Le livre du 6128

Car les oies de la roche capitulaire sont proches des tarpons aigus (ceci, c'est de l'histoire ancienne, traduction libre et ésotérique de Tacite).
Je reviens au sujet.

A force de faire des retours en arrière et en avant, très cher lecteur, je comprends que vous croisassiez vos pinceaux. Après l'été 85, vous avez eu droit à un dîner de têtes à l'Elysée en mai 84, puis une poussée en avant sur la création des journaux amstradiens depuis mai 85.

Où sommes-nous?
Où allons-nous?
Moi je le sais.
Où allez-vous donc?
Mais là où je vous emmène !

Intermède manuel

Sèvres, le 2 septembre 85

Les vacances ont été studieuses et concentrées. Un CPC 6128, plus un PCW 8256, c'est trois livres à traduire/adapter en quatrième vitesse. Le livre du 6128 ne pose pas trop de problèmes. C'est le manuel du 664, un peu remanié un peu étoffé ; et comme le manuel du CPC 664, il est produit en France.

Pour le PCW 8256, il y a deux manuels, un livre pour le traitement de texte et CP/M et un livre pour le Basic. Alan Sugar pense que le livre du Locoscript était trop technique et trop détaillé. Il a encore raison. Il m'énerve J'ai déjà insisté sur les problèmes de la documentation. Faut-il orienter vos efforts vers les néophytes et se faire reprocher la simplicité du manuel? Ou faut-il favoriser les techniciens et les experts et se faire taxer d'élitisme? Maintenant, je connais la solution. Evidente. Mais en 1985, les manuels du PCW 8256 ont été l'objet de critiques méritées et sévères.

SOMMAIRE

DEUXIEME PARTIE : (Suite).

TROISIEME PARTIE : Glossaire.

INTRODUCTION

MISE EN MARCHE DE VOTRE AMSTRAD

Vous devez connecter le moniteur couleur ou noir et blanc au clavier disquette de votre AMSTRAD.

- Ne pas brancher au secteur.

- Connecter le cable sortant du clavier dans l'entrée 12V DC.

- Connecter le cable DIN 6 broches partant de la face avant du moniteur sur la prise MONITOR à l'arrière du clavier.

- Connecter le dernier cable partant de la face avant du moniteur dans la prise 5V DC à l'arrière du clavier.

Vous pouvez maintenant brancher le moniteur au secteur.

Vous allumez le moniteur en appuyant sur le commutateur situé à l'avant du moniteur, lorsque le commutateur est enfoncé, le système est sous tension.

Vous allumez le clavier avec la touche POWER à droite de l'ordinateur, le voyant rouge s'allume .

Il faut toujours allumer le moniteur avant d'allumer le clavier.

A l'initialisation du sytème, vous avez à l'écran : BASIC 1,1.

Vous pouvez régler la brillance (BRIGHTNESS) sur tous les moniteurs et le contraste et la hauteur sur les moniteurs Noir et blanc avec les boutons CONTRAST et VERTICAL HOLD.

CONNECTION DES PERIPHERIQUES :

JOYSTICK (manettes de jeu):

Vous devez connecter la manette de jeu sur le côté de votre clavier. C'est un élément indispensable pour utiliser les logiciels de jeu, elle sert à se mouvoir sur l'écran en pouvant en même temps appuyer sur le bouton FIRE.

Vous pouvez aussi en programmant vous-même en BASIC trouver d'autres utilisations à cette manette.

LECTEUR/ENREGISTREUR DE CASSETTES

Vous pouvez connecter un lecteur enregistreur de cassettes sur le côté gauche du clavier. Pour connecter votre lecteur, vous avez besoin du câble AMSOFT CL1 que vous branchez comme suit :

- Extrémité bleue dans le port REM ou REMOTE du lecteur.

- Extrémité verte dans le port MIC, COMPUTER IN ou INPUT.

- Extrémite blanche dans le port EAR, COMPUTER OUT ou OUTPUT.

IL est important de savoir que le paramètre le plus important dans une sauvegarde sur cassettes, est le volume d'enregistrement.

IL faut donc faire plusieurs essais à différents niveaux de volumes.

IMPRIMANTES

Vous pouvez utiliser avec l'AMSTRAD les imprimantes parallèles CENTRONIC.

Si vous voulez connecter l'imprimante AMSTRAD DMP1, vous n'avez qu'à brancher le câble sur la sortie PRINTER de l'AMSTRAD.

Si vous voulez brancher d'autres imprimantes CENTRONICS, il vous faut le câble AMSOFT PL1.

UN DEUXIEME LECTEUR DE DISQUETTE

Vous pouvez brancher un 2ème lecteur de disquette, l'AMSTRAD FD1. L'utilisation d'un 2ème lecteur est obligatoire si vous voulez utiliser des logiciels CPM qui ont tous les programmes sur une disquette et les fichiers sur une autre.

Avec un 2ème lecteur, vous pouvez aussi effectuer simplement toutes les manoeuvres de copie ou de sauvegarde, car vous n'aurez plus à interchanger les disquettes.

Pour connecter le 2ème lecteur de disquette, il vous faut le cable AMSOFT D12.

IMPORTANT: Avant de brancher ou débrancher un lecteur, vous devez vous assurer qu'il n'y a pas de disquette dedans et que le système n'est pas sous tension.

AMPLIFICATEUR EXTERNE/HAUT-PARLEUR

Vous pouvez connecter votre AMSTRAD sur un amplificateur stéréo et sur des haut-parleurs externes, pour profiter pleinement des capacités musicales de l'AMSTRAD.

Le câble de connection de votre amplificateur doit avoir une prise jack stéréo de 3,5mm, que vous branchez dans la prise STEREO de votre AMSTRAD.

L'AMSTRAD donne un signal à voltage constant. Vous devez ensuite régler ce signal en utilisant les contrôles de votre amplificateurs pour le volume, la balance et la tonalité.

Vous pouvez brancher aussi directement des haut-parleurs à haute impédance, mais vous ne pourrez pas régler le volume avec le bouton de votre ordinateur. Les haut-parleurs en basse impédance (ENCEINTES HI-FI) ne peuvent pas être utilisés directement sans amplificateur.

PERIPHERIQUE D'EXTENSION

Vous pouvez étendre les capacités de votre AMSTRAD avec d'autres périphériques tels qu'une interface série, un modem, un crayon lumineux, des ROM, etc... en utilisant le port d'extention marqué EXPANSION.

Le synthétiseur de voix AMSOFT SSA2 peut aussi se connecter sur ce port de sortie.

PREMIERE PARTIE :

Présentation

PREMIERE UTILISATION _____

LES DISQUETTES

L'AMSTRAD utilise des disquettes au format 3 pouces.

Nous recommandons d'utiliser pour transférer des données, l'AMSOFT CF2. Les disques fabriqués par d'autres manufacturiers peuvent néanmoins aussi être utilisés.

- Les disquettes sont double face, mais le lecteur est mono-face, vous pouvez donc utiliser les deux faces de vos disquettes. La disquette doit toujours être insérée en gardant de votre côté l'étiquette, et avec la face qui vous intéresse.

- Vous pouvez protéger les données de votre disquette en empêchant l'écriture. Il suffit d'enfoncer le petit curseur que vous avez à l'opposé de l'étiquette. Vous pouvez protéger les faces séparement.

Il suffit de repousser ce curseur pour rendre la disquette de nouveau utilisable pour l'écriture. D'autres systèmes de protections peuvent exister sous d'autres marques de disquettes, mais le principe est le même.

En protection, vous pouvez toujours lire les données sur votre disquette (fichiers ou programme), mais vous ne pouvez plus en écrire. Vous êtes alors sûr de ne pas effacer des fichiers ou des programmes par inadvertance.

N.B.: Assurez-vous que la disquette CPM est bien protégée.

- Sur la face avant du lecteur, vous avez un voyant lumineux et un bouton d'éjection.

Le voyant s'allume quand le lecteur lit ou écrit sur la disquette. Sur le 2ème lecteur externe, ce voyant reste en permanence allumé.

Quand vous appuyez sur le bouton d'éjection, vous sortez la disquette du lecteur.

CONNAISSANCE DU CLAVIER

Avant de faire quoi que ce soit avec votre AMSTRAD, vous devez apprendre à vous servir du clavier.

Ceux qui ont déjà une expérience dans le domaine peuvent sauter ce chapitre.

Nous allons tester directement les différentes touches :

ENTER

Les deux touches ENTER et RETURN ont la même fonction. Après l'appui sur la touche ENTER, une nouvelle ligne apparaît sur l'écran. Chaque commande doit être suivie de ENTER pour être validée.

DEL

Cette touche est utilisée pour supprimer le caractère précédent immédiatement le curseur.

SHIFT

Il existe deux touches SHIFT.
En appuyant simultanément sur une des touches SHIFT et sur une autre touche, s'affiche à l'écran, ou bien une majuscule, ou bien le symbole supérieur de la touche.

CAPS LOCK

En tapant sur CONTROL puis sur CAPS LOCK, vous obtenez le même effet, qu'en appuyant sur SHIFT, vous resterez en mode majuscule jusqu'à ce que vous appuyez de nouveau sur CAPS LOCK.

CLR

En appuyant sur cette touche, le caractère qui se trouve sous le curseur sera effacé Pour vous positionner sur le caractère à supprimer, utilisez les flèches -> et <-.

ESC

En appuyant une fois sur cette touche, un arrêt momentané dans le processus en cours, s'effectue, et ceci jusqu'à ce que vous tapiez sur une une autre touche.
En tapant deux fois de suite sur ESC, on sort du programme en cours. De nouvelles instructions peuvent alors être acceptées.

PROGRAMMATION

PROGRAMMATION :

Maintenant que votre AMSTRAD est installé, que vous savez vous servir des touches, charger un programme, il faut passer aux choses sérieuses, c'est à dire LA PROGRAMMATION.

Dans le cas présent, nous allons apprendre à programmer en BASIC.

Le BASIC est le langage le plus répandu sur micro-ordinateur, il permet à l'aide de mots clés simples (en anglais) de donner des ordres à votre ordinateur.

Mais si le BASIC est simple à apprendre, il ne supporte aucune déformation de langage et à la moindre erreur de votre part, il vous le fera savoir. De toute façon ce n'est pas grave, il vous suffira de corriger l'erreur et tout rentrera dans l'ordre.

Pour bien commencer, il faut un plan de travail net. Nous allons donc effacer l'écran.

Vous tapez:

CLS <return>

Une fois return appuyé, l'écran se vide et un READY apparaît sur le coin en haut à gauche.

Essayons autre chose:

Vous voulez faire apparaître sur l'écran le mot "salut"
Très facile il vous suffit de taper:

PRINT"salut" <return>

Vous verrez sur l'écran :

salut

Vous remarquez que le texte doit toujours être entre guillemets.

Jusqu'à présent, nous nous sommes contentés d'exécuter directement des commandes BASIC. L'ennui et qu'à chaque fois que vous voulez avoir "salut" à l'écran il vous faut retaper toute la ligne. Il faudrait donc la stocker en mémoire.

Pour cela tapez:

10 PRINT "salut" <return>

Une fois la touche return appuyée, rien ne se passe, ce qui est tout à fait normal car un numéro devant une instruction signifie à l'ordinateur qu'il s'agit d'une ligne qu'il faut garder en mémoire.

Remarque : le signe ? est équivalent à la commande PRINT, ainsi l'exemple ci-dessus peut s'écrire : 10 ? "salut" <return>

Mais essayez de taper:

RUN <return>

et vous verrez le mot "salut" à nouveau s'afficher à l'écran.

Le mot RUN indique à l'ordinateur de lancer le programme qu'il a en mémoire. Vous pouvez taper autant de fois que vous voulez le mot RUN suivit de return et vous aurez toujours le mot "salut" à l'écran.

Lorque vous voulez savoir ce que vous avez en mémoire vous tapez:

LIST <return>

Essayez et vous aurez:

10 print "salut"

ce qui est bien le programme que vous avez stocké.

Vous voulez avoir à l'écran toute une série de "salut", mais taper à chaque fois <return>,c'est pas marrant.

Simple, il suffit de rajouter au programme la ligne suivante:

20 GOTO 10 <return>

et

RUN <return>

Vous aurez alors à l'écran toute une série de "salut".

Pour arrêter le programme, tapez une fois sur <ESC>. Pour le reprendre, vous tapez sur n'importe quelle touche.

Pour l'arrêter complètement, tapez <ESC> deux fois de suite.

Pour voir le mot "salut" s'afficher sur tout l'écran, il suffit de taper la ligne 10:

```
10 PRINT "salut"; <return>
RUN <return>
```

Le point-virgule commande à l'ordinateur d'afficher le prochain groupe de caractères à la suite du texte déjà présent à l'écran.

Maintenant essayons autre chose , vous allez remplacer le point-virgule de la ligne:
10 par une virgule:

```
10 PRINT "salut",<return>
RUN <return>
```

Vous vous apercevez que la virgule aprés un PRINT oblige l'ordinateur à afficher les données avec une séparation de 13 blancs entre elles. Vous pouvez , si vous le voulez mofifier cette écart à l'aide de la commande ZONE.

Vous savez maintenent maintenant entrer une ligne de programme, nous allons voir comment rentrer simplement des données.

En BASIC vous avez deux sortes de données :

Les données de forme numérique:

10,10.50,1000 etc...

Elles sont affectées à une variable quelconque :

ex a=10: B=100: salaire=8000

Les données de forme alphanumérique:

"salut", "bonjour", "taxe =18.60";

ces données sont toujours affectées à des variables ayant le signe $ (dollar) à la fin de leur nom, elles peuvent contenir des lettres et des chiffres mais toujours considérés comme du texte.
Leur longueur peut atteindre 255 caractères. Les caractères qui leur sont affectés sont toujours entre "guillemets"

ex: a$="bonjour":nom$="albert"

Pour rentrer des données en mode direct, il suffit donc de choisir un nom et d'affecter à ce nom les données

Dans un programme, il faut vous servir de la commande INPUT

ex: NEW <return> (efface le programme en mémoire)
10 INPUT "votre nom s.v.p",nom$ <return>
RUN <return>

En lançant le programme, vous vous apercevrez que le texte qui se touve entre guillemets s'affiche à l'écran.
La commande INPUT, permet tout comme PRINT, d'afficher du texte, leur seule différence est que INPUT attends que vous tapiez quelque chose.

