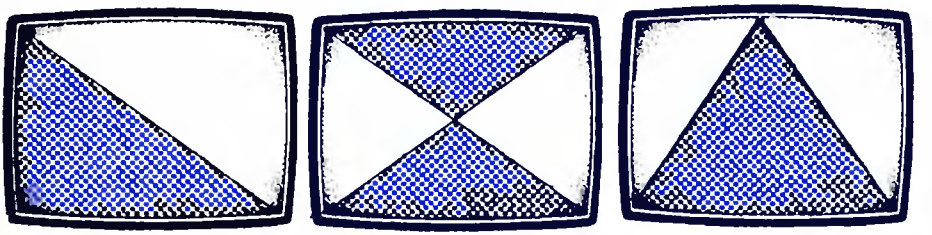
```
1 CLS
2 FOR I=0 TO 3
3  FOR J=0 TO 3
4   COLOR I+4*J
5   BOXF(10+30*I,10+30*J)
        -(30+30*I,30+30*J)
6  NEXT J
7 NEXT I
```



Ici les 2 boucles utilisées sont "imbriquées"

```
FOR I=0 TO 3
 FOR J=0 TO 3
  ...
  ...
 NEXT J
NEXT I
```

3. Dessiner les figures suivantes :



4. Comparer l'effet du programme des "carrés géométriquement emboîtés" avec l'effet du suivant :

```
10  CLS
20  C=100
30  FOR K=1 TO 10
32   BOX(0,0)-(C,C)
34   C=C-10*K
36  NEXT K
40  LOCATE 0,15
50  END
```

5. Reprendre les programmes de la leçon 5 en transformant toutes les boucles commandées par un GOTO en boucles commandées par FOR... NEXT, de manière que l'exécution s'arrête sur une instruction END.

Attention

■ Sur les listes de vos programmes repérez les boucles FOR... NEXT par un signe schématique du genre suivant, cela vous aidera à mieux voir la structure de vos programmes :

```
┌ FOR K = . . . . . .
│  . . . . . . . . . . . . . .
│  . . . . . . . . . . . . . .
└ NEXT K
```

Vous pouvez aussi décaler les instructions internes à une boucle comme nous l'avons fait dans les exercices 2 et 4

■ L'instruction END n'est pas obligatoire dans un programme. Mais cela fait plus "propre"... et évite les ennuis ultérieurs.

■ L'instruction

```
IF C<8 THEN GOTO 30 ELSE GOTO 15
```

peut être abrégée en

```
IF C<8 THEN 30 ELSE 15
```

En fait, vous verrez en consultant la fiche de référence que l'instruction IF... THEN... ELSE peut prendre des formes très diverses.

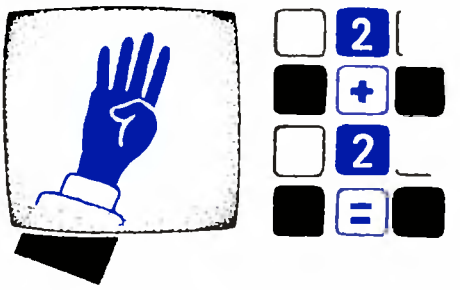
Dans un premier temps nous vous conseillons de n'utiliser que la forme abrégée ci-dessus.

Et si, comme la chèvre de monsieur Seguin, vous vous aventurez trop loin en solitaire, il se peut que vous ayez à lutter une nuit entière contre les "bugs" qui risquent de vous harceler.

Fiches de références pouvant être consultées :

* FOR... NEXT, page 161
** IF... THEN... ELSE..., page 166
** ON... GOTO, page 165
* RUN, STOP, END, CNT-C, CONT, GOTO, page 153

Page blanche



7

Nombres et opérations

Vous pensiez peut-être que les ordinateurs étaient faits pour manipuler des chiffres et cela n'a pas été tellement le cas jusqu'ici.
Il est temps de combler cette lacune.

■ Le signe =

Nous avons déjà rencontré des instructions permettant "d'affecter" des valeurs numériques à des variables représentées par une ou plusieurs lettres.
Ainsi par exemple :

```
ANS=39
A=0
I=1
C=100
```

Nous avons aussi rencontré et utilisé des "instructions d'affectation de valeurs" un peu plus complexes :

```
HEURES=(365*ANS+30*MOIS+JOURS)*24        leçon 3
A=A+1                                    leçon 4
B=B+10                                   leçon 5
C=0.90*C                                 leçon 6
```

Il est temps maintenant d'expliciter le comportement de la machine dans l'exécution d'une telle instruction :
Pour exécuter l'instruction :

```
A1=B+A1+C
```

— *D'abord*, la machine calcule le résultat du calcul décrit après le signe =.
Cela nécessite que les valeurs des variables qui y figurent soient toutes connues (ici B, A1 et C ont précédemment reçu une valeur).
— *Ensuite*, la machine donne ce résultat pour valeur à la variable écrite avant le signe =.

Remarquez bien que, dans ce processus, rien ne s'oppose à ce que le même nom de variable figure avant et après le signe =. En fait cette opportunité est exactement ce qui fait la puissance de la plupart des programmes...

■ Un programme de tables numériques

Les opérations et les fonctions "classiques" sont disponibles sur le MO 5 (voir, pour le détail, la fiche de référence).
A titre d'exemple, voici un programme très simple qui édite une table des carrés et des cubes des nombres entiers.

```
5 CLS
10 N=1
20 PRINT N,N*N,N*N*N
30 N=N+1
40 GOTO 20
50 END
```

A l'exécution, la table se met à défiler très vite, on peut la consulter en frappant sur STOP au moment opportun.

Remarques

• On peut aussi ralentir l'exécution en faisant exécuter une "boucle vide" comme ceci :

```
25 FOR K =1 TO 500
26 NEXT K
```

Essayez aussi en changeant 500 en 100 ou en 1000.

• En remplaçant la ligne 20 par :

```
20 PRINT N,LOG(N)
```

on édite une table de logarithmes.

• En remplaçant la ligne 20 par :

```
20 PRINT N,COS(N),SIN(N)
```

on édite une table trigonométrique (attention à l'unité d'angle : voir la fiche de référence des fonctions).

■ Un programme pour pousser les divisions aussi loin qu'on le veut

Rappelez-vous la technique de la division en usage chez nous.

Le "programme" de son exécution n'est-il pas le suivant ?

• Soit à diviser A par B

• Écrire le quotient (entier) de A par B ; appelons-le QUOTIENT

Le signe de la division entière est le signe @.

• Soustraire B× QUOTIENT de A ; on obtient le RESTE.

• Abaisser un zéro c'est-à-dire "multiplier le reste par 10".

• Faire comme si RESTE était le nouveau nombre à diviser par B (autrement dit traiter RESTE comme A).

et ainsi de suite...

Ce programme se traduit très bien en BASIC (quasiment mot à mot).

```
1 CLS
2 A=365
3 B=7
4 QUOTIENT=A@B
5 PRINT QUOTIENT
6 RESTE=A-B*QUOTIENT
7 RESTE=10*RESTE
8 A=RESTE
9 GOTO 4
```

Remarque : L'affichage du résultat est très spectaculaire si on termine l'instruction 5 par un point virgule (;) qui indique à la machine de ne pas aller à la ligne après avoir écrit la valeur de Q.
Voici le résultat de l'exécution de ce programme :

RUN ENTREE

```
52  1  4  2  8  5  7  1  4  2  8  5  7
1  4  2  8  5  7  1  4  2  8  5  7  1
4  2  8  5  7  1  4  2  8  5  7  1  4
2  8  5  7  1  4  2  8  5  7  1  4  2
8  5  7  1  4  2  8  5  7  1  4  2  8
5  7  1  4  2  8  5  7  1  4  2  8  5   ...
```

■ Tirer des nombres au hasard

La machine que vous utilisez offre une possibilité à laquelle vous ne vous attendiez pas :
Elle peut tirer des nombres au hasard.
Ainsi lorsque vous commandez l'instruction :

```
A=RND
```

... vous êtes strictement incapable de savoir quelle sera la valeur de A.
En fait A peut prendre n'importe quelle valeur (décimale) entre Ø.ØØØØØØ et Ø.999999.
Pour vous en persuader, exécutez le programme suivant :

```
10 FOR I =1 TO 10
20 A=RND
30 PRINT A
40 NEXT I
```

Et vous obtenez :

```
.594972
.512679
.58108E-02
.428832
.16661
.62274
.906189
.430424
4.96011E-02
9.94663E-02
```

nombre égal à Ø.Ø158108 (voir page 182)

Le nom RND est l'abrévation de l'anglais *rand*om qui signifie "hasard".

■ Un programme simulant le jet d'un dé

Cela peut vous permettre, par exemple, de "simuler" le jet d'un dé à 6 faces par la machine.
Pour cela il faudrait que RND ait une valeur qui ne soit pas comprise entre Ø et 1 mais soit égal à l'un des 6 nombres entiers : 1, 2, 3, 4, 5 ou 6.

Heureusement nous disposons de la fonction INT qui prend la "partie entière" d'un nombre (par exemple : INT (4.356) vaut 4). Pour obtenir, au hasard, le numéro de la face d'un dé, il suffit alors de s'apercevoir que :
RND est un nombre compris entre Ø et Ø.999999
6 * RND est un nombre compris entre Ø et 5.99999
1 + 6 * RND est un nombre compris entre 1 et 6.99999
INT (1 + 6 * RND) est un des nombres 1, 2, 3, 4, 5 ou 6.
Voici donc un programme simulant le jet d'un dé et son tournoiement au centre de l'écran.
Lorsque vous frappez sur STOP, il s'arrête de "rouler".

```
1 CLS
2 ATTRB 1,1
3 LOCATE 20,12
4 D=INT(1+6*RND)
5 PRINT D
6 GOTO 2
```

Note : L'instruction ATTRB 1, 1 commande l'écriture de caractères de grandeur double par rapport à la grandeur habituelle. Voir la fiche de référence.

Exercices

1. L'inflation étant de 10 % par an et la baguette de pain coûtant 2 F en 1981, vérifier que le programme suivant affiche le prix de la baguette jusqu'à l'an 2000.

```
1 CLS
2 INFLATION=0.10
3 PRIX=2
4 ANNEE=1981
5 FOR N=1 TO 20
6 PRINT ANNEE,PRIX
7 ANNEE=ANNEE+1
8 PRIX=PRIX*(1+INFLATION)
9 NEXT N
```

Remarque : La variable N ne sert ici que pour "compter" de 1 à 20 et n'est pas utilisée dans les calculs.

2. Quel rapport le programme suivant a-t-il avec l'astronomie ?

```
1 CLS
2 X=319*RND
3 Y=199*RND
4 PSET(X,Y)
5 GOTO 2
```

3. Le programme suivant provoque l'affichage d'une table de multiplication des nombres de 4 à 9.

```
1 CLS
10 FOR I=4 TO 9
20 FOR J=4 TO 9
30 PRINT I*J;
40 NEXT J
50 PRINT
60 NEXT I
```

Analysez les effets des instructions 50 et 30 et comparez avec l'instruction 6 du programme de l'exercice 1. Quels peuvent être les rôles de la virgule et du point virgule ?

Attention

■ Le même signe "=" est utilisé dans les instructions comme...

```
P=2
A=A+1
IF X=0 THEN 20 ELSE 80
FOR N=1 TO 20
```

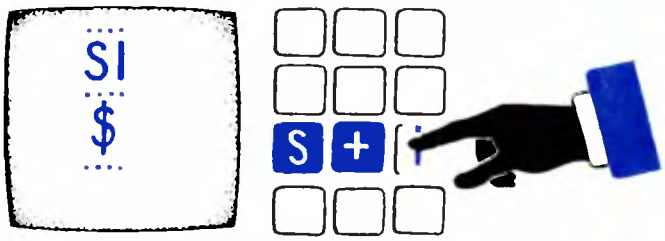
... avec des significations bien différentes.
Relisez attentivement le premier paragraphe de cette leçon.

■ Lorsque vous employez une variable (par exemple PRIX, ANS, I ou N), pensez qu'il doit y avoir une instruction où cette variable prend une valeur numérique pour la première fois.
Par exemple pour le programme de l'exercice 1 :
Dans les instructions 1 à 4, les variables prennent leurs valeurs "initiales".
Puis, dans les instructions 7 et 8, certaines variables changent de valeur ; INFLATION reste constante pendant toute l'exécution alors que N varie de 1 à 20 pour servir de "compteur".

Fiches de référence pouvant être consultées :

* +, −, *, /, @ , MOD, ∧, page 172
* RND, page 176
* SIN, COS..., page 174
* PRINT, page 192

Page blanche



8

Données et résultats

Jusqu'à maintenant, le programme une fois lancé, l'ordinateur ne vous demande plus jamais rien : vous ne pouvez plus intervenir sur les calculs et, il n'accepte pas de s'interrompre en cours de travail pour entamer le moindre échange bilatéral d'information.
Mettons un terme à cette anomalie.

■ Un programme pour calculer l'inflation

Reprenons, par exemple, l'exercice de la leçon précédente sur l'évolution du prix de la baguette de pain.
Nous voulons faire un programme plus général.
Supposons que l'utilisateur de notre programme veuille connaître le prix d'un certain objet dans quelques années, connaissant le prix de cet objet aujourd'hui et le taux d'inflation à prévoir. L'idéal serait que l'exécution de notre programme se passe comme ceci :

- RUN ENTREE (c'est parti !)
- La machine demande que l'on frappe au clavier le taux d'inflation prévu.

• La machine demande que l'on frappe le prix de l'objet qui nous intéresse.
• La machine calcule et donne le prix de l'objet pour chacune des années suivantes.
Nous vous donnons ce programme, avant d'analyser le fonctionnement des nouvelles instructions qu'il utilise. Le voici :

```
10 CLS
20 INPUT "TAUX D'INFLATION PREVU (EN POU
RCENTAGE)";I
30 INPUT "PRIX DE L'OBJET AUJOURD'HUI (E
N FRANCS)";PRIX
40 ANNEE=0
50 ANNEE=ANNEE+1
60 PRIX=PRIX*(1+I/100)
70 PRINT "DANS ";ANNEE;" ANS, L'OBJET CO
UTERA ";PRIX;"FRANCS"
80 GOTO 50
```

Note : sur le clavier du MO5 le caractère guillemet " est situé sur la même touche que le chiffre 2. Remarquez que tout a été fait pour vous simplifier l'écriture : les guillemets ouvrant et fermant sont les mêmes !
(Voir aussi l'exercice 1 de la leçon précédente)

■ INPUT

INPUT signifie "entrée" en anglais.
C'est donc une instruction permettant d'entrer des données dans la machine. Précisément, l'exécution de l'instruction 20 provoque le comportement suivant :
— tout ce qui est écrit entre guillemets est intégralement réécrit sur l'écran (le point virgule après les guillemets est obligatoire).
— puis la machine affiche un point d'interrogation et s'arrête.
— la machine reprend la suite de l'exécution lorsque l'utilisateur frappe ENTREE ; normalement avant de frapper sur ENTREE l'utilisateur *a dû* frapper la suite des chiffres d'un nombre.
C'est ce "nombre frappé" qui est donné comme valeur à la variable indiquée après le point virgule.
L'exécution du programme précédent donne donc lieu au dialogue suivant (les frappes de l'utilisateur sont indiquées touche par touche).

```
RUN ENTREE
TAUX D'INFLATION PREVU (EN POURCENTAGE)?
 14 ENTREE
PRIX DE L'OBJET AUJOURD'HUI (EN FRANCS)?
 83 ENTREE
DANS  1  ANS, L'OBJET COUTERA  94.62
FRANCS
DANS  2  ANS, L'OBJET COUTERA  107.867
FRANCS
DANS  3  ANS, L'OBJET COUTERA  122.968
FRANCS
DANS  4  ANS, L'OBJET COUTERA  140.184
FRANCS
DANS  5  ANS, L'OBJET COUTERA  159.809
FRANCS

STOP
```

Note : Si le message "Redo" apparaît sur l'écran, c'est que l'utilisateur n'a pas frappé la suite de chiffres attendue par la machine.

■ Le dialogue à l'entrée

En fait, l'instruction INPUT est une instruction d'arrêt. La machine s'arrête et attend la valeur d'une (ou plusieurs) variable(s) dont la liste des noms est donnée dans la suite de l'instruction.
Les indications entre guillemets n'ont rien de nécessaire mais elles aident l'utilisateur du programme à se souvenir de la signification des nombres introduits.
Ainsi, dans le programme précédent, on pourrait remplacer les instructions 20 et 30 par la seule instruction :

```
15 INPUT I,PRIX
```

L'exécution ressemblerait alors à :

```
RUN ENTREE
?
14,83 ENTREE
DANS 1 ANS, L'OBJET COUTERA 94.62
FRANCS
```

La simplification de l'écriture oblige à ne pas oublier le nombre et l'ordre d'entrée des données ; ici le taux d'inflation et le prix (qui doivent être séparés par une virgule).

■ Le dialogue à la sortie

De la même manière l'affichage de commentaires avec les résultats permet de soulager sa mémoire.
Il est plus agréable de lire :

```
DANS 3 ANS L'OBJET COUTERA 122.968
FRANCS
```

plutôt que de devoir traduire un laconique

```
122.968
```

Inversement, cela oblige à réfléchir un peu dans l'écriture des instructions PRINT : la manipulation des " et des : n'est, en effet, pas aussi triviale qu'on le voudrait. Témoin l'instruction suivante qui donne aussi une bonne information :

```
60 PRINT "ANS=";ANNEE;"PRIX=";PRIX
```

Exercices

1. En utilisant l'instruction IF... THEN... ELSE, modifiez le programme "inflation" de manière à éviter la faute d'orthographe : "1 ans" dans l'affichage des résultats.
2. Reprenez le programme de la 3e leçon, calculant le nombre d'heures vécues en faisant entrer, par une instruction INPUT, l'âge d'une personne.
3. Voici un programme de tracé d'un triangle dont les sommets sont définis par l'utilisateur :

```
10 INPUT "SOMMET NUMERO 1";X1,Y1
20 INPUT "SOMMET NUMERO 2";X2,Y2
30 INPUT "SOMMET NUMERO 3";X3,Y3
40 CLS
50 LINE(X1,Y1)-(X2,Y2)
60 LINE -(X3,Y3)
70 LINE -(X1,Y1)
80 END
```

Observez l'effet des séparateurs ; , et - .
La syntaxe du MO 5 est très précise !

4. Réalisez un programme coloriant en vert un rectangle dont l'utilisateur définit deux sommets opposés et traçant en rouge ses diagonales. Pour éviter les "bavures" de couleur, vous pouvez colorier le rectangle en couleur de fond, grâce à un numéro de couleur négatif (voir les fiches sur les instructions graphiques).
Exemple : `BOXF(X1,Y1)-(X2,Y2),-3`

5. Réalisez un programme dessinant un drapeau à 3 bandes dont les codes des 3 couleurs sont entrés par l'utilisateur à l'aide d'une instruction "INPUT".

Attention

■ Lorsque le message :
`?Redo`
apparaît sur l'écran c'est que la machine attend la valeur d'une variable dont le nom figure dans une instruction INPUT et que la suite de touches que vous venez de frapper est incompréhensible pour elle.
Il faut alors tout simplement frapper sur une bonne suite de touches ; par exemple une suite de chiffres suivie de ENTREE.
Si vous ne comprenez pas du tout ce qui se passe, vous pouvez toujours interrompre le programme en frappant :
CNT—C

Fiches de référence pouvant être consultées

* INPUT, page 196
* PRINT, page 192
** PRINT USING, page 194

Page blanche

9

Lettres et messages

Vous avez sûrement déjà tenu avec certains ordinateurs de véritables "dialogues" et vous voudriez bien maintenant pouvoir traiter de véritables textes et messages. Voici le moment venu.

■ Un programme de bienvenue

Vous savez déjà que :
Dans une instruction PRINT ou INPUT, tout ce qui est écrit entre guillemets est affiché tel quel lors de l'exécution du programme.
Il est possible d'aller plus loin en donnant un nom à un message (ou à un morceau de message) tout comme on donne un nom à une valeur numérique.
La seule chose à savoir pour l'instant est que le nom d'un message doit se terminer par le caractère "$". (Eh oui ! tout se paye en dollars aujourd'hui !).

Voici par exemple un petit programme de bienvenue.

```
1 CLS
2 A$="BONJOUR, MES"
3 B$="DAMES !"
4 C$="SIEURS !"
5 PRINT A$;B$
6 PRINT A$;C$
7 END
```

En informatique, on parle plutôt de "chaîne de caractères" au lieu de "suite de caractères". Ainsi "DAMES !" est une chaîne de caractères de longueur 7. Et B$ est une "variable-chaîne".

■ Entrez votre nom !

Dans le programme précédent, la machine est seule à "parler". La possibilité d'interroger l'utilisateur par une instruction INPUT ouvre la voie du dialogue.
On peut faire réagir la machine, par exemple par la répétition indéfinie de la réponse de l'utilisateur.
A défaut de signification, c'est au moins spectaculaire :

```
1 CLS
2 X$="BONJOUR JE M'APPELLE MO5 ET VOUS ?
"
3 PRINT X$
4 INPUT V$
5 PRINT "SUR CET ECRAN ";V$;" J'ECRIS TO
N NOM"
6 GOTO 5
7 END
```

Note : Dans l'instruction 4, la machine attend une suite de caractères ; lorsque l'utilisateur frappera sur ENTREE, elle donnera le nom V$ à la suite de caractères qu'il aura préalablement frappés.

■ Vers un dialogue (truqué)

Pour progresser vers un véritable dialogue, il y a au moins deux moyens :
— afficher des questions et des réponses stéréotypées (voir le programme suivant jusqu'à l'instruction 5Ø)
— varier la réaction de l'ordinateur en fonction de la réponse de l'utilisateur grâce aux instructions du type :
IF... THEN... ELSE...
(voir le programme suivant à la suite de l'instruction 5Ø et particulièrement l'instruction 8Ø)

```
10 CLS
20 X$="BONJOUR JE M'APPELLE MO5 ET VOUS
?"
30 PRINT X$
40 INPUT V$
50 PRINT "J'AI CONNU UN ";V$
60 PRINT "MAIS IL ETAIT MOINS SYMPA QUE
VOUS !"
70 INPUT"VOULEZ-VOUS ALLER PLUS LOIN AVE
C MOI    ";R$
80 IF R$="OUI" THEN A$="AH ! MON CHER "
B$="ALLONS FAIRE" ELSE A$="DOMMAGE !" B$
="AURIONS FAIT"
90 FOR I=0 TO 15
100 FOR J=0 TO 15
110 COLOR I,J
120 ATTRB1,1
130 PRINT A$;V$;
140 ATTRB0,0
150 PRINT" NOUS ";B$;" DE"
160 PRINT"GRANDES ET BELLES CHOSES ENSEM
BLE":PRINT
170 NEXT J
180 NEXT I
190 END
```

Nous aurions pu nous contenter des instructions 13Ø, 15Ø et 16Ø mais, finalement, un spectacle coloré ne coûte pas cher sur le MO 5.

■ Les mots bout à bout

Il est possible de faire des "opérations" sur les mots comme il est possible de faire des "opérations" sur les nombres.
La plus simple des opérations est la mise bout à bout. Le signe de cette opération est le signe +.
En voici une utilisation simplette :

```
10 P$="BI"
20 Q$="POLY"
30 A$="CYCLE"
40 B$="NOME"
50 X$=Q$+A$
60 Y$=P$+A$
70 Z$=Q$+B$
80 T$=P$+B$
90 PRINT X$,Y$
100 PRINT Z$,T$
```

■ Découper les mots

Une "opération" plus délicate consiste à découper les mots en sous-mots.
On peut ainsi prendre les N derniers caractères d'un mot X$ grâce à la fonction RIGHT$ (X$, N) ; ou bien les N premiers grâce à la fonction LEFT$ (X$, N) ; ce qui permet de jolis effets ; surtout à l'aide de la fonction LEN (X$) qui donne la longueur du mot X$.
Voici un exemple :

```
1 CLS
2 INPUT "QUEL EST VOTRE NOM ";NOM$
3 LONGUEUR=LEN(NOM$)
4 FOR N=1 TO LONGUEUR
5 P$=LEFT$(NOM$,N)
6 PRINT P$
7 NEXT N
```

Voici un effet possible de ce programme :

```
RUN ENTREE
QUEL EST VOTRE NOM ?
UNTEL
U
UN
UNT
UNTE
UNTEL
```

Remarque : En remplaçant l'instruction `4` par l'instruction

```
4 FOR N=LONGUEUR TO 1 STEP -1
```

on peut obtenir comme résultat :

```
ABRACADABRA
BRACADABRA
RACADABRA
ACADABRA
CADABRA
ADABRA
DABRA
ABRA
BRA
RA
A
```

Vous constatez qu'il est possible de choisir le "pas" de la "variable de boucle" en ajoutant STEP PAS à une instruction FOR. (voir la fiche de référence FOR... NEXT).

Exercices

1. Faire un programme qui écrive successivement :

```
ANDRE
ANDR
AND
AN
A
```

2. Faire un programme qui écrive successivement :

```
HELENE
ELENEH
LENEHE
ENEHEL
NEHELE
EHELEN
```

3. Faire un programme qui demande un mot à l'utilisateur et qui l'écrive

```
V
E
R
T
I
C
A
L
E
M
E
N
T
```

puis en

```
D
 I
  A
   G
    O
     N
      A
       L
        E
```

4. Écrire un programme qui met les mots au pluriel :
— si le mot se termine par S, X ou Z, l'afficher tel quel.
— si le mot se termine par AU, EU, EAU, lui ajouter un X.
Pour un programme plus performant, voyez le corrigé (après avoir consulté la fiche de référence DATA, READ) dans le chapitre conseils de programmation.

Attention

■ Les guillemets ont une grande importance !
17 est une valeur numérique, 89 aussi.
L'exécution de l'instruction :

```
PRINT 17+89
```

provoque l'impression de :

```
106
```

Par contre "17" et "89" sont des suites de caractères.
L'exécution de l'instruction :

```
PRINT "17+89"
```

provoque l'impression de :

```
17+89
```

et l'exécution de l'instruction :

```
PRINT "17"+"89"
```

provoque l'impression de :

```
1789
```

Tout ce qui figure entre guillemets n'est pas "traité" par la machine mais simplement mémorisé ou recopié.

Fiches de référence pouvant être consultées :

* "...", +, page 177
* MID$,LEFT$,RIGHT$, page 178
* LEN,INSTR, page 179
** VAL,STR$, page 180

Page blanche

10

Musique

Il est temps maintenant de jouer au compositeur électronique.

■ La gamme

La commande

```
PLAY"DOREMIFASOLASI" ENTREE
```

a pour effet de faire jouer à la machine la suite de 7 notes constituant la gamme de DO majeur.

Remarquez que :

— chaque note est nommée par un mot de deux lettres (SOL est donc appelée seulement SO).

— Un ensemble de notes se présente exactement comme une chaîne de caractères. Ainsi le programme :

```
1 A$="DOFALA"
2 PRINT A$
3 PLAY A$
4 END
```

a pour effet de faire imprimer le mot DOFALA puis de jouer les notes correspondantes. En effet :

La commande PRINT déclenche l'écriture d'une suite de caractères sur l'écran.
La commande PLAY déclenche l'émission d'une suite de sons par le haut parleur.
Évidemment,une commande comme

```
PLAY "MOZART"
```

déclenche l'affichage d'une erreur puisque ni MO, ni ZA, ni RT ne sont interprétés comme des notes jouables !

■ Les octaves

En fait, la machine émet les notes correspondant à une octave particulière.
Et, donc, si l'on commande :

```
PLAY"DOREMIFASOLASI DO"
```

Le premier et le dernier DO seront les mêmes.
Si l'on veut que le DO suivant le SI soit une octave plus haut que le DO initial, il faut commander un changement d'octave, par exemple comme ceci :

```
PLAY"DOREMIFASOLASI O5DO"
```

Vous disposez, sur le MO5, de 5 octaves numérotées de 1 à 5, le choix de l'une d'entre elles s'effectuant en insérant dans la suite des notes un "O" suivi d'un chiffre de 1 à 5 :
O1 fait choisir l'octave la plus grave.
O4 fait choisir la même octave que celle qui est implicitement choisie à l'allumage de la machine.
O5 fait choisir l'octave la plus aiguë.
En guise d'exemple, voici un programme de montée et de descente des gammes très impressionnant :

```
1 B$="DOREMIFASOLASI":C$="SILASOFAMIREDO
"
2 PLAY"O1"+B$+"O2"+B$+"O3"+B$+"O4"+B$+"O
5"+B$
3 PLAY C$+"O4"+C$+"O3"+C$+"O2"+C$+"O1"+C
$
4 END
```

En remplaçant B$ et C$ par une simple note au lieu d'une gamme entière on entend très bien le changement d'octave. Essayez par exemple de remplacer la ligne 1 par :

```
1 B$="LA":C$="LA"
```

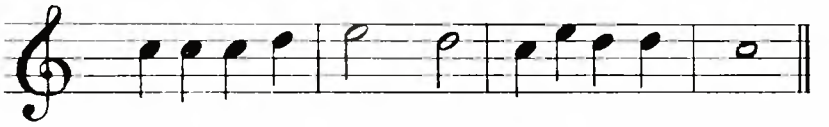
■ La durée des notes

Il est aussi possible de choisir la longueur pendant laquelle une note est jouée. Pour cela on insère dans la suite des notes un "L" suivi d'un nombre compris entre 1 et 96 :

L96 fait jouer des rondes :
L48 fait jouer des blanches :
L24 fait jouer des noires :
(C'est la longueur implicitement choisie lors de la mise en fonctionnement de la machine).
L12 fait jouer des croches :
L6 fait jouer des doubles croches :
L3 fait jouer des triples croches :

Mais toute longueur intermédiaire est jouable. Ainsi L18 fait jouer une "croche pointée" :

Voici par exemple la traduction des 4 premières mesures de "Au clair de la lune, mon ami Pierrot !".



```
1 PLAY "DDDDDRE"
2 PLAY "L48MIRE"
3 PLAY "L24DUMIRERE"
4 PLAY "L96DU"
5 PLAY "L24PPPP":GOTO 1
```

Remarque : A la ligne 5, avant de rejouer le même air, nous avons commandé 4 "pauses" indiquées chacune par la lettre P.
L24P ne dure que le temps d'un "soupir" !

■ Le tempo

Indépendamment des durées relatives de chaque note, on peut jouer un même air plus ou moins vite.

La vitesse d'exécution peut être choisie en insérant dans la suite des notes un "T" suivi d'un nombre compris entre 1 et 255. Bien que cela soit un peu approximatif, on peut risquer une correspondance que vous voudrez bien corriger selon votre humeur :

T1 fait jouer Prestissimo
T2 fait jouer Presto
T4 fait jouer Allegro
T5 fait jouer Allegretto
(C'est le tempo implicitement choisi à l'allumage de la machine).
T8 fait jouer Moderato
T12 fait jouer Andantino
T16 fait jouer Andante
T32 fait jouer Adagio
T48 fait jouer Lento
T64 fait jouer Largo
T256 fait jouer très très très très lentement.

Pour vous rendre compte de l'effet du tempo, jouez par exemple des séries de gammes sur des tempos différents.

```
1 B$="DOREMIFASOLASI"
2 PLAY "T1"+B$+"T2"+B$+"T3"+B$+"T4"+B$+"
T5"+B$
```

Ou bien séparez l'émission d'une note par des pauses :

```
1 PLAY "T1LAPT2LAPT4LAPT8LAPT16LAPT32LAP
T64LAPT128LAPT255LA"
```

■ Complément

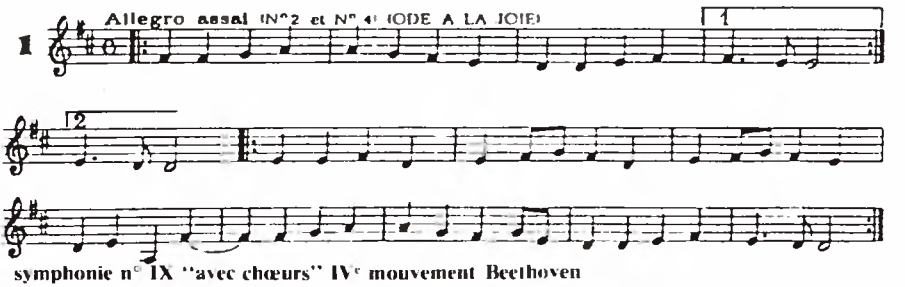
• Les dièses et les bémols s'écrivent naturellement après le nom des notes . Exemples :

```
PLAY"DO#FASO#"
PLAY"DObMIbSOb"
```

• Pour choisir une "attaque" particulière, on insère dans la suite des notes un "A" suivi d'un nombre de Ø à 255.
AØ fait jouer un son continu.
A2ØØ fait jouer une note très rapidement amortie de sorte qu'un silence précède l'émission de la note suivante. Le mieux est de faire vos propres expériences !
• On peut insérer des espaces entre les notes si l'on veut rendre la chaîne plus lisible.

Exercices

1. Traduire l'air suivant sur votre MO5 :



symphonie n° IX "avec chœurs" IV^e mouvement Beethoven

2. Quelle tragédie évoque pour vous le programme suivant ?

```
1 PLAY"L96S1SO"
2 GOTO 1
```

3. Parmi ces deux programmes, l'un d'eux évoque le ronflement d'un moteur, l'autre des tirs de fusée.
Reconnaissez-les en faisant jouer :

```
1 PLAY "T1L2A1001DO"
2 GOTO 1
```

```
10 A$="DODO#RERE#MIFAFA#SOSO#LALA#SI"
20 PLAY "T1L4A404"+A$+"O5"+A$+"PP"
30 GOTO 20
```

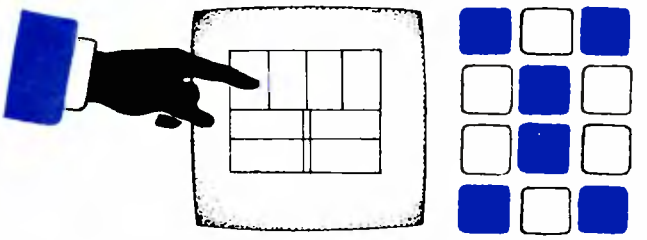
4. Faites un programme jouant des notes au hasard mais affichant simultanément leur nom écrit en noir sur un fond de couleur différente selon la note (DO sur rouge, RE sur vert, MI sur jaune, etc.)

Attention

■ Dans un programme jouant de la musique, l'appui sur CNT-C ne semble pas toujours arrêter l'exécution ; c'est que la machine ne "comprend" l'appui sur CNT-C qu'entre deux instructions. Si vous appuyez seulement pendant l'exécution de notes, rien ne se passe ; il faut donc garder les deux touches CNT et C appuyées au moins jusqu'à la fin de l'émission d'une suite de notes commandée par un ordre PLAY.

Fiche de référence pouvant être consultée :

* PLAY, page 207



11

Indices et tableaux

■ Statistiques

Faire des statistiques c'est traiter des tableaux de nombres pour en extraire des renseignements.
Au niveau familial, par exemple, on peut vouloir faire des statistiques permettant de mieux contrôler les dépenses. Imaginons un possesseur de MO5, M. Dupont qui note la nature de chacune de ses dépenses ; ainsi sur son carnet de chèques note-t-il :
le chiffre 1 pour les dépenses d'alimentation.
le chiffre 2 pour les dépenses d'habillement.
le chiffre 3 pour les dépenses de transports.
le chiffre 4 pour les dépenses courantes diverses.
le chiffre 5 pour les dépenses d'investissements (meubles,appareils...).
le chiffre 6 pour les dépenses de logement (loyer, électricité...).
le chiffre 7 pour les impôts et les assurances.

Voici le "tableau" représentant la répartition de ces dépenses au mois de janvier :

1	2	3	4	5	6	7
2130 F	1100 F	320 F	710 F	800 F	2650 F	1800 F

■ Tableaux et indices

Pour effectuer des calculs avec ses dépenses, M. Dupont est obligé de leur donner un nom.
Pour cela il est pratique d'utiliser une "variable" avec un "indice". Ainsi :
DEPENSE (1) représente les dépenses d'alimentation de M. Dupont (ici DEPENSE (1) vaut 2130 F)
DEPENSE (2) représente les dépenses d'habillement de M. Dupont (ici DEPENSE (2) vaut 1100 F)
DEPENSE (I) représente les dépenses relatives au "poste" n° I.
I est un *indice* ; DEPENSE est un nom de *tableau* et DEPENSE (I) vaut une certaine valeur numérique.

■ Programme de M. Dupont

• Pour entrer ses différentes dépenses de janvier dans le MO5, M. Dupont écrit alors le programme suivant :

```
10 FOR I=1 TO 7
11 INPUT DEPENSE(I)
12 NEXT I
```

Et chaque fois que MO5 affiche un point d'interrogation, M. Dupont frappe successivement 2130, 1100, 320, 710, 800, 2650 et 1800.

• Pour connaître le total de ses dépenses, M. Dupont complète son programme par les instructions suivantes :

```
20 SOMME=0
21 FOR I=1 TO 7
22 SOMME=SOMME+DEPENSE(I)
23 NEXT I
24 PRINT"Ce mois-ci j'ai dépensé";SOMME;
"Francs"
```

Et, à l'exécution, la machine affichera :

```
Ce mois-ci, j'ai dépensé 9510 Francs
```

• Pour comparer sa manière de vivre avec celle de ses amis, M. Dupont voudrait connaître les pourcentages relatifs à chacun de ces postes de dépenses. Les instructions suivantes effectuent ce travail :

```
30 FOR I=1 TO 7
31 PRINT"Poste numéro";I;
32 P=INT(100*DEPENSE(I)/SOMME+0.5)
33 PRINT" :";P;"%"
34 NEXT I
```

Et, à l'exécution, la machine affichera :

```
Poste numéro 1 : 22 %
Poste numéro 2 : 12 %
Poste numéro 3 : 3 %
Poste numéro 4 : 7 %
Poste numéro 5 : 8 %
Poste numéro 6 : 28 %
Poste numéro 7 : 19 %
```

Note : Pour effectuer ce dernier travail, l'ensemble des instructions de 10 à 34 doit être présent dans la machine.

■ Remarques

• Les instructions 20 à 24 valent la peine d'être bien comprises ; elles montrent comment *calculer la somme des éléments d'un tableau* dans le cas général :

— Donner un nom à cette somme (ici SOMME) et lui affecter la valeur : 0

— Ajouter successivement à SOMME chacun des éléments du tableau grâce à une boucle commandée par un "indice" parcourant tous les numéros du tableau.

— A la fin de la boucle, SOMME vaut exactement la somme des éléments du tableau.

• Pour bien comprendre les affichages prévus dans les instructions 31 à 33, relisez la fiche de référence PRINT précisant le rôle des guillemets et des points virgules.

• Si M. Dupont veut affiner l'analyse de ses dépenses, il décidera peut-être de considérer plus de 10 postes. Il faut alors savoir ceci : Lorsque l'indice d'un tableau doit dépasser 10, il est nécessaire de l'indiquer au MO5 pour qu'il prévoit le rangement d'une longue suite de nombres.
Précisément l'instruction :

```
1 DIM D(20)
```

placée en tête de programme, vous permet de parler des variables D(0), D(1), D(2), D(3), ...D(19) et D(20), dont il pourra être question par la suite. Remarquez bien que le MO5 réserve alors 21 mémoires.
En l'absence de l'instruction 1, l'exécution de l'ordre INPUT D(11) entraînerait l'affichage d'un message d'erreur.
• L'instruction 32 permet classiquement d'arrondir *un résultat à l'entier le plus proche.*
Le résultat intéressant est ici R = 100 * DEPENSE(I)/SOMME
L'instruction P = INT (R + 0,5) permet de se débarrasser des décimales.
• Avec un peu d'habitude, vous vous lancerez certainement dans une *représentation graphique* des dépenses de M. Dupont...

■ Tableaux à 2 dimensions

Si Monsieur Dupont tient ses statistiques tous les mois de l'année, il dispose en fin d'année d'un tableau à 2 dimensions représentant l'analyse de ces dépenses :

poste / mois	1	2	3	4	5	6	7
1	2 130	1 100	320	710	800	2 650	1 800
2	1 840	720	340	820	1 560	2 500	1 320
3	2 050	810	310	1 010	0	2 710	1 320
....	...	...	...	...	...	...	...
12	2 110	380	250	1 820	0	2 850	2 910

Pour donner un nom à chacune des dépenses, il est évidemment pratique d'utiliser une variable avec 2 indices. Ainsi :
DEPENSE (4, 3) représente les dépenses courantes diverses (indice 4) de M. Dupont pendant le mois de mars (3e mois).
DEPENSE (I, J) représente les dépenses relatives au poste numéro I pendant le mois numéro J.

• L'entrée des données oblige Monsieur Dupont à frapper 84 nombres ; c'est évidemment ennuyeux mais il n'y a pas d'autres moyens que de les frapper au moins une fois. (La saisie des données est le goulot d'étranglement classique de l'utilisation de l'informatique). Le programme n'est pourtant pas complexe :

```
1 DIM DEPENSE(7,12)
9 FOR J=1 TO 12
10 FOR I=1 TO 7
11 INPUT DEPENSE(I,J)
12 NEXT I
19 NEXT J
```

• Mais M. Dupont (ou vous-même) pourra s'en donner à cœur joie en effectuant des totaux par mois comme précédemment, mais aussi par poste sur une année grâce à des morceaux de programmes ressemblant à celui ci :

```
39 FOR I=1 TO 7
40 SOM=0
41 FOR J=1 TO 12
42 SOM=SOM+DEPENSE(I,J)
43 NEXT J
44 PRINT "EN UN AN, POUR LE POSTE";I;"J'
A1 DEPENSE :";SOM;"Francs"
49 NEXT I
```

Et vous pourrez aussi calculer tous les pourcentages qui vous intéressent, déterminer les moyennes de pourcentages, le mois le plus dépensier, le mois le plus "normal", les écarts de dépenses trop importants...

Exercices

1. En 1975 il y avait en France :
4,9 millions de "ménages" de 1 personne.
5,5 millions de "ménages" de 2 personnes.
3,4 millions de "ménages" de 3 personnes.
2,7 millions de "ménages" de 4 personnes.
1,5 millions de "ménages" de 5 personnes.
1,3 millions de "ménages" de 6 personnes ou plus.
Écrire un programme permettant d'afficher successivement :
— le nombre total de personnes vivant en France.
— le nombre de ménages.
— les pourcentages de ménages correspondant à chacune des catégories.

2. Écrire un programme permettant de :
— rentrer un nombre : N
— rentrer une suite de N notes (c'est-à-dire, ici de N nombres compris entre Ø et 2Ø)
— afficher la moyenne de ces notes.

3. Écrire un programme, utilisable à la caisse d'un magasin en permettant de :
— rentrer les prix unitaires de 1Ø produits numérotés de 1 à 1Ø.
— rentrer les quantités achetées de chacun de ces dix produits.
— afficher le prix total à payer.
Aide : l'une des instructions du programme ressemblera sûrement à celle-ci : `S=S+P(I)*Q(I)`

4. Que fait le programme suivant ?

```
1 FOR I=1 TO 6
2 N(I)=0
3 NEXT I
10 FOR J=1 TO 100
11 D=INT(6*RND+1)
12 N(D)=N(D)+1
13 NEXT J
20 FOR I=1 TO 6
21 PRINT "SUR 100 JETS DE DES, LE";I;"ES
T SORTI";N(I);"FOIS"
22 NEXT I
99 END
```

Attention

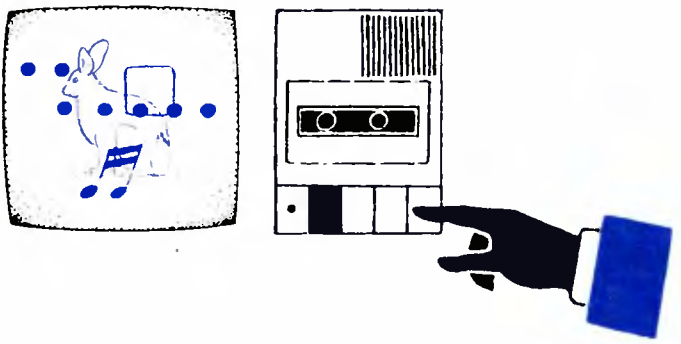
■ Nous pouvons parler entre nous du "tableau DEPENSE" mais la machine ne "comprend", elle, que des noms de variables : DEPENSE(4) vaut quelque chose. Si I vaut 3, DEPENSE(I) vaut DEPENSE(3)... Mais une instruction telle que "PRINT DEPENSE" provoquera l'affichage de quelque chose qui n'aura rien à voir avec le tableau DEPENSE.

■ Si, au lieu de "SOMME", on essaie d'appeler la variable "TOTAL", une erreur sera affichée. En effet, il existe des *mots clefs* dans le langage BASIC : ce sont les mots interprétés comme faisant partie d'une instruction : ainsi PRINT, FOR, TO, NEXT, IF... Aucun nom de variable ne doit commencer par l'un de ces mots (Voir la liste de ces mots en annexe 4) et TOTAL commence par TO !.

Fiche de référence pouvant être consultée :
*** DIM, page 189

Attention ! Lorsque vous souhaitez enregistrer sur cassette un de vos programmes (voir p. 98) ou de vos fichiers (voir p. 109), n'utilisez que des cassettes neuves ou préalablement effacées ; pour effacer, il suffit de faire défiler toute la bande en mode d'enregistrement après avoir frappé MOTORON (voir p. 212) au clavier.
Une fois la bande effacée, frappez MOTOROFF (voir p. 212).

Remarque : il est cependant possible d'enregistrer un nouveau programme ou un nouveau fichier par-dessus un ancien sans l'effacer au préalable, à condition de se positionner bien avant le début de celui-ci.
Pour ce faire, il est recommandé de laisser de larges blancs entre vos différents programmes (MOTORON, MOTOROFF).



12

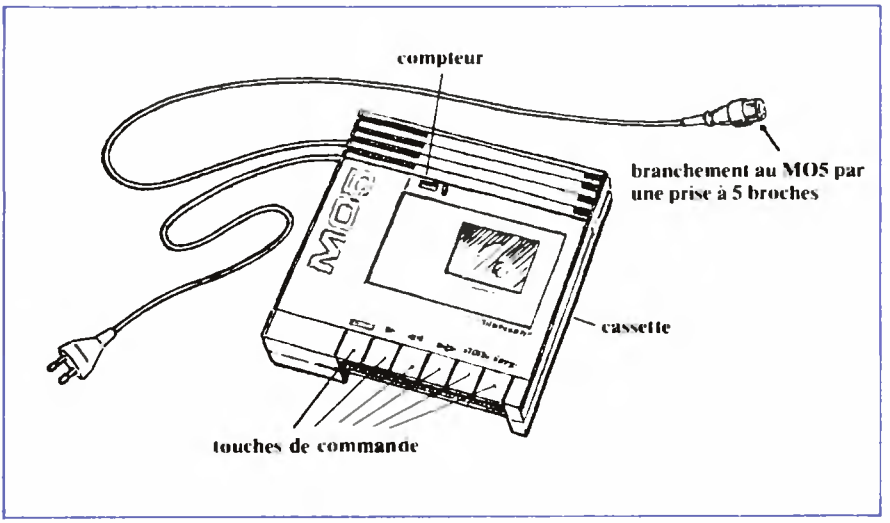
Enregistrer et charger un programme

■ Le lecteur-enregistreur de programmes

Vous commencez maintenant à écrire des programmes un peu longs. Malheureusement, lorsque vous éteignez la machine, le programme enregistré est effacé et vous êtes obligé de le refrapper le lendemain si vous voulez encore l'utiliser.

Il est donc temps d'apprendre à transférer un programme depuis la mémoire du MO5 vers un support plus sûr (par exemple une bande magnétique) et sur lequel il se conservera.

L'appareil le moins cher permettant cette conservation est une sorte de magnétophone nommé *lecteur-enregistreur de programmes* spécifique au MO5 fonctionnant avec les mêmes cassettes magnétiques que celles sur lesquelles vous enregistrez de la musique. (N'utilisez cependant pas de cassettes de durée supérieure à C6Ø et choisissez les cassettes dites "qualité-ordinateur".)



Avant de vous lancer dans son utilisation, il y a de nombreux points à bien étudier :
— Les branchements
— Les touches de commande
— Le placement de la cassette.
Si vous n'avez pas l'habitude, suivez bien l'ordre et les conseils ci-dessous, cela vous évitera quelques crises de nerfs.

■ Les branchements

— Mettez le lecteur-enregistreur (en abrégé LEP) sous tension.
— Connectez le LEP au MO5 grâce au fil et à la prise DIN 5 broches prévue à cet effet.

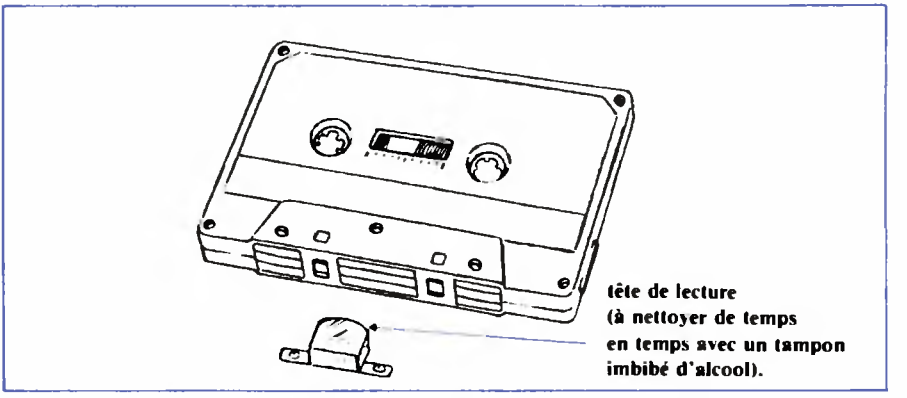
■ Les touches

La touche ▫ , *stop/eject.*, permet d'ouvrir le couvercle afin de placer ou d'enlever la cassette. Elle sert aussi à interrompre l'effet de n'importe quelle touche.
La touche ⊙ , *enreg.*, sera enfoncée lorsqu'on voudra *enregistrer* quelque chose sur la cassette (c'est une touche dangereuse puisqu'elle efface le contenu précédent de la cassette).

La touche ▷ sera enfoncée lorsque l'on voudra *lire* quelque chose (préalablement enregistré) sur la cassette.
Les touches ◁◁ et ▷▷ permettent de faire défiler rapidement la bande de la cassette soit pour la rembobiner (◁◁), soit pour la faire avancer (▷▷) dans le sens de la lecture.

■ Le placement de la cassette

Attention : Il y a deux côtés (et donc deux pistes d'enregistrement possibles) sur une cassette ! Commencer par repérer ces deux côtés en les marquant très lisiblement ; par exemple A d'un côté, B de l'autre.
Choisissez le côté qui vous intéresse et...
... *placer la cassette* dans le LEP (la bande doit passer devant la tête de lecture du LEP, c'est-à-dire vers l'avant).



... *rembobiner la cassette* (touche ◁◁) complètement.
(une cassette est en début de lecture lorsque la partie droite est vide, c'est-à-dire prête à enrouler de la bande).
... *mettre le compteur à zéro*. Cela vous permettra de repérer les numéros où commencent et où finissent les divers enregistrements de la cassette.
... *faire un peu avancer la bande* (touche ▷▷) de 1 ou 2 numéros.
En effet le début de la bande (dite "amorce") est absolument inutilisable pour l'enregistrement.
Votre bande, votre LEP, et vous-même êtes maintenant prêts pour l'enregistrement (ou la lecture) d'un programme.

■ Pour enregistrer ("sauver") un programme

• Assurez-vous d'abord de sa présence en mémoire ; par exemple, commander LIST ENTREE
• Choisissez-lui un nom ; par exemple : TRUC.
• Appuyez simultanément sur *les* touches [o] et [▷] du LEP.
Commandez sur le clavier du MO5 :

```
SAVE"TRUC" ENTREE
```

Le LEP se met alors à faire avancer la bande de la cassette. Votre programme est en train de s'enregistrer sur la bande. Lorsque la bande s'arrête, le message OK apparaît sur l'écran.
Nous vous conseillons alors de :
— noter le numéro du compteur correspondant à la fin de l'enregistrement du programme "TRUC".
— rembobiner la cassette avant de l'éjecter du LEP.

■ Pour charger ("lire") un programme dans le MO5

• Placer dans le LEP la bande sur laquelle est enregistré le programme qui vous intéresse (n'oubliez pas de la rembobiner si elle ne l'est pas déjà).
• Appuyez sur la touche "lecture" : [▷] du LEP.
• Si le programme que vous voulez charger s'appelle "TRUC", commandez sur le clavier du MO5 :

```
LOAD"TRUC" ENTREE
```

Le LEP se met alors à faire avancer la bande de la cassette. Le MO5 affiche alors :

```
Searching
```

Si d'autres programmes sont enregistrés avant TRUC, ils ne sont pas lus mais simplement "parcourus" et le MO5 affiche leur nom, par exemple :

```
Skip: TOTO      BAS
```

("skip" veut dire "sauté" et "BAS" est l'abréviation de "BASIC".)
Lorsque la partie où est enregistrée le programme "TRUC" vient devant la tête de lecture, le message suivant est affiché :

```
Found: TRUC     BAS
```

Laissez faire : votre programme est en train de se charger dans la mémoire du MO5.
Lorsque la bande s'arrête, le message OK apparaît sur l'écran : vous pouvez vérifier que TRUC est maintenant présent en mémoire en commandant LIST ou RUN.

■ Quelques conseils pour gérer vos cassettes

Lorsque vous avez enregistré quelques-uns de vos programmes et recopié quelques programmes de vos nombreux amis, vous risquez de vous mélanger les bandelettes ! Un seul moyen de vous en sortir : soyez très strict sur les méthodes de classement.
Pour vous aider, voici réunis quelques conseils que, bien sûr, vous ne suivrez pas... avant d'avoir compris que vous ne pouvez pas faire autrement...

• Numérotez ou donnez un nom à vos cassettes et marquez leurs deux côtés.
Attention : Notez ce nom *sur* la cassette (avec un autocollant blanc) et non sur la boîte (les boîtes sont interchangeables)

• Tenez un cahier où vous notez pour chaque cassette (repérée par son nom), la suite des enregistrements et les numéros-compteurs correspondants.

• Rembobinez toujours une cassette avant de la sortir du LEP.

• Enregistrez vos programmes sur 2 cassettes : l'une dont vous vous servez, l'autre que vous stockez par précaution.

• Placez vos cassettes dans des boîtes.

• Régulièrement : récupérez les cassettes contenant des enregistrements (de travail) dont vous ne vous servez plus et enregistrez du "blanc".

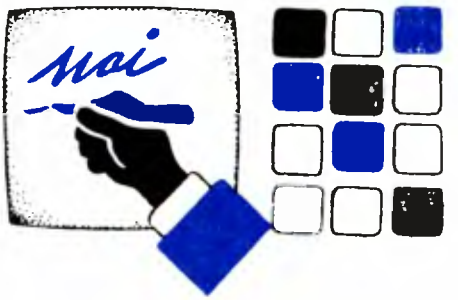
• Si vous tenez à un programme, enregistrez le deux fois sur une même cassette et une autre fois sur une cassette à laquelle vous touchez le moins possible.

• Si des erreurs se produisent, gardez votre calme et réfléchissez.

• Nettoyez régulièrement la tête de lecture de votre LEP avec un tampon imbibé d'alcool.

Fiche de référence pouvant être consultée
* SAVE, LOAD, RUN", MERGE, SKIPF, MOTORON MOTOROFF, page 210

Page blanche



13

Le crayon optique

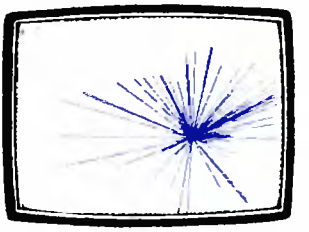
Jusque-là, vous ne pouviez rentrer des données dans votre MO5 que par le clavier (ou le LEP). Si vous disposez de *l'extension "crayon optique"*, vous allez pouvoir utiliser l'écran comme périphérique d'*entrée* (et non plus seulement comme *sortie d'image*). Le crayon optique peut, en effet, détecter le passage d'un spot lumineux sur l'écran de télévision. Voici, par exemple, un premier programme utilisant cette possibilité.

■ L'oursin optique

```
10 CLS
20 INPUT PEN X,Y
30 LINE (160,100)-(X,Y)
40 GOTO 20
```

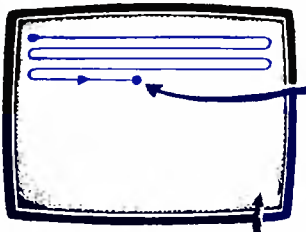
Après RUN ENTREE, prenez votre crayon optique et appuyez le bien perpendiculairement quelque part sur l'écran.
Comme l'instruction 30 le demande, une ligne se trace alors, joignant le centre de l'écran au point où vous avez appuyé. En appuyant à dif-

férents endroits de l'écran vous obtenez alors un magnifique "oursin" étoilé du plus bel effet.
Comment cela fonctionne-t-il ?



■ L'instruction INPUT PEN X, Y

Comme l'instruction INPUT A, l'instruction INPUT PEN X, Y est une instruction d'arrêt : l'exécution est interrompue et la machine attend un appui sur l'extrémité du crayon.
Remarquez en effet que l'extrémité noire du crayon cède à la pression comme un interrupteur.
Lorsque cette pression est exercée, le rayon lumineux qui passe devant l'extrémité du crayon est intercepté ; le MO5 est alors capable de calculer les coordonnées, sur l'écran, du point où ce rayon a été intercepté. Ces coordonnées sont affectées aux variables X et Y indiquées dans l'instruction INPUT PEN et l'exécution continue à l'instruction suivante.
La manière dont le MO5 calcule les coordonnées du point d'appui du crayon est intéressante : vous savez peut-être que l'image télévision est fabriquée par un pinceau lumineux qui parcourt l'écran 25 fois par seconde. Ce pinceau parcourt donc chacun des 64 ØØØ points de l'écran en 1/25 de secondes.



Il suffit donc, pour repérer la position du crayon, de repérer l'instant où le pinceau lumineux part du haut de l'écran et l'instant où il passe devant le crayon : (une horloge du MO5, synchronisée avec le balayage, suffit à ce repérage) ; un simple règle de trois et quelques calculs permettent alors de retrouver la ligne et la colonne où le crayon a capté le pinceau lumineux.

■ Quelques sources d'erreur

• Le pinceau lumineux ne sera pas reçu par le crayon lorsque *la luminosité* de l'image n'est pas suffisante, mais aussi lorsque l'endroit où vous pointez est *rouge ou noir* (ces couleurs sont trop "opaques").
Les variables X et Y prennent alors la valeur − 1 et une erreur se produit lorsque vous essayez de tracer une figure utilisant ces coordonnées, puisqu'elles sont négatives.
Pour pallier cette difficulté il vaut mieux s'assurer une bonne lecture en bouclant sur l'instruction INPUT PEN tant que X et Y n'ont pas de valeur positive. Par exemple comme ceci :

```
20 INPUTPEN X,Y : IF X<0 THEN 20
```

• Il n'est pas recommandé de faire suivre (sans le vouloir explicitement) deux instructions INPUT PEN sans s'assurer qu'un temps suffisant s'est écoulé entre leurs exécutions respectives. Un bon moyen consiste à faire suivre chaque lecture du crayon par l'émission d'une note de musique (ou d'un petit air plus long). L'utilisateur a alors l'impression que la lecture est accompagnée d'un bruit et on est assuré qu'elle a bien eu lieu...
Voici un petit programme nécessitant une telle précaution pour dessiner 15 rectangles de toutes les couleurs.

```
10 CLS:SCREEN 0,7,7
20 FOR COUL=1 TO 15
30 INPUT PEN A,B:IF A<0 THEN 30:IF B<0 T
HEN 30
31 PLAY "DO"
40 INPUT PEN C,D:IF C<0 THEN 40:IF D<0 T
HEN 40
41 PLAY "MI"
50 BOXF(A,B)-(C,D),COUL
60 NEXT COUL
99 END
```

Essayer le même programme après avoir supprimé les instructions 31 et 41 ! Ce qui ne marche pas alors est dû au fait que la machine effectue *deux* lectures du crayon optique alors que l'utilisateur appuie sur l'un des sommets du rectangle désiré... et le rectangle défini par ces "deux" lectures est ridiculement petit.

■ Un crayon ! deux organes !

En fait le crayon optique est un double organe des sens pour le MO5. Nous avons vu en effet qu'il était capable de "sentir" deux phénomènes physiques : l'appui sur son extrémité (c'est donc un organe de toucher) et le passage d'un rayon lumineux (c'est donc un organe de la vue).
Pour plus de détails voir la fiche de référence.
Et, pour votre plaisir, voici un programme qui fait de votre crayon optique un véritable stylo sur écran, puisque le MO5 trace des traits qui suivent exactement le mouvement de votre crayon.

Un programme pour dessiner sur l'écran :

```
5 PRINT "ANNONCEZ LA COULEUR"
10 INPUT C:CLS:SCREEN C,7,7
20 INPUTPEN X,Y:IF X<0 THEN 20:IF Y<0 TH
EN 20
30 PSET(X,Y)
40 INPUTPEN X,Y:IF X<0 THEN 40:IF Y<0 TH
EN 40
50 LINE -(X,Y)
60 GOTO 40
```

(Avec l'instruction 40 INPEN..., vous pouvez même dessiner en "pointant" votre crayon jusqu'à environ 20 cm de l'écran).

Exercices

1. Que fait le programme-gag suivant ?

```
10 CLS
20 INPUTPEN X,Y
30 LOCATE X/8,Y/8
40 PRINT "Aie";
50 GOTO 20
```

Il est surtout "gag" lorsque vous appuyez le crayon sur la partie droite de l'écran, vers la 39e position !

2. Faites un programme qui écrive "BONJOUR" à partir de l'endroit où l'utilisateur pointe le crayon. Arrangez-vous pour qu'aucune erreur de lecture ne soit possible.

3. Faites un programme dessinant des carrés dont l'utilisateur pointe successivement le centre puis un sommet.

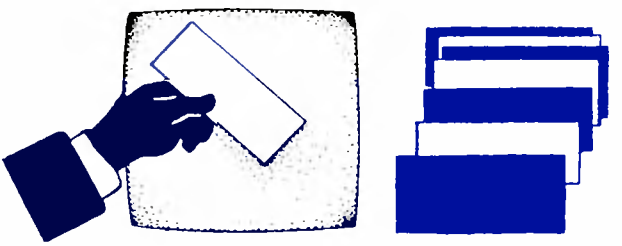
4. Faites un programme qui choisit une ligne au hasard entre Ø et 39 puis qui répond "plus bas SVP" ou "plus haut SVP" chaque fois que l'utilisateur pointe son crayon à une certaine hauteur sur l'écran.

Attention

■ Tenez bien compte du paragraphe "Quelques sources d'erreur".
Réglez la luminosité de votre téléviseur à un niveau suffisant.
Tenez bien votre crayon perpendiculairement à l'écran l'interrupteur étant au dessus du petit trou "capteur" du rayon.

Fiche pouvant être consultée :
** TUNE,INPUT PEN,INPEN,PTRIG, page 215

Page blanche

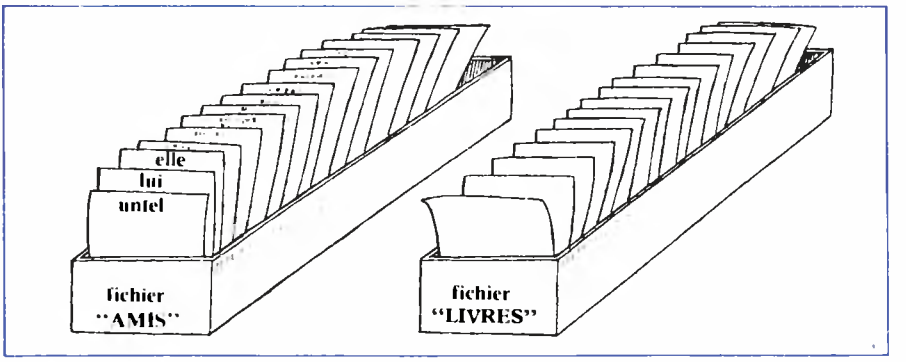


14

Les fichiers

■ Un ensemble de fiches

Un "fichier" de données est une suite de données rangées les unes derrière les autres comme des "fiches" dans une boîte.



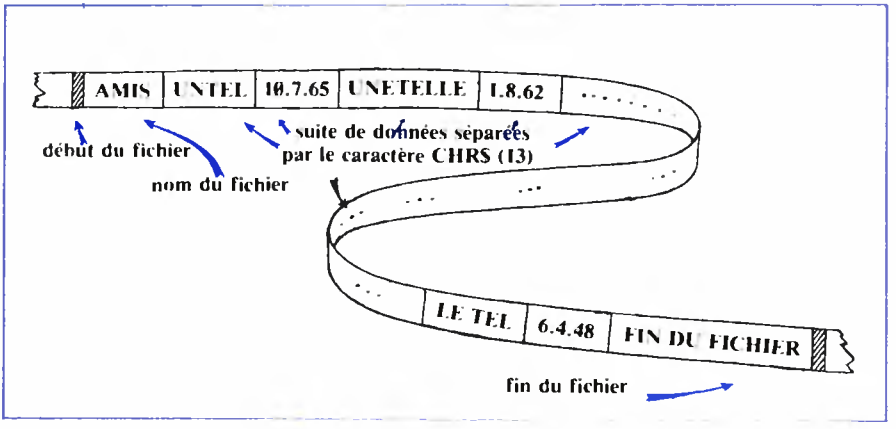
Chacune des "fiches" contient un certain nombre d'informations relatives à l'"article" correspondant.

Il peut s'agir, par exemple, du fichier de vos amis, chaque fiche contenant son adresse, son numéro de téléphone et sa date de naissance ; ou bien du fichier des clients d'une entreprise, chaque fiche contenant ses caractéristiques, l'état de ses commandes et de son compte ; ou bien du fichier des livres d'une bibliothèque... C'est en tout cas un

ensemble de fiches destiné à être ou bien feuilleté ou bien modifié (addition ou soustraction d'une fiche). Pour consulter un fichier ou pour écrire dedans, il faut successivement :
— Trouver la bonne boîte (grâce à son "nom", par exemple) et "l'ouvrir".
— Feuilleter les fiches jusqu'à ce que l'on trouve celle qui nous intéresse.
— Prendre la fiche et l'amener devant celui qui doit la traiter (par le "canal" approprié)
— Traiter les informations contenues dans la fiche ou la modifier ; puis la remettre dans le fichier.
— Ne pas oublier de "fermer" la boîte (sinon les couvercles se mélangeraient.)

■ Fichiers sur cassettes

Les fichiers traités par un ordinateur se constituent et se consultent un peu comme les ensembles de fiches dans des boîtes ; leur "support" physique est, en général, une bande ou une disquette magnétique.
Grâce au LEP, il est ainsi possible de stocker sur une cassette, à partir du MO5, des "fichiers" ou des listes de données destinées à être relues plus tard les unes après les autres.
Pratiquement un fichier sur cassette est constitué :
— d'un premier enregistrement contenant le nom du fichier
— d'une liste de données alphabétiques ou numériques
— d'un code signalant la fin des enregistrements de ce fichier.



Les fichiers ici décrits sont dits *séquentiels*.
Cela signifie qu'il n'est pas possible d'atteindre une donnée quelque part au milieu du fichier sans avoir fait défiler la suite de toutes les données précédentes. En effet, aucune des données n'a de "nom" permettant de l'"appeler" ; pour consulter un tel fichier il faut avoir bien noté la structure des enregistrements au moment de sa constitution afin de "reconnaître" chacune des données au fur et à mesure de leur défilement. Dans l'exemple du schéma précédent, chaque article du fichier AMIS est constitué de : un nom, une suite de chiffres représentant la date de naissance.

■ Pour enregistrer un fichier sur une cassette, il faut :

1. S'assurer que le lecteur-enregistreur est en position d'enregistrement avec la cassette positionnée à l'endroit où vous désirez que s'effectue l'enregistrement.
2. Ouvrir un canal, en sortie, pour permettre le passage des données du MO5 vers le lecteur-enregistreur.
L'instruction

```
OPEN"O",#1,"AMIS"
```

ouvre le fichier de nom "AMIS"en *O*utput (c'est-à-dire que l'on va écrire dessus), le *canal* réservé à sa manipulation par le MO5 portant le numéro 1.
3. Commander le passage des données sur la cassette.
L'instruction :

```
PRINT #1,A$,T$
```

commande l'envoi des variables A$ et T$ par le canal 1 (lequel a été affecté à un certain nom de fichier par l'instruction OPEN)
4. Ne pas oublier de fermer le canal après son utilisation.
L'instruction CLOSE # 1 commande cette fermeture.