Dans cette exemple vous tapez votre nom:

albert <return> (ici pas besoin de guillemets)

pour vérifier la bonne marche essayez:

PRINT nom$ <return>

et vous verrez apparaître :

albert

Tout marche donc bien, nous allons maintenant rentrer des nombres, vous allez rajouter les lignes suivantes:

```
20  INPUT "votre âge";age
30 PRINT " donc ";nom$;" vous avez ";age;" ans"
RUN < return>
```

Vous devez donc rentrer votre nom et ensuite votre âge, et vous verrez que les deux données ont été bien prises en compte.

En tapant les exemples ci-dessus vous avez peut-être fait quelques erreurs de frappes:

```
ex:10 INPU "votre nom s.v.p";nom$ <return>
20 INPUT"votr age";;age
```

Vous voyez qu'en ligne 10 , nous avons oublié le T de la commande INPUT, en ligne 20 nous avons omis le "e" dans "votre" . Il faut donc corriger les fautes. Pour cela il y a trois méthodes:

1) Retapez la ligne entiérement, quand vous rentrez une ligne avec un numéro déja existant la ligne présente en mémoire est remplacée. Méthode un peu archaïque.

2) METHODE D'EDITION A L'AIDE DU CURSEUR:

Pour corriger la ligne 10, tapez:

```
EDIT 10 <return>
```

La ligne apparaît à l'écran le curseur placé sur le I de INPUT.
Pour rajouter le T manquant vous appuyer sur la touche fléche droite jusqu'à ce que le curseur soit juste après le U, vous appuyez ensuite sur le T, voila la correction est faite, en appuyant sur <return> la ligne est mise en mémoire.

3) METHODE PAR COPIE DU CURSEUR:

Le curseur de copie est le deuxième curseur qui s'affiche lorsque vous appuyez simultanément sur la touche <SHIFT> et sur l'une des touches du curseur.
Pour corriger la ligne 20, actionnez la touche <SHIFT> et avec la touche flèche haute, amenez le curseur au début de la ligne 20. Le curseur principal est resté en bas, seul le curseur de copie a bougé. Vous pressez la touche copie jusqu'à que le curseur soit situé après le "r" de "votre".Vous tapez alors la lettre manquante, ensuite il vous reste à appuyer sur la touche copie jusqu'au bout de la ligne. En appuyant sur <return> vous validez la ligne.

Maintenant que vous savez corriger une ligne, nous allons pouvoir faire des choses beaucoup plus complexes.

Ce qui caractérise l'ordinateur, c'est en fait la possibilité de faire des tests.

ex: If 2*2=4 THEN PRINT "ok tout va bien"

Vous voyez qu'ici l'ordinateur teste si 2*2=4, et si c'est le cas, il exécute les instructions qui suivent le THEN.

Le mot clé ELSE permet de faire une action si le test est faux
ex:

```
If 2*2=4 THEN PRINT "ok tout va bien" ELSE PRINT"erreur de
calcul"
```

Le PRINT aprés le ELSE ne sera exécuté que si le test est faux .

Essayons:
ex:

```
NEW <return>
10 INPUT" votre nom s.v.p";nom$ <return>
20 INPUT"votre age";age <return>
30 IF age <10 then 100 <return>
40 IF age <18 THEN 200 <return>
50 IF age >18 THEN 300 <return>
100 PRINT"vous êtes vraiment jeune":END <return>
200 PRINT"bientôt la majorité !":END <return>
300 PRINT"Vivement la retraite":END <return>
```

Vous vérifiez le programme en tapant LIST <return>.
Si tout est correct vous tapez RUN <return>.
Nous avons été obligés de mettre des END (fin du programme) à la fin des lignes 100,200 et 300. Si nous ne l'avions pas fait la programme aurait afficher les lignes qui suivent.

Vous avez remarquez que l'instruction END se trouve sur la même ligne que l'instruction PRINT. Vous pouvez en effet mettre plusieurs instructions par ligne en les séparant simplement par deux points ":".

Avec le BASIC, vous avez la possibilé de répéter plusieurs fois la même partie du programme, pour cela on utilise les commandes FOR et NEXT

```
ex:     NEW <return>
        10 CLS <return>
        20 FOR x=1 TO 5 <return>
        30 PRINT " l'action a été répétée ";x;" fois" <return>
        40 NEXT X <return>
        RUN <return>
```

La ligne 30 va être exécutée 5 fois, comme le spécifie la commande FOR de la ligne 20.

La variable x est incrémentée de 1 à chaque passage.
Si vous voulez que l'incrément soit supérieur à 1 il suffit d'utiliser la commande STEP qui permet de définir le pas.

Changez la ligne 20 par:

```
        20 FOR x=1 TO 10 STEP 2 <return>
        RUN <return>
```

Ici le pas sera donc de 2.
Mais on peut aussi choisir un pas négatif:

```
ex:     20 FOR x=10 TO 0  STEP-2
```

Imaginons que vous tapiez un programme où il y ait plusieurs lignes identiques:

```
ex:     NEW <return>
        10 INPUT "votre nom ";nom$
        20  PRINT " en validant par la  touche RETURN"
        30 INPUT "votre age";age
        40  PRINT " en validant par la  touche RETURN"
        50 INPUT"votre no de téléphone";tel
        60 PRINT " en validant par la  touche RETURN"
        70 END
```

Vous voyez ici que les lignes 20,40 et 60 font exactement la même chose. D'où l'intérêt de ce qu'on appelle le SOUS-PROGRAMME .

Pour exécuter un sous-programme il faut l'appeler par un GOSUB suivi d'un numéro de ligne .

Un sous-programme doit toujours se terminer par l'instruction RETURN.

```
ex:     NEW <return>
        10 INPUT "votre nom ";nom$:GOSUB 100
        20 INPUT "votre age";age:GOSUB 100
        20 INPUT"votre no de téléphone";tel:GOSUB 100
        30 END
        100 PRINT " en validant par la  touche RETURN"
        110 RETURN
```

Remarquez la différence de taille ! Les sous-programme sont des outils très utiles en programmation, il permettent de faire des programmes structurés.
IL est possible dans un sous-programme d'avoir plusieurs lignes d'entrée. Si vous avez un sous-programme qui occupe les lignes de 100 à 150, il pourra être abordé aux lignes comprises entre 100 à 150.

Nous avons vu que l'ordinateur peut avec votre aide faire plein de choses. Il peut aussi servir de grosse calculatrice.

Pour bien vous familiariser avec le clavier voici quelques exemples:

ADDITION:

tapez:

```
?4+5 <return>
9
```

(le signe + s'obtient en tapant: SHIFT et ;)

SOUSTRACTION:

```
?4-5 <return>
-1
```

MULTIPLICATION:

```
?4*5 <return>
20
```

(le signe * s'obtient en appuyant sur SHIFT et :

DIVISION:

```
?20/4
5
```

(le signe / s'obtient en appuyant sur ?)

DIVISION ENTIERE:

```
?10\6
1
```

MODULO
(utilisez MOD pour avoir le reste de la division entière)

```
?10 MOD 4
2
```

RACINE CARREE

```
?SQR(4)  (équivaut à √4)
2
```

PUISSANCES

```
?4^4
16
```

RACINE CUBIQUE

Il est possible d'avoir la racine cubique par la méthode suivante.
ex: racine cubique de 27

```
?27^(1/3)
3
```

CALCULS COMPOSES

Il vous est possible d'effectuer des calculs mélangeant addition, multiplication etc...

Il vous faut cepandant faire attention aux priorités qui sont:

↑	Elévation à la puissance
MOD	Modulo
-	Moins unaire (donne le négatif d'un nombre)
* /	Multiplication et division
	Division entière
+ -	Addition et soustraction

Ex: le calcul

```
?5+8*4/3-2
 13.6666667
```

Pour changer cet ordre de priorité, il suffit d'isoler les opérations à l'aide de parenthèses.

CHARGEMENT ET SAUVEGARDE

CHARGEMENT DES PROGRAMMES :

Vous avez commencé à programmer, il vous faut maintenant apprendre à bien vous servir du lecteur de disquette.

En appuyant simultanément sur les touches ESC, CONTROL et SHIFT vous réinitialisez complétement votre ordinateur.

Une fois fait, vous insérez dans le lecteur la disquette n°4 de CP/M et vous tapez:

RUN "rointime .dem" <return>

Au bout de quelques secondes, si tout s'est bien passé le message suivant doit apparaître:

ARE YOU USING A GREEN SCREEN ?
PRESS Y OR N

Ce qui signifie que vous devez appuyer sur Y, si vous possédez un écran vert ou dans le cas contraire vous appuyez sur N.

Une fois la touche appuyée, la démonstration d'un jeu commence. Ce qui veut aussi dire que tout s'est bien passé.

Mais admettons que ce ne soit pas le cas:
et que vous avez à l'écran le message d'erreur suivant:

Drive A:Disc missing
Retry,Ignore or Cancel

A se stade plusieurs hypothèses sont envisageables:

-Soit vous n'avez pas inséré la disquette
-Soit vous possédez deux unités de disquettes et vous avez insérez la disquette dans l'unité B.
-Ou alors plus grave, votre lecteur ne fonctionne pas correctement.

Dans ce cas la meilleure solution et de refaire les opérations depuis le début. Si par malheur vos problèmes persistent, nous vous conseillons de consulter votre vendeur qui lui trouvera certainement la cause de tous vos maux.

Si par contre comme message d'erreur vous avez:

rointime.dem not found

Indique que vous n'avez pas inséré la bonne disquette, ou la bonne face, ou alors que vous mal orthographié le nom.

Le message :

Bad commande

Signifie que dans le nom, vous avez mis des signes de ponctuation superflus.

Type mismatch

Indique que vous avez oublié les guillemets .

Syntax error

Vous avez sans doute fait une erreur en tapant RUN.

drive A: read fail
Retry, Ignore or Cancel

Signifie que la lecture de votre disquette est impossible, vérifiez que vous avez inséré la bonne disquette et tapez R pour RETRY (recommencer), si le même message apparaît, c'est que votre disquette est endommagée par une mauvaise utilisation.

IL NE FAUT JAMAIS ETEINDRE OU ALLUMER L'ORDINATEUR SI UNE DISQUETTE EST INSEREE DANS LE LECTEUR.

SAUVEGARDE D'UN PROGRAMME:

Vous avez tapé un programme en mémoire et vous désirez le garder sur une disquette, il vous suffit de taper:

Save"nom" <return>

Vous devez donner un nom à votre programme.

Le nom se compose de deux parties, la première obligatoire peut contenir jusqu'à huit caractères, lettres ou nombres, mais les espaces et les signes de ponctuation sont interdits. La première zone est le nom que vous donnez à votre programme.

La deuxième zone est souvent appelée l'attribut, elle indique la nature du programme. Si celui-ci est un fichier BASIC, elle sera .BAS.

Si le programme est un fichier binaire, l'attribut sera .BIN. Si vous sauvegardez un fichier de données l'attribut sera seulement un point.
Lors d'une sauvegarde, si le nom existe déjà, le basic crée automatiquement un nouveau fichier comportant l'attribut .BAK, ce qui évite les écrasements de fichiers intempestifs.

CATALOGUE

Pour vérifier la bonne marche de votre sauvegarde, AMS-DOC, vous pouvez faire un catalogue de votre disquette.

cat <return>

CHARGEMENT D'UN PROGRAMME SUR DISQUETTE

Deux choix sont possibles:

Soit: Load"nom" <return>

Une fois READY à l'écran votre programme est en mémoire vous pouvez travailler dessus en le listant.

Soit: run"nom" <return>
pour une exécution immédiate.

DEUXIEME PARTIE :

Perfectionnement

ABS

ABS <expression numérique>

ex: PRINT ABS(-56.30)
56.30

Donne la valeur ABSolue de l'expression qui se trouve entre les parenthèses.
Cette fonction retourne toujours une valeur positive ou égale à zéro.

AFTER

AFTER <délai,(numéro),GOSUB <n° de ligne>

ex :
```
10 AFTER1000 GOSUB 100:CLS
15 IF f=1 THEN END
20 PRINT" vous avez 20 secondes pour trouver la capitale du PEROU"
30 LOCATE 10,10:INPUT a$
40 IF a$<> CHR$(108)+CHR$(105)+CHR$(109)+CHR$(97) THEN 30
50 LOCATE 10,10:PRINT "bravo vous êtes très fort"
60 SOUND 1,378: SOUND 1,258:END
100 PRINT "désolé mais le temps est écoulé !"
110 SOUND 1,1000:f=1:RETURN
RUN
```

La commande AFTER appelle un sous-programme au bout d'un certain délai, celui-ci est un multiple de 0.02 seconde.
Les numéros vont de 0 à 4.
Chacun peut être associé à un sous-programme.
Le n° 0 est toujours pris par défaut.

AND
<argument> AND <argument>

ex: IF x<10 AND x>8 THEN PRINT "x est donc compris entre 8 et 10"

Expression booléenne qui est vraie que si les deux arguments sont tous les deux exacts.

ASC
ASC <chaîne alphanumérique>

ex: PRINT ASC("a")
97

Retourne la valeur ASCII du premier caractère d'une chaîne alphanumérique.

ATN
ATN <expression numérique>

ex: PRINT ATN(2)
1.10714872

Calcule l'Arc TaNgente de l'expression numérique mais la réduisant à un nombre réel en radians compris entre -PI/2 à +PI/2.
Vous devez utiliser les commandes DEG et RAD pour spécifier dans quel mode sera donné le résultat.

AUTO
AUTO (n° de ligne),(incrémentation)

ex: auto 100,2

Dans la saisie d'un programme cette commande vous donne AUTOmatiquement les numéros de ligne.
Si vous ne précisez pas de numéro, les lignes sont générées à partir de 10

L'incrémentation, elle aussi falcultative, fixe l'intervalle entre les numéros, si elle n'est pas précisée, 10 est pris comme défaut.
Si un numéro de ligne déja existante est créé la ligne s'affiche et peut être modifiée.
Pour arrêter la numérotation, il suffit d'appuyer sur la touche ESC.

BIN$
BIN$ (nombre entier), (nombre entier)

ex: PRINT BIN$(255,8)
11111111

Retourne une série de chiffres BINaires représentant la valeur du premier nombre entier (toujours sans signe), sur autant de caractères qu'indique le deuxième nombre dans la mesure où il n'est pas trop petit, si c'est le cas le résultat se compose d' autant de chiffres nécessaires.
De plus le premier nombre entier ne doit pas être supèrieur à 65535.

BORDER
BORDER <couleur>, (couleur)

ex:
```
10 FOR couleur=0 TO 26
20 BORDER couleur
30 FOR temps=1 TO 500:NEXT temps
40 NEXT couleur:END
RUN
```

Change la couleur du bord de l'écran. Vous pouvez indiquer deux couleurs qui alterneront à la vitesse définie par la commande SPEED INK.

CALL
CALL <ADRESSE>, (paramètres)

ex: CALL 0

Permet à partir du BASIC de lancer une routine en langage machine son emploi demande une bonne connaissance de l'ordinateur. L'exemple donné réinitialise votre AMSTRAD.

CAT

Affiche sur l'écran le CATalogue de la disquette, ce qui comprend tous les noms de fichiers présents avec leurs longueurs ainsi que la place disponible et les identificateurs de la disquette et de l'utilisateur.

CHAIN
CHAIN <nom fichier>, (numéro de ligne)

ex: CHAIN "deuxieme pro",150

Charge un programme à partir de la disquette, remplacant le programme précédent tout en conservant les variables et tableaux déja existants.
Le nouveau programme peut commencer à partir d'un numéro de ligne.

Cette commande ne fonctionnera pas si les fichiers sont protégés
(sauvegardés par SAVE,p).

CHAIN MERGE
CHAIN MERGE <nom fichier>, (numéro de ligne), (DELETE <groupe de lignes>)

ex: CHAIN MERGE "deuxième pro",350,DELETE 100-200

Fusionne un programme sur disquette avec un programme en mémoire et lance le programme obtenu à partir d'un numéro de ligne si celui-ci est précisé. Vous pouvez détuire des lignes du programme initial en vous servant de l'option DELETE. Les numéros de lignes communs aux deux programmes seront remplacés par ceux du deuxième programme. Il faut impérativement que les programmes ne soient pas protégés.

CHR$
CHR$ <nombre entier>

```
10 FOR i=32 to 255
20 PRINT i;chr$(i),
30 NEXT i
RUN
```

Convertit un code ASCII en son caractère équivalent. Sur AMSTRAD, les codes de 0 à 31 sont des caractères de contrôle, c'est pourquoi l'exemple ne commence qu'à partir de 32.

CINT
CINT <expression numérique>

ex:
```
10 x=2.5999
20 PRINT CINT(n)
RUN
3
```

Donne l'entier arrondi, compris entre -32768 et 32767, de l'expression numérique

CLEAR
CLEAR

Remet à zéro toutes les variables, fichiers ouverts, tableaux et fonctions, le mode de calcul s'effectue en radians

CLEAR INPUT
CLEAR INPUT

ex:
```
10 CLS
20 PRINT "appuyer sur plusieurs touches"
30 FOR i=1 TO 2000:NEXT i
40 CLEAR INPUT
RUN
```

Efface le tampon mémoire du clavier. Lancez le programme ci-dessus puis supprimez la ligne 40 (en tapant son n°) et regardez la différence.

CLG

CLG (encre)

ex: CLG 3

Efface l'écran graphique avec la couleur spécifiée.

CLOSEIN

CLOSEIN

Ferme un fichier ouvert en lecture sur la disquette.

CLOSEOUT

CLOSEOUT

Ferme un fichier ouvert en écriture sur la disquette.

CLS

CLS (#<numéro de canal>)

```
ex :    10 PAPER #1,2
        20 CLS #1
        RUN
```

Efface la fenêtre d'écran donnée par le n° de canal en lui donnant sa couleur de papier. Le n° 0 est toujours pris par défaut.

CONT
CONT

Permet de CONTinuer l'exécution d'un programme après un STOP, ou deux pressions sur la touche ESC si le programme n'a bien sûr pas été, ni modifié ni protégé.

COPYCHR$
COPYCHR$ (#<numéro de canal>)

ex:
```
10 CLS
20 PRINT "hello"
30 FOR i=1 to 5
40 LOCATE 1,1
50 a$=a$+COPYCHR$(#0)
60 NEXT i
70 LOCATE 10,20
80 PRINT a$:END
RUN
```

Copie dans une variable alphanumérique le caractère qui se trouve à la position du curseur. Le numéro de canal doit toujours être spécifié. Si le caractère n'est pas reconnu, la variable recoit une chaîne nulle.**COS**
COS <expression numérique>

ex:
```
DEG:PRINT COS(90)
0
```

Retourne le COSinus de l'expression. Vous devez spécifier avec les commandes DEG et RAD, si le résultat sera en degrés ou en radians.

CREAL
CREAL <expression numérique>

ex:
```
10 x=78/3.14
20 PRINT CREAL(x)
run
24.8407643
```

Convertit une expression numérique en nombre réel.

CURSOR
CURSOR (<système>), (<utilisateur>)

ex:
```
10 CLS:y=10:y=10:CURSOR 1
20 PRINT " a allume le curseur n l'éteint"
30 LOCATE x,y:a$=inkey$
40 x=int(RND*100):if x<=0 or x>40 then 40
50 y=int(RND*100):if y<=0 or y>24 then 50
50 IF a$="n" THEN CURSOR 0
70 IF a$="a" THEN CURSOR 1
80 GOTO 30
RUN
```

Active ou désactive le curseur en mettant 0 ou 1 pour le systéme et l'utilisateur. La commande INPUT active automatiquement le curseur, la commande INKEY$ le désactive.
Pour afficher un texte, il faut que le curseur soit éteint. Si les paramètres sont omis, l'état du curseur reste inchangé.

DATA

DATA <liste de constantes>

ex:
```
10 CLS:FOR I=1 to 5
20 READ prenom$,age
30 PRINT prenom$;" a ";age;" ans"
40 NEXT I
50 DATA "william","bob","marcel","edith","simone"
60 DATA 10,8,30,65
RUN
```

Déclare des constantes à l'intérieur d'un programme. Ces données sont lues par la commande READ qui les affecte à une variable et passe à la suivante.

DEC$

DEC$ <expression numérique>, <modèle de format>

ex:
```
PRINT DEC$((78/3.14),"+** ****")
+24.8408
```

Donne la représentation DECimale de l'expression numérique dans le format indiqué. L'emploi de ce format est décrit dans la définition de la commande PRINT USING.

DEF FN

DEF FN <nom> (<paramètres>)=<expression>

ex:
```
10 CLS:t=time/300
20 DEF FNchrono=int(time/300-t)
30 PRINT "vous devez taper la lettre à l'écran le plus vite
possible"
40 GOSUB 200
50 Q=int(rnd*120)
60 IF q<97 OR q>122 THEN 50
70 LOCATE 20,10:PRINT CHR$(q)
80 A$=INKEY$:IF A$="" THEN 80
90 IF ASC(a$)<>q THEN 80
100 PRINT "gagné en ";fnchrono ;" secondes"
110 GOSUB 200:T=time/300:CLS:GOTO 50
200 PRINT "si vous êtes prêt tapez ESPACE"
210 k$=INKEY$:if k$<>" " THEN 210 ELSE RETURN
RUN  (pour arrêter le programme vous tapez deux fois sur ESC)
```

Permet de DEFinir une FoNction retournant une valeur unique. Dans le programme d'exemple la fonction est constamment mise à jour, même si le programme est suspendu par ESC ou arrêté par double ESC, puis relancé.

DEFINT

DEFINT <liste de variables concernées>

ex:
```
10 DEFINT e
20 exemple=pi
30 PRINT exemple
RUN
3
```

Definit le type des variables par DEFaut,ici le type est entier. Lorsqu'une variable intervient sans marqueur (! % $), le type par défaut est automatiquement appliqué. Cette commande définit le type des variables par leur première lettre du nom. Elle peut se définir comme ceci:
DEFINT a,b,c
ou par une fourchette:
DEFINT a-z

DEFREAL
DEFREAL <liste de variables concernées>

Même chose que pour DEFINT, sauf que le type par défaut est REel

DEFSTR
DEFSTR <liste de variables concernées>

ex:
```
10 DEFSTR N
20 nom="AMSTRAD 6128"
30 PRINT nom
RUN
AMSTRAD 6128
```

Même chose que pour DEFINT, sauf que le type par défaut est une chaîne de caractères

DEG
DEG

Etablit le mode de calul en DEGrés. Par défaut le mode est en radians. Cette commande reste valable jusqu à ce qu'on utilise les commandes RAD, NEW, CLEAR ou RUN.

DELETE
DELETE (numéros de lignes)

ex: DELETE 50-150

Efface une partie du programme définie par les numéros de lignes. Vous n'êtes pas obligés de définir le début ou la fin de l'effacement.

ex: DELETE -200
Efface du début jusqu'à la ligne 200 incluse.

ex: DELETE 50-
Efface de la ligne 50 incluse jusqu'à la fin .

ex: DELETE
Efface tout le programme

DERR
DERR

EX: LOAD"inconnu"
inconnu.bas not found
READY
PRINT DERR
146

Donne le dernier code d'ERReur envoyé par la Disquette.
Reportez-vous à la liste des messages d'erreurs pour consultation.

DI
DI

EX:
```
10 CLS:TAG
20 EVERY 10 GOSUB 100
30 x1=RND*320:x2=RND*320
40 y=200+RND*200:c$=CHR$(RND*255)
50 FOR x=320-x1 TO 320*2 STEP 4
60 DI
70 MOVE 320,0,1:MOVE x-2,y:MOVE x,y
80 PRINT " ";c$:FRAME
90 EI:NEXT x:GOTO 20
100 MOVE 320,0:DRAW x+8,y-16,1:RETURN
RUN
```

Désactive une interruption , autre que ESC, jusqu'à ce qu'elle soit réactivée par la commande EI ou indirectement par un RETURN à la fin du sous-programme d'interruption GOSUB.

L'entrée dans un sous-programme d'interruption désactive automatiquement les interruptions de priorité égale ou inférieure.

DIM

DIM <variable indicée>

ex:
```
10 DIM nom$(5),note(5)
20 FOR i=1 TO 5
30 PRINT "éléve no ";i
40 INPUT "entrez son nom$";nom$(i)
50 INPUT "entrez sa note";note(5)
60 PRINT
70 NEXT i
80 CIS:FOR i=1 TO 5
90 PRINT I;NOM$(I),note(I)
100 NEXT I
```

Cette commande DIMensionne un tableau, elle alloue l'espace requis et donne les valeurs d'indices maximales. En l'absence de spécification le BASIC met 10 comme valeur par défaut. La valeur minimale d'un indice est zéro.

Les tableaux peuvent être à plusieurs dimensions, chaque élément est dans ce cas référencé par sa position .

ex: DIM essai$(10,10,10)

Un élément sera référencé par : essai$(1,3,5)

DRAW
DRAW <coordonnée x>, <coordonnée y>, (encre), (mode d'encre)

ex:
```
10 MODE 0:BORDER 0:PAPER 0:INK 0,0
20 X=RND*640:y=RND*400:z=RND*15
30 draw x,y,z
40 GOTO 20
RUN
```

Trace une ligne à l'écran entre la position du curseur jusqu'à une position déterminée par les coordonnées x et y.
L' encre falcultative varie entre 0 et 15.
Le mode d'encre, falcultatif lui aussi, précise l'intéraction de l'encre sur l'affichage présent à l'écran, les quatre modes d'encre sont:

0: Normal
1: XOR (OU exclusif)
2: AND (ET)
3: OR (OU)

DRAWR
DRAWR <décalage x>, <décalage y>, (encre), (mode d'encre)

ex:
```
10 CLS:MOVE 200,200:DRAW 0,200:MOVE 200,200:FOR N=1 TO 15
20 DRAWR 10,0:DRAW 0,-10
30 NEXT
40 DRAWR 200,0
50 GOTO 50
RUN
```

Même chose que pour DRAW sauf que DRAWR part à partir du curseur graphique et effectue un décalage spécifié par x et y.

EDIT
EDIT <numéro de ligne>

ex: EDIT 10

Affiche à l'écran la ligne spécifiée, prête à être modifiée

EI
EI

Active une interruption désactivée par la commande DI.

ELSE (se reporter à la commande IF).

END
END

Termine l'exécution d'un programme et rétablit le mode direct. Cette commande est implicite à la fin de tout programme du BASIC.

ENT
ENT <numéro d'enveloppe> (5 sections d'enveloppe)

ex : 10 ENT 1,25,-10,20,25,10,20
20 SOUND 1,150,200,1,1
RUN

Définit l'Enveloppe de Tonalité spécifiée par le numéro d'enveloppe, entre 0 et 15, utilisée avec la commande SOUND. Si le numéro d'enveloppe est négatif (entre -1 et-15) l'enveloppe se répète jusqu'à la fin de la durée spécifiée par SOUND.
Chaque section d'enveloppe peut contenir 2 ou 3 paramètres. Ceux-ci sont :

A) Dans le cadre de trois paramètres:

1 nombre de pas
2 amplitude de pas
3 durée de pas

Nous allons les examiner plus en détails:

NOMBRE DE PAS:
Spécifie le nombre de pas de variation de tonalité à l'intérieur de la section d'enveloppe. Exemple: dans une section de note durant 5 secondes , vous pouvez fixer 5 pas de 1 seconde chacun, ainsi le nombre de pas sera donc de 5.
Sachez que le nombre de pas varie de 0 à 239.

AMPLITUDE DE PAS:
Est compris entre -128 à +127. Les pas positifs abaissent la hauteur de la note, les pas négatifs l'augmentent. La période minimale est zéro.