Voici un exemple d'enregistrement de données dans un fichier sur cassette. On y voit qu'une suite d'instructions "DATA", dans un programme, n'est pas autre chose qu'un fichier séquentiel. Ce fichier est consulté grâce à l'instruction "READ" (Le "pointeur-des-DATA" joue le même rôle que la tête de lecture du LEP : à un instant donné, il est prêt à lire la donnée suivante dès que l'ordre lui en

sera communiqué). Notez que "DATA" signifie "données".

```
10 DATA UNTEL,345 26 33,LUI,235 26 47,MO
I,010 37 48
11 DATA ELLE,720 14 10,PERSONNE,456 78 9
0
19 DATA Z,0
100 '
101 ' *** ENREGISTREMENT ***
110 OPEN"O",#1,"TELEPHON"
120 READ A$,T$
130 IF A$="Z" THEN 190
140 PRINT #1,A$,T$
150 GOTO 120
190 CLOSE #1
199 STOP
```

Note : les données de l'instruction 19 permettent de repérer la fin de la liste des données.
Évidemment, avant d'exécuter ce programme, il faut que le LEP soit en position d'enregistrement avec une bande magnétique dans le lecteur. Vous devez avoir bien repéré le début de l'enregistrement qui va avoir lieu.

■ Pour lire un fichier, d'une cassette vers la mémoire du MO5, il faut :

— s'assurer que le lecteur-enregistreur est en position de lecture avec la cassette positionnée avant l'endroit où sont enregistrés les fichiers qui seront lus.
— ouvrir un canal, en entrée, pour permettre le passage des données du lecteur-enregistreur vers le MO5.
L'instruction

```
OPEN"I",#1,"AMIS"
```

ouvre le fichier "AMIS" en *I*NPUT (c'est-à-dire que l'on va lire dessus), le canal réservé à sa manipulation portant le numéro 1.
— commander la lecture des données enregistrées sur la cassette :

```
INPUT#1,A$,B$
```

— ne pas oublier de fermer le canal après son utilisation :

```
CLOSE#1
```

```
1 CLEAR 600
2 DIM A$(30),T$(30)
200 '
201 ' ***** LECTURE *****
210 OPEN"I",#1,"TELEPHON"
215 I=1
220 IF EOF(1) THEN 290
230 INPUT #1,A$(I),T$(I)
240 I=I+1 : GOTO 220
290 CLOSE #1
299 STOP
```

Voici, maintenant un exemple de lecture de fichier.
Au fur et à mesure de leur lecture, les données se voient nommées et rangées comme les éléments de deux tableaux, ce qui permettra ensuite de les rappeler (l'instruction 1 réserve 30 places en mémoire pour chacun de ces tableaux).
La variable EOF (1) prend la valeur VRAI lorsque la dernière donnée du fichier traitée par le canal 1 vient d'être lue (EOF est l'abréviation de *E*nd *O*f *F*ile et signifie "fin de fichier").
Note : Évidemment, avant l'exécution de ce programme, la bande doit être positionnée de manière à ce que le fichier "TELEPHON" soit après la tête de lecture du LEP.

Voici enfin un programme permettant de sortir sur imprimante le fichier lu par le programme précédent.

```
300 ' ***** IMPRESSION *****
310 OPEN"O",#2,"LPRT:"
320 FOR K=1 TO I
330 PRINT #2,"NOM : ";A$(K),"TELEPHONE :
 ";T$(K)
340 NEXT K
350 CLOSE #2
399 END
```

Fiches de référence pouvant être consultées :
* OPEN, CLOSE, EOF, PRINT #, INPUT #, LINE INPUT #, page 219
* "LPRT :", PRINT #, page 217

Page blanche

Conseils de programmation

■ Comprendre ce que l'ordinateur peut faire...

□ *Savoir programmer* c'est d'abord *savoir ce qu'il est possible de programmer*, c'est-à-dire ce que votre ordinateur est exactement capable de faire.
Vous commencez certainement à avoir une bonne idée de ce que vous pouvez demander à votre MO5. En voici un rapide résumé :
— déplacer un curseur sur l'écran, écrire quelques caractères à tout endroit voulu, dessiner quelques segments de droites, colorier des rectangles, les effacer et donc les faire clignoter ou se déplacer...
— tirer des nombres au hasard, jouer quelques notes de musique, faire des opérations et modifier la valeur de certaines variables.
— s'arrêter pour laisser l'utilisateur frapper quelques touches et donc répondre à des questions ou donner des éléments de calculs...
— répéter telle séquence ou comparer des nombres ou des mots et faire telle ou telle chose en fonction du résultat de la comparaison...

■ ... et "structurer son programme".

□ Après avoir appris la syntaxe des différentes instructions acceptées par une machine dans un certain langage, on croit souvent que la programmation n'est plus qu'un tricotage patient consistant à aligner des instructions numéro après numéro.
Or, écrire les instructions les unes après les autres de la première à la dernière est une très mauvaise méthode de programmation, on aboutit alors souvent (surtout après quelques améliorations) à un style de programme dénommé "style spaghetti" ; en effet, les sauts, conditionnels ou non, ont une fâcheuse tendance à se chevaucher les uns les autres de sorte que même l'auteur du programme perd le fil de son Ariane dans un inextricable lacis d'instructions.
Nous vous suggérons plutôt d'opérer avec méthode et sagesse en suivant les étapes que nous développons ci-après.

Étape numéro 1 : L'ÉNONCÉ

Énoncez clairement en français et en peu de mots ce que vous attendez de votre programme.

Si cela se passe sur l'écran, précisez-en la mise en page générale.

Étape numéro 2 : LES VARIABLES ET LEURS NOMS

Avant d'entrer dans le détail des actions à commander, il vous faut en nommer les "acteurs" significatifs. Il s'agit ici des *variables* dont les valeurs vont définir la nature et la position des éléments en jeu et qui vont permettre d'en décrire l'évolution.

Nommez les variables que vous croyez devoir utiliser et précisez-en la signification.

Étape numéro 3 : LE SQUELETTE DU PROGRAMME

Il n'est pas nécessaire d'écrire tout de suite votre programme en BASIC : Vous pouvez l'écrire en français, en essayant de limiter l'énoncé des ordres à un ensemble de mots et expressions proche du BASIC de votre MO5. Par exemple :

"afficher, déplacer à droite, choisir au hasard entre 1 et 1Ø0, faire ceci pour I variant de 1Ø à 2Ø, faire cela tant que..., Selon N, faire ou bien 1° ... ou bien 2°... ou bien 3° ..., entrer A, B, C,..."

Écrivez le "squelette" de votre programme en explicitant et en isolant l'essentiel des actions à accomplir et leur ordre de succession.

Vous avez ainsi précisé les grandes lignes et les gros blocs de votre programme.

Laissez de côté pour l'instant tout ce qui fera plus tard l'agrément de votre programme : fioritures et sophistications graphiques et musicales.

Étape numéro 4 : L'ÉCRITURE

Traduisez en BASIC chacun des blocs définis à l'étape précédente.

Essayez alors de ne pas écrire de bloc dépassant une douzaine d'instructions BASIC. Si cela devait être le cas, considérez tout bloc important comme un véritable programme et reprenez pour lui les étapes 2 et 3.

A partir de maintenant n'oubliez pas de conserver sur cassette les différentes versions successives de votre "grand œuvre" !

Étape numéro 5 : LA MISE AU POINT

Vérifiez que votre programme exécute bien ce que vous voulez. Essayez les cas particuliers ; "ratissez" le en vérifiant que le recollement des blocs et l'utilisation de variables secondaires s'effectuent sans problème.

Étape numéro 6 : LA DOCUMENTATION

Lorsque tout à l'air de marcher faites un petit effort :
Documentez votre programme.
Le "documenter", c'est, si possible, lui ajouter des instructions de commentaires qui rendent sa liste la plus lisible possible. Pensez que ce programme pourra dormir pendant quelques mois, et que vous voudrez peut-être un jour le réutiliser ou l'améliorer ; et vous aurez alors oublié sa structure, la signification des variables, les valeurs pertinentes des données...
Le mieux est de conserver un petit dossier comprenant au moins :
— *la liste* où les structures apparaissent grâce à des encadrements ou des décalages ou des segments fléchés...
— *le dictionnaire des variables.*
— un ou deux *exemples pertinents d'exécution* où apparaissent les données et une copie d'écran en cours d'utilisation.

Étape numéro 7 : LA SOPHISTICATION

La sophistication peut porter sur la forme ou sur le fond.
Pour la forme, il est temps de *soigner la présentation :* pensez à l'animation des messages, aux couleurs, à la musique...
Si les choix de l'utilisateur sont un peu complexes, présenter un véritable *menu* en début d'exécution.
Pour le fond, pensez aux petites exceptions qui empêchent votre programme d'être général ; *essayez d'étendre son champ d'application* en modifiant des blocs d'instructions déjà écrits ou en en créeant de nouveaux.

Le nettoyage de votre programme est alors souvent nécessaire car les ajustements et les sophistications vous ont obligé à modifier certaines conditions, à utiliser des noms de variables divers, à compliquer un bloc trop lourdement...

Quelques conseils pour les "gros" programmes :

Vous allez très vite vous laisser aller à écrire des programmes de plus en plus importants.
Si vous n'avez pas de problème de place en mémoire, il est bon d'aérer la présentation de la liste en multipliant les commentaires.
Mais, si, par contre, vous en avez besoin ou si vous voulez gagner sur le temps d'exécution voici quelques conseils techniques :

- Ne pas utiliser de variables à nom trop long.
- Grouper plusieurs instructions par ligne numérotée.
- Supprimer les REM.

(attention ces 3 conseils ne vont pas dans le sens d'une bonne lisibilité des programmes : ici l'inutile est lié à l'agréable).

- Utiliser des variables entières quand cela est possible, (DEFINT est préférable au ... %. Voir les fiches de référence), en particulier dans les instructions FOR...NEXT.
- Pour les tableaux de moins de 10 éléments, expliciter la dimension.
- Éviter les calculs emboîtés avec parenthèses.
- Si un bloc d'instructions n'est appelé qu'une fois par un GOSUB, remplacer par deux GOTO.
- Plutôt que des constantes, utiliser des variables (d'une part cela permet un paramétrage souvent utile, d'autre part cela économise le temps de conversion en binaire.)
- Initialiser les variables en début de programme dans l'ordre de leur fréquence d'utilisation (cela économise le temps de recherche).

La programmation reste un art difficile !

En tentant de suivre les étapes précédentes vous éviterez peut être la panique qui prend parfois le "bidouilleur" ; ce genre de programmeur se lance en général, sans délai, dans l'écriture du noyau qui lui semble le plus intéressant quelque part au milieu du programme. Après l'accumulation et le greffage des instructions les unes aux autres, les variables se bousculent, les modifications provoquent des erreurs en boule de neige et le découragement finit par l'emporter sur l'enthousiasme...

■ Exemple

Prenons l'exemple du programme de "mise au pluriel" (Leçon 9 - exercice 4) et examinons ce que donnent les différentes étapes.

Étape 1 :
L'utilisateur entrera un mot au singulier ; la machine devra afficher son pluriel en tenant compte de certaines exceptions.

Étape 2 :
Les types d'exceptions seront numérotés de 1 à NEC.
(d'abord limité à 5) :
cas numéro 1 : les mots pouvant avoir deux pluriels
cas numéro 2 : les mots en AIL
cas numéro 3 : les mots en OU
cas numéro 4 : les mots en AL
cas numéro 5 : les mots en EU et AU
Le nombre d'exceptions de type I sera noté NBEXC(I). Le J^{ième} mot de la liste d'exceptions numéro I sera noté EXCEP$ (I,J).
Le mot entré au singulier sera noté MOT$. Son pluriel sera noté PLUR$.
Chaque mot pourra être découpé en un début de mot, noté DEB$, et une terminaison notée FIN$.

Étape 3

1 à 9 *Initialisation*
Effacer l'écran
NEC=5

10 à 19 *Données*
Les 5 listes d'exceptions.
Une dernière liste contient les deux formes de pluriel des mots de la liste numéro 1.

20 à 70 *Lecture des données*

80 *Entrée du mot*

— *Traitement :*

90 Si le mot se termine par S, X ou Z, alors PLUR$=MOT$. Aller à *Fin.*
Sinon PLUR$=MOT$+"S".

100 Si le mot est dans la liste numéro 1, alors PLUR$ est à prendre dans la dernière liste, et aller à *Fin.*

200 Si le mot se termine par AIL, alors PLUR$ = ...AILS.
Mais, si le mot est dans la liste numéro 2, alors
PLUR$=...AUX, et aller à *Fin.*

300 Si le mot se termine par OU, alors PLUR$ = ...OUS.
Mais, si le mot est dans la liste numéro 3, alors
PLUR$ = ...OUX, et aller à *Fin.*

400 Si le mot se termine par AL, alors PLUR$= ...AUX.
Mais, si le mot est dans la liste numéro 4, alors
PLUR$ = ...ALS, et aller à *Fin.*

500 Si le mot se termine par AU alors PLUR$ = ...AUX.
et, si le mot est dans la liste numéro 5
alors PLUR$ = ...AUS, et aller à *Fin.*
Si le mot se termine par EU alors PLUR$ = ... EUX.
et si le mot est dans la liste numéro 5
alors PLUR$ = ... EUS, et aller à *Fin.*

900 *Fin :*
Afficher le pluriel PLUR$
Retourner (en 80) pour entrer un autre mot.

Étape 4 :

Ici sont écrits des blocs de programme comme celui-ci :

```
'.....LECTURE DES EXCEPTIONS
30 FOR I=1 TO NEL: READ NBEXC(I)
40 FOR J=1 TO NBEXC(I):READ EXCEP$(I,J):
NEXT J
50 NEXT I
60 FOR J=1 TO NBEXC(1):READ PLURIEL$(J):
NEXT J
70 '
```

...ou celui là :

```
200 '  ...AIL          ??????????????????
210 FIN$=RIGHT$(MOT$,3):DEB$=LEFT$(MOT$,
LEN(MOT$)-3)
220 IF FIN$="AIL" THEN PLUR$=DEB$+"AILS"
 ELSE 300
230 FOR J=1 TO NBEXC(2):IF MOT$=EXCEP$(2
,J) THEN PLUR$=DEB$+"AUX":GOTO 900  ELSE
 NEXT J
```

Le résultat de cette étape donne quelque chose qui peut ressembler au programme dont la liste est donnée aux pages suivantes,(120 et 121).

Étape 5

Lors de la mise au point, on s'aperçoit alors par exemple que les mots de 2 lettres ne sont pas acceptés à cause de l'instruction 21Ø qui provoque une erreur.
Il faut donc rajouter l'instruction :

```
95 IF LEN(MOT$)<3 THEN 900
```

En "ratissant" le programme, on peut aussi voir que les instructions 41Ø et 51Ø sont inutiles en début de leurs blocs puisque l'instruction 31Ø, déjà exécutée dans un bloc précédent a donné leurs valeurs à FIN$ et DEB$.

Étape 6

La "documentation" de ce programme ressemblerait beaucoup à ce qui est écrit ci dessus !

Étape 7

Pour la "sophistication" on peut penser aux multiples autres cas particuliers que nous offre la langue française : les mots sans pluriel, les mots en ...UM, les cas très "singuliers" comme les mots d'origine étrangères ou les mots composés...
Remarquez que les "DATA" 11 et 19 peuvent recevoir mot par mot les cas isolés.
Nous avons aussi réservé les lignes de programmes 16, 17, 18 et 6ØØ..., 7ØØ..., 8ØØ... pour compléter le traitement des cas d'exceptions. C'est maintenant à vous de jouer !
Mais si vous allongez un peu vos listes vous aurez sûrement besoin des instructions supplémentaires :

```
2 CLEAR 1000
3 DIM EXCEP$(10,20)
```

Bon courage !

```
1 '  PLURIELPLURIELPLURIELPLURIELPLURIEL
5 CLS:NEC=5
10 '
.....................LISTE DES EXCEPTIONS
11 DATA 7,AIEUL,AIL,CIEL,IDEAL,OEIL,TRAV
AIL,VAL
12 DATA 6,BAIL,CORAIL,EMAIL,SOUPIRAIL,VA
NTAIL,VITRAIL
13 DATA 7,BIJOU,CAILLOU,CHOU,GENOU,HIBOU
,JOUJOU,POU
14 DATA 7,BAL,CAL,CARNAVAL,CHACAL,FESTIV
AL,PAL,REGAL
15 DATA 5,BLEU,EMEU,LANDAU,PNEU,SARRAU
16 '
17 '
18 '
19 DATA AIEULS & AIEUX,AILS & AUX,CIELS
 & CIEUX,IDEALS & IDEAUX,OEILS & YEUX,TR
AVAILS & TRAVAUX,VALS & VAUX
20 '
.....................LECTURE DES EXCEPTIONS
30 FOR I=1 TO NEC :READ NBEXC(I)
40 FOR J=1 TO NBEXC(I):READ EXCEP$(I,J):
NEXT J
50 NEXT I
60 FOR J=1 TO NBEXC(1):READ PLURIEL$(J):
NEXT J
70 '
.....................SINGULIER >>> PLURIEL
79 '
80 INPUT "MOT";MOT$
89 '
90 '  ...S ...X ...Z ????????????????????
91 FIN$=RIGHT$(MOT$,1)
92 IF FIN$="S" OR FIN$="X" OR FIN$="Z"
THEN PLUR$=MOT$:GOTO900
93 '
94 PLUR$=MOT$+"S"
99 '
100 '  ...  ...        ????????????????????
101 FOR J=1 TO NBEXC(1):IF MOT$=EXCEP$(1
,J) THEN PLUR$=PLURIEL$(J):GOTO 900 ELSE
 NEXT J
199 '
200 '  ...AIL          ????????????????????
210 FIN$=RIGHT$(MOT$,3):DEB$=LEFT$(MOT$,
LEN(MOT$)-3)
```

```
220 IF FIN$="AIL" THEN PLUR$=DEB$+"AILS"
 ELSE 300
230 FOR J=1 TO NBEXC(2): IF MOT$=EXCEP$(2
,J) THEN PLUR$=DEB$+"AUX":GOTO 900  ELSE
 NEXT J
299 '
300 '  ...OU              ??????????????????
310 FIN$=RIGHT$(MOT$,2):DEB$=LEFT$(MOT$,
LEN(MOT$)-2)
320 IF FIN$="OU" THEN PLUR$=DEB$+"OUS" E
LSE 400
330 FOR J=1 TO NBEXC(3): IF MOT$=EXCEP$(3
,J) THEN PLUR$=DEB$+"OUX":GOTO 900  ELSE
 NEXT J
399 '
400 '  ...AL              ??????????:???????
410 FIN$=RIGHT$(MOT$,2):DEB$=LEFT$(MOT$,
LEN(MOT$)-2)
420 IF FIN$="AL" THEN PLUR$=DEB$+"AUX" E
LSE 500
430 FOR J=1 TO NBEXC(4): IF MOT$=EXCEP$(4
,J) THEN PLUR$=DEB$+"ALS":GOTO 900  ELSE
 NEXT J
499 '
500 '  ...AU  ...EU    ??????????????????
510 FIN$=RIGHT$(MOT$,2):DEB$=LEFT$(MOT$,
LEN(MOT$)-2)
520 IF FIN$="AU" THEN PLUR$=DEB$+"AUX" E
LSE GOTO 540
530 FOR J=1 TO NBEXC(5): IF MOT$=EXCEP$(5
,J) THEN PLUR$=DEB$+"AUS":GOTO 900  ELSE
 NEXTJ
540 IF FIN$="EU" THEN PLUR$=DEB$+"EUX" E
LSE 600
550 FOR J=1 TO NBEXC(5):IF MOT$=EXCEP$(5
,J) THEN PLUR$=DEB$+"EUS":GOTO 900 :NEXT
J
599 '
600 '                  ??????????????????
699 '
700 '                  ??????????????????
799 '
800 '                  ??????????????????
899 '
900 PRINT"PLURIEL: ";PLUR$:PRINT:GOTO 80
999 END
```

Page blanche

correction des exercices

2e leçon

Exercice 2

En écrivant bleu sur fond bleu, on ne voit évidemment rien ! Mais en frappant...

ENTREE SCREEN 4,6,6 ENTREE

vous "démasquez" tous les textes d'un coup.

Exercice 3

```
4096
```

Exercice 4

```
1001
285714          428571          571428
714285          857142          999999
111             1111            11111
```

3e leçon

Exercice 2

```
1 SCREEN 3,0,0
2 SCREEN 1,0,0
3 SCREEN 5,0,0
4 SCREEN 7,0,0
5 SCREEN 6,0,0
6 SCREEN 2,0,0
7 SCREEN 4,0,0
```

Exercice 3

```
1 SCREEN 1,3,0
2 PRINT   17
3 COLOR 4
4 PRINT 17*17
5 COLOR 0
6 PRINT 17*17*17
```

Exercice 4

```
1 CLS
2 LOCATE19,1:PRINT2
3 LOCATE19,23:PRINT4
4 LOCATE1,12:PRINT3
5 LOCATE19,12:PRINT0
6 LOCATE38,12:PRINT1
7 END
```

4e leçon

Exercice 1

a) Le compteur démarre, non plus à Ø, mais à 2.
b) Les puissances de 2 s'affichent.
c) Les puissances de 2 s'affichent.
d) 2,4,16,256,... une suite exponentielle s'affiche.
e) Les nombres ne s'affichent plus au même endroit et les lignes défilent indéfiniment.

Exercice 2

Les changements de couleur ont lieu tant que I < 16.

Exercice 3

Les couleurs défilent jusqu'à ce que K = 16.

5e leçon

Exercice 1

```
10 SCREEN 0,10,10
20 PSET (100,60)
30 LINE -(200,60)
40 LINE -(150,5)
50 LINE -(100,60)
60 BOX (100,61)-(200,150)
70 BOX (130,100)-(170,150)
80 END
```

Exercice 2

```
1 CLS
2 FOR LIGNE=0 TO 199
3 LINE(0,LIGNE)-(319,LIGNE),1
4 NEXT LIGNE
```

```
10 FOR COL=0 TO 159
11 LINE(COL,0)-(COL,199),4
12 LINE(319-COL,0)-(319-COL,199),4
13 NEXT COL
20 END
```

Exercice 3

```
1 CLS
10 SCREEN 0,10,0
20 C=0:L=0:T=0
30 BOXF(C,L)-(199-C,199-L),T
40 C=C+8:L=L+8:T=T+1
50 GOTO 30
```

Exercice 4

Le programme dessine une "enveloppe", sans "lever le crayon". Il s'agit de la "maison d'Euler".

6e leçon

Exercice 1

```
10 CLS:SCREEN 1,0,1
20 FOR C=10 TO 310 STEP 15
30 LINE (C,00)-(C,199)
40 NEXT C
50 FOR L=10 TO 199 STEP 15
60 LINE (0,L)-(319,L)
70 NEXT L
80 END
```

Exercice 3

```
1 CLS:SCREEN1,0,0
2 FOR K=0 TO 319
3 LINE(0,0)-(K,199)
4 NEXT K
```

```
10 CLS:SCREEN1,0,0
20 FOR K=0 TO 319
30 LINE(K,0)-(319-K,199)
40 NEXT K
```

Exercice 5

```
100 CLS:SCREEN1,0,0
200 FOR K=0 TO 319
300 LINE(159,0)-(K,199)
400 NEXT K
```

7e leçon

Exercice 2

Des points sont allumés au hasard sur l'écran comme pendant une nuit étoilée.

8e leçon

Exercice 1

```
70 PRINT"DANS ";ANNEE;" AN";
71 IF ANNEE 1 THEN PRINT"s";
72 PRINT"; L'OBJET COUTERA ";PRIX;"FRANC
S"
```

Exercice 2

Remplacez les lignes 1,2,3 par :

```
1 INPUT ANS,MOIS,JOURS
```

Exercice 4

```
10 CLS:SCREEN7,0,0
20 INPUT"ENTREZ LES COORDONNES DU 1er SO
MMET      ";C,L
30 INPUT"ENTREZ LES COORDONNES DU 2eme S
OMMET    ";C1,L1
40 BOXF(C,L)-(C1,L1),-3
50 LINE (C,L)-(C1,L1),1
60 END
```

Exercice 5

```
10 CLS
20 FOR I=1 TO 3
30 PRINT "ENTREZ LA COULEUR No";I
40 INPUT C(I)
50 NEXT I
60 CLS:I=0
70 FOR P=0 TO 140 STEP 70
80 I=I+1
90 BOXF(0+P,50)-(70+P,150),C(I)
100 NEXT P
110 END
```

9e leçon

Exercice 1

```
10 CLS
20 INPUT"QUEL EST VOTRE NOM";N$
30 I=LEN(N$)
40 FOR L=I TO 1 STEP -1
50 A$=LEFT$(N$,L)
60 PRINT A$
70 NEXT L
80 END
```

Exercice 2

```
1 CLS
5 A$="HELENE"
6 LONG=LEN(A$)
10 FOR K=1 TO LONG
20 PRINT A$
30 A$=RIGHT$(A$,LONG-1)+LEFT$(A$,1)
40 NEXT K
50 END
60 END
```

Exercice 3

```
10 CLS
20 INPUT "ENTREZ LE MOT DESIRE";M$
30 I=LEN(M$):CLS
40 FOR L=1 TO I
50 A$=MID$(M$,L,1)
60 PRINT A$
70 NEXT L
80 END
```

```
10 CLS
20 INPUT"QUEL EST VOTRE PRENOM";P$
30 L=LEN (P$):CLS
40 FOR I=1 TO L
50 A$=MID$(P$,I,1)
60 LOCATE I,I:PRINT A$
70 NEXT
80 END
```

10e leçon

Exercice 1

```
10 A$="T804L24FA#FA#SOLALASOFA#MIREREMIF
A#":B$="L36FA#L12MIL48MI"
20 C$="L36MIL12REL48":D$="REL24MIMIFA#RE
MIL12FA#SOL24FA#REMIL12FA#SOL24FA#MI"
30 E$="REMI03LA04FA#FA#FA#SOLALASOFA#L12
SOMIL24REREMIFA#L36MIL12REL48RE"
100 PLAY A$+B$:PLAY A$+C$:PLAY D$:PLAY E
$:PLAY D$:PLAY E$
```

Exercices 2 et 3

Sirène des pompiers, bruit de moteurs et tirs de fusée.

Exercice 4

```
10 CLS:SCREEN 7,0,1
20 DATA DO,1,RE,2,MI,3,FA,4,SO,5,LA,6,SI
,7
30 A=1+INT(7*RND)
40 FOR I=1 TO A
50 READ N$,C
60 NEXT I
70 PLAY N$
80 COLOR 0,C:PRINT N$;
90 RESTORE:GOTO 30
```

11e leçon

Exercice 1

```
10 CLS:SCREEN0,10,10:T=0:TP=0
20 DATA 4.9,1,5.5,2,3.4,3,2.7,4,1.5,5,1.
3,6
30 FOR I=1 TO 6
40 READ M(I),P:T=T+M(I):TP=TP+(M(I)*P)
50 NEXT I
60 PRINT"TOTAL DES PERSONNES";TP;"MILLIO
NS"
70 PRINT"TOTAL DES MENAGES";T;"MILLIONS"
80 FOR I=1 TO 6
90 PRINT"POURCENTAGE DES MENAGES DE";I;"
PERSONN";
100 IF I=1 THEN PRINT"E :", ELSE PRINT"E
S.:";
110 PC=100*M(I)/T:PRINT PC;"%"
120 NEXT I
130 END
```

Exercice 2

```
10 CLS:SCREEN 0,10,10:S=0
20 INPUT N:DIM A(N)
30 FOR I=1 TO N
40 PRINT"ENTREZ LA NOTE No";I
50 INPUT A(I):S=S+A(I)
60 CLS
70 NEXT I
80 PRINT"MOYENNE DES";N;"NOTES :";S/N
90 END
```

Exercice 3

```
10 CLS:SCREEN 0,10,10
20 S=0
30 FOR I=1 TO 10
40 PRINT"PRIX DU PRODUIT No";I:INPUT P(I
)
50 INPUT"QUANTITE ACHETEE";Q(I)
60 S=S+P(I)*Q(I):CLS
70 NEXT I
80 PRINT"PRIX TOTAL A PAYER :";S;"FRANCS
"
90 END
```

Exercice 4

Le programme "jette" 100 dés successifs et compte le nombre d'apparition de chaque numéro.

13e leçon

Exercice 2

```
10 SCREEN 0,10,10
20 PRINT"POINTEZ LE CRAYON OPTIQUE OU VO
US"
30 PRINT"DESIREZ VOIR S'ECRIRE LE TEXTE"
40 INPUTPEN C,L
50 CLS:IF C<0 OR C>319 OR L<0 OR L>199 T
HEN PRINT "RECOMMEMCEZ LE POINTAGE"
60 CLS:A$="BONJOUR"
70 LOCATE C/8,L/8:PRINT A$
80 END
```

Exercice 3

```
10 CLS
20 SCREEN 0,10,10
30 PRINT"POINTEZ LE CENTRE DU CARRE":INP
UTPEN C,LC:IF PTRIG THEN 30
40 IF C<1 OR C>318 OR LC<1 ORLC>198 THEN
 PRINT"IMPOSSIBLE, RECOMMENCEZ":GOTO 30
50 PRINT"POINTEZ UN SOMMET DU CARRE":INP
UTPEN S,LS
60 IF S<1 OR S>318 OR LS<1 OR LS>198 THE
N PRINT "IMPOSSIBLE, POINTEZ UN AUTRE SO
MMET":GOTO 50
70 CLS:IF S<C THEN S1=C+(C-S) ELSE S1=C-
(S-C)
80 IF S1<1 OR S1>318 THEN PRINT "TRACE E
N DEHORS DE L'ECRAN, RECOMMENCEZ":GOTO 3
0
90 IF S<S1 THEN D=S1-S ELSE D=S-S1
100 IF LS<LC THEN L1=LS+D ELSE L1=LS-D
110 IF L1<1 OR L1>198 THEN PRINT"TRACE E
N DEHORS DE L'ECRAN, RECOMMENCEZ":GOTO 3
0
120 BOX(S,LS)-(S1,L1)
130 INPUT"VOULEZ-VOUS RECOMMENCER";R$
140 IF R$="OUI" THEN 10 ELSE END
```

Exercice 4

```
10 CLS:SCREEN 10,10,0
20 X=1+INT(39*RND)
30 CLS:INPUTPEN C,L:L1=INT(L/8)
40 IF L1<X THEN PRINT"PLUS BAS S.V.P.":G
OTO 30
50 IF L1>X THEN PRINT"PLUS HAUT S.V.P.":
GOTO 30
60 PRINT "BRAVO !!"
70 END
```

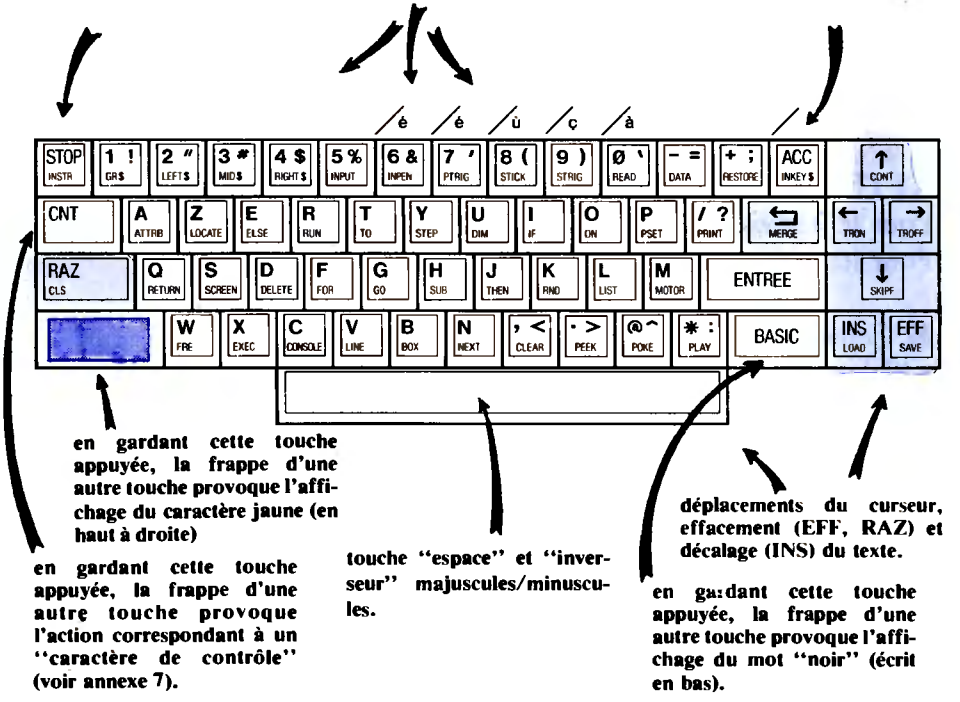
Fiches de référence

Page blanche

Affichage

Affichage

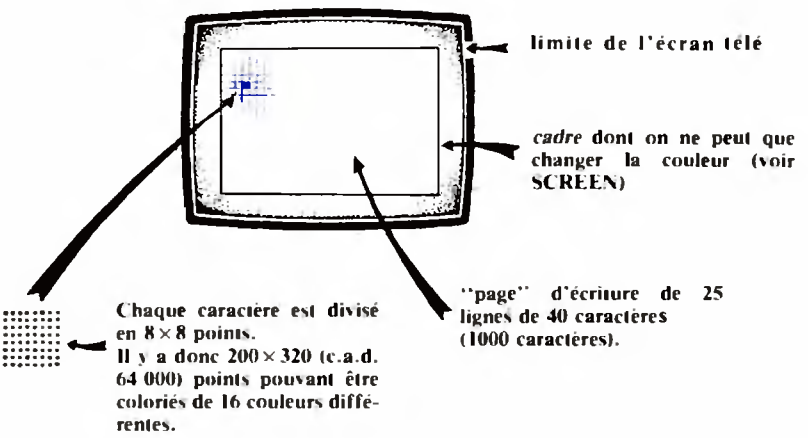
Le clavier ★



La touche ENTREE équivaut au "point" terminant toute instruction, commande ou liste de données énoncées par l'utilisateur. C'est un "point d'exécution".

La leçon 1 détaille les précisions et conseils élémentaires relatifs au clavier et à l'écran.

L'écran ★



Voir en annexe, *les grilles représentant l'écran*. Pour le choix des *"attributs" d'affichage* (forme et couleur), voir COLOR, SCREEN, ATTRB.
Pour le déplacement du *curseur*, voir LOCATE,...
Pour les *instructions graphiques*, voir PSET,...

■ Consulter aussi les fiches :
• "Ligne d'écriture" pour la manipulation des 40 caractères d'une ligne.
• "CONSOLE" pour la définition de la "fenêtre" d'écriture.
• "Fond et écriture" pour savoir comment le système MO5 colorie chaque point.