DUREE DE PAS:
Donne la durée d'un pas en unités de 0,01 seconde, peut varier de 0 à 255. La durée maximale d'un pas est donc de 2,56 secondes.

B) AVEC DEUX PARAMETRES SEULEMENT:

PERIODE SONORE:
Donne la nouvelle valeur de la période .

DUREE DE PAS:
Même définition que pour trois paramètres.
La commande ENT peut être accompagnée par 5 sections d'enveloppe différentes, qui peuvent bien sûr avoir deux ou trois paramètres.
Le premier pas d'une enveloppe de tonalité s'exécute immédiatement.
Si un numero d'enveloppe est déja attribué une nouvelle attribution efface l'ancienne définition.

ENV
ENV <numéro d'enveloppe> (5 sections d'enveloppe)

```
ex :    10 ENV 1,25,-10,20,25,10,20
        20 SOUND 1,150,200,1,1
        RUN
```

Définit l'ENVeloppe de Volume correpondant au numéro d'enveloppe, entre 1 et 15 utilisé par la commande SOUND.

Comme pour ENT, trois paramètres peuvent êtres définis.

A) Dans le cadre de trois paramètres :

1 nombre de pas
2 amplitude du pas
3 durée du pas

NOMBRE DE PAS:
Même définition que pour la commande ENT sauf qu'ici le paramètre varie de 0 à 127.

AMPLITUDE DE PAS:
Peut faire varier le volume de 0 à 15 par rapport au pas précédent. Les 16 volumes différents sont les mêmes que ceux de la commande SOUND. Le paramètre amplitude peut varier de -128 à +127, le volume revenant à 0 après avoir dépasser 15.

DUREE DU PAS:
Exactement le même emploi que pour ENT.

B) Avec deux paramètres seulement:

1 enveloppe matérielle
2 période de l'enveloppe

ENVELOPPE MATERIELLE:
Spécifie la valeur à envoyer au registre d'enveloppe contenu dans le générateur sonore.

PERIODE DE L'ENVELOPPE:
Spécifie la valeur à envoyer aux registres de période d'enveloppe.
L'utilisation d'enveloppes matérielles suppose la connaissance du matériel. Si vous ne l'avez pas, nous vous conseillons d'utiliser une enveloppe logicielle intégrant un paramètre durée de pas adéquat.

La commande ENV peut contenir 5 sections d'enveloppes différentes avec deux ou trois paramètres.

EOF
EOF

```
ex:     10 OPENIN "essai"
        20 WHILE NOT EOF
        30 INPUT#9,A$
        40 PRINT A$
        50 WEND:CLOSEIN
        RUN
```

Permet de lire un fichier dont on ne connait pas la longueur.
EOF (End Of File) donne vrai quand la fin du fichier est détectée.

ERASE
ERASE <liste de variables>

```
ex:     10 CLS:PRINT FRE(0)
        20 DIM ESSAI$(100)
        20 PRINT"avant ERASE ";FRE(0)
        30 ERASE essai$
        40 PRINT"après ERASE ";FRE(0)
        50 END
        RUN
```

Permet de récupérer la place en mémoire d'un tableau, quand celui-ci est devenu inutile.

ERL
ERL

ex:
```
10 ON ERROR GOTO 100
20 GOTO 200
100 PRINT " l'erreur se trouve en ";ERL
110 END
RUN
```

Retourne le numéro de Ligne où une ERreur a été détectée.

ERR
ERR

ex: Dans l'exemple de programme pour ERL, vous rajoutez la ligne suivante:

```
105 PRINT"erreur no "; ERR
RUN
```

Retourne le numéro de le dernière ERReur. Dans notre exemple le n° est 8, ce qui correspond à "line does not exist".

ERROR
ERROR <nombre entier>

ex:
```
10 ON ERROR GOTO 50
20 INPUT a$:a=ASC(a$)
30 IF a<97 OR a>123 THEN ERROR 100
40 GOTO 20
100 PRINT "erreur de saisie ":RESUME 20
RUN
```

Cette commande simule une erreur de BASIC et agit comme si elle était réelle en rapportant les valeurs appropriées d'ERR et ERL.

EVERY
EVERY <temps>, (numéro), GOSUB <numéro de ligne>

```
ex:     10 EVERY 15,1:GOSUB 30
        20 SOUND 1,45:GOTO 20
        30 SOUND 1,35
        40 RETURN
```

Appelle à intervalles réguliers un sous-programme spécifié par le GOSUB. Le temps est indiqué en unités de 0,02 seconde. Quatre numéros peuvent être définis, de 0 à 3, avec une priorité supérieure pour 3 et inférieure pour 0.

EXP
EXP <expression numérique>

```
ex:     PRINT EXP(5.85)
        347.23438
```

Retourne "e" à la puissance donnée par l'expression numérique où "e" est égal à 2.7182818 environ, le nombre dont le logarithme naturel est 1.

FILL
FILL <encre>

```
ex:     10 MODE 0:FOR i=1 TO 500
        20 PRINT "q";:NEXT i
        30 couleur=2+RND*13:FILL couleur
        40 GOTO 30
        RUN
```

Colorie une zone de l'écran graphique. Les bords de la zone sont délimités par les lignes dessinées avec l'encre du stylo en cours ou avec l'encre du fond, entre 0 et 15. Si le curseur graphique se trouve sur un bord de l'écran, rien n'est rempli

FIX
FIX <expression numérique>

ex: FIX (7.899999)
7

Retourne la partie entière de l'expression numérique en arrondissant toujours par défaut.

FOR
FOR <variable numérique>=<début> TO <fin> (STEP <incrément>)

ex: 10 k=10
20 FOR I=1 TO k STEP 2
30 PRINT I;
40 K=K+10
50 PRINT K
60 NEXT

Exécute les lignes qui se trouvent entre les mots FOR et NEXT autant de fois que la variable augmentée de l'incrément n'est pas égale au paramètre de fin. L'incrément prend toujours par défaut la valeur 1.

Si l'incrément est négatif, la valeur de début doit être supérieure à la valeur de fin; si ce n'est pas le cas la variable ne peut être incrementée. Les boucles FOR NEXT peuvent être imbriquées.

ex: 10 FOR I=1 TO 10
20 For j=1 TO 2
30 PRINT I,j
40 NEXT J
50 NEXT I

Il n'est pas nécessaire de mettre le nom de la variable dans la commande NEXT, le BASIC la détermine automatiquement .

FRAME
FRAME

EX:
```
10 MODE 0:PRINT "sans frame"
20 TAG
25 MOVE 0,200
30 FOR x=0 TO 500 STEP 4
40 IF f=1 THEN FRAME
50 MOVE X,200
55 PRINT " ";CHR$(143);
60 NEXT X
65 IF F=1 THEN RUN
70 CLS
75 TAGOFF
80 PRINT "avec FRAME"
85 f=1
90 goto 20
RUN
```

Synchronise l'écriture des graphiques avec les trames vidéo pour éviter les distorsions ou le scintillement.

FRE
FRE (0)
FRE (" ")

Donne en octets l'espace mémoire disponible.
L'option FRE (" ") force l'ordinateur, à effacer les variables et tableaux inutilisés, avant de donner la valeur.

GOSUB
GOSUB 1000

Appelle un sous-programme en se branchant sur la ligne indiquée. La fin du sous-programme doit toujours comporter un RETURN renvoyant à l'instruction suivant la commande GOSUB.

GOTO
GOTO 1000

Saut à la ligne indiquée, sans condition.

GRAPHICS PAPER
GRAPHICS PAPER <encre>

ex:
```
10 MODE 0
20 MASK 15
30 GRAPHICS PAPER 4
40 DRAW 500,0
RUN
```

Permet de déterminer l'encre du fond. Lors du traçage de lignes, le fond n'est pas visible.
L'encre de fond, qui va de 0 à 15, fait office de valeur par défaut lors de l'effacement par la commandes CLG.

GRAPHICS PEN
GRAPHICS PEN <encre>,(type du fond)

```
ex      10 MODE 0
        20 GRPHICS PEN 10
        30 MOVE 100,0
        40 DRAW 100,200
        50 MOVE 639,0
        60 FILL 10
        RUN
```

Permet de fixer la couleur pour le dessin des lignes, entre 0 et 15.
Deux types de fond vous sont proposés :

0: fond opaque
1: fond tansparent.

Dans cette commande une seule option est obligatoire.

HEX$
HEX$ <nombre entier sans signe>, <largeur de la zone>

```
ex:     PRINT HEX$(255,4)
        00FF
```

Retourne une série de chiffres HEXadécimale représentant la valeur du nombre entier, sur autant de caractères qu'indique la largeur de la zone dans la mesure où elle n'est pas trop petite, si c'est le cas le résultat se compose d'autant de caractères que nécessaires.

HIMEM
HIMEM

ex: PRINT HIMEM
42619

Retourne l'adresse la plus Haute de la mémoire utilisée en BASIC.

IF
IF <expression logique> THEN <action> (ELSE action)

ex:
```
10 CLS:x=CINT(RND*100):x=0:y=100
20 PRINT "vous devez trouvez un nombre"
30 LOCATE 20,10:PRINT "compris entre ";x; " et ";y:c=c+1
40 INPUT n
50 IF N=a THEN "BRAVO TROUVE EN "; c;" COUPS":END
60 IF n<a AND n>x THEN x=n:GOTO 30
70 IF N>a AND n<y THEN y=n
80 GOTO 30
RUN
```

Teste si l'expression logique est vraie, si c'est le cas elle exécute la première option, dans le cas contraire elle exécute l'action placée après le ELSE, en son absence le BASIC passe à la ligne suivante.

Chaque commande IF THEN peut être imbriquée mais doivent se terminer à la fin de la ligne. Si le résultat nécessite un saut de ligne, plusieurs syntaxes sont possibles:

IF x=1 THEN 20
ou
IF x=1 GOTO 20
ou encore:
IF x=1 THEN GOTO 20

INK

INK <encre>, <n° de couleur>(,<n° de couleur>)

Ex:
```
10 MODE 1
15 PAPER 0
17 PEN 1
20 FOR J=0 TO 1:REM  ENCRE
30 FOR I=0 TO 26: REM COULEUR
40 INK J,I
50 LOCATE 14,10
55 PRINT "ENCRE";J;",";I
60 FOR K=1 TO 500
65 NEXT K,I,J:REM BOUCLE ATTENTE
70 INK 0,1
75 INK 1,24
80 CLS:REM RETOUR AUX VALEURS INITIALES
```

Détermine la ou les couleurs d'une encre donnée.
Le paramètre <encre> donne la référence de l'encre (un entier de 0 à 15), correspondante aux commandes PEN ou PAPER.
Le premier paramètre <n° couleur> donne une valeur de couleur (entier de 0 à 26).
Le second paramètre facultatif détermine la 2ème couleur et l'encre passera d'une couleur à l'autre selon la vitesse définie par la commande SPEED INK.

INKEY
INKEY (<nombre entier>)

Ex : 10 IF INKEY(55) = 32 THEN PRINT "vous venez d'appuyer sur la touche SHIFT et V"
20 CLEAR INPUT :REM ANNULE BUFFER
30 GOTO 10

Le clavier est analysé tous les 60èmes de seconde. Cette fonction est utilisée pour la détection de la position haute ou basse des touches, ainsi que l'état des touches SHIFT et CONTROL.

Dans l'exemple ci-dessus, le numéro de touche correspond à la touche V (vous avez le numéro des touches sur le boîtier de votre ordinateur), et la valeur 32 correspond à l'enfoncement de la touche SHIFT.

VALEUR	SHIFT	CONTROL	TOUCHE
-1	INDIFFERENT	INDIFFERENT	RELEVEE
0	RELEVEE	RELEVEE	ENFONCEE
32	ENFONCEE	RELEVEE	ENFONCEE
128	RELEVEE	ENFONCEE	ENFONCEE
160	ENFONCEE	ENFONCEE	ENFONCEE

INKEY$
INKEY$

Ex : 10 CLS: REM EFFACE ECRAN
20 PRINT "TAPEZ O(ui) ou N(on) "
30 A$=INKEY$: IF A$ = "" THEN 30
40 IF A$ = "o" OR A$ = "O" THEN 70
50 IF A$ = "n" OR A$ = "N" THEN 80
60 GOTO 30
70 PRINT "Vous avez tapé O(ui)":END
80 PRINT "Vous avez tapé N(on)

Cette fonction renvoie les caractères tapés au clavier. Si aucune touche n'est tapée, la fonction renvoie une chaîne vide.

INP
INP(<n° du port>)

Ex : PRINT INP(&FF77)
255

Cette fonction lit la valeur contenue dans un port d'entrée/sortie dont l'adresse est le numéro de port.

INPUT

INPUT (#<n° de canal>,)(;)(<chaîne> <séparateur>)<variable>(<variables>)

Ex : 10 MODE 1
20 INPUT "Tapez deux nombres à ajouter en les séparant par une virgule " ;a,b
30 PRINT a;" plus ";b;" = ";a+b
40 goto 20

Cette commande reçoit du canal spécifié les données (Canal #0 par défaut).

Le point-virgule ";" aprés INPUT supprime le passage à la ligne aprés exécution de la commande.

Le <séparateur> est soit la virgule ou le point-virgule. Le point-virgule fait apparaître un point d'interrogation.

Si l'entrée ne correspond pas avec le type de variable (une chaîne de caractères pour une variable numérique), vous aurez le message suivant :

?REDO FROM START (Vous pouvez modifié ce message avec le ONERROR.)

Toute saisie au clavier doit se terminer par RETURN pour l'INPUT.

INSTR
INSTR (<position de départ>, <chaîne>, <chaîne à rechercher>)

Ex :
```
10 CLS
20 A$="essai de recherches"
30 B$="es"
40 PRINT "La première séquence des lettres 'es' est";
INSTR(A$,B$)
50 PRINT "La deuxième séquence des lettres 'es' est";
INSTR(2,A$,B$)
```

Cette fonction permet de retrouver l'existence et la place d'une chaîne de caractères dans une autre chaîne.
La position de départ est facultative et sa valeur par défaut est 0.
Si l'AMSTRAD ne trouve pas la chaîne de recherche, la fonction prend la valeur 0.