COLOR SCREEN ★

COLOR X	X devient la couleur d'écriture de tout ce qui sera écrit et dessiné.
COLOR X, Y	Y devient la couleur de fond d'écriture de tout caractère qui sera écrit.
COLOR, Y	Y devient la couleur de fond d'écriture, et la couleur d'écriture ne change pas.
SCREEN X	Tous les caractères déjà écrits et ceux qui seront écrits prennent la couleur X.
SCREEN X, Y	La couleur du fond d'écran devient Y.
SCREEN X, Y, Z	La couleur du pourtour devient Z.
SCREEN,Y, Z	
SCREEN , , Z	

Lorsqu'un paramètre est omis, il garde sa valeur présente ; son omission est repérable grâce à la présence de la virgule séparatrice.

Voir le code des couleurs en Annexe 6.

Attention :
SCREEN modifie la couleur de tout ce qui figure déjà sur l'écran (caractères, fond d'écran et pourtour).
COLOR ne modifie rien de déjà écrit, mais choisit les couleurs d'écriture et de fond qui seront utilisées au prochain affichage.
Exemple : Encre "sympathique" et "démasquage".
Il peut paraître inintéressant d'écrire, par exemple, en vert sur un fond vert. Ainsi le programme :

```
10 COLOR 2,2
20 PRINT "COUCOU"
30 END
```

provoque l'affichage d'une bande verte sur laquelle on ne lit aucun texte.
La surprise est donc totale lorsqu'on commande alors...

```
SCREEN 1,6,6
```

et on voit apparaître le mot COUCOU en rouge sur fond ciel. L'instruction SCREEN peut donc faire tout d'un coup apparaître un texte ou un dessin ayant été initialement "écrit" sur tout l'écran de la couleur du fond.

Code des couleurs

Ø	Noir	8	Gris
1	Rouge	9	Rouge pâle (vieux rose)
2	Vert	10	Vert pâle
3	Jaune	11	Jaune pâle
4	Bleu	12	Bleu pâle
5	Violet (Magenta)	13	Violet pâle
6	Bleu clair (Cyan)	14	Bleu très pâle
7	Blanc	15	Orange

Paramètre de permutation

• COLOR accepte un 3e paramètre et SCREEN un 4e (égal à Ø ou 1) ; Sa présence fait permuter les couleurs de fond et d'écriture.
Exemple d'utilisation :

```
1 LOCATE 10,10:"*":COLOR,,1:GOTO 1
```

Paramètre d'incrustation

— SCREEN accepte un 5e paramètre permettant l'incrustation de l'image TV dans l'image MO5 !

Attention : cette possibilité ne peut être utilisée qu'en présence de l'*interface d'incrustation,* qui est un périphérique se branchant sur le connecteur arrière. Utilisée sans l'interface, cette commande déplait beaucoup au MO5.

SCREEN ,,,,1 affichage de l'image TV sur tous les points de couleur noire de l'image MO5. Tout se passe comme si le noir devenait transparent.

La possibilité d'incrustation vous permet un "bricolage" de l'image-télévision assez fantastique. Vous pouvez par exemple préparer à

l'avance un programme qui vous permette de placer des moustaches à tout endroit de l'écran (par exemple sur le visage de votre commentatrice favorite).
Mieux, vous pouvez programmer ou exécuter votre programme, tout en suivant votre émission préférée, à la télévision. Pour plus de confort, vous pouvez par exemple regarder la télévision sur les 20 premières lignes et utiliser les 5 dernières pour travailler, en commandant :

```
SCREEN 4,0,0,,1
CONSOLE 20,24
COLOR 4,6
```

image télé

ATTRB ★

ATTRB Ø,1 les caractères seront écrits double hauteur et simple largeur.
ATTRB 1,Ø ..... double largeur, simple hauteur.
ATTRB 1,1 ..... double grandeur.
ATTRB , 1 ..... double hauteur, la largeur ne change pas.
ATTRB Ø,Ø ..... simple grandeur.

(La position du curseur est en bas à gauche du caractère).

• Dans la séquence :

```
10 ATTRB 1,1
11 PRINT
```

l'instruction 11 provoque le saut de 2 lignes d'écriture.

• **Attention** : Si le curseur est en haut de l'écran, une commande d'écriture avec ATTRB 1,1 provoquera un effet inattendu.

• Les instructions CNT-C, END, INPUT et LINE INPUT provoquent l'exécution implicite de ATTRB Ø,Ø. Si l'on veut entrer des données tout en les affichant en plus gros caractères, il faut donc utiliser des instructions comme INKEY$ ou INPUT$ (voir les fiches correspondantes).

Note : ATTRB est le squelette du mot "Attribut" au sens (anglo-saxon) de "qualité" ; la couleur, la grandeur,... sont des *attributs* du caractère

Ligne d'écriture EFF INS CNT-X CNT-W ★

Le curseur sur tout l'écran

Le MO5 dispose d'un "éditeur pleine page".
Cela signifie que le curseur d'écriture peut être positionné n'importe où sur la page afin d'y écrire ou d'y modifier une écriture préalable.
La ligne d'écriture du MO5 comporte 40 caractères.

EFF Efface le caractère situé au dessus du curseur et déplace d'une position vers la gauche toute la fin de la ligne.

INS Déplace d'une position vers la droite toute la fin de la ligne et marque un espace au dessus du curseur.

Lorsque le curseur est positionné sur le caractère d'une *instruction de programme*, la frappe des touches INS / EFF agit sur la totalité de cette instruction de programme (et donc éventuellement sur plusieurs lignes d'écriture).

CNT—X Efface toute la fin de la ligne à partir de la position du curseur.

CNT—W Provoque le lien, en une seule instruction de programme, de la ligne d'écriture du curseur avec la ligne suivante.

Voir la fiche CNT

Espace

Les espaces non significatifs peuvent être frappés ou non.
Cependant les frappes suivantes provoquent une erreur de syntaxe :

`3 297 - 19`	car la machine interprète des constantes séparées par un espace, ce qui n'a pas de signification.
`FOR I=ATO B`	car la machine interprète ATO comme une variable et ne trouve pas le "TO" attendu.
`IF I=1 TONTHEN`	même raison : NTHEN est interprétée comme une variable (par contre TON est bien compris comme TO suivi de N)

L'interpréteur BASIC crée un espace entre le numéro de ligne et le premier caractère (lettre) de l'instruction.

Attention :

Entre des guillemets les espaces sont évidemment significatifs.

★★

Minuscules et accents ACC

Majuscules/Minuscules

Après la mise sous tension, les lettres sont affichées en majuscules.
Après la frappe simultanée de [SHIFT]+[ACC], les lettres sont affichées en minuscules.
Pour afficher A il suffit alors de frapper sur [A]
Pour afficher a il faut alors frapper sur [SHIFT]+[A]
Pour revenir à l'affichage en majuscule frapper [SHIFT]+[ACC]

Attention :
Pour savoir si, à un moment donné, vous êtes en mode "majuscules" ou en mode "minuscules", il vous faut écrire une lettre !

Mots clés et variables

Ils peuvent être écrits en majuscules ou en minuscules ; mais l'interpréteur BASIC les transforme automatiquement en majuscules.

Chaîne de caractères

Entre guillemets, les caractères minuscules et majuscules sont différents. Ainsi l'instruction

```
LOAD"TATA"
```

ne chargera pas le programme enregistré sous le nom "tata". D'autre part "NOM" est différent de "nom" (Attention aux tests portant sur les chaînes de caractères !).

Accents et cédilles

Pour afficher des lettres accentuées les plus courantes, il suffit de frapper successivement sur :
[ACC] puis [6], et l'on affiche é
[ACC] puis [7], et l'on affiche è
[ACC] puis [8], et l'on affiche ù
[ACC] puis [9], et l'on affiche ç
[ACC] puis [0], et l'on affiche à

Ces cinq lettres sont d'ailleurs marquées sur le boîtier de votre MO5 afin de soulager votre mémoire.

Mais il est aussi possible de frapper un "tréma" ou un accent quelconque sur une voyelle quelconque. Pour cela :
— Il vous faut d'abord être en mode "minuscules".
— Frappez alors successivement sur :

- ACC
- une touche pour l'accent (le curseur n'avance pas)
  - 0 pour l'accent grave,
  - 7 pour l'accent aigu,
  - @ pour l'accent circonflexe,
  - " pour le tréma,
  - c pour la cédille.
- la lettre

Attention : que les accents ou la cédille soient frappés en 2 ou 3 touches, une lettre accentuée compte pour 3 caractères.
Essayez, par exemple la séquence suivante :

```
A$="à"
?LEN(A$)
 3
OK
```

Le caractère "ESCAPE"
Un caractère, de code ASCII 27, donne accès à une grille d'*attributs* permettant (par exemple) de glisser des changements de couleur à l'intérieur d'une chaîne de caractères, selon le code suivant :

ESC @	caractères noirs	ESC P	fonds noirs
ESC A	caractères rouges	ESC Q	fonds rouges
ESC B	caractères verts	ESC R	fonds verts
ESC C	caractères jaunes	ESC S	fonds jaunes
ESC D	caractères bleus	ESC T	fonds bleus
ESC E	caractères magenta	ESC U	fonds magenta
ESC F	caractères cyan	ESC V	fonds cyan
ESC G	caractères blancs	ESC W	fonds blancs

Exemple :

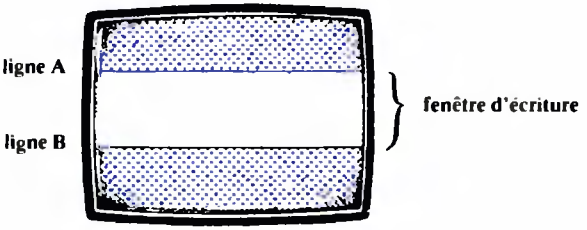
```
C$=CHR$(27)+"A"+"ROUGE"+CHR$(27)+"B"
+"VERT":?C$
```

CLS CONSOLE

★★

CLS — Efface le contenu de la fenêtre d'écriture et positionne le curseur en haut à gauche.

CONSOLE A, B — Définit la fenêtre d'écriture pour les instructions PRINT ; A et B sont les lignes supérieures et inférieures d'écriture.
(A ≤ B, A et B étant compris entre Ø et 24).



CONSOLE A, B, 1 — Les couleurs ne changent plus dans la fenêtre d'écriture jusqu'à une autre instruction CONSOLE (même l'effet d'une instruction COLOR ne sera pas visible).

CONSOLE A, B, Ø — Utilisation des couleurs courantes.

CONSOLE A, B,,Z — Z = Ø : La fenêtre est en mode "rouleau" c.a.d. que lorsque l'on croit écrire sur la ligne B+1, l'ensemble des lignes [A, B] remonte d'une ligne ; et la ligne A disparaît.
Z=1 : la fenêtre est en mode "rouleau" avec une vitesse de défilement faible.
Z=2 : la fenêtre est en mode "page" c.a.d. que lorsqu'elle est pleine, la ligne suivante apparaît sur la première ligne.

Le retour au mode-commande (par exemple par CNT-C ou END), fait revenir au mode rouleau, mais toujours dans la fenêtre (A, B). Pour éviter des désagréments, il vaut mieux programmer l'instruction :

```
CONSOLE 0,24,0,0
```

(qui fait revenir à la fenêtre de travail standard) avant l'instruction END d'un programme.

Actions dans la fenêtre

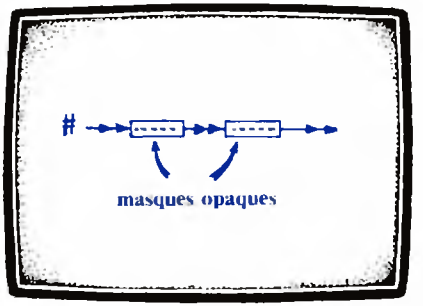
Les instructions graphiques (PSET, LINE, BOX) ne sont pas affectées par l'instruction CONSOLE ; on peut donc positionner le curseur par LOCATE à tout endroit hors de la fenêtre, mais le passage du curseur à la ligne (après un PRINT) le fera alors revenir dans la fenêtre.

Les instructions CLS et SCREEN (et la touche RAZ) ont leur effet limité à la fenêtre. Cependant le changement de couleur affectera toute écriture même en dehors de la fenêtre.

L'effet caméléon

Le troisième paramètre de CONSOLE permet de produire des effets intéressants.

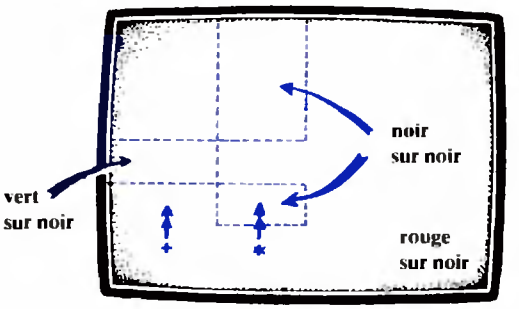
Voici, par exemple, un programme simulant le passage d'un # "dessus" ou "dessous" un "masque"




```
5 CLS:SCREEN1,0,0:ATTRB1,1
6 LOCATE 10,10
7 COLOR4,4:PRINT"  ";
8 COLOR1,4:PRINT"  ";
9 COLOR4,4:PRINT"  ";
15 CONSOLE0,24,1
20 FOR I=0 TO 30
21 LOCATE I,10
22 PRINT" #";
23 FORK=1TO100:NEXT
25 NEXT I
30 CONSOLE0,24,0,0
40 END
```

Voici un autre programme ou le mode "rouleau" permet de cacher ou de faire apparaître un dessin qui monte.

```
1 CLS:SCREEN1,0,0:ATTRB1,1
5 BOXF(9,0)-(16,9)" ",0
6 BOXF(9,15)-(16,20)" ",0
7 BOXF(0,10)-(8,14)" ",3
10 LOCATE 0,24:CONSOLE 0,24,1
20 FOR K=1 TO 10
30 PRINT"  +   *"
35 FOR I=1 TO 100:NEXT I
40 NEXT K
50 CONSOLE0,24,0
60 FOR I=1 TO 1000:NEXT I
70 SCREEN4,6,6
```



LOCATE CSRLIN POS SCREEN ★★

Positionnement du curseur

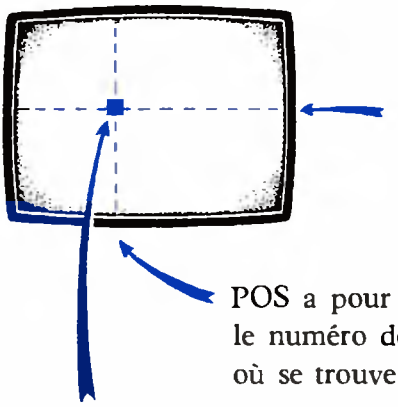
LOCATE X, Y — positionne le curseur sur la position de la ligne numéro Y.

LOCATE X, Y, Ø — le curseur se positionne en (X, Y) et devient invisible.

LOCATE X, Y, 1 — le curseur se positionne en (X, Y) et devient clignotant.

Exemple : Pour ne pas détruire l'effet d'un joli programme on peut le terminer par l'instruction LOCATE Ø,Ø,Ø.

Fonctions dont la valeur dépend du curseur



CSRLIN a pour valeur le numéro de la ligne où se trouve le curseur.

POS a pour valeur le numéro de la colonne où se trouve le curseur.

SCREEN (C, L) a pour valeur le code ASCII du caractère affiché sur la colonne C de la ligne L.

(Voir annexe 7)

Plus généralement POS (#2) donne la position du paramètre d'écriture sur le périphérique du canal numéro 2.

Les caractères de code ASCII 32 à 148 sont seuls reconnus, étant entendu que (pour cette instruction) les codes 128 à 148 sont ceux de :
ç à á â ä è é ê ë ì í î ï ò ó ô ö ù ú û ü
La fonction prend la valeur Ø pour un caractère non reconnu.

Ligne de programme

★★

Lorsque le premier caractère d'une ligne d'écriture est un chiffre la suite des caractères frappés jusqu'à la frappe de ENTREE *est considérée comme une "ligne de programme".*
Une ligne-de-programme peut donc s'étendre sur plusieurs lignes d'écriture (le passage à la ligne d'écriture suivante étant automatique). Cependant :
La longueur d'une ligne-de-programme ne peut pas dépasser 255 caractères (soit 6 lignes et 15 caractères).

■ Une ligne-de-programme peut contenir plusieurs instructions séparées par le caractère séparateur "deux points".

Exemple :

```
40FOR I=1 TO N:? I,I^2:NEXT I
```

■ Tout ce qui suit une apostrophe dans une ligne-de-programme n'est pas interprété.
Cela permet d'introduire des commentaires souvent utiles, ou des espaces entre les lignes listées.

```
60 GOSUB400  ' DESSINS DES BRAS
80 X=1  ' initialisation
85 '
```

Entrée
Lorsque le curseur est positionné sous un caractère d'une ligne-de-programme affichée, la frappe de ENTREE provoque la mise en mémoire de cette ligne dans sa totalité (avec comme numéro celui figurant au début de la ligne-de-programme).

Lignes semblables : le curseur "pleine-page"
Lorsque deux lignes de programme se ressemblent et que la première a été frappée et enregistrée en mémoire, il est possible de frapper la deuxième en utilisant l'affichage de la première.
Si par exemple la ligne 40 ci-dessus est enregistrée et affichée, alors

pour enregistrer la ligne :

```
90 FOR I=1 TO M:PRINTI,I^3:NEXT I
```

il suffit d'amener le curseur sur la ligne 40 :

```
40 FOR I=1 TO M:PRINTI,I^3:NEXT I
```

puis de modifier les caractères voulus avant de frapper sur ENTREE

Attention :
la ligne 40 existe toujours et la ligne 90 est maintenant enregistrée.

CNT ★★★

Arrêt

La touche CNT, abrégé de "Contrôle", a un fonctionnement analogue à celui de la touche : En la gardant appuyée et en frappant une autre touche, on déclenche une certaine action.
Ainsi :

CNT-C provoque l'arrêt de toute action en cours (exécution d'un programme, listage sur écran ou imprimante).
La machine affiche alors, par exemple :

```
Break in 160
OK
```

Effacement de fin de ligne

CNT-X provoque l'effacement de tous les caractères situés après le curseur sur la ligne d'écriture (depuis la position suivant le curseur jusqu'à la position 39).

Exemple :

```
20 IF Z=0 THEN PRINT "BIEN" ELSE
   PRINT"CELA EST MAUVAIS"
```

Supposons que le curseur soit sur le premier E de ELSE et que l'on appuie sur CNT-X; alors d'une part la fin de la ligne d'écriture est effacée, d'autre part le "lien" avec la ligne d'écriture suivante est supprimé. L'instruction devient donc :

```
20 IF Z=0 THEN PRINT "BIEN"
```

Moins 64

La frappe de CNT-■ équivaut à la frappe du caractère ■ dont le code ASCII a été diminué de 64 (le 7e bit !).
Voir en annexe les codes et leur correspondance.
Ainsi CNT-V a pour code ASCII 22 et équivaut à la frappe de ACC
CNT-H a pour code ASCII 8 et équivaut à la frappe de ←

Création de lien

CNT-W provoque le lien (en une seule ligne de programme) de la ligne d'écriture où se trouve le curseur avec la ligne d'écriture suivante.

Exemple :

```
10 X=5
20 Y=10
```

Supposons que le curseur soit après le 5 et que l'on appuie sur CNT W ; alors, en appuyant trente trois fois sur EFF, on voit la ligne d'écriture 20 venir se coller derrière X = 5.
En frappant alors : puis EFF pour effacer l'étiquette "20", on obtient l'instruction :

```
10 X=5:Y=10
```

Le code ASCII

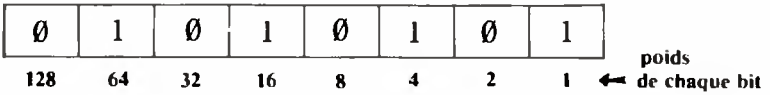
★★★

Les cinq lettres ASCII sont les initiales de "American Standard Code for Information Interchange".

Les mots du code ASCII sont des "octets" transportant une information que l'on peut interpréter comme un nombre décimal compris entre Ø et $2^8 - 1$, c'est à dire entre Ø et 255.

Un octet est constitué de 8 "bits" ayant chacun un "poids" de 1 à 128 égal à l'une des huit premières puissances de 2 .

Exemple :



L'octet 01010101 correspond au nombre décimal
64 + 16 + 4 + 1, c'est à dire 85.
85 est le "code ASCII" de la lettre U.

Cet octet représente donc le caractère U.

(Note : ici le bit de poids 128 n'est pas significatif : 7 chiffres binaires seulement sont utilisés).

Les caractères de codes ASCII compris entre Ø et 127 sont donnés en annexe 7.

VOIR la fiche CHR$ et ASC.

Dans un programme, la commande de l'impression du caractère de code ASCII égal à 85 (par exemple),est équivalente à la commande d'impression de U.

```
PRINT CHR$(85)
```
équivaut à
```
?"U"
```

Ce qui ne semble pas intéressant pour une lettre peut le devenir pour un "caractère de contrôle" qui provoque une action. Par exemple :

Application : L'instruction ? CHR$ (11) provoque la montée du curseur d'un caractère. (Sans effacement ni autre action sur les caractères déjà écrits).

Cela peut permettre entre autres choses, l'affichage de dessins composés de plusieurs caractères définis par l'utilisateur (voir fiche GR$). Ainsi pour afficher un animal dont le haut est défini par GR$(1) et le bas par GR$(Ø) on utilisera les instructions :

```
10 HOMME$=GR$(0)+CHR$(11)+CHR$(8)
   +GR$(1)
80 PRINT HOMME$
```

Certains caractères spéciaux, ne correspondant à aucune touche, peuvent cependant être affichés sur l'écran.
Ainsi :
CHR$(123) est une accolade ouvrante : {
CHR$(127) est le caractère "plein" : ■
CHR$(92) est le signe : \
Voir aussi la fiche "minuscules-accents" et l'annexe 7.

Bip
?CHR$ (7) provoque l'émission d'un "bip" analogue à celui que l'on entend quand on appuie sur une touche.
Cette dernière instruction équivaut à l'instruction BEEP.

Passage à la ligne
CHR$(13) fait revenir le curseur en début de ligne.
CHR$ (1Ø) fait descendre le curseur d'une ligne.
Dans une chaîne de caractère à afficher, CHR$(13)+CHR$ (1Ø) équivaut donc à la frappe de ENTREE
Par exemple la commande

```
H$="UN"+CHR$(13)+CHR$(10)+"DEUX":?H$
```

provoque l'affichage de :

```
UN
DEUX
OK
```

Initialisation

★★★

A l'initialisation, c'est-à-dire à la mise sous tension de la machine ou après appui sur le bouton en haut du clavier à gauche, tout se passe comme si l'on avait commandé :

- SCREEN 4, 6, 6

(et donc COLOR 4, 6)

- CONSOLE Ø, 24, Ø, Ø
- la mise à la valeur Ø pour toutes les variables numériques éventuelles et la valeur " " (vide) pour toutes les chaînes éventuelles.
- DIM (1Ø) pour tous les tableaux éventuels.
- Le repositionnement de tous les "pointeurs" (en particulier pour le choix d'un nombre au hasard, le rangement des variables et des lignes de programmes, la lecture des DATA,...)
- la fermeture de tous les fichiers
- PLAY "AØL2404T5"
- ATTRB Ø,Ø
- PRINT "OK"

Attention :
Lorsque, dans une instruction, certains paramètres sont omis, on ne revient pas aux valeurs standard initiales, mais on garde les valeurs en cours ("courantes").
Note : La touche initialisation s'appelle "RESET" en anglais.

RUN STOP END CNT-C CONT GOTO ★

Exécution

RUN exécute le programme à partir de la première instruction de numéro supérieur ou égal à 1.

RUN 1Ø exécute le programme à partir de la première instruction de numéro supérieur ou égal à 1Ø.

L'instruction RUN efface les valeurs des variables.

Si on ne veut pas les effacer on peut commander GOTO 1Ø.

GOTO 1Ø exécute les instructions à partir de l'instruction numéro 1Ø

(Provoque une erreur si l'instruction 1Ø n'existe pas)

Arrêts et reprise d'exécution

La touche STOP arrête l'exécution. Celle-ci est reprise par la frappe d'une touche quelconque.

L'instruction STOP, rencontrée dans un programme, arrête l'exécution.

END arrête l'exécution définivement et ferme les fichiers.

CNT-C arrête l'exécution sans fermer les fichiers.

Après l'intervention de CNT-C ou l'exécution des instructions STOP ou END, on peut demander l'écriture des valeurs de certaines variables, puis reprendre l'exécution.

Par exemple, si un programme manipulant les variables A, B(J), I,... est arrêté par une instruction STOP, la machine affiche :

```
Break in 150
```

On peut alors frapper :

```
PRINT A,B(I),I
```

ENTREE et la machine affichera par exemple

```
14   7.23   2
```

Si on frappe alors :

CONT ENTREE .... l'exécution reprend.

CONT reprend l'exécution à l'endroit où elle a été interrompue, après STOP ou CNT-C (au début d'un INPUT si les données n'avaient pas été entrées).

LIST DELETE NEW ★

Effacer des instructions en mémoire

DELETE — efface l'ensemble des instructions en mémoire.

DELETE 20 efface l'instruction 20 en mémoire.
DELETE — 20 efface les instructions 20 et les précédentes.
DELETE 20 — efface les instructions 20 et les suivantes.
DELETE 20 — 30 efface les instructions 20 à 30.

Attention :
Delete efface les instructions en mémoire et n'a aucun effet immédiat sur l'affichage à l'écran.
— Inversement la disparition de l'écriture d'une instruction sur l'écran ne signifie pas que cette instruction est effacée de la mémoire.

NEW efface le programme et détruit les variables.
Cette instruction ne ferme pas les fichiers et ne modifie pas les options d'une éventuelle instruction CLEAR ou CONSOLE.
Il est recommandé d'exécuter l'instruction NEW ENTREE lorsque vous même ou un autre utilisateur avez travaillé préalablement sur la machine.

Lister les instructions en mémoire

LIST Liste sur l'écran l'ensemble des instructions en mémoire.
(Même syntaxe pour LIST que pour DELETE)
LIST "CASS : TOTO" enregistre sur la cassette sous le nom de TOTO, et sous forme ASCII, le programme en mémoire.
Voir aussi "LPRT :" dans les fiches "périphériques".

TRON TROFF ★★

TRON passage en mode "trace"
TROFF retour au mode normal d'exécution.
Dans le mode "trace", la machine écrit entre crochets le numéro de toute instruction exécutée.
Cette écriture est effectuée sur la position du curseur avec les attributs de couleur et de forme au moment de l'exécution de l'instruction.
(Cela peut donner de curieux effets).

Exemple :
Voici la copie de l'écran d'un utilisateur ayant utilisé ces instructions.

```
LIST

10 INPUT A
20 I=0
30 PRINT I;
40 IF I<A THEN I=I+1:GOTO 30
50 PRINTA
60 TROFF
70 PRINTA*I
80 END
OK
TRON
OK
RUN
[10]? 4
[20][30] 0 [40][30] 1 [40][30] 2 [40
]   [30] 3 [40][30] 4 [40][50] 4
[60] 16
OK
RUN
? 4
 0  1  2  3  4  4
 16
OK
```

ON ERROR GOTO ERR ERL RESUME ★★

ERR numéro de la dernière erreur détectée.
ERL numéro de la ligne où a été détectée la dernière erreur.

Il n'est possible d'utiliser ces valeurs que dans une expression algébrique mais on ne peut jamais placer ERR ou ERL avant le signe = d'une instruction d'affectation.
Elles peuvent être utilisées, dans un programme, à la suite d'une instruction ON ERROR GOTO...

ON ERROR GOTO 30 branche l'exécution à l'instruction 30 dès qu'une erreur est détectée dans l'exécution d'un programme.
ERR et ERL prennent alors des valeurs non nulles.
Aucun message d'erreur n'est alors affiché et l'exécution n'est pas interrompue.

Exemple :

```
10 ON ERROR GOTO 50
20 INPUT A
30 PRINT 1/A
40 GOTO20
50 IF ERR=11 THEN PRINT"DIVISEUR NUL!"
51 GOTO 20
RUN
? 2
 .5
? 0
DIVISEUR NUL!
? 4
 .25
? 0
Error 11 in 30
OK
?ERR,ERL
 11            30
```

ON ERROR GOTO Ø annule l'effet de ON ERROR GOTO.
Cette instruction permet, par exemple, de limiter la possibilité d'une erreur non "punie" à l'entrée des données ; si une autre erreur apparait dans la suite du programme, elle sera détectée et signalée.

RESUME	branche l'exécution au début de l'instruction ayant provoqué l'erreur.
RESUME NEXT	branche l'exécution à l'instruction suivant celle qui a provoqué l'erreur.
RESUME 2Ø	branche l'exécution en 2Ø.

Normalement, tout branchement par ON ERROR GOTO... devrait se terminer par une instruction RESUME (de même que tout GOSUB... devrait se terminer par un RETURN). En effet :
La détection d'une deuxième erreur après une première, sans exécution de RESUME provoque une erreur
Le programme donné en exemple devrait se transformer, ainsi :

```
51 RESUME 20
```

ERROR X Simule une erreur de numéro X.
Si une instruction ON ERROR GOTO 2Ø a été exécutée, alors l'exécution est branchée à l'instruction 2Ø avec ERR égal à X.
Sinon, un message d'erreur numéro X est affiché.
Voir les numéros de code des erreurs en Annexe 8.

EXEC VARPTR

★★★

EXEC 32624 — Branche l'éxécution vers un sous-programme en langage machine préalablement chargé en mémoire centrale à partir de la mémoire numéro 32624.

EXEC ADRESSE — Le numéro de mémoire peut être une variable

Évidemment, pour utiliser cette instruction, il faut savoir programmer en langage assembleur. Si vous êtes dans ce cas, sachez que votre sous programme doit préserver le contenu des registres S et DP du microprocesseur et se terminer par l'instruction RTS qui renvoie au programme BASIC.

VARPTR (A) — La valeur de VARPTR (A) est l'adresse en mémoire du premier octet de la variable A.

L'adresse est donnée sous la forme d'un entier décimal compris entre −32768 et 32767 (si l'adresse est négative, ajouter 65536 pour avoir l'adresse effective).

Exemple : Essayez, juste après l'initialisation, ces quelques instructions :

```
A%=3
?VARPTR(A%)
 9645
FOR I=9640 TO 9650:?PEEK(I):NEXT
0
0
0
33   <---------------------------------- entier
65   <----------------------------------------- A
0  <------------------------------ VARPRT(A%)
3
65
73
142
22
```

Vous pouvez ainsi vérifier qu'un entier est codé sur deux octets (Ø et 3), l'octet encore précédent indique qu'il s'agit d'un entier(32)et que son nom comporte 1 lettre (33=32+1).
En remplaçant A% par A vous verrez qu'un réel est codé sur quatre octets, à partir de VARPTR(A).
Et en remplaçant A par A$ vous constaterez que la mise en mémoire d'une chaîne fait l'objet d'un adressage indirect : VARPTR pointe sur la longueur de la chaîne, les deux octets suivants indiquant l'adresse de stockage.

Exemple : Entrée d'un programme en assembleur.
Grâce à la fonction VARPTR(−), il est possible de rentrer les codes des instructions-assembleur par des DATA lues en BASIC :

```
10 DATA 52,54,198,.....
20 DATA ..............
100 'chargement du programme assembleur
110 ASS$=""
120 FOR I=1 TO NOMBINST:READ A :
130 ASS$=ASS$+CHR$(A ):NEXT
140 ADRESSE=256*PEEK(VARPTR(ASS$)+1)+PEE
K(VARPTR(ASS$)+2)
150 ..................
1000 'appel du programme assembleur
1010 EXEC ADRESSE
```

FOR ... NEXT ★

```
FOR K = X TO Y STEP H
:
:
NEXT K
```

Les instructions situées entre FOR et NEXT sont exécutées pour K = X, puis pour K = X + H, puis pour K = X + 2H... tant que K reste inférieur ou égal à Y.

```
FOR K = X TO Y
:
:
NEXT
```

on peut sous entendre STEP 1

si la variable de boucle est omise, il s'agit de celle de la dernière boucle commencée.

```
FOR I = 1 TO N
FOR J = 1 TO M
...
NEXT J, I
```

on peut emboîter des boucles mais non les entrelacer.

équivaut à NEXT J : NEXT I
Le seul mot NEXT ne ferait ici que "fermer" la boucle J. On pourrait écrire NEXT : NEXT

A l'exécution de l'instruction FOR K=X TO Y STEP H :
- K prend la valeur X
- Les nombres K, Y et H sont mémorisés.

* Si Y − K est nul ou du signe de H :
les instructions de la boucle sont exécutées.
Sinon la machine "oublie" les valeurs de K,Y,H et exécute l'instruction qui suit l'instruction NEXT K suivante.

A l'exécution de l'instruction NEXT :
- La valeur de K est augmentée de H et l'exécution reprend en * au début de la boucle commandée par K.

Il est interdit :

• d'emboîter des boucles utilisant la même variable de boucle. Attention, lorsque, dans une boucle, on appelle des sous-programmes qui utilisent eux mêmes des boucles : les noms de variables de boucle des sous programmes doivent être surveillés.

Exemple :

```
10 FOR I=1 TO 10
20 GOSUB 100
30
40
50 NEXT I
60 ...
99 END
100 ' SOUS PROGRAMME
110 FOR I=1 TO 5
120 ...
130 NEXT I
149 RETURN
```

• d'entrelacer des boucles.

• de faire exécuter NEXT K sans que FOR K= ... ait été exécuté

• de faire exécuter un FOR I... alors qu'aucun NEXT I ou NEXT n'existe dans la suite des instructions.

• de modifier dans la boucle la valeur de la variable de boucle. (Ou alors c'est à vos risques et périls !)

Attention à l'écriture : FOR K = XTO Y. En effet XTO est pris comme nom de variable et une erreur est détectée : absence de TO.

Il est conseillé de ne pas :

• sortir d'une boucle avant d'atteindre la valeur maximale de la variable. Aucune erreur ne serait détectée mais, si cela arrive trop souvent, la mémoire est inutilement encombrée. (Peut être jusqu'au dépassement de capacité).

• modifier la valeur de Y ou de P à l'intérieur de la boucle qui commence par FOR K = X TO Y STEP P.

La valeur maximale de la variable de boucle reste pourtant la valeur initialement donnée, car elle est conservée (dans une pile) dès le début de la première boucle.

Il est bon de savoir que :

• A la sortie de la boucle, la variable de boucle a la valeur immédiatement supérieure à sa valeur maximale.

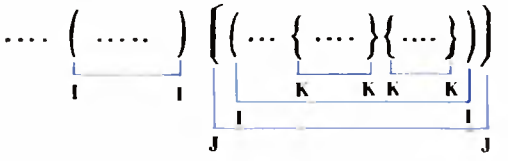
• Pour mieux visualiser l'association, FOR K... NEXT K dans le listing d'un programme, on peut décaler d'un ou plusieurs espaces les instructions situées entre elles.

• Les boucles du type : FOR K=X TO Y ... avec X>Y, ne sont pas exécutées (mais K prend la valeur X).

• Pour vérifier la bonne imbrication des boucles, on peut comparer un programme à une expression algébrique et remplacer chaque FOR K... par une parenthèse ouvrante (de style K) et chaque NEXT K par une parenthèse fermante (de même style).

Exemple :

```
10 CLS:SCREEN12,0,0
20 PRINT"     Voici  les ";
22  FOR I=3 TO 14
24  COLOR I-2
26  PRINT" ";MID$(STR$(I),2);
    "-some,";
28  NEXT I
29 PRINT CHR$(8);".":LOCATE 0,4,0
30  FOR J=1 TO 4
40   FOR I= 1 TO 3
42   X=-40+80*J:Y=-10+60*I
44   N=4*(I-1)+J:COLOR N
46   IF I>1 THEN 56
50    FOR K=0 TO 6.3 STEP .1
52    PSET(X+20*COS(K),Y+20*SIN(K))
54    NEXT K
56   PSET(X+20,Y)
60    FOR K= 0 TO 6.4 STEP 6.3/(N+2)
62    LINE-(X+20*COS(K),Y+20*SIN(K))
64    NEXT K
70   NEXT I
80  NEXT J
90 END
```



• Pour ralentir l'exécution d'un programme, on peut insérer une "boucle de temporisation". Ainsi, par exemple, vous pouvez vérifier que la boucle suivante occupe le MO5 pendant 1 seconde environ :

```
FOR K=1 TO 500:NEXT K
```

GOSUB ... RETURN ★★

35 GOSUB 200 — branche l'exécution à l'instruction 200 (le numéro de ligne indiqué doit exister) et garde l'emplacement de l'instruction d'appel dans la "pile-des-appels".

RETURN — branche l'exécution à l'instruction qui suit la dernière instruction GOSUB exécutée. (L'emplacement de cette dernière instruction est au sommet de la pile-des-appels).

(L'instruction GOSUB peut ne pas être la première instruction d'une ligne de programme ; le retour s'effectue sans problème à l'instruction suivante, éventuellement sur la même ligne).

En termes de sous-programme :

— plusieurs points d'entrée peuvent exister pour un "même" sous-programme.

— plusieurs instructions RETURN peuvent exister dans un "même" sous-programme.

Un sous-programme peut en appeler un autre. Il peut même se rappeler lui-même (c'est ce qu'on appelle la récursivité). La pile-des-appels n'est limitée que par la mémoire du MO5.

Conseil : La première instruction d'un sous-programme devrait être une REM. Et l'instruction immédiatement précédente devrait être RETURN, STOP ou END (pour éviter les entrées intempestives).

Exemple :

```
1 CLS
2 INPUT N:NN=N
3 GOSUB 10
4 PRINT"FACTORIELLE";NN;"=";F
5 END
10 'SP.FACTORIELLE
11 F=N
12 IF N=1 THEN RETURN
13 N=N-1
14 F=F*N
15 GOSUB 12
16 RETURN
```

ON ... GOTO... ON ... GOSUB... ★★

ON I GOTO 10, 20, 30, 40, 50
ON I GOSUB 10, 20, 30

branche l'exécution à la ligne dont le numéro est le I^e^ dans la liste des numéros de ligne suivant le GOTO ou le GOSUB.

I doit être compris entre 0 et 255.
Si I est nul ou supérieur au nombre de numéros de la liste, l'exécution se poursuit à la ligne suivante.

Exemple :

```
40 INPUT I
45 ON I GOSUB 101,102,103,104,105,106,10
7
50 PRINT J$
60 '
99 END
101 J$="LUNDI":RETURN
102 J$="MARDI":RETURN
103 J$="MERCREDI":RETURN
104 J$="JEUDI":RETURN
105 J$="VENDREDI":RETURN
106 J$="SAMEDI":RETURN
107 J$="DIMANCHE":RETURN
```

(L'éditeur pleine-page du MO5 permet une écriture rapide des instructions 101 à 107).

IF ... THEN ... ELSE ... ★★

```
30 IF X<0 THEN 40
31...
40...
```

Si X est négatif l'instruction 40 est exécutée ; sinon l'instruction suivante (31) est exécutée.

Attention : après l'instruction 31 il se peut que l'instruction 40 soit exécutée si rien ne vient interrompre la séquence.
L'instruction 30 peut aussi s'écrire :

```
30  IF X<0 GOTO 40
```

```
50 IF X<0 THEN X = X + 1
51...
```

Si X est négatif, X est augmenté de 1. Dans tous les cas, l'exécution reprend à l'instruction suivante (51).

```
60 IF X<0 THEN X = X + 1 ELSE 70
61...
70...
```

Si X est négatif, X est augmenté de 1 et l'instruction suivante (61) est exécutée. Sinon l'instruction 70 est exécutée.

Syntaxe générale de l'instruction :

IF *condition* THEN (*numéro de ligne* ou *suite d'instructions*) ELSE (*numéro de ligne* ou *suite d'instructions*) (facultatif)

- *un numéro de ligne* équivaut à *GOTO ce numéro de ligne.*
- *Une suite d'instructions* est constituée d'instructions BASIC séparées par :

L'exécution de cette instruction provoque :
. L'évaluation de la *condition*.
. Si la condition est *vraie* (*vraie* est ici équivalent à *non faux* c'est-à-dire *non nulle* (voir fiche "Vrai-Faux")), la liste d'instructions figurant après THEN est exécutée ; puis, s'il n'y a pas de branchements, l'instruction suivante est exécutée.
. Si la condition est *fausse*, la liste d'instructions figurant après ELSE est exécutée ; si le mot ELSE est absent de l'instruction, le branchement se fait à l'instruction suivante.

Exemple :

```
10 INPUT A,B,C: D=B*B-4*A*C
20 IF D<0 THEN PRINT"PAS DE SOLUTION!"
ELSE IF D=0 THEN X=-B/(2*A):PRINT"UNE SO
LUTION:";X ELSE X1=(-B+SQR(D))/(2*A):X2=
(-B-SQR(D))/(2*A):PRINT"DEUX SOLUTIONS:"
;X1;"ET";X2
30 GOTO10
```

Les suites d'instructions ne sont limitées que par la longueur totale de la ligne débutant par IF (c'est-à-dire 255 caractères).

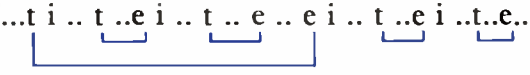
Emboîtements

Les instructions IF... peuvent s'emboîter les unes dans les autres ; selon l'explication précédente, chaque THEN est associé au ELSE qui le suit le plus proche, non déjà associé à un THEN.

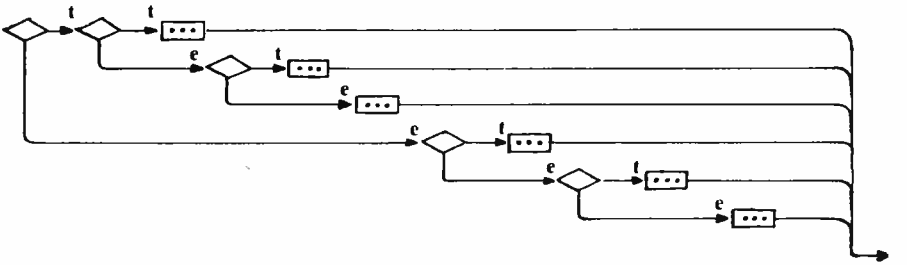
Exemple suicidaire :

```
10 INPUT A,B,C
20 IF A<B THEN IF B<C THEN PRINT A,B,C
   ELSE IF A<C THEN PRINT A,C,B ELSE
   PRINT C,A,B ELSE IF C<B THEN PRINT
   C,B,A ELSE IF C<A THEN PRINT B,C,A
   ELSE PRINT B,A,C
30 GOTO 10
```

En ne retenant que les initiales de *if, then, else,* on voit que l'instruction 20 a la structure suivante qui s'analyse comme une "expression parenthésée".



... et correspond au schéma suivant :



Égalité de réels

Il n'est pas prudent de faire des tests d'égalité sur les nombres non définis explicitement comme des entiers. Par exemple :

Si C est le résultat d'un calcul, il se peut que C vale Ø.999999 ; le test C = 1, prendra alors la valeur FAUX. Si on veut éviter ce phénomène, il vaut mieux parler de C%, ou bien commander au préalable : DEFINT C.

Vrai faux
AND OR NOT XOR IMP EQV

★★★

Conditions

• Une *condition simple* est une expression de l'un des types suivants,

X=Y X<>Y X<Y X>Y X<=Y X>=Y

<> signifie "différent de"

<= signifie "inférieur ou égal à".

• Lorsque X et Y sont des constantes, variables ou expressions numériques, ces conditions prennent la valeur *vrai* ou la valeur *faux* comme l'on s'y attend. Ainsi :

2 = 1 + 1 prend la valeur *vrai*

3<2 prend la valeur *faux*

• Lorsque X et Y sont des chaînes de caractères, les comparaisons se font selon l'ordre "lexicographique" habituel.

Plus généralement, lorsque les caractères ne sont pas des lettres majuscules, l'ordre alphabétique des caractères est donné par l'ordre (numérique) de leur code ASCII.

Ainsi :

"ABC"<"ABCD"	prend la valeur *vrai*
"a"<"B"	prend la valeur *faux*
"="<"$"	prend la valeur *vrai*
"BON JOUR"<"BON JOUR"	prend la valeur *faux*
"X0"<"X1"	prend la valeur *vrai*
"11"<"2"	prend la valeur *vrai*
"<"<">"	prend la valeur *vrai*

Attention aux "espaces" qui comptent pour un caractère. Ainsi "ABC ">"ABC" et " "> " ".

Les conditions simples se combinent en conditions plus complexes grâce aux opérateurs logiques ET, OU et NON selon les règles suivantes.

NOT

Si *cond 1* est *vrai*, alors *NOT (cond 1)* est *faux*

Si *cond 1* est *faux*, alors *NOT (cond 1)* est *vrai*

AND
Si *cond 1* est *vrai* et si *cond 2* est *vrai* , alors *cond 1 AND cond 2* est *vrai*
Dans les trois autres cas : *cond 1 AND cond 2* est *faux.*

OR
Si *cond 1* est *faux* et si *cond 2* est *faux* : alors *cond 1 OR cond 2* est *faux*
Dans les trois autres cas (c'est-à-dire si l'une des deux conditions est *vraie*) : *cond 1 OR cond 2* est *vrai.*

Remarque : On peut fabriquer des conditions très compliquées en utilisant des parenthèses. Dans les expressions ne contenant pas de parenthèses, NOT a la priorité sur AND qui a la priorité sur OR.

La vérité sur vrai et faux
En réalité, dans le MO5, les valeurs prises par les *conditions* ne sont pas autre chose que des *nombres entiers.*
Une condition est donc, en fait, un nombre compris entre −32768 et 32767 et codé par une suite de 16 bits.
1111 1111 1111 1111 représente la condition *vrai* (mais aussi le nombre −1; voir la fiche "mémoire")
0000 0000 0000 0000 représente la condition *faux.*

• *On peut donc commander sans problème :*
```
C=X<0
```
ou même :
```
A=B=C
```
A prend la valeur −1 si B égale C et la valeur 0 si B est différent de C. (Attention, les deux signes = ne sont pas interprétés de la même manière)

• Dans les tests (if...) *faux* vaut donc 0 et tout autre valeur entière non nulle est considérée comme vraie.

• D'autre part on remarque avec curiosité que :
```
IF X<0 THEN 40 ELSE 50
```
est équivalent à
```
ON 2+(X<0)GOTO 40,50
```
... mais il n'est pas conseillé du tout de se livrer à ce genre de gymnastique.

Il est encore plus extraordinaire de savoir que AND, OR et NOT sont en fait des *opérations sur les nombres entiers.*
Le MO5 effectue, sur chacun des 16 bits des entiers représentés en binaire, les opérations logiques commandées sur les Ø et les 1.
Ainsi :

• 13 AND 25 vaut 9, en effet :

```
      ØØØØØØØØØØØØ11Ø1
AND   ØØØØØØØØØØØ11ØØ1
      ------------------
égale ØØØØØØØØØØØØ1ØØ1
```

• 13 OR 25 vaut 29

• A % AND (−2) vaut le plus grand nombre pair inférieur ou égal à A %.
(Voir la notation des nombres négatifs dans la fiche "mémoire").

■ XOR, EQV, IMP

Les opérations logiques XOR (soit..., soit...), EQV (équivalence) et IMP (implication) sont aussi disponibles sur MO5.
Voici les résultats de leurs actions sur les valeurs logiques −1 (vrai) et Ø (faux) :

```
-1 IMP -1 =-1
-1 IMP  Ø = Ø
 Ø IMP -1 =-1
 Ø IMP  Ø =-1

-1 EQV -1 =-1
-1 EQV  Ø = Ø
 Ø EQV -1 = Ø
 Ø EQV  Ø =-1

-1 XOR -1 = Ø
-1 XOR  Ø =-1
 Ø XOR -1 =-1
 Ø XOR  Ø = Ø

-1  OR -1 =-1
-1  OR  Ø =-1
 Ø  OR -1 =-1
 Ø  OR  Ø = Ø

-1 AND -1 =-1
-1 AND  Ø = Ø
 Ø AND -1 = Ø
 Ø AND  Ø = Ø
```

★

+ − * / @ **MOD** ∧

Priorité

Dans une expression algébrique, en l'absence de parenthèses précisant l'ordre des opérations, la machine effectue :

- d'abord les fonctions
- puis les élévations à une puissance A∧B
- puis les prises d'opposés −A
- puis les multiplications et divisions A * B, A/B
- puis les calculs de quotient entier A @ B
- puis les calculs de reste de division A MOD B
- puis les additions et les soustractions A + B − C

Dans les cas de priorités égales, les opérations se font de gauche à droite.

Exemples :

5/2/5	vaut Ø.5
5*2/5	vaut 2
5/2 * 5	vaut 12.5
2∧3	vaut 8
4∧0.5	vaut 2
5∧−1	vaut Ø.2
31 @ 7	vaut 4
31 MOD 7	vaut 3 (MOD est l'abréviation de "modulo")

Attention : @ et MOD opèrent sur des *entiers*, c'est-à-dire que les 2 opérandes sont arrondis avant l'opération et doivent être inférieurs à 32768.

Attention aux calculs impossibles

• Dans A∧B, A doit être positif ou nul (à moins que B ne soit un entier)

• Une division par Ø provoque une erreur numéro 11.

• Si le résultat d'un calcul dépasse $1Ø^{38}$ (en valeur absolue) une erreur numéro 6 est affichée.

Précision

Le MO5 traite 7 chiffres significatifs. Cependant une erreur de l'ordre de 10^{-6} est parfois possible. Ainsi par exemple :

```
?7^7
 823544
?7^6
 117649
?117649*7
 823543
```

SIN COS TAN EXP LOG SQR FIX INT CINT SGN ABS. ★

Trigonométrie

COS(A) vaut cos A
SIN(A) vaut sin A
TAN(A) vaut tg A

Dans les fonctions trigonométriques, l'argument est exprimé en radians.

COS(1) vaut Ø.54Ø3Ø2

Pour passer de la mesure D d'un angle en degré à sa mesure en radians, il faut multiplier D par $\pi/180$.

Exemple :

```
PI=3.141593
FOR D=Ø TO 90 STEP 5
A=D*PI/18Ø
PRINT A,COS(A),TAN(A)
NEXT D
```

Exponentielles et logarithmes

SQR(A) vaut racine carrée de A : $\sqrt{A}$
Si A est négatif une erreur est affichée (numéro 5).

EXP(A) vaut exponentielle de A : e^A
EXP(1) vaut 2.71828
EXP(Ø) vaut 1

LOG(A) vaut logarithme népérien de A : Log (A)
ou Ln (A)
LOG(1) vaut Ø
LOG(Ø.367879) vaut −1
Si A est négatif une erreur est affichée (numéro 5).

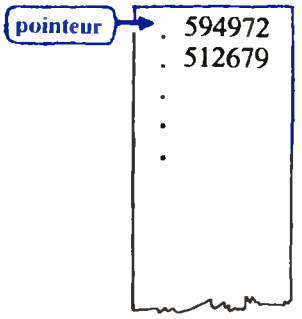
Partie entière et signe

ABS(A)	vaut valeur absolue de A : |A|
SGN(A)	vaut −1 si A est négatif, Ø si A est nul, 1 si A est positif. (SGN est l'abréviation de "signe")
INT (A)	est la "partie entière" de |A| INT(3.52) vaut 3 INT(−3.52) vaut −4
FIX(A)	est la "partie entière" de |A| munie du signe de A FIX(3.52) vaut 3 FIX(−3.52) vaut −3
CINT(A)	est le *nombre entier* le plus proche de A, c'est-à-dire (si $\|A\| \leq 2^{15}$) : INT(A + Ø.5)

Attention : CINT(A) est de type entier.
Par contre INT(A) et FIX(A) sont de type réel.
Brièvement : CINT(A) est égal à A arrondi au nombre entier le plus proche. FIX(A) est exactement A privé de sa partie décimale.

RND ★

RND
ou RND (I) Vaut un nombre compris entre .ØØØ ØØ1 et .999 999.
Tout se passe comme si la machine disposait d'une liste de nombres (pseudo-aléatoires) uniformément répartis entre Ø et 1.




```
?RND
 .594972
```

En début d'exécution, le pointeur est positionné sur le premier nombre de la liste.
RND (7) avance le pointeur de 7 positions et lit le nombre correspondant.
RND (−5) lit toujours le même nombre : le numéro 5 de la liste.
RND équivaut à RND (1)

En conséquence :
Avec RND (X), X négatif, on est assuré de retrouver le même nombre aléatoire pour le même argument, (Cela permet de répéter à l'identique une simulation estimée intéressante).
Avec RND (X), X positif, on appelle successivement des nombres différents mais deux passages d'un même programme donneront la même suite aléatoire, à partir de RND(Ø). Le premier nombre au hasard tiré après l'initialisation est aussi Ø.594972.
Pour éviter ce phénomène (qui peut être désagréable) on peut faire tirer en début de programme un certain nombre de RND.
Par exemple en faisant :

```
10 PRINT"FRAPPEZ UNE TOUCHE!"
20 U=RND: IF INKEY$="" THEN 20
```

Des nombres aléatoires sont tirés tant que l'utilisateur n'appuie pas sur une touche (voir la fiche "INKEY$") ; ce qui assure un choix "aléatoire" du premier tirage qui suivra, ce choix étant lié au temps de réponse de l'utilisateur.

"..." +

★

Variable "chaîne"

A$ = "BONJOUR" la variable alphabétique A$ prend la valeur de la chaîne de caractères entre guillemets.

Le nom d'une variable alphabétique se termine par $ à moins qu'une instruction DEF STR... ait été exécutée.

Une chaîne de caractères comporte de Ø à 255 caractères.

Attention :

- Les minuscules accentuées et ç comptent pour 3 caractères.
- La chaîne vide "" comporte Ø caractère.

Si un programme utilise plus de 3ØØ caractères sous forme de chaînes, utiliser l'instruction CLEAR.

Concaténation (+)

```
A$="BON"
B$="JOUR"
C$=A$+B$+"A TOUS" C$ vaut "BONJOURA TOUS"
```

L'opération ainsi définie entre chaînes s'appelle la *concaténation.*

Comparaison (<)

Les chaînes peuvent être comparées selon "l'ordre lexicographique". Précisément :

- "" est la chaîne la plus "petite".
- La comparaison se fait ensuite caractère par caractère à partir de la gauche : Un caractère est "plus petit" qu'un autre si son code ASCII est plus petit. Voir annexe 7.
- Dès qu'un caractère d'une chaîne est plus petit que le caractère correspondant de l'autre chaîne, ou bien dès qu'une chaîne ne contient plus de caractères, la comparaison est terminée.

Exemples :

```
"TRAC"<"TRUC"
"BAS"<"BASSE"
"A"<"a"
"DEUX"<"UN"
"12"<"2"
"1"<"2"
"BON.JOUR"<"BONJOUR"
```

MID$ LEFT$ RIGHT$ ★

Sous chaînes

A$ = LEFT$ (X$, I) — A$ prend la valeur de la chaîne constituée des I premiers caractères de la chaîne X$.

B$ = RIGHT$ (X$, I) — B$ prend la valeur de la chaîne constituée des I derniers caractères de la chaîne X$.

Si I = Ø, A$ est vide (B$ aussi)
Si I est plus grand que la longueur de X$, A$ vaut X$ (B$ aussi)
Si I<Ø ou I >255, une erreur est affichée (numéro 5).

C$ = MID$ (X$, K, L) — C$ prend la valeur de la sous-chaîne de X$ commençant à son K^e caractère et de longueur L.

Si K≤Ø, ou K>255, ou L <Ø, ou L>255, une erreur est affichée (numéro 5)
Si K est plus grand que la longueur de X$, alors C$ est vide.
Si L est absent ou trop grand, la sous-chaîne s'arrête à la fin de X$.

Exemples :

```
Z$="INFORMATIQUE"
LEFT$(Z$,4)     vaut "INFO"
RIGHT$(Z$,3)    vaut "QUE"
MID$(Z$,3,6)    vaut "FORMAT"
MID$(Z$,3)      vaut "FORMATIQUE"
```

LEN INSTR ★

Longueur d'une chaîne

A = LEN (X$) A vaut le nombre de caractère de la chaîne X$.

```
LEN("")          vaut Ø
LEN("TROIS")     vaut 5
LEN("été")       vaut 7
                 (une lettre accentuée compte pour 3 caractères)
LEN(" ")         vaut 1
```

Test de présence d'un caractère ou d'une chaîne dans une chaîne

K=INSTR (X$,A$) Si la chaîne A$ est une sous-chaîne de X$ débutant au I^{ième} caractère, K prend la valeur I.
Si A$ est vide alors que X$ ne l'est pas, K prend la valeur 1.
Sinon K prend la valeur Ø.

K=INSTR (J,X$,A$) Même résultat, mais on recherche A$ à partir du J^{ième} caractère de X$.

Exemples :

```
INSTR("BONJOUR","JOUR")   vaut 4
Z$="INFORMATIQUE"
INSTR (Z$,"MA")           vaut 6
INSTR (Z$,"Y")            vaut Ø
INSTR (4,Z$,"F")          vaut Ø
```

VAL STR$ ★★

A=VAL (X$) — A prend la valeur du nombre dont X$ est une écriture.

```
VAL("+17.4")      vaut  17.4
VAL("PI")         vaut  0
VAL("15E-2")      vaut  0.15
```

Si le premier caractère de X$ n'est ni un chiffre ni + ni − ni un espace alors VAL (X$) vaut 0.

X$ =STR$ (A) — X$ est la chaîne de caractères égale à l'écriture du nombre A.

```
STR$(-147)              vaut  "-147"
STR$(0)                 vaut  " 0"
STR$(9000*123000)       vaut  " 1.107E+09"
```

Attention : Si A est un nombre positif, STR$ (A) est constitué de la suite de ses chiffres *précédée d'un blanc.*
Cela peut vous gêner si, par exemple, vous voulez écrire STR$ (A) sur une couleur fond-de-caractère différente du fond d'écran (le premier "blanc" fait un peu drôle) ou bien si vous voulez écrire une suite de nombres à un chiffre sans aucun espace entre les chiffres.
Pour ne pas écrire le blanc précédent il faut afficher :

```
MID$(STR$(A),2)
```

ASC CHR$

★★★

ASC(L$) est le nombre entier égal au code ASCII du premier caractère de la chaîne L$

CHR$ (N) est le caractère de code ASCII égal à la valeur de N arrondie.
N doit être compris entre Ø et 255.

Voir la signification des codes ASCII, en annexe 3.

Exemples :

`ASC("ANDRE")`	est égal à 65
`ASC("é")`	est égal à 22 En effet, "é" est la chaîne égale à CHR$ (22) + CHR$ (55) + CHR$ (1Ø1)
`CHR$(74)`	est égal à "J"
`CHR$(74+32)`	est égal à "j".
`?CHR$(30)`	déplace le curseur en haut à gauche de l'écran.
`?CHR$(127)`	fait afficher un carré plein.
`?CHR$(128+I)`	fait afficher le caractère GR$(I) défini par l'instruction DEF GR$(I).

Exemple :

```
10 FOR I=1TO10
20 PRINTCHR$(127)+CHR$(127)+CHR$(10)+CHR
$(127)+CHR$(11);
30 NEXTI

RUN
OK
```

Types de variables

★★

Les variables traitées par le MO 5 peuvent être de 3 types :
• type "nombre entier"
• type "nombre réel"
• type "chaîne de caractères".
En l'absence d'autres renseignements sur le type, MO 5 considère une variable comme étant du type "nombre réel".
Il est possible de préciser qu'une variable est de type-entier ou de type-chaîne de deux façons différentes :
— soit par le dernier caractère de son nom.
— soit par une instruction DEFSTR ou DEFINT ou DEFSNG.
(Voir p. 184, "noms de variables".)

Type réel :

• Les nombres traités par le MO 5 ont une valeur comprise entre 10^{-38} et 10^{38}.
Lorsque le résultat d'un calcul est, en valeur absolue, supérieur à 10^{38}, l'erreur numéro 6 est détectée. Ainsi :
— 1 E 20 * 1 E 20 provoque une erreur.
— Le calcul des factorielles successives s'interrompt à 23 !
• A l'écriture, si la valeur absolue est plus petite que 0.000001 ou plus grande que 999999, l'affichage se fait automatiquement en notation "exponentielle" (dite aussi "scientifique")
0, 237 E 8 se lit 23700000
(le "point décimal" est déplacé de 8 positions vers la droite).
0,237 E − 4 se lit 0.0000237
• le "point décimal" est déplacé de 4 positions vers la gauche).
• le MO 5 conserve d'ailleurs les nombres en mémoire sous la forme "exponentielle" avec 6 chiffres significatifs.
Ainsi le résultat de 12345 * 6789, qui vaut exactement 838102055, est conservé sous la forme 838102 E9 (c'est à dire 838 102 000).

Type entier :

• Il s'agit d'un nombre dont la valeur est comprise entre −32768 et 32767 :
Le choix du type entier pour une variable permet :
• d'économiser de la place en mémoire (voir fiche "mémoire")
• d'éviter les ennuis dans l'évaluation d'expressions intervenant dans les conditions logiques.

• Lorsqu'une valeur réelle est attribuée à une variable entière, cette valeur est arrondie à l'entier le plus proche :
A % = B équivaut à A %=INT(B+Ø.5)
A % = 3.14 équivaut à A % = 3
(Voir fiche INT, FIX, CINT)
• Si une variable de type entier figure dans une expression algébrique où figurent d'autres variables de type réel, elle est transformée en variable de type réel dans le calcul.

Constantes hexadécimales

Le préfixe & H transforme en un entier la suite de caractères qui suit (4 caractères au maximum parmi les chiffres et A, B, C, D, E ou F). Cette suite de caractères est interprétée comme l'écriture d'un nombre en base 16 (Voir annexe 5).
Par exemple :

&H12	représente le nombre 18
&HA	représente le nombre 1Ø
&HF	représente le nombre 15
&HFFFF	représente le nombre 65535.

...$ ...% DEFINT DEFSTR DEFSNG ★

Noms de variables

Un nom de variable est un mot commençant par une lettre.
Exemple : A A1B XXX ANTICONSTITUTIO
Sa longueur maximum est de 15 caractères (les 14 derniers étant soit des chiffres soit des lettres).
Cependant, le début d'un nom de variable ne doit coïncider avec aucun mot clef du BASIC-MO 5. Ne peuvent donc pas être des noms de variables, les mots suivants :

IMPOT	car IMP est un mot-clef.
GOUT	car GO est un mot-clef.
TOILE	car TO est un mot-clef.
*FORMIDABLE*s	car FOR est un mot-clef.
BOXEUR	car BOX est un mot-clef.

Voir la liste des mots-clefs en annexe 4

Fin de noms

Si le nom d'une variable se termine par %, il s'agit d'un "entier"
S'il se termine par un !, il s'agit d'un "réel".
S'il se termine par $, il s'agit d'une "chaîne" (voir fiche "chaîne")
Dans tous les autres cas, il s'agit d'un "réel".

Début de noms

Il est possible de préciser que tout nom de variable commençant par telles lettres de l'alphabet est de type-entier ou chaîne.
C'est l'objet des instructions suivantes :

DEFSTR A, B	Toutes les variables dont le nom commence par A ou B seront de type-chaîne.
DEFINT I – N, P, W	Toutes les variables dont le nom commence par une lettre de I à N (ordre alphabétique) ou par P ou W seront de type-entier.

DEFSNG U – Z Toutes les variables dont le nom commence par une lettre de U à Z sont de type réel ("SNG" vient de "single" signifiant "simple précision").

Les types des variables dont les noms se terminent par %, $ ou ! ne sont pas affectés par les instructions DEF STR DEF INT et DEFSNG.

Occupation des mémoires ★★★

Carte de la mémoire

La mémoire du MO5 est composée de 64 K octets, numérotés de 0000 à FFFF (en héxadécimal, voir annexe 2), et organisés de la manière suivante :

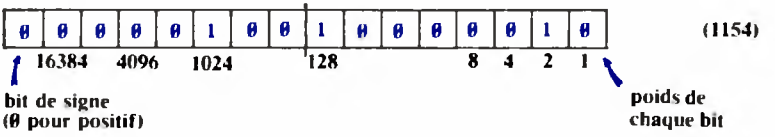
de 0000 à 1FFF	8K octets définissant l'état de chacun des points de l'écran. (Voir explication dans la fiche "couleur de fond")
de 2000 à 9FFF	32K octets à la disposition de l'utilisateur. Sauf la zone 2000 - 21FF utilisée comme zone de données.
de A000 à A800	2K octets pour le contrôleur de disquette et les ports d'entrée/sortie.
de A800 à AFFF	2K libres
de B000 à BFFF	4K réservés pour les cartouches (ROM) d'application.
de C000 à FFFF	16K octets pour le moniteur et l'interpréteur BASIC.

Mémoires occupées par les variables

L'utilisation d'une variable fait occuper des octets d'une part pour stocker son nom, d'autre part pour stocker sa valeur.
Pour son nom il faut un octet de plus que son nombre de caractères.
Pour sa valeur cela dépend de son type.

Chaîne de caractères :
1 octet par caractère (voir "code ASCII") plus 3 octets (1 pour la longueur et 2 pour les "pointeurs").

Entiers :
2 octets.



15 bits permettent de coder un nombre entre 0 et 32767 (c'est-à-dire $2^{15}-1$)
Le 16e bit code le signe.
Le nombre codé ci-dessus en exemple est le nombre 1154 (1024+128+2) ; son bit de signe (positif) vaut Ø.
Si le bit de signe vaut 1, le nombre représenté est négatif ; sa valeur absolue est obtenue en changeant tous les bits (Ø en 1 et 1 en Ø) et en ajoutant 1 au résultat.
Voici, par exemple, la représentation de −2 :

Réel :
4 octets.
Un réel se présente extérieurement sous la forme :
Ø.123456 E 30
"mantisse" "exposant"
En fait, il est codé, non en décimal mais en binaire.
• Le MO5 calcule avec 7 chiffres significatifs ; la "mantisse" reste donc inférieure à 10^7 c'est-à-dire environ 2^{23} à 2^{24}. Or $23=3\times8-1$. La mantisse est donc stockée sous la forme de 3 octets, un bit étant utilisé pour le signe.
• L'"exposant" reste inférieur à 38 car il est stocké sous la forme d'un octet dont un bit est pris pour le signe. Les 7 bits significatifs permettent en effet de coder jusqu'à 127 (2^7-1). On peut donc atteindre 2^{127} c'est-à-dire environ 10^{38}.

Pour les tableaux, chaque élément prend la place prévue par son type ; il faut ajouter quelques octets pour mémoriser les caractéristiques du tableau.

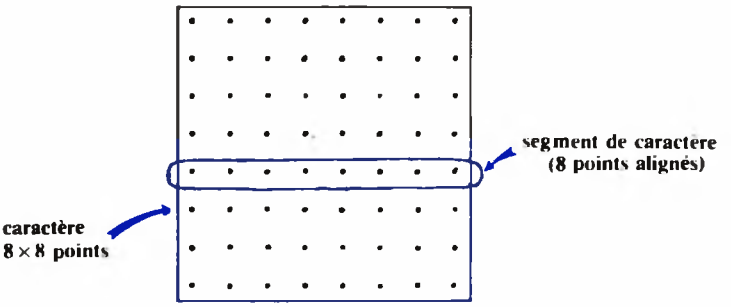
Remarque : Consulter aussi les fiches CLEAR, FRE et VARPTR.

Mémoires occupées par le programme
• Chaque ligne utilise 5 octets, plus 1 ou 2 octets par mot BASIC plus les octets nécéssaires aux variables et opérations énoncées.
• Pendant leur exécution :
— un couple de parenthèses dans un calcul occupe 3 octets
— une boucle FOR−NEXT occupe 19 octets.
— Un appel de GOSUB occupe 7 octets.

Mémoires occupés par la page-écran

Si chacun des 64000 points (200×320) pouvait être colorié de l'une des 16 couleurs ($16=2^4$), il faudrait 4 fois 64K bits pour mémoriser son état (il faut mémoriser cet état car, sur l'écran-télé, l'image est régénérée 25 fois par seconde) : cela ferait 32K octets. Afin d'économiser de la mémoire, on a ramené à 16K octets la mémoire occupée par l'écran ; mais, en contrepartie, il n'est pas possible de traiter chaque point séparement de tous les autres. Précisément :

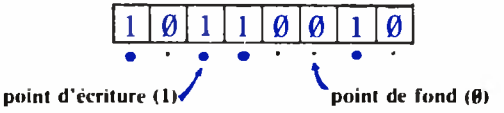
• Chacun des 1 000 caractères est divisé en 8 segments horizontaux.



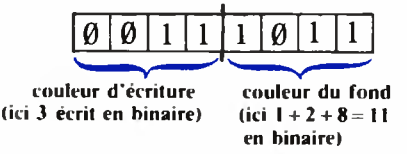
Il y a donc environ 8K segments et on dispose de 8K octets pour stocker l'état de la page-écran.

• Pour chaque segment :

— 1 octet indique l'appartenance de chacun des points à la catégorie "écriture" ou à la catégorie "fond".



— 1 octet, divisé en 2×4, précise la couleur du fond et la couleur de l'écriture.



Sur 4 bits on peut repérer 2^4, c'est-à-dire 16 couleurs !