INT
INT (<expression numérique>)

Ex :
```
PRINT INT(-2.774)
-3
```

Cette fonction arrondit à l'entier immédiatement inférieur. Identique à FIX pour les nombres positifs, l'arrondi pour un nombre négatif est de 1 de moins que FIX.

JOY
JOY (<nombre entier>)

Ex : 10 PRINT "ACTIONNEZ LA MANETTE DE JEU"
20 IF JOY(0) = 0 THEN 10
30 PRINT "VOUS AVEZ ACTIONNE LA MANETTE DE JEU"

Cette fonction permet de connaître l'état de la manette de jeu (JOYSTICK) spécifiée par le chiffre 0 ou 1.

Le résultat a une signification binaire :

DECIMAL	BIT
1	0 : Haut
2	1 : Bas
4	2 : Gauche
8	3 : Droite
16	4 : Tir 2
32	5 : Tir 1

Si JOY(0) a la valeur 24, la décomposition binaire donne 16 (Tir 2) + 8 (Droite), donc vous avez appuyé sur le bouton de tir de la première manette en la déplaçant vers la droite.

KEY

KEY <n° logique de touche>, <chaîne alphanumérique>

Ex : KEY 11,"BORDER 13: PAPER 0: PEN 1: INK 1,0: MODE 2. LIST "+ chr$(13)
Appuyer sur ENTER.

Cette commande associe une chaîne de caractères à une touche du clavier. Il y a 32 numéros logiques de touche (0 à 31), occupant les touches 128 à 159.

Les touches 128 (0 du clavier numérique) à 140 (CONTROL ENTER du clavier numérique) sont associées par défaut aux chiffres 0 à 9, au point décimal, à RETURN et à RUN RETURN , mais peuvent être associées à d'autres chaînes si nécessaire.

Les autres numéros logiques de touche 13 à 31 (touche 141 à 159) sont affectés à des chaînes vides par défaut mais peuvent être modifiés et associés à d'autres touches avec la commande KEY DEF.

Le parametre <n° logique de touche> doit être compris entre 0 et 31 ou entre 128 et 159 pour correspondre aux numéros physiques des touches du clavier numérique.

La chaîne associée ne doit pas dépasser 120 caractères sinon vous aurez une erreur "IMPROPER ARGUMENT" (argument incorrect)

KEY DEF
KEY DEF <n° de touche>, <répétition>(, <normal>(, <shift>(, <control>)))

Ex : KEY 159,"touche TAB"
KEY DEF 68,1,159
Appuyez sur la touche TAB

Cette commande définit la valeur logique d'une touche (KEY) par son numéro physique (0 à 79).

Les paramètres <normal>, <shift> et <control> doivent contenir les numéros logiques correspondant aux valeurs à envoyer, selon que la touche est enfoncée seule (normal), avec SHIFT ou avec CONTROL. Ces paramètres sont facultatifs.

Le paramètre <répétition> active ou désactive la fonction d'auto répétition (1 ou 0). La vitesse de celle-ci est réglable avec la commande SPEED KEY. Dans l'exemple, la touche 159 (numéro logique 31) est d'abord associée à une chaîne et ensuite avec la commande KEY DEF, on définit que la touche n° 68 (TAB) enverra la chaîne définie sous le n° 159 quand elle sera enfoncée seule et que l'auto répétition est active.

Pour revenir au mode normal : KEY DEF 68,0,9
9 est la valeur ASCII normale de TAB

LEFT$
LEFT$ (<chaîne>, <longueur>)

Ex : 10 CLS
20 A$="AMSTRAD 6128"
30 FOR K=1 TO 12 : PRINT LEFT$(A$,K): NEXT

Cette fonction extrait la partie gauche d'une chaîne alphanumérique en spécifiant le nombre de caractères voulus. Si la longueur donnée est plus longue que la chaîne, celle-ci est utilisée entièrement.

LEN
LEN (<chaîne>)

Ex : 10 LINE INPUT "DONNEZ UNE SUITE DE MOTS";a$
20 PRINT"La phrase est longue de",len(a$)," caractères."

Cette fonction donne le nombre exact de caractères de la chaîne.

LET
LET <variable>=<expression>

EX : LET X=100

Cette commande est un reste des premiers BASIC, pour la compatibilité avec les programmes antérieurs. En AMSTRAD BASIC la commande LET est inutile. X=100

LINE INPUT
LINE INPUT*(<n° canal>,)(;)(<chaîne><séparateur>)<variable chaîne>

Ex : 10 LINE INPUT "TAPEZ UNE LIGNE DE TEXTE AVEC DES VIRGULES";a$
20 CLS
30 PRINT "LA LIGNE EST :"
40 PRINT A$

Cette commande reçoit une ligne entière en provenance du canal indiqué (canal 0 par défaut). Le point-virgule ";" facultatif enlève le saut de ligne suivant l'exécution de l'instruction.

Le <séparateur> est soit un point-virgule ou une virgule, le premier donne l'affichage d'un point d'interrogation. La commande se termine après la frappe de la touche RETURN.

LINE INPUT avec le canal 9 de la disquette (ou cassette) se termine par un retour chariot ou par l'affectation de 255 caractères au plus dans la variable.

LIST
LIST (<lignes>) (,<n° canal>)

Ex : LIST 100-200,#1

Cette commande liste le programme en mémoire sur le canal désiré. 0 pour l'écran (valeur par défaut), 8 pour l'imprimante. La liste peut être interrompue avec la touche ESC, puis reprise avec la barre d'espacement.

Si vous appuyez deux fois sur ESC, la liste est arrêtée et vous revenez au mode direct.

Vous pouvez ne mettre que le premier numéro ou le dernier avec le "-" devant, si vous voulez lister le programme jusqu'à la fin ou depuis le début. Ex : LIST -200 ou LIST 100-

LOAD
LOAD <nomfich>(, <adresse>)

Ex : LOAD "FICESSAI.ABC",&3ABC

Cette commande charge en mémoire un programme BASIC de la disquette en effaçant tout programme en place.

L'option <adresse> permet de charger un fichier binaire à l'adresse indiquée et non pas à l'adresse où se trouvait ce fichier lors de sa sauvegarde.

Un programme protégé ne peut pas être chargé par la commande LOAD. IL faut utiliser RUN ou CHAIN.

LOCATE

LOCATE (#n° canal>,)<coordonnée X>, <coordonnée Y>

Ex : 10 MODE 1
20 FOR K=1 TO 20
30 LOCATE N,N: PRINT CHR$(143);"POSITION";n;",";n
40 NEXT

Cette commande déplace le curseur texte vers une nouvelle position définie par les coordonnées relatives au coin supérieur gauche de l'écran.
0 est le canal par défaut.

LOG

LOG (<expression numérique>)

Ex : PRINT LOG(9999)
9.21024037

Cette fonction correspond au logarithme naturel d'une expression numérique (>0).

LOG10

LOG10 (<expression numérique>)

Ex : PRINT LOG10(9999)
3.99995657

Cette fonction calcule le logarithme en base 10 de l'expression numérique (>0).

LOWER$
LOWER$ (<chaîne>)

Ex : 10 A$="CHANGEMENT MAJUSCULE/MINUSCULE"
20 PRINT LOWER$(A$)

Cette fonction change toutes les majuscules en minuscules.

MASK
MASK (<nombre entier>)(, <premier point>)

Ex : 10 MODE 0
12 INK 5,21
14 INK 8,16
20 MOVE -100*RND,400*RND
30 WHILE XPOS < 640
40 FOR X=1 TO 8
50 MASK 2^(8-x)
60 DRAWR 32,0,X,1
65 MOVER -32,0
70 NEXT
80 MOVER 34,0
90 WEND:
100 GOTO 20

Cette commande définit le modèle à utiliser pour le tracé des lignes. La valeur binaire du nombre entier (0 à 255) active (1) ou désactive (0) les bits dans chaque groupe contigu de 8 pixels.

Le paramètre premier point détermine si le premier point de la ligne doit être tracé (0 ou 1).

Vous devez spécifier au moins l'un des deux paramètres.

MAX
MAX (<liste de:<expression numérique>)

Ex : PRINT MAX(1,2,55,77,5,10)
77

Cette fonction donne la plus grande valeur de la liste.

MEMORY
MEMORY <adresse>

Ex : MEMORY &2ABC

Cette commande sert à définir la limite supérieure utilisable par la mémoire BASIC, en donnant l'adresse de l'octet le plus élévé.

REMARQUE : Le basic n'exploite que bloc 0 de la mémoire.

MERGE
MERGE <nomfich>

Ex : MERGE "ficnom.bas"

Cette commande charge un programme de la disquette et le mélange au programme déjà en mémoire.

Les numéros de lignes du premier programme identiques au nouveau sont écrasés et remplacés par le nouveau.

Les fichiers protégés ne peuvent pas être fusionnés avec un autre programme.

MID$
MID$ (<chaîne>, <position départ>(,<longueur>))

Ex : PRINT MID$("ABCDEF",3,2)
CD

Cette fonction envoie une nouvelle sous-chaîne commençant de la position de départ de la chaîne et contenant le nombre de caractères correspondant à la longueur.

Si le dernier paramètre n'est pas spécifié, la fonction renvoie le reste de la chaîne depuis la position de départ.

Si la position de départ est supérieure à la longueur de la chaîne, une chaîne vide est renvoyée. La position de départ est comprise entre 1 et 255, la longueur entre 0 et 255.

MID$
MID$ (<variable chaîne>,<position d'insertion>(,<longueur>))=<nouvelle chaîne>

Ex : 10 A$="BONJOUR"
20 MID$(a$,3,2)="YY"
30 PRINT a$
BOYYOUR

Cette commande insère dans la chaîne de départ une nouvelle chaîne à partir du point d'insertion. Lorsque vous utilisez MID$ en tant que commande, vous devez utiliser une variable chaîne et non une constante.

MIN
MIN (<liste de:<expression numérique>)

```
Ex :    PRINT MIN(3,57,45,999,1,56)
        1
```

Cette fonction donne la valeur minimale de la liste.

MOD
<argument 1> MOD <argument 2>

```
Ex :    PRINT 10 MOD 3, 10 MOD 5
        1            0
```

L'opérateur MOD donne le reste de la division entière de l'argument 1 par l'argument 2.

MODE
MODE <nombre entier>

```
Ex :    10 FOR M = 0 TO 2
        20 MODE M
        30 PRINT "CECI EST LE MODE";M
        40 PRINT "APPUYEZ SUR UNE TOUCHE"
        50 IF INKEY$ = "" THEN GOTO 50
        60 NEXT
```

Cette commande modifie le mode d'écran (0,1 ou 2) et rétablit sur l'écran l'encre 0. Toutes les fenêtres et curseurs sont réinitialisés.

MOVE
MOVE <coordonnée X>, <coordonnée Y>(, (<encre>)(, <mode d'encre>))

Ex :
```
10 MODE 1: TAG
20 X=RND*800-100:Y=RND*430
30 MOVE X,Y
40 PRINT "HELLO";
50 GOTO 20
```

Cette commande positionne le curseur graphique au point absolu spécifié. Le paramètre facultatif <encre> (0 à 15) permet de modifier la couleur du stylo graphique.

Le paramètre facultatif <mode d'encre> détermine l'interaction de l'encre sur l'affichage en place à l'écran. Il existe 4 modes d'encre.

0: normal 1: XOR (OU exclusif) 2 : AND (ET) 3 : OR (OU)

MOVER
MOVER <décalage x>,<décalage y>(,(<encre>)(,<mode d'encre>))

Ex :
```
10 MODE 1: TAG: MOVE 0,16
20 PRINT "ESSAI VERS LE";
30 FOR N=1 TO 10
40 MOVER -45,16
50 PRINT "HAUT";:NEXT:PRINT" ET ";
60 FOR N=1 TO 10
70 MOVER -64,-16
80 PRINT"LE BAS";:next
```

Cette commande positionne le curseur graphique en coordonnées relatives par rapport à la position actuelle. Les paramètres facultatifs <encre> et <mode d'encre> ont la même utilisation que dans la commande MOVE.

NEW
NEW

Cette commande efface le programme et les variables en mémoire. Les définitions des touches ne sont pas effacées et le mode d'affichage est inchangé.

NEXT
NEXT (<liste de :<variables>)

Ex :
```
10 FOR K=1 TO 10
20 FOR L=0 TO 20
30 MODE 1
32 PEN K
34 BORDER L
40 PRINT "PEN";K;"BORDER";L
50 FOR M=1 TO 500
60 NEXT M
70 NEXT L,K
```

Cette commande indique la fin d'une boucle FOR.

La commande NEXT peut être seule ou accompagnée de variables se rapportant au FOR.

La liste de variables doit apparaître en sens inverse des FOR, afin d'éviter des chevauchements.

NOT
NOT <argument>

```
Ex :    PRINT NOT -1, NOT 0
        0             -1
        IF NOT "ESSAI" <"BONJOURS" THEN PRINT "vrai" ELSE PRINT "faux"
        vrai
```

Cet opérateur exécute des opérations binaires sur des entiers.
Il inverse chaque bit de l'argument.

ON BREAK CONT
ON BREAK CONT

```
Ex :    10 ON BREAK CONT
        20 PRINT " APPUYER SUR LA TOUCHE ESC"
        25 PRINT
        30 FOR T=1 TO 1000
        40 NEXT
        50 GOTO 20
```

Cette commande désactive la touche ESC, empêchant l'arrêt du programme. Vous devez utiliser cette commande avec précaution en phase d'essai d'un programme car vous ne pouvez plus interrompre le programme en cas de bouclage infini que par la réinitialisation complète de l'ordinateur.

Vous pouvez désactiver ON BREAK CONT par ON BREAK STOP.