DIM CLEAR FRE ★★★

Dimensionner les tableaux

```
1 INPUT N
2 DIM A % (N), B(100,3), C$(15)
```

Réserve des mémoires pour :
— le tableau d'entiers A% dont l'indice pourra varier de Ø à la valeur de N.
— le tableau de réels B dont les indices varieront de Ø à 1ØØ pour le premier et de Ø à 3 pour le second.
— le tableau de chaînes C$ dont l'indice pourra varier de Ø à 15.

• Le nombre d'indices et leur valeur dépendent de la place mémoire restante.
• Avant tout autre calcul les variables indicées ainsi créées prennent les valeurs zéro ou vide.
• Si l'indice d'un tableau (de trois dimensions au plus) ne dépasse pas 1Ø, l'instruction DIM n'est pas nécessaire ; dès que l'on parle de KIM (3), tout se passe comme si l'on avait commandé

```
DIM KIM(10)
```

Réservation de mémoires

CLEAR 5ØØ réserve 5ØØ octets (c'est-à-dire 5ØØ caractères) pour le stockage des variables chaînes.

En l'absence de réservation explicite pour les chaînes, le système MO5 réserve 3ØØ caractères.

ATTENTION : Si A$ a 2ØØ caractères, l'instruction A$=A$+"X" a besoin de 4Ø2 octets pour s'exécuter.

CLEAR, 33ØØØ, Interdit à l'interpréteur BASIC d'utiliser les mémoires d'adresse supérieures à 33ØØØ.

Cette instruction permet la réservation de mémoire pour stocker des sous-programmes en assembleur.

CLEAR ,, 7 réserve la place nécessaire à la définition de 7 caractères graphiques par l'utilisateur.

Cette instruction doit précéder la séquence DEFGR$(6).
Le nombre de réservations est limité à 128.

CLEAR efface les valeurs de toutes les variables et de tous les tableaux.

Occupation des mémoires

FRE (Ø) vaut le nombre d'octets encore utilisables en mémoire.

La commande ? FRE (Ø) après l'exécution d'un programme permet de se rendre compte de la place occupée en mémoire et de la place libre (s'il n'y a plus beaucoup de places libres, sa sophistication va vite déboucher sur une erreur numéro 6).

FRE (X$) vaut le nombre d'octets encore disponible pour stocker des chaînes de caractères (variables).

Exemples :

```
?FRE(Ø)
 31003
?FRE(X$)
 300
```

PEEK POKE ★★★

PEEK (I) vaut un entier compris entre Ø et 255, égal à la valeur décimale de l'octet numéro I de la mémoire du MO5. I doit être compris entre Ø et 65535 (64K−1).

POKE I,J place la valeur binaire de J dans l'octet numéro I. J doit être compris entre Ø et 255. I doit être compris entre Ø et 65535. Les nombres I et J sont arrondis s'ils ne sont pas entiers.

PEEK est réservé aux petits curieux ; l'utilisation de POKE est dangereuse si on ne connaît pas exactement les contenus des mémoires du MO5 et leur signification.

Par exemple l'octet numéro &HA7CØ contient quelques enseignements relatifs à l'état de la mémoire écran (voir la fiche SAVEM, LOADM).

Vous pouvez aussi vous amuser à changer les formes ou les couleurs affichées avec des POKE I, J où la variable I prend des valeurs de ØØØØ à 1F4Ø.

Et si vous avez l'âme d'un Sherlock Holmes, faites afficher le contenu des mémoires grâce à un programme du genre :

```
INPUT A,B
FOR I=A TO B
PRINTPEEK(I);
NEXT I
```

Clavier, écran

Clavier, écran

PRINT SPC TAB ★

L'instruction PRINT... est l'instruction courante d'affichage. Elle permet d'afficher tel quel tout texte écrit entre guillemets ou bien d'afficher les valeurs des variables nommées dans l'instruction. Les différents éléments à afficher sont séparés par des , ou des ; selon l'effet désiré.
L'affichage s'effectue à partir de la position où se trouve alors le curseur. Il vaut donc mieux savoir où il se trouve à tout moment de l'exécution et/ou alors commander une instruction LOCATE... avant l'affichage.

PRINT A, A∧2, A∧3 — provoque l'impression des valeurs de A, de son carré et de son cube sur chacun des tiers d'une même ligne, puis le curseur va à la ligne.

PRINT A ; B ; C ; — provoque l'impression des valeurs de A, B, C séparées par un espace et le curseur reste un espace après le dernier chiffre de C.

PRINT — le curseur va au début de la ligne suivante.

PRINT "A=" ; A — provoque l'impression de la chaîne A = suivie de la valeur de la variable A.

```
1 A=12:B=-3:C=4
2 PRINTA,B,C
3 PRINTA;B;C
4 PRINT"A=";A;
5 PRINT"B=";B
6 PRINT"C vaut",C
7 PRINT"ET";" VOILA"+"!"
RUN
 12             -3             4
 12 -3  4
A= 12 B=-3
C vaut         4
ET VOILA!
```

```
PRINTTAB(I) A,X$;W SPC(12)"BON"
```

espace

positionne le curseur sur le I^eme^ caractère de la ligne (modulo 40). Si I est négatif, en début de ligne.
Si I est plus petit que POS, à la ligne suivante.

positionne le curseur sur le caractère suivant.

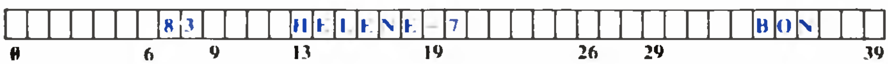
Positionne le curseur à la 0^e^, 13^e^, 26^e^ et 39^e^ position suivant la position précédente.

Les valeurs positives sont précédées d'un espace.
Les valeurs hors de l'intervalle $]10^{-7}, 10^{+6}[$ sont écrites avec exposant.

Si la fin de l'instruction n'est ni un ; ni une , le curseur passe au début de la ligne suivante.

déplace le curseur de 12 positions.

Avec I=6, A=83, X$="HELENE" et W=−7 on obtient l'impression suivante :



Remarques :

— Après ATTRB, 1 le passage à la ligne laisse un interligne.

— L'instruction PRINT permet de passer à la ligne.

— PRINT peut être remplacé par ? .

PRINTUSING ★★

PRINTUSING A$; U, V, W — les variables numériques U,V et W sont affichées selon les spécifications données par la chaîne de caractères A$.

Remarques :

• Si l'instruction ne se termine pas par ; , le curseur va à la ligne après son exécution.

• La chaîne A$ contient une liste de *descripteurs* séparés par des virgules ou des messages alphabétiques qui seront reproduits à l'affichage.

Exemple :

```
10 A$="## OBJETS A ##.## FRANCS: ###.##
FRANCS"
20 INPUT U,V
30 W=U*V
40 PRINTUSING A$;U,V,W
50 END
RUN ENTREE
?3,4.2 ENTREE
 3 OBJETS A   4.20 FRANCS :   12.60 FRANCS
```

Descripteurs

— Chaque # du descripteur représente un chiffre du nombre à afficher et le . représente la place de la virgule.

• les zéros non significatifs à gauche de la virgule ne sont pas affichés.

• les zéros dans les décimales sont affichés.

• le dernier chiffre affiché est arrondi.

• le signe - prend la place d'un chiffre.

Si la place prévue pour l'écriture n'est pas suffisante, le MO5 affichera % suivi de l'écriture la plus compacte possible.

— Si la suite des # est précédée de:

+ ... le signe sera écrit devant tout nombre positif affiché.

* ... les espaces à gauche seront remplacés par des astérisques

Si la suite des # est suivie de :
^ ... le nombre s'affichera en notation exponentielle (l'affichage du nombre prend alors 4 caractères de plus que les dièses et le point)

Exemple :

```
10 PRINTUSING"##.######^";1234.56
```

provoque l'affichage de :

```
 1.234560E+03
```

— Si le nombre de variables est supérieur au nombre de descripteurs, l'interprétation est reprise au début de A$.

INPUT LINEINPUT INPUT$ ★

INPUT A,B,C — provoque l'arrêt de l'exécution et l'affichage d'un point d'interrogation; l'utilisateur doit frapper :
un *nombre* , *un nombre* , *un nombre*
et l'exécution reprend.

INPUT A$ — L'utilisateur peut frapper :
- soit " *n'importe quoi* " ENTREE
- soit n'importe quelle suite de caractères ne comprenant ni , ni : puis ENTREE (les espaces de début et de fin ne sont alors pas considérés).

La frappe de CNT-C interrompt l'introduction des données et l'exécution du programme.
Si les entrées frappées par l'utilisateur ne correspondent pas à celles attendues (en nombre et en type) le message *Redo* est affiché et l'entrée doit être reprise complètement.

Exemple :

```
1 INPUTA,B$,C
2 PRINTA;B$;C
OK
RUN
? 1,OUI
?Redo
? A,OUI,3
?Redo
? 1,2,3
 1  2  3
OK
RUN
? 1,OUI          on a frappé
                 sur CNT - C
Break in 1
OK
```

INPUT "MESSAGE", A,B,C$ provoque l'arrêt de l'exécution et l'affichage, au lieu d'un "?", du message entre guillemets.

Pour rentrer une donnée sans voir apparaître le point d'interrogation on peut donc commander :

```
INPUT"",A
```

INPUT "MESSAGE"; A,B,C$ même effet mais le point d'interrogation apparaît.

LINE INPUT A$ provoque l'arrêt de l'exécution. L'utilisateur peut alors frapper une suite quelconque de caractères suivi de ENTREE.

Attention : Aucun "?" n'est affiché.

On peut aussi commander :
LINE INPUT "message", A$

X$=INPUT $ (I) provoque l'arrêt de l'exécution.
X$ prend pour valeur la suite des I caractères frappés par l'utilisateur au clavier.
Reprend l'exécution dès que I caractères ont été frappés.
(il n'y a aucun autre moyen de relancer l'exécution ou de reprendre la main).

Attention : Les caractères frappés ne sont pas affichés.

Exemple :

```
1 PRINT"QUEL EST LE MOT DE PASSE ?"
2 X$=INPUT$(6)
3 IF X$="SESAME" THEN 10 ELSE CLS:GOTO 1
10 PRINT"SOYEZ LE BIENVENU"
```

X$=INPUT$ (1) permet d'attendre l'appui sur une seule touche du clavier sans que l'utilisateur ait à frapper ENTREE.

A$ = INPUT$ (I, # 2) attend l'entrée d'une suite de I caractères sur le canal numéro 2.

INKEY$ ★★

A$=INKEY$ A$ prend la valeur du caractère frappé au clavier au moment où l'instruction est exécutée.

Cette fonction permet de tester l'appui d'une touche sans arrêter l'exécution.

Attention : l'appui sur une touche doit se faire au moment de l'exécution de l'instruction (ni avant, ni après).

Exemple :

```
40 A$=INKEY$:IF A$="" THEN 40
```

— Si aucune touche n'est frappée pendant l'exécution de l'instruction 40, celle ci est répétée indéfiniment.

— Lorsqu'une touche a été frappée, l'instruction suivante est exécutée et A$ a pris la valeur du caractère frappé.

DATA READ RESTORE ★★

Les données

• L'instruction DATA contient une liste de constantes numériques ou alphabétiques séparées par des virgules.
Les données-chaînes peuvent ne pas être encadrées de guillemets si elles ne contiennent ni virgule ni deux points, ni espaces significatifs en début ou fin.
Les instructions DATA peuvent être éparpillées dans le programme (mais il n'est pas mauvais de les grouper).

• Après l'exécution de RUN, un pointeur se positionne devant la première donnée ; puis tout se passe comme si il n'y avait qu'une seule instruction DATA contenant la liste de toutes les données.

Exemple :

```
10 DATA 4,PRINTEMPS
20 READ NB%
30 DATA ETE,AUTOMNE
40 FOR I=1 TO NB%:READ SAISON$(I):NEXT
50 DATA"HIVER"
```

Leur lecture

• L'exécution d'une instruction READ provoque :
— l'affectation, à la variable (ou aux variables) nommée(s) dans READ, de la valeur repérée par le "pointeur" des données
— l'avance du pointeur vers la donnée suivante :
Une erreur est détectée si une variable nommée dans READ ne correspond pas au type de la donnée pointée ou bien si la liste des DATA est épuisée alors qu'il reste des variables à lire.

• Lorsqu'on veut pouvoir modifier le nombre de données à lire après une utilisation, on peut programmer un "point final" numérique ou alphabétique. Par exemple, pour faire lire une suite de notes comprises entre Ø et 2Ø, on peut programmer la séquence suivante :

```
30 DATA 5,7,14,8,-1
31 K=0
32 K=K+1:READ A(K): IF A(K)>0 THEN 32
```

Si de nouvelles notes sont obtenues, l'utilisateur pourra modifier l'instruction 30 :

```
30 DATA 5,7,14,8,19,19,15,-1
```

La gestion du pointeur de données

40 RESTORE 30 positionne le pointeur des données devant la première donnée figurant dans la première instruction DATA suivant l'instruction 30.

Cela permet soit de relire une séquence de données, soit de lire telle ou telle séquence selon le résultat d'un test.

RESTORE sans numéro de ligne, positionne le pointeur des données sur le premier DATA du programme.

PSET LINE BOX BOXF ★

Point, ligne, boite

PSET (X,Y), 3	Allume le point (X,Y) en jaune.
LINE (X1,Y1)−(X2,Y2)	Trace le segment de droite d'extrémités (X1,Y1) et (X2,Y2).
LINE−(X2,Y2)	Si le point initial est omis c'est le dernier point colorié par une instruction graphique ou, par défaut, le point (Ø,Ø).
BOX (X1,Y1)−(X2,Y2)	Trace les côtés du rectangle dont une diagonale est [(X1,Y1), (X2,Y2)].
BOXF (X1,Y1)−(X2,Y2)	Colorie l'intérieur du rectangle précédent.

• BOX, BOXF et LINE ont même syntaxe.

• Les coordonnées sont arrondies à l'entier le plus proche et doivent être comprises dans [Ø,319] × [Ø,199].
Si une coordonnée est négative, l'erreur numéro 5 est affichée ;
Si une coordonnée dépasse 319 (ou 199), les points allumés sont ceux qui sont au bord droit (ou bas) de l'écran.

• Le curseur n'est pas déplacé par une instruction graphique.

Paramètre de couleur

PSET (X,Y) ,N N précise la couleur prise pour le coloriage du point (X,Y) ($-16 \leq N \leq 15$)
(idem pour BOX, BOXF et LINE)

Les couleurs courantes d'écriture et de fond ne sont pas modifiées. Si N est positif le point est considéré comme un "point-graphique".
Si N est négatif le point est considéré comme un "point-fond".
(Voir la fiche de référence : couleur de fond)

Mode "caractère"

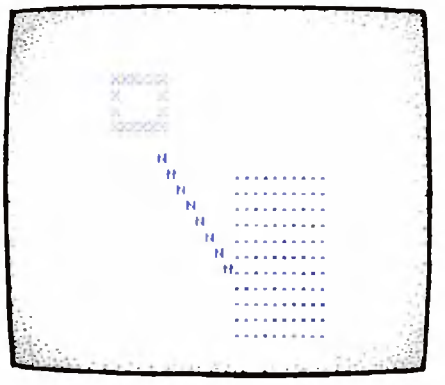
PSET (X,Y) "K"
PSET (X,Y) "K", U, V, W — L'allumage du point est remplacé par l'écriture du caractère "K" sur la position de caractère contenant le point de coordonnée (X,Y).
(idem pour BOX, BOXF et LINE)

Les coordonnées X et Y sont ici les numéros de colonne et de ligne ($0 \leq X \leq 39$, $0 \leq Y \leq 24$)
Les paramètres U,V,W agissent exactement comme ceux de l'instruction COLOR.

Exemples

```
1 CLS
2 LINE -(100,1000),2
3 LINE (0,100)-(100,0),1
4 LINE (50,0)-(75,100),4
6 LINE (200,0)-(300,100),2
7 LINE (200,100)-(300,0),-2
8 LINE (250,0)-(300,0),-2
9 LINE (200,80)-(300,80),-4
```

```
1 BOX(8,8)-(13,11)"%"
2 LINE (13,13)-(20,20)"NON"
3 BOXF(21,14)-(30,24)"."
```



Fond et écriture POINT

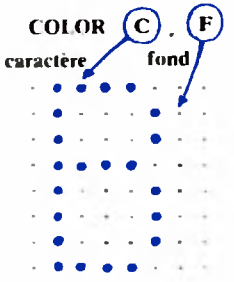
★★

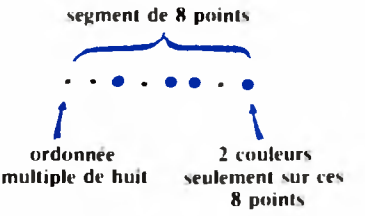
Chacun des 64 000 points de l'écran appartient à un bloc de 64 points formant un "caractère". A l'initialisation, chacun de ces points est un *point-de-fond* (comme un papier sur lequel on va pouvoir écrire) colorié de la couleur du fond définie par SCREEN, ..., (... = 4 par défaut) Chacun de ces points peut devenir un point-écrit lors de l'exécution d'une instruction PRINT ou d'une instruction graphique (PSET,LINE, BOX, BOXF) .

(Voir la fiche de référence "occupations des mémoires").

Caractère

• Lors de l'écriture d'un caractère, la couleur de fond du caractère est celle qui a été définie dans la dernière instruction SCREEN ou COLOR exécutée.





Fond et écriture

• Lors du coloriage d'un point, il faut savoir que les 8 points appartenant à une même ligne d'un même caractère ne peuvent prendre que 2 couleurs différentes : une couleur "fond" et une couleur "écriture".

En conséquence ;
— Si les 8 points appartenant au même caractère que le point considéré sont de la couleur d'écriture ou de la couleur de fond dernièrement choisies, le coloriage se fait normalement selon la commande.
— Sinon, il y a deux cas possibles :
• le paramètre de couleur de l'instruction graphique exécutée est absent ou positif ; le point à colorier et les points "écriture" précédemment coloriés sur le segment prennent ensemble la couleur d'écriture.
• Le paramètre de couleur de l'instruction graphique exécutée est négatif ; le point à colorier est considéré comme un point "fond" ; l'ensemble des points "fond" sur le segment est alors recolorié de la couleur indiquée.

• **Code des couleurs de fond** indiqué comme dernier paramètre des instructions graphiques BOX, BOXF, LINE, PSET :
La couleur d'écriture C et la couleur de fond − (C+1) sont les mêmes.

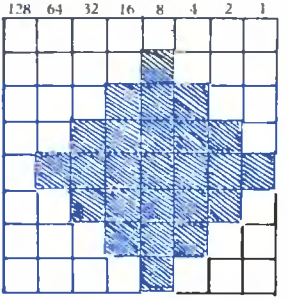
Exemple :
−4 (fond) est la couleur 3 (écriture)

Point (X, Y) a la valeur du code de la couleur du point (X,Y).
(Si le point est un point "fond", cette valeur est négative).

DEFGR$ GR$ ★★

Un caractère est défini sur une grille de 8 segments de 8 points.
Chaque segment du caractère est codé par un nombre de Ø à 255, déterminé de la manière suivante :
— A chacune des huit colonnes est attribué un "poids" de 1 à 128 (indiqué en couleur sur l'exemple ci-dessous) de la droite vers la gauche.
— Pour déterminer le code d'une ligne, on doit additionner les poids de chaque point dessinant le caractère.
Par exemple la troisième ligne est codée 28, car les points du caractère sont dans les colonnes correspondant à 4, 8 et 16.
(4+8+16=28).

Exemple :
Le caractère "as de carreau"



Avant de définir un caractère, il faut avoir réservé la place permettant de le mémoriser. Cela se fait par l'instruction CLEAR,,

```
5 CLEAR,,1
```
réserve 1 place en mémoire

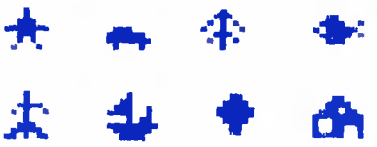
```
10 DEFGR$(0)=0,8,28,62,127,62,28,8
```
définit le caractère GR$(Ø)

```
20 PRINT GR$(0)
```
imprime le caractère GR$ (Ø)

GR$ (Ø) est le nom du caractère défini par DEFGR$(Ø). On l'emploie comme un nom de chaîne de caractères normal.
On peut définir 128 caractères au maximum.
Le J[ième] caractère défini s'appelle GR$ (J − 1) ou bien CHR$ (128 + J).
Les caractères doivent être définis dans l'ordre à partir de Ø : Ø,1,2...

Autre exemple :

```
5  CLEAR   ,,8
10 DEFGR$(0)=8,8,28,127,28,20,34,0
11 DEFGR$(1)=0,0,0,56,126,127,34,0
12 DEFGR$(2)=16,56,84,146,56,84,16,0
13 DEFGR$(3)=0,48,121,190,125,56,0,0
14 DEFGR$(4)=16,16,60,82,16,56,124,214
15 DEFGR$(5)=16,48,114,242,22,255,126,60
16 DEFGR$(6)=24,60,126,126,60,24,24,0
17 DEFGR$(7)=24,60,118,255,159,153,249,0
20 ATTRB1,1
21 FOR K=0 TO 7
22 PRINT GR$(K)+" "; IF K=3 THEN PRINT
23 NEXT
30 END
```



Périphériques

PLAY ★

Note, octave, longueur, tempo, attaque

La seule instruction provoquant l'émission de son se présente sous la forme : PLAY suivi d'une chaîne de caractères.

```
10 A$="A10L4801T10"
20 B$="DOMISOP"
30 PLAY "DO# FA LAb"+"O5 SI SI SI."+"L12
A100 LA SO FA MI"
```

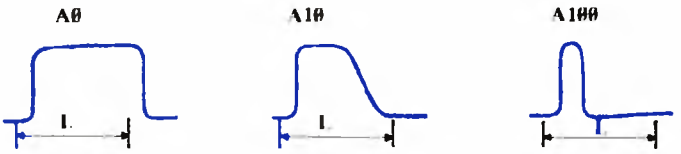
(On peut mettre plusieurs chaînes de caractères séparées par une virgule, mais cette virgule est équivalente au "+" concaténant les chaînes).

— La rencontre de DO RE MI FA SO LA ou SI,suivis éventuellement de # ou b, provoque l'émission de la note correspondante. Attention : "sol" s'écrit SO car une note est une chaîne de 2 caractères.

— La rencontre de P provoque un silence de la longueur d'émission d'une note.

— La rencontre de O1, O2, ou... O5 change la hauteur de l'octave d'émission : 1 est l'octave la plus grave,
4 est l'octave standard à l'initialisation

— La rencontre de L1 L2 ou... L96 change la longueur d'émission des notes :
1 est la longueur la plus courte
24 est la longueur standard à l'initialisation (note "noire")
— La rencontre de T1, T2 ou... T255 change le "tempo" c'est-à-dire, par exemple, le nombre de noires émis en 1 minute.
1 est le tempo le plus rapide.
5 est le tempo standard à l'initialisation.
— la rencontre de AØ A1 ou... A255 change "l'attaque" lors de l'émission d'une note.
Ø provoque l'émission d'une note continue. (c'est l'attaque standard à l'initialisation)
1Ø provoque l'émission de la note pendant une durée plus courte que la longueur prévue puis elle s'amortit jusqu'au silence avant l'émission de la note suivante.



Erreurs

Une erreur est détectée si les chaînes suivant PLAY contiennent d'autres lettres ou mots que ceux attendus (voir ci-dessus) ou si les nombres suivants O, L, T et A ne sont pas des entiers compris dans les intervalles prévus (1 à 5 pour O, 1 à 96 pour L, 1 à 255 pour T, Ø à 255 pour A). Cependant,on peut séparer les notes par des espaces ou par des points virgules.

Notes variables

Il n'est pas possible de faire suivre O, L, T ou A d'une variable. Cependant, on peut provoquer l'émission d'une note, d'octave, de longueur, de tempo ou d'attaque variables en utilisant la fonction STR$. Par exemple :

```
30 PLAY L$,"DOREMI"
```

L'instruction L$="L"+STR$ (LONG) provoquerait une malencontreuse erreur due à la présence d'un blanc devant les chiffres de l'entier LONG.

On a ainsi rendu variable la "longueur-d'une-note" : L$.

• De même, il peut être commode d'associer à chaque note, de DO à SI, un nombre de Ø à 6 pouvant être variable...

Par exemple, en utilisant un tableau de chaînes :

```
10 DATA DO,RE,MI,FA,SO,LA,SI
20 FOR I=0 TO 6
30 READ A$(I)
40 NEXT I
```

..... et A$ (I) est la I^{ème} note de la gamme.

..... on peut alors jouer par exemple :

```
50 FOR I=0 TO 6:PLAY A$(I)+"P":NEXT I
```

SAVE LOAD RUN MERGE MOTORON MOTOROFF SKIPF ★

Sauver un programme

SAVE "TRUC" — sauvegarde sur la cassette le programme situé dans la mémoire du MO5. Le programme est enregistré sous le nom "TRUC".
Attention : sur le lecteur-enregistreur, les touches d'enregistrement doivent être enclenchées. ⏺ ▶.
Attendre le message "OK" qui annonce la fin de l'enregistrement sur la cassette.

SAVE "TRUC", A — le programme TRUC est sauvegardé sous forme codée en ASCII.
(pour la récupération on pourra alors commander : MERGE)

SAVE "TRUC", P — le programme est sauvegardé et "protégé". Attention, il ne pourra plus jamais être modifié ou listé et, après sa récupération en mémoire, les instructions POKE et PEEK ne pourront pas être exécutées en mode immédiat.

Charger un programme

LOAD "TRUC" — provoque la recherche sur la cassette du programme du nom "TRUC" (à condition que la touche ▶ du lecteur-enregistreur soit enclenchée).

Pendant la recherche, le message :

```
Searching
```

est affiché.

Dès que l'enregistrement d'un programme de nom CHOSE (différent

de TRUC) est trouvé, le message

```
Skip : CHOSE     BAS
```

est affiché ; et cet enregistrement est parcouru sans être lu par le MO5.

Lorsque le programme TRUC est trouvé, le message :

```
Found : TRUC     BAS
```

est affiché. Alors :

— le programme précédemment en mémoire du MO5 est effacé

— les fichiers sont fermés

— le programme TRUC est chargé dans la mémoire du MO5 et le message "OK" apparaît.

(On peut alors commander LIST ou RUN...)

LOAD	Le programme chargé en mémoire est le premier rencontré sur la cassette.
LOAD "TRUC", R	Charge le programme "TRUC" et lance son exécution. Les variables sont détruites. (les fichiers ne sont pas fermés)

Note : Les instructions SAVE A$ et LOAD "TRUC" peuvent être des instructions de programme.

Voir, page 220, le paragraphe "descripteur de fichier".

Exécuter un programme sur cassette

RUN "TRUC"	équivaut à la succession de commandes : LOAD "TRUC" et RUN
RUN "TRUC", R	équivaut à LOAD "TRUC", R
RUN"	équivaut à LOAD" et RUN

Charger plusieurs programmes

MERGE "TRUC"	provoque la recherche du programme TRUC et son chargement, *à condition* qu'il ait été enregistré sous forme codée par la commande SAVE "TRUC",A
MERGE "TRUC", R	avec l'option R, l'exécution est lancée à partir de la première instruction.

Le chargement est exécuté *sans destruction du programme* précédem-

ment en mémoire (à condition que les numéros d'instructions soient différents), mais les variables sont effacées.
Les instructions LOAD''... MERGE''... et RUN''... peuvent être programmées, ce qui permet le *chaînage* des programmes.

Faire avancer la bande

SKIPF — Provoque la recherche et le saut du premier programme ou fichier enregistré sur la cassette.

SKIPF ''TRUC'' — Provoque la recherche et le saut du programme TRUC enregistré sur la cassette.

Remarque :

• La commande SKIPF ''TRUC'' permet de positionner la bande juste après le programme TRUC.
(Cela permet un enregistrement ultérieur sur la suite de la bande).

• Pour connaître le contenu des enregistrements figurant sur une cassette, on peut :
— rembobiner la cassette
— commander LOAD''X'', si l'on est certain qu'aucun enregistrement ne s'appelle X.
— noter les programmes successivement "sautés", ainsi que les numéros-compteur correspondant au début de leur enregistrement (au moment où le message "Found..." apparaît à l'écran).
• Si le message :

```
Error 53
OK
```

apparaît en cours de lecture ou de chargement, il faut rembobiner la bande avant le début de l'enregistrement pour tenter un nouvel essai de chargement.

MOTORON — provoque le défilement de la bande, à condition que la touche ▷ soit enclenchée.
Si la touche *enreg* est aussi enclenchée, il y a effacement de la bande.
Pendant ce défilement, on peut frapper toute commande voulue sur le clavier, en particulier la suivante:

MOTOROFF — arrête le défilement de la bande.

SAVEM LOADM

★★★

Il est possible de sauver sur cassette l'image figurant à l'écran à un instant donné ou bien les valeurs des variables en mémoire. Pour cela on dispose d'une instruction qui sauve, en fait, une partie de la mémoire du MO5.
Cette instruction est :

SAVEM "TRUC", &HØØØØ, &H2ØØØ
sauve sur la cassette et sous le nom "TRUC", les mémoires dont les adresses sont comprises entre les numéros &HØØØØ et &H2ØØØ (voir en annexe 5 la conversion hexadécimal-décimal).

LOADM "TRUC" recharge en mémoire le contenu de la cassette qui a été enregistré sous le nom "TRUC".
(La tête de lecture de la bande doit évidemment être positionnée avant le début de l'enregistrement de "TRUC").
Le chargement a lieu dans le bloc de mémoires nommées lors de l'instruction SAVEM "TRUC".

SAVEM "TRUC", &HØØØØ, &H2ØØØ, &H3ØØ
le troisième parametre de SAVEM indique l'adresse où le programme devra être repris et exécuté dès que son chargement aura été réalisé après un LOADM.

Exemple : sauver une image

La mémoire-écran du MO5 est constituée de deux blocs de 8K chacun contenant respectivement les informations relatives à la "forme" (distinction entre fond et écriture) et à la "couleur" (du fond et de l'écriture).
Ces deux blocs ont une même "adresse" : de &HØØØØ à &H1FFF.
Si la mémoire &HA7CØ contient un nombre impair, ces adresses font accéder aux informations de forme, si cette mémoire contient un nombre pair, elles font accéder aux informations de couleur.

Voici donc un programme permettant de stocker une image-écran à un moment donné :

```
100 POKE&HA7C0,PEEK(&HA7C0) OR 1
110 SAVEM"FORME",&H0000,&H1F40
120 POKE&HA7C0,PEEK(&HA7C0) AND 254
130 SAVEM"COULEUR",&H0000,&H1F40
```

La séquence de rechargement en mémoire sera :

```
200 POKE&HA7C0,PEEK(&HA7C0) OR 1
210 LOADM"FORME"
220 POKE&HA7C0,PEEK(&HA7C0) AND 254
230 LOADM"COULEUR"
```

INPUTPEN INPEN TUNE PTRIG ★★

Réglage

TUNE provoque l'apparition d'un trait vertical au milieu de l'écran.
Appuyez alors le crayon sur ce trait... et le crayon optique est réglé.

Si l'on n'exécute pas l'instruction TUNE avant d'utiliser le crayon optique, il se peut que sa lecture soit entachée d'une légère erreur de position.

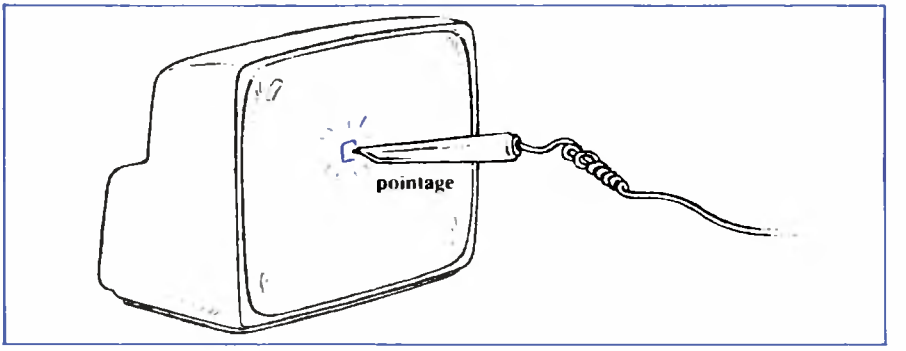
Attention :

Pour une lecture correcte :

— le crayon optique doit être pointé bien perpendiculairement à l'écran.

— la luminosité de l'écran doit être suffisante (légèrement supérieure à la normale)

— le crayon ne doit pas être pointé sur une zone noire ou rouge (ces couleurs ne sont pas perçues par le crayon optique).



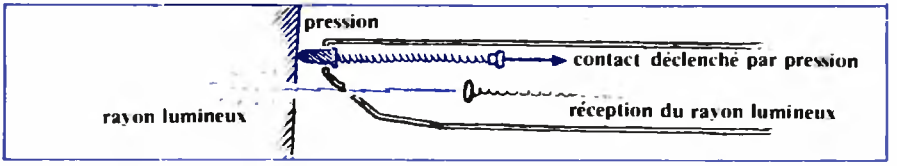
Pointage

INPUTPEN X,Y provoque l'arrêt de l'exécution et l'attente du pointage du crayon sur l'écran.
Lorsque le pointage est effectué sur le point de coordonnées (123,45) la variable X reçoit la valeur 123 et la variable Y reçoit la valeur 45 et l'exécution continue à l'instruction suivante.

INPEN X,Y provoque la lecture des coordonnées du point montré par le crayon. (La machine n'attend pas de pression sur l'interrupteur).

X reçoit une valeur entière comprise entre Ø et 319.
Y reçoit une valeur entière comprise entre Ø et 199.
Si le crayon pointe en dehors de l'écran ou si la luminosité est mauvaise, X et/ou Y prennent la valeur −1.

Le crayon optique comporte en fait 2 organes d'entrée :
— un petit trou ("capteur") recevant le rayon envoyé par le pinceau lumineux du moniteur T.V.
— un interrupteur déclenché par l'appui de l'extrémité du crayon sur une surface dure.



Dans l'instruction...

INPEN X, Y Le toucher physique de l'écran n'est pas nécessaire mais,si le crayon ne montre pas un point de l'écran,X et Y prennent des valeurs négatives.

Dans l'instruction...

INPUTPEN X,Y l'exécution reprend après pression sur l'interrupteur.
(Il faut donc bien appuyer le crayon contre l'écran).

PTRIG cette variable prend la valeur "VRAI" lorsque le bout du crayon est appuyé et "FAUX" sinon.

Exemple :

```
10 CLS
20 PRINT"APPUYEZ VOTRE DOIGT SUR LE BOUT
":PRINT"DU CRAYON"
30 IF PTRIG THEN 40 ELSE 20
40 PRINT "OUF ! MERCI !"
49 END
```

"LPRT :" PRINT # SCREENPRINT ★★

Copie d'écran

SCREENPRINT provoque la recopie intégrale de l'écran sur l'imprimante "graphique"
les points-écriture sont recopiés noirs et les points-fond restent blancs.

L'imprimante "graphique" est une imprimante à "interface parallèle" écrivant sur un papier thermo-sensible à 40 caractères par ligne.
L'autre modèle d'imprimante à interface parallèle utilisable comme sortie du MO5 est une imprimante "à impact".
Elle n'accepte pas l'instruction SCREENPRINT mais elle permet l'écriture de 80 caractères par lignes et un sous-programme en langage machine peut vous permettre une copie d'écran.

Pour lister des instructions sur imprimante

LIST "LPRT:"	provoque le listage du programme sur l'imprimante connectée.
LIST "LPRT: (80)"	provoque le listage sur l'imprimante à raison de 80 caractères maximum par ligne.
LIST "LPRT:", 10 -20	provoque le listage sur l'imprimante (à raison de 40 caractères par ligne) des instructions de numéros compris entre 10 et 20.
LIST "LPRT:(80)", -100 LIST "LPRT:(20)", 1000 -	voir l'instruction LIST pour les différentes options pouvant suivre la virgule.

Pour sortir des résultats sur l'imprimante.

Il faut, avant toute commande d'impression par PRINT, "ouvrir" le canal permettant le passage des données du MO5 vers l'imprimante. Comme pour un fichier, il faut penser à "fermer" ce canal après son utilisation.

OPEN "O", # 1, "LPRT:" provoque l'ouverture du canal numéro 1 pour la sortie ("O" utput) vers l'imprimante (LPRT:)

Le numéro du canal est choisi par l'utilisateur entre 1 et 16, c'est lui qui devra être rappelé dans l'instruction d'impression.

PRINT # 1, A ; B provoque l'impression, par le canal 1, des valeurs des variables A et B. Voir l'instruction PRINT pour les différentes options pouvant suivre la virgule.

CLOSE # 1 provoque la fermeture du canal 1.

Les instructions suivantes sont correctes et montrent quelques autres possibilités :

```
OPEN"O",#2,"LPRT:(80)"
?#2,USING A$;B,C,D
CLOSE
```

OPEN CLOSE EOF SKIPF PRINT # INPUT # LINE INPUT

★★★

Créer un fichier

OPEN "O", #3, "MACHIN" provoque l'ouverture du canal numéro 3 pour la sortie ("O"utput) vers le lecteur-enregistreur de cassettes, du fichier de nom MACHIN.

Le numéro du canal est choisi par l'utilisateur entre 1 et 16 (attention de ne pas utiliser un même canal pour deux fichiers différents ou pour l'imprimante).
C'est ce numéro qui devra être rappelé dans l'instruction de passage des données vers la cassette.

PRINT # 3, U, V, W$ provoque l'écriture, par le canal 3, des variables U, V, W$.

Si le canal 3 est associé en sortie au fichier "MACHIN" les valeurs de U, V, W sont écrites à la suite des données déjà entrées dans MACHIN par le lecteur-enregistreur de cassettes.

CLOSE # 3 provoque la fermeture du canal 3.
CLOSE provoque la fermeture de tous les canaux.

Remarque :
Les instructions END, NEW et RUN ferment automatiquement tous les fichiers.

Consulter un fichier

OPEN "I", #4,"MACHIN" provoque l'ouverture du canal n°4 pour l'entrée ("I"nput), à partir du lecteur-enregistreur de cassettes, du fichier de nom MACHIN ; et recherche le début du fichier MACHIN sur la bande.

Le numéro du canal compris entre 1 et 16 doit être rappelé dans l'instruction d'entrée des données de la cassette vers le MO5.

INPUT #4, X, Y$, Z — provoque la lecture de données successives enregistrées sur la cassette.

Les types de données doivent évidemment se correspondre comme pour les instructions DATA et READ (le pointeur des DATA correspond ici à la tête de lecture devant la bande de la cassette).

LINEINPUT #4, U$ — provoque la lecture d'une chaîne de caractères sur la cassette.

Remarque : Lors de l'enregistrement par PRINT #, le caractère de code ASCII 13 (ENTREE) est enregistré entre chaque variable. C'est lui qui joue le rôle de séparateur à la lecture par INPUT # ; ce caractère ne fait évidemment pas partie de la valeur affectée à la variable lue.

EOF (3) — est une variable ayant la valeur "vrai" (−1) si la fin du fichier correspondant au canal 3 est atteinte et la valeur "faux" (Ø) sinon.

Descripteur de fichier

Dans l'instruction :

```
OPEN "O",#3,DF$
```

DF$ peut être variable et prendre la valeur PERIF$+ NOMF$. PERIF$ peut être égal à SCRN : (pour sortie sur écran), LPRT : (pour sortie sur imprimante) ou CASS : (pour sortie sur cassette).
Si PERIF$ n'est pas précisé, et si l'unité de disquette n'est pas connectée, c'est la sortie sur cassette qui est prise par défaut. NOMF$ est un "nom de fichier" constitué de 8 caractères au maximum, éventuellement suivis d'un point et de 1 à 3 caractères pouvant être : DAT (pour données), BAS (pour programme) et BIN (pour binaire).
Dans l'instruction :

```
OPEN "I",#1,PERIF$+NOMF$
```

PERIF$ peut être égal à KYBD : (pour entrée par clavier) ou CASS : (pour entrée par cassette).

Annexes

ANNEXE 1

Renseignements techniques - MO5

Présentation :

Coffret : matière moulée.
Clavier : Type à déplacement "silicone".
Dimensions : L.291 - H.51 - P.190 mm.
Masse : 1,1 kg.

Particularités techniques :

Alimentation :	Extérieur 17 volts courant continu - 750 mA.
Microprocesseur :	6809 E - 1 MHz.
Mémoire :	64 k dont : Utilisateur : 32 k RAM. Écran graphique : 16 k RAM. BASIC : 12 k ROM. Moniteur : 4 k ROM.
Écran :	Sortie RVB+Son par prise péritélévision. 320×200 points - 16 couleurs. 25 lignes×40 caractères. Majuscules et minuscules accentuées.
Clavier :	57 touches AZERTY avec accents.
Musique :	Synthèse musicale sur 5 octaves.
Crayon optique :	Résolution 320×200 points.
Interface cassette :	1 200 bauds avec télécommande pour lecteur/enregistreur de programmes. Voie sonore sur la deuxième piste.
Extension disques :	Contrôleur-lecteur disquettes 5 1/4 pouces. Maximum 4 unités de 80 K octets ou 2 unités de 320 K octets.

Extension communication :	Interface parallèle type « CENTRONICS ».
Extension musique et jeux :	2 leviers de jeux digitaux. Générateur d'enveloppe incorporé pour synthèse sonore jusqu'à 4 voix simultanées sur 7 octaves.
Imprimantes :	Imprimante matricielle 50 caractères par seconde, 80 colonnes, graphique. Imprimante thermique, 2 lignes par seconde, 40 colonnes, graphique.

Le moniteur du MO5

SOMMAIRE

1.0 INTRODUCTION

1.1 Le moniteur du MO5

Le moniteur du MO5 est formé de différentes "routines" (programmes) ayant chacune une fonction bien précise: gestion de l'écran, du clavier, lecture du crayon optique, des manettes de jeu, génération de musique, gestion de l'interface de communication, du lecteur-enregistreur de programmes, etc...

Les programmes d'application en ROM, comme le Basic résident ou l'Assembleur, utilisent ces routines pour leurs propres besoins et vous pouvez faire de même. Le principe est le suivant:

1) Vous chargez avec des valeurs ayant une signification précise certains registres du 6809, et éventuellement certains registres en RAM (mots-mémoire sur un ou deux octets): ce sont les paramètres d'entrée.

2) Vous appelez la routine du moniteur qui effectue l'opération demandée, en utilisant le mécanisme explicité au § 1.2

3) Vous récupérez s'il y a lieu le résultat de l'opération dans certains registres du 6809, et/ou dans des registres en RAM: ce sont les paramètres de retour.

Dans certains cas, il n'y a pas de paramètres d'entrée, ou pas de paramètres de retour, mais le schéma général est toujours le même.

Dans ce qui suit, certains nombres ou adresses-mémoire seront écrits en hexadécimal pour plus de commodité: dans ce cas ils seront suivis par la lettre "H". Sauf avis contraire, tout nombre non suivi par un "H" est à lire en décimal.

Exemple: 10H = 16, 16H = 22, etc...

Les registres en RAM sur deux octets seront notés de la façon suivante: XXXXH-XXXXH où X représente un chiffre hexadécimal. Pour une meilleure compréhension, nous avons donné des noms à certains registres-mémoire: à chacun d'entre eux correspond l'adresse en mémoire du premier octet du registre.

Exemple: mettre la valeur 4B0H dans le registre 205AH-205BH signifie: mettre la valeur 04 dans le mot-mémoire d'adresse 205AH et mettre la valeur B0H dans le mot-mémoire d'adresse 205BH.

Les nombres en binaire seront précédés du symbole "%". Les bits notés bi sont numérotés de droite à gauche, le bit de poids le plus faible étant le bit 0, et celui de poids le plus fort étant le bit 7.

Exemple: %11001100

b7 = 1

b0 = 0

1.2 Accès aux routines du moniteur

L'accès normalisé à une routine du moniteur se fait en utilisant l'instruction "SWI" suivie d'un code (sur un octet). Ces codes sont au nombre de 58, répartis en deux groupes différenciés par l'état du bit 7, les 6 bits de poids faible formant le code proprement dit de la routine à laquelle on veut accéder. Le bit 7 a la signification suivante: s'il est à 0, un "JSR" à la routine est simulé, c'est-à-dire que le retour se fait à l'instruction suivant l'appel: s'il est à 1, un "JMP" est simulé, c'est-à-dire que le retour se fera à l'instruction qui suit l'appel du programme où se trouve l'appel au moniteur.

EXEMPLE

```
PROG1   EQU     *
        SWI
        FCB     08H
        Instruction suivante  1
```

Le bip sonore se fera entendre, puis "instruction suivante 1" sera exécutée.

Mais si l'on rencontre dans un programme:

```
PROG1   EQU     *
        JSR     PROG2
        Instruction suivante  1
        . . . . . .
PROG2   EQU     *
        SWI
        FCB     88H
        Instruction suivante  2
```

Le bip sonore se fera entendre, puis "instruction suivante 1" sera exécutée, et non "instruction suivante 2".

L'instruction SWI sauvegarde tous les registres du 6809. Au retour, tous les registres sont remis dans l'état qu'ils avaient lors de l'appel, sauf le registre code condition du 6809 et bien sûr les registres devant contenir des paramètres de retour (le plus souvent B, X et Y).

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

La liste des codes des routines du moniteur est résumée dans le § 15.2.

2.0 GESTION DES INTERRUPTIONS

Les interruptions IRQ et FIRQ du 6809 sont programmables, c'est-à-dire qu'à chacune correspond un registre en RAM contenant l'adresse du programme qui doit les traiter. A la mise sous tension ou après un redémarrage "à chaud", ces registres ont été initialisés avec l'adresse d'un programme du moniteur.

Le moniteur utilise les interruptions IRQ (50 Hertz) pour gérer le clignotement du curseur et la répétition du clavier. Ces interruptions peuvent être aiguillées sur vos propres programmes en mettant dans le registre 2063H une valeur quelconque non nulle, et en mettant dans le registre TIMEPT (2061H-2062H) l'adresse de votre programme de traitement.

Pour gérer les FIRQ, vous devez mettre l'adresse de votre programme dans FIRQPT (2067H-2068H).

Attention cependant: certaines routines du moniteur sont interruptibles, et vos programmes ne doivent pas modifier leurs paramètres.

3.0 INITIALISATION

A la mise sous tension et après un "reset", le moniteur exécute un certain nombre de tests et d'initialisations:

3.1 Test de démarrage à chaud ou à froid

Ce test est effectué pour ne pas modifier, en cas de redémarrage:

* le réglage du crayon optique,

* le mot de code de mise en mode graphique, spécifique de l'imprimante utilisée,

* la durée de la latence du clavier.

Et dans les deux cas:

Les registres d'aiguillage d'interruptions sont initialisés avec les adresses des routines moniteur correspondantes.

Les registres contenant les adresses des différentes tables (table de décodage du clavier, générateur de caractères standard, générateur de caractères utilisateur) sont réinitialisés de manière à pointer sur les tables standard.

Tous les autres registres sont remis aux valeurs standard (le plus souvent remis à zéro).

3.2 Initialisation du PIA système et du PIA jeux

Le PIA système est initialisé comme suit:

* port A:

  - entrée: bouton du crayon optique, lecture cassette.
  - sortie: sélection de la mémoire écran, écriture cassette, couleur du tour de l'écran.

* port B:

  - entrée: sortie clavier.
  - sortie: son, matriçage/dematriçage clavier.

* registre de contrôle:

  - entrée: crayon optique, interruptions 50 Hertz.
  - sortie: moteur du LEP, commande d'inscrustation.

Les ports de données du PIA jeux sont initialisés en entrée.

4.0 GENERATEURS DE CARACTERES

4.1 Alphabet standard G0

Le générateur de caractères G0 est une suite de caractères affichables, correspondant au standard ASCII.

Chaque caractère est composé de 8 octets, qui forment une matrice de 8x8 points représentant le caractère. Le premier octet du caractère correspond à la ligne inférieure de la matrice.

Exemple: matrice définissant le "A"

```
0 0 0 0 0 0 0 0
0 0 0 1 1 0 0 0
0 0 1 0 0 1 0 0
0 1 0 0 0 0 1 0
0 1 1 1 1 1 1 0
0 1 0 0 0 0 1 0
0 1 0 0 0 0 1 0
0 0 0 0 0 0 0 0
```

Dans le générateur, A sera donc défini par la liste d'octets suivants: 00H, 42H, 42H, 7EH, 42H, 24H, 18H, 00H

L'adresse du début du générateur G0 est contenue en RAM dans le registre GENPTR = 2073H-2074H. Vous pouvez redéfinir l'alphabet standard en créant votre propre générateur selon le principe énoncé ci-dessus, puis en chargeant le registre GENPTR avec l'adresse de ce générateur.

4.2 Alphabet G2

L'alphabet G2 contient 5 caractères, qui sont les accents aigu, grave et circonflexe, le tréma et la cédille. Ces caractères sont définis suivant le principe décrit ci-dessus, et destinés à être combinés avec des caractères minuscules.

4.3 Caractères utilisateur

Outre le registre contenant l'adresse du générateur standard, le registre USERAF = 2070H-2071H est destiné à contenir l'adresse de début d'un générateur de caractères utilisateur. Ce générateur sera organisé comme expliqué ci-dessus, 8 octets étant nécessaires pour définir un caractère. Quant aux codes, ils seront pris séquentiellement à partir de 80H. Cela veut dire que le code 80H correspondra au caractère défini par les 8 premiers octets, le code 81H au caractère défini par les 8 octets suivants, et ainsi de suite jusqu'au code FFH correspondant aux 8 derniers octets du générateur. Vous pouvez ainsi définir 128 caractères en plus de l'alphabet standard. L'adresse contenue par défaut dans le registre USERAF est celle du générateur G0.

5.0 PRIMITIVES DE GESTION D'ECRAN

Le MO5 peut gérer un écran pleine page de 320x200 points, avec 16 couleurs de forme, 16 couleurs de fond, et 16 couleurs de tour. En mode alphanumérique, il peut afficher 25 lignes de 40 caractères, définis par une matrice de 8x8 points, et ce dans une couleur au choix parmi 16, quelle que soit la couleur du fond. En mode graphique, l'unité centrale contrôle 8000 segments de 8 points chacun, et à chaque segment correspondent deux octets, situés à la même adresse logique. L'un est situé en mémoire dite "caractère", l'autre en mémoire dite "couleur", un bit de PIA permettant d'adresser physiquement l'une ou l'autre mémoire.

Dans un segment, un point peut prendre au choix deux couleurs: une couleur dite de "forme", auquel cas le bit correspondant dans l'octet en mémoire caractère est à 1, et une couleur dite de "fond", auquel cas le bit correspondant en mémoire caractère est à 0. L'octet se trouvant à la même adresse mais en mémoire "couleur" complète cette information: les 4 bits de poids fort donnent la couleur de la "forme", c'est-à-dire la couleur que devront prendre les points mis à 1, et les 4 bits de poids faible donnent la couleur du "fond", c'est-à-dire la couleur que devront prendre les points mis à 0. Les 4 bits donnant la couleur sont pBVR où p est le bit de pastel (couleurs claires, sauf pour le orange) et où B, V, R indiquent la présence de bleu, de vert, ou de rouge (les trois couleurs de base pour la T.V.) dans la couleur.

Il est donc parfois imprécis de parler de points "allumés" ou "éteints": l'on peut avoir des points clairs appartenant au "fond" (donc de codage 0) et des points foncés appartenant à la "forme" (donc de codage 1).

Pour mémoire, sachez que:

Rouge + vert = jaune

Rouge + bleu = magenta

Bleu + Vert = cyan

Rouge + vert + bleu = blanc

et bien sûr ni rouge, ni bleu, ni vert donne du noir.

Exemple: 1000

R = 0: pas de rouge

V = 0: pas de vert

B = 0: pas de bleu

p = 1: couleur claire

Le résultat est "noir clair", soit gris...

Autre exemple: 1111

R = 1: rouge présent

V = 1: vert présent

B = 1: bleu présent

p = 1: couleur claire

Le résultat devrait être "blanc clair". Cette couleur faisant double emploi avec le blanc, une exception a été faite: c'est le orange.

Le MO5 offre également la possibilité d'incrustation de l'image T.V. dans l'image MO5: si votre MO5 est équipé de l'extension incrustation, la mise en mode "incrustation" (expliquée au § 5.4.4) laissera apparaître l'image T.V. à la place de toutes les zones noires, comme si la couleur noire devenait transparente.

5.1 Mise en mémoire couleur

* code du point d'entrée: 04 pour un JSR, 84H pour un JMP.

Cette routine vous permet d'aller sélectionner la mémoire couleur: tout ce que vous écrirez entre l'adresse 0 et l'adresse 1FFFH modifiera la couleur des points correspondants.

5.2 Mise en mémoire caractère

* code du point d'entrée: 06 pour un JSR, 86H pour un JMP.

Cette routine vous permet d'aller sélectionner la mémoire caractère: tout ce que vous écrirez entre l'adresse 0 et l'adresse 1FFFH modifiera l'état (fond ou forme) des points correspondants.

5.3 Bip sonore

* code du point d'entrée: 08 pour un JSR, 88H pour un JMP.

* paramètre d'entrée:

- registre STATUS (2019H)

Cette routine teste l'état du bit 3 du registre STATUS. S'il est à 0, elle génère le bip sonore, sinon elle retourne sans rien faire.

5.4 Gestion de caractères alphanumériques

* code du point d'entrée: 02 pour un JSR, 82H pour un JMP

* paramètre d'entrée:

- registre 6809 B

5.4.1 Séquence normale

5.4.1.1 Codes affichables

Les codes affichables sont compris entre 20H et 7FH. Le caractère est affiché à la position courante du curseur. Si vous avez modifié le contenu du pointeur sur le générateur de caractères standard, les caractères affichés seront ceux que vous avez définis (entre 20H et 7FH).

5.4.1.2 Codes interprétables

Les codes interprétables sont compris entre 7 et 1FH. A la réception du code, l'une des opérations suivantes est effectuée:

- 07 : BEL--> "Bip" sonore.
- 08 : BS --> Déplacement d'une position vers la gauche ou recopie à gauche du caractère courant, si le registre COPCHR (202FH) contient la valeur FFH.
- 09 : HT --> Déplacement d'une position vers la droite ou recopie à droite du caractère courant, si le registre COPCHR (202FH) contient la valeur FFH.
- 0AH : LF --> Descente d'une ligne.
- 0BH : VT --> Remontée d'une ligne.
- 0CH : FF --> Effacement de toute la fenêtre.
- 0DH : CR --> Retour au début de la ligne courante.
- 0EH : SO --> Rien.
- 0FH : SI --> Rien.
- 10H : DLE--> Rien.
- 11H : DC1--> Allumage du curseur.
- 12H : DC2--> Rien.
- 13H : DC3--> Rien.
- 14H : DC4--> Extinction du curseur.
- 15H : NAK--> Rien.
- 16H : ACC--> Séquence accent.
- 17H : ETB--> Rien.
- 18H : CAN--> Effacement de la fin de la ligne à partir de la position courante du curseur.
- 19H : EM --> Rien.
- 1AH : SUB--> Rien.
- 1BH : ESC--> Séquence "d'échappement".
- 1CH : FS --> Rien.
- 1DH : GS --> Rien.
- 1EH : RS --> Retour du curseur dans le coin supérieur gauche de la fenêtre courante.
- 1FH : US --> Séquence de positionnement curseur ou de définition de fenêtre.

5.4.1.3 Caractères utilisateur

Les codes compris entre 80H et FFH sont ceux des caractères utilisateur. Avant d'appeler la routine pour faire afficher le caractère correspondant au code, vous devrez avoir chargé le registre USERAF avec l'adresse de votre générateur de caractères, comme expliqué au § 4.3.

5.4.2 Séquence de curseur ou de fenêtre

Le code 1FH définit une séquence de positionnement du curseur ou de définition de la fenêtre. Trois appels à la routine sont nécessaires, et peuvent être schématisés comme suit: US/N1/N2.

5.4.2.1 Positionnement du bas de la fenêtre (N1,N2 C <10H,19H>)

Cette séquence définit le bas de la fenêtre courante, où le numéro de ligne est égal à 10 x n1 + n2, n1 et n2 (chiffre des dizaines et chiffre des unités) étant respectivement les quatre bits de poids faible de N1 et N2.

EXEMPLE:

Pour définir le bas de la fenêtre en ligne 21, l'on doit envoyer dans B:

1° appel: code 1FH dans B

2° appel: code 12H dans B

3° appel: code 11H dans B

5.4.2.2 Positionnement du haut de la fenêtre (N1,N2 C <20H,29H>)

Cette séquence définit le haut de la fenêtre courante, où le numéro de ligne est égal à 10 x n1 + n2, n1 et n2 étant respectivement les quatre bits de poids faible de N1 et N2.

5.4.2.3 Positionnement du curseur (N1,N2 C <40H,70H>)

Le curseur est positionné en ligne n1, colonne n2, où n1 et n2 sont respectivement les 6 bits de poids faible de N1 et N2, avec n1 compris entre 0 et 24 et n2 compris entre 1 et 40.

EXEMPLE:

Pour positionner le curseur en ligne 3, colonne 20, il faudra mettre successivement dans B avant chaque appel:

- 1° appel: code 1FH dans B
- 2° appel: code 43H dans B
- 3° appel: code 54H dans B

5.4.3 **Séquence accent**

La séquence ACC est utilisée pour permettre l'affichage d'une minuscule accentuée. On peut schématiser cette séquence comme suit: ACC/CODE/L.

1° appel: envoyer ACC, code d'une séquence accent (16H)

2° appel: envoyer CODE qui est le code de l'accent :

41H pour l'accent aigu

42H pour l'accent grave

43H pour l'accent circonflexe

48H pour le tréma

4BH pour la cédille

3° appel: envoyer L qui est le code de la minuscule à accentuer. Si ce code ne correspond pas à un code de minuscule, alors la lettre dont le code est L est affichée, et l'accent est effacé.

5.4.4 **Séquence "d'échappement"**

Cette séquence est utilisée pour positionner les attributs de couleur de l'écran, et d'autres attributs de la vidéo comme la taille des caractères, la mise en mode incrustation pour les MO5 équipés de l'extension correspondante, le type de scroll, etc... Ces attributs peuvent être de deux types: courant ou plein écran. Dans le cas d'attributs de type courant, la séquence est de la forme: ESC/ATT, et dans le cas d'attributs de type plein écran, la séquence est de la forme: ESC/ESPACE/ATT où ESC est le code de la séquence d'échappement, ESPACE le code de la barre d'espacement (20H), et ATT le code de l'attribut. Les codes d'attributs sont identiques, qu'ils soient de type courant ou plein écran, mais certains attributs (couleur du tour, attributs divers sauf la vidéo inverse) n'ont pas de signification en plein écran, et sont alors ignorés.

5.4.4.1 Attributs de couleur

Si ATT compris entre 40H et 6FH, la zone de l'écran qui sera modifiée est sélectionnée par les 4 bits de poids fort de l'octet "ATT", tandis que les 4 bits de poids faible sélectionnent l'une des 16 couleurs MO5.

* 4 bits de poids fort:

  4 --> modification de la couleur de la forme.

  5 --> modification de la couleur du fond.

  6 --> modification de la couleur du tour.

* 4 bits de poids faible:

  00 --> noir

  01 --> rouge

  02 --> vert

  03 --> jaune

  04 --> bleu

  05 --> magenta

  06 --> cyan

  07 --> blanc

  08 --> gris

  09 --> rouge clair

  10 --> vert clair

  11 --> sable

  12 --> bleu clair

  13 --> rose

  14 --> bleu ciel

  15 --> orange

EXEMPLE:

Après la séquence suivante, vous obtiendrez un fond orange:

```
LDB     #1BH    code ESC
SWI
FCB     02
LDB     #5FH    5 pour le fond, F pour l'orange.
SWI
FCB     02
```

5.4.4.2 Attributs divers

Si ATT est compris entre 70H et 7FH, il a la signification suivante:

70H --> caractères en taille normale.

71H --> double hauteur, largeur normale.

72H --> double largeur, hauteur normale.

73H --> double taille.

74H --> écriture du caractère dans la couleur courante.

75H --> écriture du caractère sans modifier la couleur.

76H --> suppression de l'incrustation.

77H --> mise en mode incrustation.

78H --> scroll à vitesse normale.

79H --> scroll doux.

7AH --> mode page (pas de scroll).

7BH --> inversion de la vidéo.

Si un code supérieur à 7BH est envoyé, tout se passera comme si l'on avait envoyé le code 7BH. Si l'on envoie un code inférieur à 40H, rien ne se passera, de même que si l'on envoie dans une séquence plein écran le code d'un attribut qui n'a de sens qu'en tant qu'attribut courant.

EXEMPLE:

Après la séquence suivante, et si le MO5 est équipé de l'extension incrustation, toutes les zones noires de l'écran seront remplacées par l'image T.V.

```
LDB     #1BH    code de la séquence d'échappement.
SWI
FCB     02
LDB     #77H    code de la mise en mode incrustation.
SWI
FCB     02
```

6.0 CLAVIER

6.1 Lecture rapide du clavier

* code du point d'entrée: OCH pour un JSR, 8CH pour un JMP.

* paramètres de retour:

- registres 6809 A, B et CC

Cette routine effectue une lecture rapide du clavier, pour tester si une touche a été enfoncée ou non. Si aucune touche n'a été enfoncée, le bit Z du registre CC est mis à 1, sinon il est mis à 0. Dans ce cas, B contient le code de la touche enfoncée, et l'état des bits 0, 1 et 2 de A indique si la touche jaune, ou BASIC, ou CNT a été enfoncée:

- bit0: touche BASIC

- bit1: touche CNT

- bit2: touche JAUNE

Si le bit correspondant à la touche est à 1, la touche a été enfoncée, sinon le bit est à 0.

6.2 Décodage du clavier

* code du point d'entrée: 0AH pour un JSR, 8AH pour un JMP.

* paramètres d'entrée:

- registres CHRPTR (206DH-206EH), STATUS (2019H), LATCLV (2076H)

* paramètres de retour:

- registres 6809 B et CC

- registre KEY (2037H)

Cette routine effectue le décodage du clavier, et gère la répétition d'une touche après un certain délai. Ce délai est programmable via le registre LATCLV: en modifiant le contenu de ce registre, vous obtenez une latence du clavier. De la même facon, la répétition clavier peut être inhibée en chargeant ce registre avec une valeur suffisamment élevée. Le bip sonore qui se fait entendre chaque fois qu'une touche est frappée peut être inhibé lui-aussi, de deux façons: la première, classique, consiste à baisser le son de votre récepteur T.V., la seconde à forcer à 1 le bit 3 du registre STATUS.

B retourne soit le code ASCII du caractère, soit le code numérique de la touche (au cas où la touche BASIC a été enfoncée). Si aucune touche n'a été enfoncée, ou si l'une des trois touches BASIC, JAUNE, ou CNT a été enfoncée seule, ou si plusieurs touches parmi les touches BASIC, JAUNE, ou CNT ont été enfoncées simultanément, B retourne la valeur zéro.

La touche jaune sélectionne le caractère imprimé en jaune sur la touche. La touche CNT force à 0 le bit 6 du code ASCII du caractère imprimé en blanc sur la touche. La touche BASIC retourne le code numérique de la touche (qui dépend du matriçage du clavier), tout en forçant le bit 7 à 1 (codes retournés de 80H à B9H).

L'accès aux minuscules accentuées nécessite en général trois frappes consécutives:

- 1° frappe: touche accent.

- 2° frappe: touche jaune en même temps que l'accent.

- 3° frappe: lettre minuscule.

Cependant, l'accès à certaines minuscules accentuées couramment utilisées en Français peut être fait en deux touches:

* ACC/6: é

* ACC/7: è

* ACC/8: ù

* ACC/9: ç

* ACC/0: à

Mais dans tous les cas, vous devrez faire trois appels consécutifs à cette routine:

- le 1° appel retourne dans B le code de la touche ACC, soit 16H.

- le 2° appel retourne dans B le code de l'accent ou de la cédille.

- le 3° appel retourne dans B le code de la minuscule.

Remarque: bien que ACC suivi de 0 produise un à, et ACC suivi de touche jaune / 0 produise un accent grave, la frappe de touche jaune / 0 non précédée de la frappe de ACC retournera 0 dans B (pas de caractère connu), car la quote inverse n'est pas reconnue par le M05.

Le code ASCII du caractère est lu dans une table dont l'adresse se trouve dans le registre CHRPTR. Si vous voulez redéfinir votre clavier, il suffira de mettre dans ce registre l'adresse de la table contenant vos propres codes ASCII, mais les séquences "accent" peuvent donner des résultats imprévisibles ...

Voici la liste des valeurs retournées quand la touche BASIC est enfoncée simultanément avec une touche autre que la touche jaune ou CNT:

CODE RETOURNE	TOUCHE ENFONCEE
82H	STOP
83H	ACC
85H	ENTREE
86H	RAZ
87H	C
88H	^
89H	W
8AH	1
8BH	+
8CH	A
8DH	*
8EH	Q
8FH	V
90H	<
91H	X
92H	2
93H	-
94H	Z
95H	/
96H	S
97H	B
98H	v
99H	Barre d'espacement
9AH	3
9BH	O
9CH	E
9DH	P
9EH	D
9FH	M
AOH	>
A1H	
A2H	4
A3H	9
A4H	R
A5H	O
A6H	F
A7H	L
A8H	
A9H	.
AAH	5
ABH	8
ACH	I
ADH	I
AEH	G
AFH	K
BOH	INS
B1H	,
B2H	6
B3H	7
B4H	Y
B5H	U
B6H	H
B7H	J
B8H	EFF
B9H	N

7.0 PRIMITIVES GRAPHIQUES

7.1 Allumage ou extinction d'un point

* code du point d'entrée: 10H pour un JSR, 90H pour un JMP.

* paramètres d'entrée:

- registres 6809 X et Y

- registres FORME (2029H), CHDRAW (2036H), COLOUR (202BH)

* paramètres de retour:

- registres PLOTX (2032H-2033H), PLOTY (2034H-2035H)

7.1.1 Mode graphique

Cette routine met en couleur le point de coordonnées passées par X et Y, où X est compris entre 0 et 319 et Y entre 0 et 199. La couleur du point est déterminée par le contenu du registre FORME, qui doit être compris entre -16 et +15. Si le code de la couleur est négatif, le point sera écrit en "fond", c'est-à-dire que le bit correspondant dans l'octet en mémoire "caractère" sera mis à zéro. Si le code est positif, le point sera écrit en "forme", c'est-à-dire que le bit correspondant dans l'octet en mémoire "caractère" sera mis à 1. Si le bit 4 du registre STATUS est à 1, seul le bit en mémoire caractère sera modifié, la couleur ne sera pas changée.

Voici les codes des couleurs "forme" ou "fond":

COULEUR	CODE "FORME"	CODE "FOND"
NOIR	0	-1
ROUGE	1	-2
VERT	2	-3
JAUNE	3	-4
BLEU	4	-5
MAGENTA	5	-6
CYAN	6	-7
BLANC	7	-8
GRIS	8	-9
ROSE	9	-10
VERT CLAIR	10	-11
SABLE	11	-12
BLEU CLAIR	12	-13
PARME	13	-14
BLEU CIEL	14	-15
ORANGE	15	-16

Un code de couleur "fond" se déduit donc d'un code de couleur "forme" en ajoutant 1 et en prenant l'opposé.

Le registre CHDRAW doit être mis à zéro, sinon l'on est dans le mode caractère, explicité ci-dessous.

7.1.2 Mode caractère

Si le contenu du registre CHDRAW est non nul, il est supposé être le code ASCII d'un "point caractère" à afficher à l'endroit défini par les registres X et Y. Dans ce cas, on doit avoir X C <1,40> et Y C <0,24>.

La couleur du "point-caractère" est fournie non plus par le registre FORME, mais par le registre COLOUR qui contient les couleurs courantes de "fond" et de "forme".

On peut accéder directement à l'écriture du caractère du registre CHDRAW en appelant la routine CHPL$ de numéro 12H pour un JSR ou 92H pour un JMP.

Dans les deux modes, les registres X et Y seront recopiés respectivement dans les registres PLOTX et PLOTY qui gardent l'abscisse et l'ordonnée du dernier point "allumé".

7.2 Tracé d'un segment de droite

* Code du point d'entrée: 0EH pour un JSR, 8EH pour un JMP.

* paramètres d'entrée:

- registres 6809 X et Y

- registres PLOTX (2032H-2033H), PLOTY (2034H-2035H), FORME (2029H), COLOUR (202BH), CHDRAW (2036H)

* paramètres de retour:

- registres PLOTX (2032H-2033H), PLOTY (2034H-2035H)

7.2.1 Mode graphique

Cette routine trace un segment de droite entre le point défini par son abscisse en PLOTX et son ordonnée en PLOTY (dernier point allumé si vous n'avez pas modifié ces registres depuis), et le point dont les coordonnées sont passées par X et Y, avec X compris entre 0 et 319 et Y entre 0 et 199. La couleur du segment est définie par le contenu du registre FORME, suivant les conventions explicitées ci-dessus.

Si le bit 4 du registre STATUS est mis à 1, la couleur ne sera pas mise à jour. Le contenu du registre CHDRAW doit être nul, sinon le segment tracé est un "segment caractère".