ON BREAK GOSUB
ON BREAK GOSUB <n° de ligne>

Ex :
```
10 ON BREAK GOSUB 40
20 PRINT "ça tourne"
30 GOTO 20
40 CLS
45 PRINT "APPUYER 2 FOIS ESC,";
50 PRINT"APPELLE LE SOUS PROGRAMME"
60 FOR T=1 TO 2000
65 NEXT
70 RETURN
```

Cette commande passe la main au sous-programme commençant par le n° de ligne spécifié si vous appuyez sur ESC deux fois.

ON BREAK STOP
ON BREAK STOP

Ex :
```
10 ON BREAK GOSUB 40
20 PRINT "ça tourne"
30 GOTO 20
40 CLS
50 PRINT "VOUS AVEZ APPUYE 2 FOIS SUR ESC"
60 FOR T=1 TO 1000
65 NEXT
70 ON BREAK STOP
80 RETURN
```

Cette commande désactive les commandes ON BREAK CONT et ON BREAK GOSUB pour permettre l'arrêt du programme.
Dans l'exemple ci-dessus, ON BREAK GOSUB ne fonctionne qu'une seule fois, car elle est désactivée en ligne 70 dans le sous-programme ON BREAK.

ON ERROR GOTO
ON ERROR GOTO <n° de ligne>

Ex :
```
10 ON ERROR GOTO 50
20 CLS
25 PRINT "En cas d'erreur, le programme est listé"
30 FOR T=1 TO 4000
35 NEXT
40 GOTO 100
50 PRINT "Erreur ligne ";ERL
60 PRINT
70 LIST
```

Cette commande détourne le programme vers la ligne spécifiée, dès qu'une erreur est détectée.

La comande ON ERROR GOTO 0 met hors fonction le déroutement du programme en cas d'erreur et rétablit le traitement normal des erreurs par BASIC.

Voir également la commande RESUME.

ON GOSUB

ON GOSUB <sélecteur> GOSUB <liste de:<n° de ligne>

Ex:
```
10 PAPER 0: PEN 1: INK 0,1
20 CLS: PRINT " MENU ": PRINT
30 PRINT "1 : CADRE": PRINT
40 PRINT "2 : STYLO": PRINT
50 PRINT "3 : MODE": PRINT
60 INPUT "Votre choix";K
70 ON K GOSUB  100, 200, 300
80 GOTO 20
100 B=B-1: IF B = -1 THEN B = 26
110 BORDER B: RETURN
200 P=P-1: IF P < 2 THEN P = 26
210 INK 1,P: RETURN
300 M=M-1: IF M = -1 THEN M = 2
310 MODE M: RETURN
```

Cette commande séléctionne le sous-programme en fonction de la valeur du sélecteur (0 à 255). La valeur du sélecteur détermine le numéro de ligne du sous-programme en fonction de la place de ce numéro dans la liste.

Dans l'exemple ci-dessus : 1 provoque le passage à la ligne 100, 2 le passage à la ligne 200 et 3 le passage à la ligne 300.

Si cette expression est égale à 0 ou si elle est supérieure au nombre de lignes de la liste spécifiée dans la commande, le programme passe à l'instruction suivante.

ON GOTO
ON <sélecteur> GOTO <liste de:<n° de ligne>

Ex :
```
10 PAPER 0: PEN 1: INK 0,1
20 CLS: PRINT " MENU ": PRINT
30 PRINT "1 : CADRE": PRINT
40 PRINT "2 : STYLO": PRINT
50 PRINT "3 : MODE": PRINT
60 INPUT "Votre choix";K
70 ON K GOTO  100, 200, 300
80 GOTO 20
100 B=B-1: IF B = -1 THEN B = 26
110 BORDER B: GOTO 10
200 P=P-1: IF P < 2 THEN P = 26
210 INK 1,P:  GOTO 10
300 M=M-1: IF M = -1 THEN M = 2
310 MODE M: GOTO 10
```

Cette commande est identique à la commande ON GOSUB, à la différence prés que vous ne sélectionnez pas un sous-programme mais seulement un branchement.

ON SQ GOSUB
ON SQ (<n° canal>) GOSUB <n° ligne>

Ex :
```
10 ENV 1,15,-1,1
20 ON SQ(1) GOSUB 60
30 MODE 0: ORIGIN 0,0,200,440,100,300
40 FOR X=1 TO 13: FRAME: MOVE 330,200,X
50 FILL X: NEXT: GOTO 40
60 READ S: IF S=0 THEN RESTORE: GOTO 60
70 SOUND 1,S,25,15,1
80 ON SQ(1) GOSUB 60: RETURN
200 DATA 50, 60, 90, 100, 35, 200, 24, 500, 0
```

Cette commande détourne le BASIC au n° de ligne spécifié en cas de place dans une file sonore (SOUND QUEUE). Le numéro de canal est un nombre entier avec les valeurs suivantes : 1 pour le canal A, 2 pour le canal B et 3 pour le canal C.

OPENIN
OPENIN <nomfic>

```
Ex :    10  REM OUVERTURE ET FERMETURE D'UN FICHIER DISQUETTE
        20 OPENIN "FICNOM"
        25 INPUT #9,A,A$
        30 CLOSEIN:
        35 PRINT "LES DEUX VALEURS SONT : ";a,a$
```

Cette commande ouvre un fichier existant sur la disquette en écriture.

Le fichier doit être écrit en ASCII.

L'exemple ne fonctionne que si vous avez au préalable créé le fichier avec la commande OPENOUT.

OPENOUT
OPENOUT <nomfic>

```
Ex :    10 REM OUVERTURE ET ECRITURE FICHIER DISQUETTE
        20 INPUT "NOMBRE ";A
        30 INPUT "MOT ";A$
        40 OPENOUT "FICNOM"
        50 WRITE #9,A,A$
        60 CLOSEOUT
        70 PRINT "LES DONNEES SONT SAUVEES SUR    DISQUETTE"
```

Cette commande ouvre un fichier sur disquette pour l'écriture.

OR
<argument> OR <argument>

Ex : PRINT 1 OR 1, 0 OR 0, 1 OR 0
1 0 1
IF "BONJOUR" > "DEMAIN" OR "HIERS" > "AUJOURD'HUI" THEN
PRINT "VRAI" ELSE PRINT "FAUX"
VRAI

Cet opérateur exécute des opérations booléennes sur des entiers.

Le résultat de la comparaison est 1 sauf si les deux bits sont égaux à 0.

ORIGIN
ORIGIN <X>,<Y>(,<gauche>, <droite>, <haut>, <bas>)

Ex :
```
10 MODE 1
12 BORDER 13
14 TAG
20 ORIGIN 0,0,100,540,300,100
30 GRAPHICS PAPER 3: CLG
40 FOR X=5502 TO -340 STEP -10
50 MOVE X,206
60 PRINT " FENETRE GRAPHIQUE";
70 FRAME
80 NEXT
90 GOTO 40
```

Cette commande détermine le point d'origine du curseur graphique aux coordonnées X et Y.

Vous pouvez fixer les dimensions de la fenêtre graphique en spécifiant les 4 paramètres facultatifs. Si les définitions de la fenêtre sont en dehors de l'écran alors les bords de l'écran sont les limites de la fenêtre.

OUT
OUT <n° port>,<nombre entier>

Ex : OUT &F8F4, &FF

Cette commande envoie la valeur du nombre entier vers le port de sortie spécifié par son adresse. Ne pas utiliser sans approfondissement.

PAPER
PAPER (<n° canal>,)<encre>

```
Ex :    10 MODE 0
        12 PEN 0
        14 INK 0,13
        20 FOR P=1 TO 15
        30 PAPER P
        35 CLS
        40 LOCATE 7 , 12
        45 PRINT "PAPER ";P
        50 FOR T = 1 TO 5000
        60 NEXT T, P
```

Cette commande établit la couleur du fond des caractères.

A l'affichage d'un caractère, la matrice est d'abord remplie avec l'encre du papier et ensuite le caractère est affiché (sauf en cas de mode transparent).

Le canal #0 est le canal par défaut. Le nombre de couleurs disponibles dépend du mode choisi.

PEEK
PEEK (<adresse>)

Ex : PRINT PEEK(100)
32

Cette fonction lit le contenu des cases mémoires dont l'adresse est indiquée entre parenthèses. Cette adresse va de &0000 à &FFFF (0 à 65535).

PEEK ne lit que la RAM (mémoire vive), non la ROM (mémoire morte) et donne des valeurs allant de &00 à &FF (0 à 255).

PEN
PEN (<n° canal>,)(<encre>)(,<mode du fond>)

Ex :
```
10 MODE 0
12 PAPER 0
14 INK 0, 13
20 FOR P=1 TO 15
30 PEN P:
35 PRINT SPACE$(47);"PEN ";P
40 FOR T=1 TO 500
50 NEXT T,P
60 GOTO 20
```

Cette commande sélectionne l'encre utilisée pour écrire sur le canal indiqué (=0 par défaut).

Le paramètre <mode du fond> est soit transparent (1), soit opaque (0).

Il faut préciser un des deux derniers paramètres. La valeur antérieure est conservée en cas d'ommission.

PI
PI

Ex : PRINT PI
3.14159265

Cette fonction donne la valeur du nombre PI.

PLOT
PLOT <X>,<Y>(, (<encre>)(, <mode d'encre>))

Ex :
```
10 MODE 1: BORDER 0: PAPER 0: PEN 1
20 INK 0,0: INK 1,26: INK 2,13,26: DEG
30 FOR X=1 TO 360: ORIGIN 320,200
40 DRAW 50*COS(X),50*SIN(X),1
50 PLOT 100*COS(X),25*SIN(X):NEXT
60 ORIGIN 0,0: T=TIME + 700
70 WHILE TIME < T
80 PLOT RND*640, RND*400
90 WEND
100 PLOT RND*640,RND*400,2
110 GOTO 110
```

Cette commande affiche en mode graphique, le point de coordonnées X et Y. On définit l'encre de ce point avec une valeur entre 0 à 15.

Le paramètre facultatif <mode d'encre> détermine l'interaction entre la couleur utilisée et celle de l'écran : 0 le mode normal, 1 le mode XOR (OU exclusif), 2 le mode AND (ET) et 3 le mode OR (OU).

PLOTR

PLOTR <décalage X>,<décalage Y>(,(<encre>)(,<mode d'encre>))

Ex :

```
10 REM UTILISATION DU PAVE CURSEUR POUR DESSINNER
20 BORDER 0
25 GRAPHIC PEN 1
30 MODE 1
35 PLOT 320,200
40 IF INKEY (0) = 0 THEN PLOTR 0,1 :GOTO 40
50 IF INKEY (1) = 0 THEN PLOTR 1,0 :GOTO 40
60 IF INKEY (2) = 0 THEN PLOTR 0,-1 :GOTO 40
70 IF INKEY (8) = 0 THEN PLOTR -1,0 :GOTO 40
80 IF INKEY (9) = 0 THEN  30 : REM COPY = CLS
```

Cette commande est identique à la commade PLOT, à la différence que les paramètres des coordonnées X et Y ne sont plus absolus mais relatifs à la position du curseur.

POKE
POKE <adresse>, <nombre entier>

Ex : 10 for K=49152 TO 65535
20 POKE M,100
30 NEXT

Cette commande inscrit la valeur du nombre entier (0 à 255) directement dans la case mémoire de la RAM du Z80 dont l'adresse est indiquée.

A utiliser avec précaution.

POS
POS (#<n° de canal>)

Ex : PRINT POS(#0)

Cette fonction donne la position du curseur de texte sur l'axe horizontal à partir du bord gauche de la fenêtre. Le n° de canal doit être obligatoirement précisé.

POS(#8) donne la position horizontale du chariot de l'imprimante par rapport à la marge gauche. POS(#9) donne la position logique de l'unité de disquette, c'est-à-dire le nombre de caractères depuis le dernier retour chariot.

PRINT

PRINT (#<n° de canal>,)(<liste de:<article à imprimer>)

Ex :
```
10 A$="essai"
20 B$="ESSAI D'UNE LONGUE CHAINE DE CARACTERES"
30 PRINT A$;A$
40 PRINT A$,A$
50 PRINT
60 PRINT B$;B$
70 PRINT B$,B$
```

Cette commande envoie la liste d'articles à imprimer sur le canal indiqué (canal 0 par défaut).

Le point-virgule ne laisse aucun espace à l'impression entre deux articles.

Si l'article est trop long, l'ordinateur force le passage à la ligne.

La virgule positionne le prochain article sur la tabulation suivante.

Si l'article précédent déborde sur cette tabulation, l'ordinateur va sur la prochaine.

PRINT SPC
PRINT TAB

PRINT (#<n° canal>,)(liste de :<article à imprimer>)(;)(SPC(<nombre entier>)) (<liste de :<article à imprimer>)

PRINT (#<n° canal>,)(liste de :<article à imprimer>)(;)(TAB(<nombre entier>)) (<liste de :<article à imprimer>)

Ex :
```
10 PRINT "EXEMPLE POUR SPC "
20 FOR K=6 TO 15:
25 PRINT SPC(5)"A";SPC(K);"B": NEXT
30 PRINT "EXEMPLE POUR TAB "
40 FOR K=6 TO 15:
50 PRINT TAB(5)"A";TAB(K);"B": NEXT
```

SPC crée le nombre d'espaces vides indiqué par le nombre entier, à condition que l'article suivant tienne complètement sur la ligne.

TAB compte à partir de la marge gauche le nombre d'espaces indiqué par le nombre entier à condition que l'article suivant tienne intégralement sur la ligne.

Si le curseur a déjà dépassé la position, il y a changement de ligne.

PRINT USING

PRINT (#<n° de canal>)(<liste de :<article à imprimer>)(;)(USING <modèle de format>)(<séparateur> <extpression>)

Ex : 10 FOR X=1 TO 10
20 N=10000* (rnd^5)
30 PRINT "QUANTITE";USING "*******,**";n
40 NEXT

PRINT USING définit le format d'impression ou d'affichage d'expressions envoyées par la commande PRINT. I

I faut définir le modèle de format que l'on veut avoir.

Le séparateur est soit la virgule ou le point-virgule.

Le <modèle de format> est une chaîne de caractères composée d'indicateurs de champs.

FORMAT NUMERIQUE :

* Chaque signe * indique l'emplacement d'un chiffre.