7.2.2 Mode caractère

Si le contenu du registre CHDRAW n'est pas nul, il est interprété comme le code ASCII du caractère servant au tracé. Dans ce cas, la couleur est la couleur courante donnée par le registre COLOUR, et les coordonnées X et Y doivent être comprises respectivement entre 1 et 40 et 0 et 24.

Dans les deux modes, les registres X et Y seront recopiés respectivement dans les registres PLOTX et PLOTY qui gardent l'abscisse et l'ordonnée du dernier point allumé, ce qui permet de tracer un contour polygonal sans avoir à préciser à chaque fois le point de départ.

7.3 Segments horizontaux

Dans le cas où le segment à tracer est horizontal, un algorithme de tracé rapide est mis en oeuvre, ce qui peut être utile pour remplir des contours par une série de segments horizontaux.

7.4 Lecture de la couleur d'un point

* code du point d'entrée: 14H pour un JSR, 94H pour un JMP.

* paramètres d'entrée:

- registres 6809 X et Y

* paramètre de retour:

- registre 6809 B

Cette routine retourne dans l'accumulateur B la couleur d'un point : de -16 à -1 si le point est en "fond", et de 0 à +15 si le point est en "forme". Les coordonnées du point sont passées par X compris entre 0 et 319, et Y compris entre 0 et 199. La couleur est codée comme décrit au § 7.1.1

8.0 LECTURE DE L'ECRAN

* code du point d'entrée: 1AH pour un JSR, 9AH pour un JMP

* paramètres d'entrée:

- registres 6809 A et X

- registre SS3GET (205DH)

* paramètres de retour:

- registre 6809 B

- registres SS2GET (205CH), SS3GET (205DH)

Cette routine retourne dans l'accumulateur B le code ASCII du caractère dont les coordonnées sont passées par X compris entre 1 et 40, et A compris entre 0 et 24.

S'il n'y a pas de caractère connu en (X,A) alors B retourne la valeur zéro. Sinon, deux cas peuvent se produire:

8.1 Caractère normal

Si le caractère appartient à l'alphabet G0, alors B retourne le code ASCII du caractère reconnu, et les registres SS2GET et SS3GET sont remis à zéro.

8.2 Minuscule accentuée ou ç

Dans ce cas, trois appels à la routine sont nécessaires:

- 1º appel: une minuscule accentuée est détectée: B retourne le code ACC (accent), soit 16H, le registre SS3GET est chargé avec le code de l'accent (qui appartient à l'alphabet G2), et le registre SS2GET est chargé avec le code ASCII de la minuscule.

- 2º appel: B retourne le code de l'accent (qui se trouve maintenant dans le registre SS3GET), le registre SS3GET est chargé avec la valeur 80H, et le registre SS2GET n'est pas modifié.

- 3º appel: B retourne le code ASCII de la minuscule (qui se trouve dans le registre SS2GET), et les registres SS2GET et SS3GET sont remis à zéro.

ATTENTION: cette routine ne fonctionne correctement qu'avec l'alphabet standard G0. Si vous avez changé le contenu du registre GENPTR, vous devez restituer son ancien contenu avant d'appeler la routine, c'est-à-dire le charger avec l'adresse de début du générateur G0.

9.0 GENERATION DE MUSIQUE

* code du point d'entrée: 1EH pour un JSR, 9EH pour un JMP.

* paramètres d'entrée:

- registre 6809 B

- registres DUREE (203BH-203CH), OCTAVE (203EH-203FH), TIMBRE (203DH), TEMPO (2039H-203AH).

La note à jouer est passée par l'accumulateur B. Il y a 13 notes de base, de DO à UT, plus le silence. Voici la table des valeurs correspondant aux notes:

NOTE:	CODE:
SILENCE	00
DO	01
DO#	02
RE	03
RE#	04
MI	05
FA	06
FA#	07
SOL	08
SOL#	09
LA	0AH
LA#	0BH
SI	0CH
UT	0DH

Il faut ensuite préciser les paramètres suivants: tempo, octave, timbre et durée.

* l'octave: il y a 5 octaves possibles, de l'octave 1 qui est la plus grave, à l'octave 5. L'octave 4 correspond à l'octave du LA 440.

Voici la liste des valeurs à mettre dans le registre OCTAVE:

OCTAVE	VALEUR
1	16
2	08
3	04
4	02
5	01

* la durée: il s'agit de la durée relative de chaque note, pouvant aller de la ronde à la triple croche. La valeur de la durée est à charger dans le registre DUREE. La valeur pour la ronde est 96, et l'on obtient les valeurs relatives en divisant par des puissances de 2, ou de 3 pour des triolets ou les notes pointées. Voici la liste des valeurs:

NOTES:	VALEURS:
RONDE	96
BLANCHE pointée	72
BLANCHE	48
NOIRE pointée	36
NOIRE	24
CROCHE pointée	18
CROCHE	12
DOUBLE CROCHE pointée	09
DOUBLE CROCHE	06
TRIPLE CROCHE pointée	05
TRIPLE CROCHE	03

DANS UN TRIOLET:	VALEURS:	
NOIRE	16	(car 3x16 = 48)
CROCHE	08	(car 3x8 = 24)
DOUBLE CROCHE	04	(car 3x4 = 12)
TRIPLE CROCHE	02	(car 3x2 = 06)

* le tempo: c'est le mouvement auquel doit être joué le morceau. La durée réelle d'une note est égale au tempo x la durée. La valeur du tempo, de 1 à 255, doit être chargée dans le registre TEMPO.

* le timbre: cette valeur (de 0 à 5) doit être chargée dans le registre TIMBRE. Le rapport cyclique est modifié en conséquence, ce qui donne à la note une attaque différente. Pour une note continue, il faut mettre la valeur 0 dans le registre TIMBRE.

10.0 LECTURE DES MANETTES DE JEU

* code du point d'entrée: 1CH pour un JSR, 9CH pour un JMP.

* paramètre d'entrée:

- registre 6809 A

* paramètres de retour:

- registres 6809 B et CC

Le numéro (0 ou 1) de la manette de jeu dont on veut connaître l'état est passé par l'accumulateur A.

B retourne une valeur de 0 à 8 donnant l'état de la manette de jeu, suivant la convention que voici:

0 ==> Centre

1 ==> Nord

2 ==> Nord Est

3 ==> Est

4 ==> Sud Est

5 ==> Sud

6 ==> Sud Ouest

7 ==> Ouest

8 ==> Nord Ouest

Le bit de retenue du registre CC est mis à 1 si le bouton a été enfoncé, sinon il est mis à zéro.

11.C CRAYON OPTIQUE

11.1 Test du bouton du crayon optique

* code du point d'entrée: 16H pour un JSR, 96H pour un JMP.

* paramètre de retour:

- registre 6809 CC

Cette routine teste l'état du bouton du crayon optique. Si celui-ci a été enfoncé, le bit de retenue est forcé à 1, sinon il est forcé à 0.

11.2 Lecture de la zone pointée

* code du point d'entrée: 18H pour un JSR, 98H pour un JMP.

* paramètres de retour:

- registres 6809 X,Y et CC

Cette routine lit les coordonnées du point visé par le crayon optique, et retourne l'abscisse dans X (de 0 a 319) et l'ordonnée dans Y (de 0 a 199). Si la mesure est correcte, le bit de retenue du registre CC est forcé à zéro. En cas de mauvaise lecture (luminosité trop faible, crayon ne pointant pas l'écran), ce bit est forcé à 1.

12.0 GESTION DE L'INTERFACE DE COMMUNICATION

* code du point d'entrée: 24H pour un JSR, A4H pour un JMP.

* paramètres d'entrée:

- registre 6809 B

- registres PR.OPC (2042H) et GRCODE (2077H)

* paramètres de retour:

- registre 6809 CC

- registres PR.STA (2043H)

Cette routine gère l'interface de communication. Le contenu du registre sélectionne l'une des opérations suivantes:

CONTENU DE PR.OPC	OPERATION DEMANDEE
% 0 0 0 0 0 0 0 1	Ecriture d'un octet en parallèle.
% 0 0 0 0 0 0 1 0	Copie graphique d'écran.
% 0 0 0 0 0 1 0 0	Ouverture pour écriture en parallèle.
% 0 0 0 1 0 0 0 0	Fermeture.

Dans le cas d'un octet à écrire en parallèle, B doit contenir l'octet à envoyer.

En cas de copie graphique de l'écran, le registre GRCODE doit contenir le code de mise en mode graphique spécifique de l'imprimante. Ce registre contient la valeur 7 par défaut.

Le registre PR.STA retourne le code de l'opération réalisée :

CONTENU DE PR.STA	ETAT DE LA COMMUNICATION
% 0 0 0 0 0 1 0 0	Ouvert pour écriture en parallèle.
% 0 0 0 0 1 0 0 0	Périphérique non prêt
% 0 0 0 1 0 0 0 0	Fermé.

Le bit de retenue du registre CC est forcé à zéro si tout s'est passé normalement, sinon il est forcé à 1.

13.C GESTION DU LECTEUR-ENREGISTREUR DE PROGRAMMES (LEP)

13.1 Lecture et écriture de la bande

* code du point d'entrée: 20H pour un JSR, A0H pour un JMP.

* paramètres d'entrée;

- registres 6809 A, B et Y

- registres K7DATA (2040H) et K7LENG (2041H)

* paramètres de retour:

- registres 6809 A et B.

Cette routine permet de lire ou d'écrire des blocs de 253 octets de données au maximum. Ces blocs peuvent être de trois types, le type étant codé de la facon suivante:

- 00: bloc d'en-tête du fichier.

- 01: bloc de données.

- FFH: bloc de fin de fichier.

Si A est non nul, la routine effectue une lecture de la bande magnétique. Y doit contenir l'adresse de la zone mémoire où sera rangé le bloc lu. Au retour, B contiendra le type de bloc, et A le "checksum" calculé d'après les données lues (somme des données et du checksum lu). A l'adresse pointée par Y, l'on trouvera:

- un octet donnant la longueur du bloc.

- n octets de données.

- un octet donnant le "checksum" lu.

Si A = 0, il s'agit d'une écriture. B doit contenir le type du bloc, et Y doit contenir l'adresse de la zone mémoire où se trouvent les données à écrire sur la bande. Cette zone doit avoir la structure suivante:

1º octet: longueur du bloc = nombre d'octets de données +2.

2º octet: première donnée.

(Longueur du bloc - 2)données

.

nº octet: (n-1)º donnée.

n+1º octet: checksum de contrôle (opposé de la somme des données si l'on veut en cas de lecture valide avoir un checksum lu qui soit nul).

13.2 Moteur: marche/arrêt

* code du point d'entrée: 22H pour un JSR, A2H pour un JMP.

* paramètre d'entrée:

- registre 6809 A

* paramètre de retour:

- registre 6809 CC

Si le bit 0 de A est à zéro, le moteur est arrêté après une demi-seconde de délai.

Si le bit 0 et le bit 1 de A sont chacun à 1, le moteur est mis en route, et il y a ensuite une seconde de délai.

Si le bit 0 de A est à 1, et le bit 1 de A est à zéro, le moteur est mis en route mais il n'y a pas d'attente ensuite.

La fiabilité sera augmentée si vous suivez les conseils que voici:

* mise en route du moteur pour une écriture: lancement du moteur, puis attente d'une seconde, afin d'être sûr que la vitesse du moteur s'est stabilisée avant d'écrire sur la bande. A doit alors contenir la valeur %xxxxxx11.

* mise en route du moteur pour une lecture: lancement du moteur, pas d'attente. A doit alors contenir la valeur %xxxxxx01.

* arrêt du moteur après une lecture ou une écriture: attente d'une demi-seconde pour être sûr qu'aucune partie de la bande contenant des données ne se trouvera sous le cabestan, puis arrêt du moteur. A doit alors contenir la valeur %xxxxxx10.

Si le LEP n'est pas raccordé à votre MO5, le bit de retenue du registre code condition sera forcé à 1, sinon il sera forcé à zéro.

14.0 CONTROLEUR DE DISQUETTES

Le MO5 est entièrement compatible avec les disquettes TO7, simple ou double densité. Vous pouvez gérer les entrées/sorties disquette à deux niveaux: au niveau physique en utilisant le point d'entrée du moniteur, ou à un niveau logique en manipulant des fichiers au format Basic Microsoft(R).

Rappelons que les disquettes sont divisées en 40 pistes de 16 secteurs chacune. Un secteur contient lui-même 128 octets en simple densité, ou 256 en double densité.

De plus, le contrôleur double densité peut fonctionner dans les 2 modes simple ou double densité. Au démarrage, sa densité de travail doit être sélectionnée après l'initialisation du contrôleur.

14.1 Entrées/sorties disquette physiques

* code du point d'entrée: 26H pour un JSR, A6H pour un JMP.

* paramètres d'entrée:

- registres DK.OPC (2048H), DK.DRV (2049H), DK.SEC (204CH), DK.TRK (204AH-204BH), DK.BUF (204FH-2050H).

* paramètres de retour:

- registre 6809 CC

- registre DK.STA (204EH)

Le registre DK.OPC contient le code de l'opération à réaliser:

- Code 01: demande l'initialisation du contrôleur. Dans ce cas, si l'initialisation a pu avoir lieu sans erreurs, le bit de retenue du registre code condition est mis à 0, et le type de contrôleur est retourné dans le registre DK.STA: "C" pour la simple densité, et "D" pour la double densité. Sinon, le bit de retenue est mis à 1, et le code d'erreur 40H est mis dans le registre DK.STA (voir ci-après les codes d'erreur).

- Code 02: lecture d'un secteur. Les codes d'erreur possibles sont 02, 04, 08, 10H, 80H. Si une erreur a eu lieu, son code est retourné dans le registre DK.STA.

- Code 04: sur lecteur double densité, passage du contrôleur en simple densité, pas d'erreur possible. Erreur sur un controleur simple densite.

- Code 08: écriture d'un secteur. Les codes d'erreur possibles sont 01, 02, 04, 08, 10H, 20H, 80H. Si une erreur a eu lieu, son code est retourné dans le registre DK.STA.

- code 10H: sur lecteur double densité, passage du contrôleur en double densité, pas d'erreur possible. Erreur sur un controleur simple densite.

- Code 20H: recherche de la piste zéro. Les codes d'erreur possibles sont 10H et 80H.

- Code 40H: recherche de la piste dont le numéro est passé par le registre DK.TRK. Les codes d'erreur possibles sont 10H et 80H.

- Code 80H: option de vérification. Il faut faire un "OR" logique entre ce code et le code de l'opération que l'on veut vérifier. Les codes d'erreur retournés sont les mêmes que ceux de l'opération spécifiée augmentés du code 20H.

Paramètres à passer suivant l'opération demandée:

* registre DK.DRV: ce registre doit contenir le numéro du lecteur concerné, soit une valeur entre 0 et 3.

* registre DK.TRK: ce registre doit contenir 39, numéro de piste maximum autorisé, les 40 pistes étant numérotées de 0 à 39.

* registre DK.SEC: ce registre doit contenir le numéro de secteur où l'on veut lire ou écrire. Ce numéro doit être compris entre 1 et 16.

* registre DK.BUF: ce registre doit contenir l'adresse de début d'une zone tampon en RAM de 128 octets en simple densité, ou de 256 octets en double densité, soit pour y lire les données à écrire sur disquette, soit pour y écrire les données lues sur disquette.

* registre DK.STA: ce registre contient le code d'erreur, ou le type de contrôleur pour une initialisation correcte.

Voici les différentes erreurs possibles:

- code 01: disquette protégée; cette erreur ne peut apparaître qu'après une demande d'écriture.

- code 02: erreur de piste; l'identificateur de piste est correct, mais ne correspond pas à la piste demandée.

- code 04: erreur de secteur; l'identificateur de secteur est incorrect (secteur ne pouvant être lu ou erreur sur le checksum), cependant la piste peut être correcte.

- code 08: erreur sur les données; l'identificateur de secteur est correct, mais les données ne peuvent être lues, ou le checksum est incorrect.

- code 10H: lecteur non prêt; le moteur n'est pas en route, ou le lecteur spécifié est inexistant.

- code 20H: erreur sur vérification; la zone tampon en mémoire et la zone correspondante sur la disquette ne sont pas identiques.

- code 40H: contrôleur non prêt.

- code 80H: disquette non formattée. L'identificateur de piste ne peut être lu.

14.2 Le format Basic Microsoft(R)

Pour assurer la compatibilité entre les diverses applications, les fichiers créés par le MO5 suivent le standard Basic Microsoft(R).

La piste 20 est une zone réservée destinée à rendre compte de l'état de la disquette. Elle est organisée comme suit:

* secteur 1: réservé

* secteur 2: table d'allocation des fichiers ou "FAT"

* secteurs 3 à 16: catalogue.

14.2.1 La table d'allocation des fichiers

Les fichiers sont organisés en blocs de 1K octets en simple densité, ou 2K octets en double densité. L'on a donc dans tous les cas 2 blocs par piste. Les blocs sont numérotés à partir de 0. Chaque octet de la table d'allocation des fichiers, à partir de l'octet 1, représente un bloc physique.

Organisation de la "FAT":

* Octet 0: 0

* Octet 1: bloc 0, piste 0, secteurs 1 à 8.

* Octet 2: bloc 1, piste 0, secteurs 9 à 16.

* Octet 3: bloc 2, piste 1, secteurs 1 à 8.

* Octet 4: bloc 3, piste 1, secteurs 9 à 16.

* .....

* Octet 2j-1: bloc 2j-2, piste j-1, secteurs 1 à 8.

* Octet 2j: bloc 2j-1, piste j-1, secteurs 9 à 16.

* .....

* Octet 80: bloc 79, piste 39, secteurs 9 à 16.

Un octet de la "FAT" représentant un bloc physique peut avoir comme valeurs:

* FFH, qui signifie bloc non alloué.

* FEH, qui signifie bloc réservé.

* Tout nombre de 0 à BFH, signifie bloc alloué. Dans ce cas, le nombre représente le numéro du bloc logique suivant du même fichier.

* Tout nombre de C1H à C8H, signifie dernier bloc d'un fichier. Les 4 bits de poids faible indiquent le nombre de secteurs utilisés dans ce dernier bloc.

14.2.2 Le **catalogue**

Le catalogue donne la liste des fichiers, et occupe 14 secteurs. Chaque fichier est répertorié sur 32 octets. Il y a donc 4 fichiers répertoriés par secteur en simple densité, et 8 fichiers par secteur en double densité. Au total, le catalogue peut donc répertorier 56 fichiers en simple densité, et 112 en double densité.

Chaque fichier est répertorié de la facon suivante:

Octets 00 à 07: Nom du fichier, cadré à gauche, complété par des blancs.

Octets 08 à AH: Suffixe du fichier (.BAS, .BIN, etc...), cadré à gauche, complété par des blancs.

Octet 0BH: Type de fichier: 0 pour un programme Basic ASCII ou binaire, 1 pour des données Basic en ASCII, 2 pour un programme en langage machine (binaire), 3 pour un fichier assembleur édité en ASCII.

Octet 0CH: Sémaphore: FFH pour de l'ASCII, 00 pour du binaire.

Octet 0DH: Numéro du premier bloc logique du fichier.

Octets 0EH-0FH: Nombre d'octets utilisés dans le dernier secteur du fichier.

Octets 10H-1FH: Réservés.

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

* 00: entrée non allouée, pas de fichier répertorié pour cette entrée.

* 20H-7FH: code ASCII du premier caractère du nom de fichier, donc entrée allouée.

* FFH: fin logique du catalogue.

Lors de la création du catalogue, ses octets sont mis à FFH. Chaque fois qu'un fichier est créé, la fin logique du catalogue est déplacée dans le premier octet de l'entrée suivante, jusqu'à ce que le catalogue soit plein. Lorsqu'un fichier est détruit, le premier octet de son entrée est mis à zéro (entrée non allouée). Dans ce cas, tout fichier nouvellement créé se verra attribuer en priorité cette entrée.

15.0 INFORMATIONS COMPLEMENTAIRES

15.1 Organisation de la mémoire

ADRESSES (HEXADECIMAL)	
0000-1FFF	Mémoire écran 2 x 8 K.
2000-20FF	Registres du moniteur.
2100-21FF	Registres de l'application.
2200-9FFF	Mémoire utilisateur env. 32 K
A000-A7BF	1.9 K pour la gestion disquettes.
A7C0-A7C3	PIA 6821 système.
A7C4-A7CB	L I B R E . . .
A7CC-A7CF	PIA 6821 extension jeux.
A7D0-A7DF	Contrôleur de disquettes.
A7E0-A7E3	PIA 6821 interface de communication.
A7E4-A7E7	Compteurs crayon optique.
A7E8-A7FF	Extensions.
A800-AFFF	L I B R E, 2 K.
B000-EFFF	Cartouche ROM, 16 K.
F000-FFFF	Moniteur, 4 K.

15.2 Points d'entrée standard du moniteur

Les codes qui suivent sont les codes sur 7 bits. Le bit 7 doit être mis à zéro pour simuler un JSR, à 1 pour simuler un JMP.

00 : Rend la main à la cartouche assembleur.

02 : Affichage d'un caractère.

04 : Mise en mémoire couleur.

06 : Mise en mémoire forme.

08 : Emission d'un bip sonore.

0AH : Lecture du clavier.

0CH : Lecture rapide du clavier.

0EH : Tracé d'un segment de droite.

10H : Allumage ou extinction d'un point.

12H : Ecriture d'un point "caractères".

14H : Lecture de la couleur d'un point.

16H : Lecture du bouton du crayon optique.

18H : Lecture du crayon optique.

1AH : Lecture de l'écran.

1CH : Lecture des manettes de jeu.

1EH : Génération de musique.

20H : Lecture/écriture sur la cassette.

22H : Mise en route/arrêt du moteur.

24H : Gestion de l'interface de communication.

26H : Contrôleur de disquette.

Les codes de 28H à 5AH (et de A8H à DAH) inclus sont réservés à l'usage interne du moniteur.

15.3 Registres utilisés par le moniteur

Les registres sont donnés par leur nom et leur adresse en HEXADECIMAL.

* 2000-2018 (TERMIN) : Table des terminateurs de lignes logiques.
* 2019 (STATUS) : Différents sémaphores:

  b7: 0: majuscule/1: minuscule.
  b6: 1: scroll caractère sans couleur.
  b5: non utilisé.
  b4: 1: graphiques sans écriture de couleur.
  b3: sémaphore de lecture clavier.
  b2: curseur 1: visible/0: invisible.
  b1: sémaphore de répétition clavier.
  b0: touche clavier déjà lue.

* 201A-201B (TABPT) : Pointeur dans la table des terminateurs de lignes.
* 201B (RANG) : Ligne logique courante.
* 201C (COLN) : Colonne logique courante.
* 201D-201E (TOPTAB) : Pointeur sur le sommet logique de la table des terminateurs de lignes.
* 201E (TOPRAN) : Première ligne logique de la fenêtre.
* 201F-2020 (BOTTAB) : Pointeur sur la fin logique de la table des terminateurs de ligne.
* 2020 (BOTRAN) : Dernière ligne logique de la fenêtre.
* 2021-2022 (SCRPT) : Pointeur courant dans l'écran.
* 2023-2024 (STADR) : Adresse du premier octet de la fenêtre.
* 2025-2026 (ENDDR) : Adresse + 1 du dernier octet de la fenêtre.
* 2027-2028 (BLOCZ) : Registre de contenu toujours nul, pour les initialisations.
* 2029 (FORME) : Contient le code de la couleur (de -16 à +15) pour la mise en couleur d'un point ou le tracé d'un segment de droite.
* 202A (ATRANG) : Sémaphores pour la gestion d'écran:

  b7: 1: fond plein écran.
  b6: 1: forme plein écran.
  b5: réservé.
  b4: réservé.
  b3: réservé.
  b2: réservé.
  b1: largeur 0: simple/1: double.
  b0: hauteur 0: simple/1: double.

* 202B (COLOUR) : Couleur courante; les 4 bits de poids faible donnent la couleur du fond, les 4 bits de poids fort la couleur de la forme, suivant le codage vidéo pBVR où p est le bit de couleur pastel.
* 202C (PAGFLG) : Si ce registre contient la valeur 0, on est en mode scroll; s'il contient 255, on est en mode "page" (pas de scroll).
* 202D (SCROLS) : Si ce registre est à 0, le scroll est normal; s'il contient 255, le scroll est lent.
* 202E (CURSFL) : Sémaphore de mouvement curseur, qui, s'il contient la valeur 255, indique qu'il ne faut pas lier logiquement la ligne courante à la suivante.
* 202F (COPCHR) : Sémaphore qui, s'il contient la valeur 255,

indique que le déplacement à droite ou à gauche doit être interprété comme une recopie à droite ou à gauche du caractère courant.

* 2030 (EFCMPT) : Compteur d'effacement du curseur pour la gestion du clignotement.
* 2031 (ITCMPT) : Compteur d'interruptions IRQ.
* 2032-2033 (PLOTX) : Abscisse du dernier point allumé ou éteint par une routine graphique.
* 2034-2035 (PLOTY) : Ordonnée du dernier point allumé ou éteint par une routine graphique.
* 2036 (CHDRAW) : Code ASCII du caractère pour un tracé de point ou de droite en mode "caractères". Si ce registre contient la valeur 0, le tracé est fait en mode "points".
* 2037 (KEY) : Code de la dernière touche enfoncée au clavier
* 2038 (CMPTKB) : Compteur de répétitions clavier.
* 2039 : Réservé.
* 203A (TEMPO) : Tempo général pour la génération de musique.
* 203B : Réservé.
* 203C (DUREE) : Durée de la note (de 1 à 96).
* 203D (TIMBRE) : Attaque de la note.
* 203E-203F (OCTAVE) : Octave (1, 2, 4, 8 ou 16)
* 2040 (K7DATA) : Octet à écrire sur la bande magnétique du LEP.
* 2041 (K7LENG) : Nombre d'octets consécutifs à écrire sur la bande magnétique du LEP.
* 2042 (PR.OPC) : Mot de commande pour la gestion imprimante.
* 2043 (PR.STA) : Etat courant de la liaison imprimante.
* 2044-2045 (TEMP) : Registre temporaire.
* 2046-2047 (SAVEST) : Sauvegarde du pointeur de pile.
* 2048 (DK.OPC) : Mot de commande pour le contrôleur de disques.
* 2049 (DK.DRV) : Numéro de lecteur de disquettes.
* 204A-204B (DK.TRK) : Numéro de piste.
* 204C (DK.SEC) : Numéro de secteur.
* 204D (DK.NUM) : Entrelacement de secteurs.
* 204E (DK.STA) : Etat courant du contrôleur de disquettes.
* 2041 (K7LENG) : Nombre d'octets consécutifs à écrire sur la
* 204F-2050 (DK.BUF) : Pointeur sur la zone-tampon réservée aux entrées-sorties disquette.
* 2051-2052 (TRACK0) : Position de la tête, lecteur 0.
* 2053-2054 (TRACK1) : Position de la tête, lecteur 1.
* 2055-2056 (TRACK2) : Position de la tête, lecteur 2.
* 2057-2058 (TRACK3) : Position de la tête, lecteur 3.
* 2059 (SEQUCE) : Code indiquant dans quelle séquence de gestion d'écran on se trouve.
* 205A (US1) : Sémaphore pour les séquences "unit separator".
* 205B (ACCENT) : Sémaphore pour les séquences accent.
* 205C (SS2GET) : Registre pour l'impression ou la lecture d'une minuscule accentuée.
* 205D (SS3GET) : Registre pour l'impression ou la lecture d'une minuscule accentuée.
* 205E-205F (SWI1PT) : Pointeur sur la routine de traitement des appels au moniteur.
* 2060 : Réservé.
* 2061-2062 (TIMEPT) : Pointeur sur la routine utilisateur de traitement des interruptions IRQ.
* 2063 : Sémaphore d'aiguillage des IRQ.
* 2064-2065 (IRQPT) : Pointeur sur la routine moniteur de traitement des interruptions IRQ.
* 2066 : Réservé.
* 2067-2068 (FIRQPT) : Pointeur sur la routine de traitement des interruptions rapides FIRQ.

```
* 2069 :                    Réservé.
* 206A-206B (SIMUL) :       Pointeur sur  la table des points d'entrée  du
                            moniteur.
* 206C :                    Réservé.
* 206D-206E (CHRPTR) :      Pointeur sur la table de décodage du clavier.
* 206F :                    Réservé.
* 2070-2071 (USERAF) :      Pointeur  sur  le  générateur  de  caractères
                            utilisateur.
* 2072 :                    Réservé.
* 2073-2074 (GENPTR) :      Pointeur sur le générateur de caractères stan-
                            dard
* 2075 :                    Réservé.
* 2076 (LATCLV) :           Latence clavier.
* 2077 (GRCODE) :           Mot de code pour la mise en  mode graphique de
                            l'imprimante.
* 2078 (DECALG) :           Ajustement pour le crayon optique.
* 2079-207E :               Réservés., 2 0.
* 207F (DEFDST) :           Sémaphore de simple ou double densité.
* 2080 (DKFLG) :            Sémaphore de présence du contrôleur de disques
* 2081-20CC :               Pile système.
* 20CD-20E4 (LPBUFF) :      Zone-tampon pour la lecture du crayon optique.
* 20E5-20FD :               Réservés.
* 20FE-20FF (TSTRST) :      Sémaphore de démarrage à chaud ou à froid.
```

15.4 Adresses d'entrées/sorties

Les adresses qui suivent sont données en hexadécimal.

15.4.1 PIA système

*** PIA système 6821 ***

* A7C0: Registre de données, port A.

bit0 (sortie): commutation mémoire écran caractères et mémoire écran couleur.
bit1 (sortie): couleur du tour, rouge.
bit2 (sortie): couleur du tour, vert.
bit3 (sortie): couleur du tour, bleu.
bit4 (sortie): couleur du tour, pastel.
bit5 (entrée): interruption crayon optique.
bit6 (sortie): écriture cassette.
bit7 (entrée): lecture cassette.

* A7C1: Registre de données, port B:

bit0 (sortie): son
bits 1-6: (sortie): matriçage/dematriçage clavier.
bit7 (entrée): lecture clavier.

* A7C2: Registre de contrôle, port A:
CA1 (entrée): crayon optique.
CA2 (sortie): moteur du LEP.

* A7C3: Registre de contrôle, port B:
CB1 (entrée): interruptions 50 Hertz.
CB2 (sortie): commande d'incrustation.

15.4.2 PIA jeux

*** PIA JEUX 6821 ***

* A7CC: Registre de données, port A:

bits 0-7 (entrée): lecture des manettes de jeu.

* A7CD: Registre de données, port B:

bits 0-5 (entrée): convertisseur digital/analogique.
bit6 (entrée): action manette de jeu 0.
bit7 (entrée): action manette de jeu 1.

* A7CE: Registre de contrôle, port A:
CA1 (entrée): action manette de jeu 0.

* A7CF: Registre de contrôle, port B:
CB1 (entrée): action manette de jeu 1.

15.4.3 Interface de communication

*** INTERFACE DE COMMUNICATION 6821 ***

* A7EC: Registre de données, port A:
Non utilisé, disponible pour la gestion d'une liaison série.

* A7E1: Registre de données, port B:

Bits 0-7 (sortie) données en parallèle.

* A7E2: Registre de contrôle, port A:
CA1 (entrée): request to send (demande d'émission).

* A7E3: Registre de contrôle, port B:
CB1 (entrée): acknowledge.
CB2 (sortie): strobe.

15.5 Profondeur de la pile dans le moniteur

Lors de tout appel au moniteur se faisant par un SWI, il y a au moins 16 octets empilés au cours d'un appel.

Certaines routines du moniteur sont protégées contre les interruptions: c'est le cas des routines de génération de musique, de lecture/écriture de la cassette, de saisie du crayon optique, et des routines de gestion des disquettes.

Pour les autres, il faut tenir compte de la possibilité d'interruption par IRQ ou FIRQ, ce qui ajoute respectivement 30 et 3 octets à la profondeur de pile nécessaire.

Voici la liste de la profondeur de pile MINIMUM nécessaire selon les codes des différents modules:

Code 02H: de 16 à 43 octets selon les caractères:

Caractère normal affichable : minimum 21 octets, maximum 43 octets si il y a scroll, y compris pour les minuscules accentuées.

Caractère de contrôle non affichable :

07H	Bip sonore	16 octets
08H	Flèche à gauche	23 octets, 43 si scroll.
09H	Flèche à droite	23 octets, 43 si scroll.
0AH	Flèche en bas	21 octets, 43 si scroll.
0BH	Flèche en haut	21 octets, 43 si scroll.
0CH	RAZ écran	21 octets.
0DH	Retour chariot	21 octets.
11H	Curseur visible	16 octets.
14H	Curseur invisible	21 octets
16H	Séquence accent	16 octets.
18H	"Cancel"	35 octets.
1BH	Echappement	16 octets.
1EH	"RS"	21 octets.
1FH	Séquence US	16 octets.

Les codes non reconnus, ne sont pas traités et provoquent donc seulement l'empilement de 16 octets.

* séquences d'échappement (taille pour chaque appel):

- plein écran : inverse vidéo 35 octets
  couleur fond 38 octets
  couleur forme 38 octets

- courant : inverse vidéo 22 octets
  scroll 18 octets
  incrustation 18 octets
  pas incrust. 18 octets
  écr.car.seul 18 octets
  écr.car&coul 18 octets
  taille 23 octets
  tour saturé 23 octets
  couleur fond 23 octets
  car. saturé 23 octets

* séquences US :

- code < $40 : 26 octets
- code >= $40 : 25 octets

Code 04H: 16 octets.

Code 06H: 16 octets.

Code 08H: 16 octets.

Code 0AH: 32 octets.

Code 0CH: 16 octets.

Code 0EH: 23 octets si le tracé est horizontal,
32 s'il est graphique,
53 s'il est en caractères.

Code 10H: 18 octets si graphique,
40 si caractère.

Code 12H: 40 octets.

Code 14H: 18 octets.

Code 16H: 17 octets.

Code 18H: 18 octets.

Code 1AH: 36 octets.

Code 1CH: 16 octets.

Code 1EH: 18 octets.

Code 20H: 20 octets en lecture,
18 en écriture.

Code 22H: 16 octets (marche ou arrêt).

Code 24H: 16 octets pour ouverture et écriture caractère,
30 pour une copie graphique d'écran.

Code 26H: 40 octets en lecture ou écriture.

Interruption rapide FIRQ: 3 octets

Interruption normale IRQ: 30 octets

INDEX

En majuscules : les MOTS, instructions ou commandes BASIC.
En minuscules : fonctions
En italiques minuscules : les signes d'*opérations*
En italiques majuscules : les *TOUCHES* du MO5

Achevé d'imprimer par Pollina, 85400 Luçon - N° 7347
Dépôt légal : 3e trimestre 1985