, Indique l'emplacement du point décimal.
Ce signe à ne pas confondre avec la virgule décimale, doit figurer immédiatement avant le point décimal. Les chiffres avant le point décimal seront alors regroupés par trois et ces groupes seront séparés par une virgule.
Ex : ******,.**

££ Le signe £ (livre anglaise) apparaîtra immédiatement avant le premier chiffre ou le point décimal, sur un emplacement réservé aux chiffres.
Ex : ££*****,.**

** indique que tous les espaces vides situés avant le nombre seront comblés par les astérisques.
Ex : **######,**

**£ Cumule les options ££ et **, astérisques en tête et le signe £ précédant le nombre.
Ex : **£######,**

$$ Identique à l'option ££, remplace le signe £ par le signe $ (dollar).
Ex : $$#######,**

**$ Cumule les options $$ et **.
Ex : **$#######,**

+ Indique la visualisation du signe du nombre. le signe apparaîtra avant le nombre si le + est situé au début du format et après le nombre si le + est situé en fin de format.
Ex : +######.## ######.##+

- Ce signe ne peut figurer qu'à la fin du format. Il donnera la visualisation du signe - après les nombres négatifs. Sans indications, le signe - apparaît avant le nombre.
Ex : ###.##-

^^^^ Indique que le nombre doit apparaître en exposant. Ces signes doivent être en fin de format mais avant le + ou le -.
Ex : #.####^^^^+

La longueur maximale du format d'un nombre est de 20 caractères. Les nombres sont arrondis au nombre de signes indiqué.

Si un format est trop petit pour contenir les chiffres, le signe % apparaît avant les chiffres pour indiquer que le format est erroné.

FORMAT D'UNE CHAINE ALPHANUMERIQUE

Ex :
```
10 CLS: A$="azertyuiop"
20 PRINT "CHAINE = "; A$
30 PRINT
35  PRINT "AVEC ! ";:PRINT USING "!";a$
40 PRINT
45 PRINT " AVEC \ESPACES\ ";
50 PRINT USING "\            \";a$
60 PRINT
65 PRINT "AVEC & "
67 PRINT USING "&";a$
70 GOTO 70
```

! Indique que seul le premier caractère de la chaîne doit apparaître.

Ex : !

\ \ Indique que seul les X premiers caractères de la chaîne doivent apparaître, X etant le nombre de blancs entre les \.

Ex: \ \

& Indique que la chaîne doit apparaître telle-quelle
Ex : &

Le modèle de format d'une chaîne ne doit pas excéder 255 caractères et tout format peut être une variable chaîne.

Ex :
```
10 A$="FF*****,.**"
20 B$="!"
30 PRINT USING A$;11112.2365;
40 PRINT USING B$;"centimes"
```

RAD
RAD

Cette commande établit le mode de calcul en radians C'est le mode de calcul par défaut en BASIC.

RANDOMIZE
RANDOMIZE (<expression numérique>)

Ex : RANDOMIZE 123.456
PRINT RND
0.2588521139

Cette commande permet de définir la valeur de calcul du nombre aléatoire donné avec la commande RND et de la séquence associée.

Si l'expression numérique n'est pas précisée, l'ordinateur demandera à l'utilisateur une valeur.

La commande RANDOMIZE TIME fournit une séquence pratiquement imprévisible.

READ
READ <liste de variables>

ex:
```
10 FOR n=1 TO 8
20 READ note
30 SOUND 1,note
40 DATA 478,426,379,358,319,284,253,239
RUN
```

Permet de lire une série de données contenues dans une instruction DATA. Un pointeur s'incrémente à chaque instruction READ, la commande RESTORE permet de revenir au début de DATA.

RELEASE
RELEASE <canaux sonores>

ex:
```
10 SOUND 65,478,10
20 PRINT" vous appuyez sur N pour avoir la note"
30 kk$=inkey$:if kk$<>"n" THEN 30
40 RELEASE 1
RUN
```

Libére les canaux sonores bloqués par la commande SOUND.

Les canaux sonores peuvent prendre les valeurs suivantes:

1: libére le canal A
2: libére le canal b
3: libére les canaux A et B
4: libére le canal C
5: libére les canaux A et C
6: libére les canaux B et C
7: libére les canaux A, B et C.

REM
REM <texte>

ex:
```
10 REM permet de mettre une REMarque
20 REM dans un programme BASIC
RUN
```

Attention cette instruction considère comme REMarque tout ce qui la suit, même si il y a un séparateur d'instruction ":" ou une autre instruction BASIC.

Vous pouvez remplacer REM par un apostrophe ' sauf à l'intérieur d'une ligne de DATA.

REMAIN
REMAIN <numéro de chronomètre>

ex:
```
10 AFTER 100,1 GOSUB 70
20 AFTER 50,1 GOSUB 100
30 GOTO 30
70 PRINT" chrono 1, ok"
80 RETURN
100 PRINT" il restait ",REMAIN(1);"unitès de temps au chrono 1"
RUN
```

Permet de désactiver le chronomètre spécifié (de 0 à 3), en donnant le temps qui restait.

RENUM

RENUM (<nouveau numéro >),(ancien numéro),(incrément)

```
ex      10 MODE 1
        11 GOTO 13
        12 PRINT " bonjour":GOTO 11
        13 RENUM 45,10,1
        RUN
        LIST
```

Permet de RENUMéroter les lignes du programme en mémoire de la manière suivante :

Ancien numéro:
Indique à partir de quelle ligne commencera la nouvelle numérotation.

Nouveau numéro:
Donne le numéro que prendra l'ancien numéro de ligne.

Incrément:
Indique l'espace désiré entre les lignes.

En l'absence d'un ou de plusieurs paramètres la valeur 10 sera prise comme défaut.

Les numéros de ligne doivent être compris entre 1 et 65535.

Cette commande ne change pas les numéros de ligne contenus dans les chaînes des caractères apparaissant dans les commandes KEY, REM, CHAIN et CHAIN MERGE.

RESTORE
RESTORE <numéro de ligne>

ex:
```
10 FOR N=1 TO 8
20 READ note
30 SOUND 1, note:NEXT
40 RESTORE 50:GOTO 10
50 DATA 478,426,379,358,319,284,253,239
```

Permet de ramener le pointeur à l'instruction DATA indiquée. En l'absence de paramètres, le pointeur se place sur la première instruction DATA.

RESUME
RESUME <numéro de lignes>

ex:
```
10 ON ERROR GOTO 100
20 GOTO 1000
30 GOTO 1000
100 PRINT "erreur no ";ERR;" ligne ";ERL
110 RESUME 30
```

Permet de reprendre l'exécution d'un programme après une erreur détectée par l'instruction ON ERROR GOTO.

Si aucun numéro de ligne est indiqué, le BASIC reprend à la ligne de l'erreur.

RESUME NEXT
RESUME NEXT

Même usage que pour RESUME , cette commande permet cependant de reprendre à la ligne suivant l'erreur.

RETURN
RETURN

```
EX      10 GOSUB 100:PRINT" retour du sous programme":END
        100 rem sous programme
        110 FOR I=1 TO 10
        120 PRINT I;" ";
        130 RETURN
        RUN
```

Obligatoire à la fin d'un sous-programme, cette instruction permet de revenir d'un sous-programme et d'exécuter l'instruction qui suit le GOSUB.

RIGHT$
RIGHT$ <chaîne alphanumérique>,<longueur requise>

```
ex:     10 MODE 0
        20 a$=" JE VOUS SOUHAITE UNE BONNE JOURNEE "
        30 C=LEN(a$)
        40 a$=RIGHT$(a$,c-1)+left$(a$,1)
        50 LOCATE 1,12:PRINT LEFT$(a$,20);
        60 FOR I=1 TO 100:NEXT :GOTO 40
        RUN
```

Permet d'extraire à droite d'une chaîne alphanumérique, des caractères suivant la longueur requise (entre 1 et 255).

RND

RND <expression numérique>

ex:
```
10 RANDOMIZE
20 FOR X=1 TO -1 STEP -1
30 PRINT "parametres du RND ";x;" "
40 FOR n=1 TO 6
50 PRINT RND(x);" ";
60 NEXT N
70 NEXT X
80 PRINT
RUN
```

Si l'expression numérique est positive ou nulle, cette commande donne le prochain nombre de la séquence aléatoire en cours.

Si l'expression numérique est négative, une séquence est produite dont RND fournit le premier élément.

ROUND
ROUND <expression numérique>,<nombre de décimales>

```
ex:     10 FOR ND=2 TO -2 STEP-1
        20 X=ROUND(789.45,ND):PRINT X;" avec ";n;" decimales"
        30 NEXT
        RUN
```

Fournit l'arrondi de l'expression numérique avec le nombre de décimales spécifiées par le paramètre, si celui-ci est négatif l'expression est arrondie à un entier absolu, suivi par un nombre de zéros égal à sa valeur absolue.

RUN
RUN <NOM d'un programme>

```
EX      RUN "essai"
```

Permet de charger et de lancer un programme se trouvant sur la disquette insérée dans le lecteur.

RUN <numéro de ligne>
```
ex:     RUN 100
```

Lance un programme présent en mémoire, en commencant au numéro de ligne indiqué ou par défaut au début du programme.

SAVE

SAVE <nom fichier>, (type du fichier), (paramètres binaires)

```
SAVE "essai.bas"
```

Sauvegarde un programme BASIC non protégé.

```
SAVE "essai.bas",p
```

Sauvegarde un programme BASIC protégé.

```
SAVE "essai.abc"
```

Sauveagarde un fichier en mode ASCII

```
SAVE "essai.abc",b,8000,3000,8001
```

Sauvegarde un fichier en mode binaire. Ici le programme sera stocké à l'adresse 8000 et occupera 3000 octets. Le point d'entrée falcultatif est 8001.

SGN

SGN <expression numérique>

ex:
```
10 x=INT(RND*100):y=INT(RND*100):x=x-y
20 IF SGN(x)=1 THEN a$="positve":GOTO 50
30 IF SGN(x)=-1 THEN a$="négative":GOTO 50
40 IF SGN(x)=1 THEN a$="nulle"
50 PRINT"la valeur est: ";a$
RUN
```

Donne le SiGNe de l'expression numérique. Cette commande renvoie les valeurs suivantes:

1 si l'expression est positive.
-1 si l'expression est négative.
0 si l'expression est nulle.

SIN
SIN <expression numérique>

ex:
```
10 CLS:DEG:ORIGIN 0,200
20 FOR X=0 TO 720
30 Y=sin(x)
40 PLOT x*640/720,198*Y:NEXT
50 GOTO 50
RUN
```

Donne le SINus de l'expression numérique.

L'argument est soit en degrés, soit en radians en utilisant les commandes DEG ou RAD.

SOUND
SOUND <état de canal>, <période sonore>, (<durée>, <volume>(, <enveloppe de volume>(, <enveloppe de tonalité>, (<période du bruit>))))

```
10 MODE 1
20 periode =10
30 e$=periode:IF e$<>" " then période=période+10
40 LOCATE 1,1:PRiNT "periode: ";periode;" frequence ca.";
INT(125000/periode)
50 SOUND 1,periode
60 goto 30
RUN
```

Permet de programmer un son avec les paramètres suivants:

1) ETAT DU CANAL

Compris entre 1 et 255.La conversion binaire donne la signifiaction de chaque BIT , suivant les codes:

BIT 0, 1 en décimale: sort le son sur le canal A
BIT 1, 2 en décimale: sort le son sur le canal B
BIT 2, 4 en décimale: sort le son sur le canal C
BIT 3, 8 en décimale: synchronisation avec le canal A
BIT 4,16 en décimale: synchronisation avec la canal B
BIT 5,32 en décimale: synchronisation avec la canal C
BIT 7,64 en décimale: vider un canal sonore.

2) PERIODE SONORE:

Ce paramètre établit la note produite (par ex: do). Chaque note est définie par sa période sonore.

3)DUREE:

Etablit la durée du son, l'unité correspond à un centième de seconde. Cette valeur, toujours positive, prend par défaut la valeur 20 (un cinquième de seconde). Si ce paramètre est nul, la longueur sera celle de l'enveloppe de volume indiquée.

4) VOLUME:

Détermine le volume sonore d'une note. Peut prendre les valeurs de 0 à 15. La valeur 12 est prise comme défaut.

5) ENVELOPPE DE VOLUME:
Vous avez la possibilité de moduler le volume d'une note durant son exécution à l'aide de la commande ENV. Vous pouvez définir jusqu'à quinze enveloppes différentes, que vous pourrez sélectionner à l' aide du paramètre "enveloppe de volume" de la commande SOUND.

6) ENVELOPPE DE TONALITE:
Son utilisation et sa syntaxe sont similaires à l'enveloppe de volume. Elle modifie légèrement la fréquence d'un son, en créant un effet vibrato.

7) PERIODE DE BRUIT:
Permet de rajouter ou de supprimer tout un choix de bruits blancs, valeur comprise entre 0 et 31, dans le signal sonore.

SPACE$
SPACE$ <nombre entier>

```
ex:     10 MODE 1
        20 FOR i=1 TO 10
        30 LOCATE 1,12:PRINT SPACE$(i);"BONJOUR"
        50 NEXT i
        RUN
```

Permet d'écrire une chaîne d'espace de la longueur spécifiée.

SPC (voir PRINT SPC)

SPEED INK
SPEED INK <période 1>,<période 2>

Etablit la période d'alternance lors d'une utilisation de deux couleurs pour les instructions INK et BORDER.

Les durées sont indiquées en cinquantièmes de secondes par les deux paramètres période 1 et période 2.

SPEED KEY

SPEED KEY <délai>,<intervalle inter-répétitions>

ex:
```
10 MODE 1
20 PRINT "essayez de taper du texte"
30 FOR x=10 TO 1 step -2
40 SPEED KEY x,x
50 NEXT
60 PRINT "DIFFICILE N'EST CE PAS !!"
RUN
```

Permet de paramétrer la vitesse de répétition des touches, le délai fixe (en cinquantièmes de secondes) l'intervalle qui sépare l'enfoncement d'une touche et le début de la répétition automatique.

Cette instruction ne concerne que les touches où la répétition existe implicitement ou celles pour lesquelles cette fonction à été créée par la commande KEY DEF.

SPEED WRITE

SPEED WRITE <nombre entier>

Définit la vitesse de transmission des données vers l'unité de cassettes. Cette vitesse peut être de 2000 bauds (bits par seconde) dans ce cas le paramètre sera 1, ou de 1000 bauds si celui est 0.

La valeur par défaut est 1000 bauds.

SQ
SQ <numéro de canal>

```
ex      10 SOUND 65,478,100
        20 PRINT sq(1)
        run
        67
```

Donne l'état de la file d'attente (SOUND QUEUE) dans la canal spécifié par le numéro.

1: canal A
2: canal B
4: canal C

Cette commande fournit un entier correspond aux bits suivants:

BITS 0,1,2: nombres d'entrées libres dans la file
BITS 3,4,5: état de la synchronisation au début de la file
BIT 6 : la tête de la file est bloquée
BIT 7 : le canal est en activité

Une remarque se dégage de se tableau:
Si le BIT 6 est à 1, le BIT 7 ne peut l'être. De même si les BITS 3, 4 ou 5 sont à 1, les BITS 6 et7 ne peuvent pas l'être.

SQR
SQR <expression numérique>

```
        PRINT SQR(4)
        2
```

Retourne la racine carrée (square root) de l'expression numérique.

STEP (voir FOR)

STOP
STOP

EX:
```
10 PRINT"bonjour"
20 STOP
30 PRINT"et bienvenue sur AMSTRAD"
RUN
CONT
```

Permet d'interrompre le programme, tout en pouvant le reprendre par la commande CONT.

STR$
STR$ <expression numérique>

ex:
```
10 x=INT(RND*100)
20 a$="--"+str$(x)+"--"
30 PRINTa$
RUN
```

Convertit l'expression numérique en une chaîne de caractères.

SWAP (voir WINDOW SWAP)

SYMBOL

SYMBOL <numéro du caractère>, <liste de ligne>

```
ex:     10 MODE 0
        12 SYMBOL AFTER 105
        20 RANGEE1=255:rem 255=11111111 en binaire
        30 RANGEE2=129:rem 129=10000001 en binaire
        40 RANGEE3=189:rem 189=10111101 en binaire
        50 RANGEE4=153:rem 153=10011001 en binaire
        60 RANGEE5=153:rem 153=10011001 en binaire
        70 RANGEE6=189:rem 255=10111101 en binaire
        80 RANGEE7=129:rem 129=10000001 en binaire
        90 RANGEE8=255:rem 255=11111111 en binaire
        100 SYMBOL 105, RANGEE1,RANGEE2,RANGEE3, RANGEE4,
        RANGEE5, RANGEE6, RANGEE7, RANGEE8
        110 PRINT "la lettre i est maintenant redéfinie,-essayez !!"
        RUN
```

Permet de redéfinir la forme d'un caractère, les paramètres peuvent prendre une valeur entre 0 et 255.

La commande SYMBOL AFTER permet à l'ordinateur de réserver de la place en mémoire pour le caractère redéfini.

Même si le caractère n'est pas accessible par l'intermédiaire du clavier, on peut l'utiliser avec la fonction CHR$.

La commande SYMBOL s'emploie avec huit paramètres qui représentent les huit lignes constituant le caractère. C'est la représentation binaire du paramètre qui définit le motif de la ligne correspondante dans le nouveau caractère.

Par exemple, si le premier paramètre a la valeur 3, sa représentation binaire sera 00000011. Les points correspondant au 1 apparaîtront dans le caractère et seront de la même couleur que celle définie dans la commande PEN, les 0 seront affichés avec la couleur du fond, ils ne seront donc pas visibles.

Vous pouvez éviter de convertir en notation décimale les symboles binaires, il vous suffit pour cela d'introduire les paramètres directement en binaire, sans bien-sûr oublier le préfixe &.

ex :

```
SYMBOL 255,&x11111111,&x10000001, &x10000001,
&x10000001, &x10000001, &x10000001, &x10000001,
&x11111111
PRINT CHR$(255).
```

Un caractère nouvellement créé sera accepté par le BASIC comme l'équivalent du caractère remplacé, et si ce caractère était affecté à une touche, l'appui sur cette touche fournira la nouvelle version créée.

SYMBOL AFTER
SYMBOL AFTER <nombre entier>

ex:

```
10 mode 1
20 SYMBOL AFTER 105
30 PRINT "on redéfinie la lettre i"
40 SYMBOL 105,255,129,189,153,153,189,129,255
50 PRINT "pour revenir au i normal, tapez:"
60 PRINT "SYMBOL AFTER 240"
RUN
```

Permet de fixer la limite inférieure des caractères redifinissables de 0 à 255. La valeur par défaut est fixée à 240, dans ce cas, vous disposez de 16 caractères redifinissables (de 240 à 255).

ex: SYMBOL AFTER 105
Vous donne la possiblité de redéfinir les caractères situés entre 105 et 255.
La valeur 256, interdit toute redéfinition. La commande SYMBOL AFTER sans paramètre permet de revenir à la valeur par défaut.

L'utilisation de cette commande comporte une restriction:
Ne pas modifier la mémoire par une commande HIMEM ou par l'ouverture d'un fichier (OPENIN,OPENOUT).

TAG
TAG <#n° du canal)

ex:
```
10 mode 1: A$="BONJOUR": TAG
30 x=len(a$)*17:y=50+rnd*300:MOVE -x,y
40 for f=-x TO 640 STEP RND*7+3
50 MOVE f,y:PRINT a$;:FRAME:NEXT
60 FOR B=640 TO -x STEP -RND*7+3
70 MOVE b,y:PRINT a$;" ";:FRAME:NEXT
80 GOTO 30
RUN
```

Ecrit du texte à la position du curseur graphique.

Cette commande permet de dépacer le texte à écrire non plus par caractére mais par pixel. La valeur par défaut du CANAL est #0.
L'extrémité gauche de la chaîne de caractères se positionne sur le curseur graphique.

Si le canal spécifié est 0, le basic annule la commande TAG lors du retour en mode direct.

TAGOFF
TAGOFF <numèro de canal>

Annule la commande TAG, concernant le canal spécifié et dirige donc le texte sur la position du curseur de TEXTE.

TAN

TAN <expression numérique>

```
ex:     TAN (90)
        -1.99520043
```

Retourne la TANgente de l'expression numérique, comprise entre -200000 et +200000.

L'argument peut être en degrés ou en radians, si l'on emploie les fonctions DEG et RAD.

TEST

TEST <coordonnée x>,<coordonnée y>

```
ex:     10 CLS
        20 TEST 10,394:rem place le curseur
        30 PEN 2
        40 PRINT "la couleur du stylo est: ";test(10,394)
        RUN
```

Place le curseur graphique à la position spécifiée par les coordonnées x et y, indique aussi la couleur de l'encre à cet endroit.

TESTR

TESTR <décalage x >, <décalage y>

Même usage que pour la commande TEST, seulement ici le positionnement s'effectue par rapport à la position actuelle du curseur.

THEN (voir IF)

TIME
TIME

EX:
```
10 CLS
20 LOCATE 10,10
30 PRINT "vous allez rentrer l'heure sous la forme HH:MM:SS"
40 INPUT h$
50 h=VAL(MID$(h$,1,2)):IF h<0 OR h>24 THEN 1000
60 m=VAL(MID$(h$,3,2)):IF m<0 OR h>60 THEN 1000
70 s=VAL(MID$(h$,5,2)):IF s<0 OR s>60 THEN 1000
80 d=INT(TIME/300)
90 WHILE H<13
100 WHILE m<60
110 WHILE t<60
120 t=(INT(TIME/300)-d)+s
130 LOCATE 1,1
140 PRINT USING "## ";h;m;t
150 WEND
160 t=0:s=0:m=0:m=m+1
170 WEND
180 h=1
190 goto 60
```

Donne le temps écoulé depuis la dernière mise en route de l'ordinateur ou le dernier RESET effectué.

Les temps de transfert entre l'ordinateur et l'unité de disquette ne sont pas pris en compte.

Une seconde est égale à TIME/300.

TO (voir FOR)

TRON
TROFF

La commande TRON permet de suivre à la TRace le programme qui s'exécute.

Les lignes apparaissent entre crochets. La commande TROFF revient au mode normal.

UNT
UNT <expression hexadécimale>

```
ex      PRINT UNT(&bec)
        3052
```

Convertit l'argument en un nombre entier signé, en représentation complément à 2.

Le nombre doit être compris entre -32768 et 32767.

UPPER$
UPPER$<chaîne alphanumérique>

```
ex:     10 INPUT "votre nom s.v.p";nom$
        20 PRINT UPPER$(nom$)
        RUN
```

Convertit une chaîne alphanumérique comportant des minuscules en une chaîne n'ayant plus que des majuscules.

USING (voir PRINT USING)

VAL
VAL<chaîne de caractères)

```
EX:     10 a$="125.30"
        20 a=VAL(A$)+10
        30 PRINT A
        RUN
```

Retourne la VALeur numérique de la chaîne de caractères indiquée. Cette commande fournit 0 si le premier caractère n'est pas un chiffre.

VPOS
VPOS(#<numéro de canal>)

```
ex:     10 LOCATE 10,10
        20 PRINT "le curseur se trouve en :";
        30 PRINT POS(#0),VPOS(#0)
        RUN
```

Indique la position du curseur de texte sur l'axe vertical.

Le numéro de canal, est 0 par défaut.

WAIT
WAIT <numéro de port>, <masque>, (<inversion>)

ex: WAIT &ff34,20,25

Provoque une attente jusqu'à ce que le port d'entrées-sorties désigné transmette une valeur comprise entre 0 et 255; de telle sorte qu'après avoir opéré un XOR avec la masque, puis un AND avec le paramètre d'inversion un résultat nul.

N'utilisez cette commande qu'avec beaucoup de précaution .

WHILE
WEND

WHILE <expression logique>

EX:
```
10 CLS:PRINT"vous avez 10 secondes pour evacuer"
20 t=time:WHILE TIME<t+3000
30 SOUND 1,0,100,15
40 WEND:SOUND 129,40,30,15
RUN
```

Permet de répéter une section du programme contenu entre les commandes WHILE et WEND tant qu'une condition est vérifiée. WHILE indique le début de la section, l'expression définit la condition à respecter.

La commande WEND marque la fin de la section de programme.

WIDTH
WIDtH <nombre entier>

```
WIDTH 35
```

Indique le nombre de caractères par ligne pour une sortie sur imprimante et charge le BASIC de gérer automatiquement les retours chariot et les sauts de ligne nécessaires durant l'impression.

La valeur par défaut est 132.

La valeur 255 supprime tous les retours chariot et sauts de ligne, laissant ainsi l'imprimante générer seule ces caractères. Cependant la commande PRINT envoie toujours un retour chariot à moins qu'il n'y ait un point virgile ou une virgule.

WINDOW
WINDOW (#<numéro de canal>,), <gauche>, <droit>, <haut>, <bas>

ex:
```
10 MODE 1
20 WINDOW #1,30,40,10,10
30 WINDOW #2,10,20,24,24
40 PRINT#2:PRINT#1,"COUCOU"
50 FOR I=1 to 100:NEXT
60 PRINT#1:PRINT#2,"BONJOUR"
70 FOR I=1 TO 100:NEXT
80 GOTO 40
RUN
```

Permet de définir à l'écran une fenêtre de visualisation qui aura la dimension spécifiée par les valeurs des paramètres.

Il faudra veiller à ce que ces paramètres ne dépassent pas les coordonnées admises par le MODE utilisé.

Le numéro de canal est par défaut 0.

WINDOW SWAP
WINDOW SWAP <numéro de canal>, <numéro de canal>

Dans l'exemple de programme pour la commande WINDOW, tapez la ligne suivante:

```
65 WINDOW SWAP #1,#2
```

Intervertit la première fenêtre avec la seconde.

WRITE
WRITE <#numéro de canal>, (données à écrire)

ex:
```
10 CLS
20 INPUT "votre nom s.v.p";nom$
30 INPUT "votre age";age
40 OPENOUT "fichier"
50 WRITE #9,nom$,age:CLOSEOUT
60 PRINT"vous etes maintenant inscrit dans le fichier"
RUN
```

Permet d'écrire ou d'afficher des données sur le canal spécifié, les articles doivent êtres séparés par une virgule, les chaines de caractères doivent figurer entre guillemets.

Pour vérifier la bonne exécution de notre programme d'exemple, il vous suffit de taper:

```
10 OPENIN "fichier"
20 INPUT #9,NOM$,AGE
30 CLOSEIN
40 PRINT "votre nom est :";nom$
50 PRINT " vous êtes agé de ";age;" ans"
RUN
```

XOR
<argument> XOR <argument>

EX: IF "lion" > "léopard" XOR "homme" >"singe" THEN PRINT "vrai" ELSE PRINT "faux"

Effectue bit à bit l'opération booléenne XOR, OU exclusif, sur des entiers. Lorsque les bits des deux arguments ne sont pas identiques le bit résultant vaut 1.

XPOS
YPOS

```
10 mode 1
20 DRAW 300,100
30 PRINT"position du curseur graphique :";
40 PRINT XPOS,YPOS
RUN
```

Ces deux fonctions permettent de connaître les coordonnées horizontales (XPOS) et verticales (YPOS) du curseur graphique.

ZONE
ZONE

EX:
```
10 MODE 1
20 FOR zone=1 TO 10
30 ZONE zone
40 PRINT "zone =",zone
50 NEXT
```

Modifie l'espace entre la virgule dans une commande PRINT.

La valeur par défaut est 13, elle peut varier de 1 à 255.

MESSAGES D'ERREURS

MESSAGES D'ERREUR DU BASIC

1 UNEXPECTED NEXT (Next inattendu)

Une commande **NEXT** a été rencontrée sans que la commande **FOR** n'ait été exécutée ou la variable suivant le **NEXT** ne correspond pas à celle du **FOR**.

2 SYNTAX ERROR (Erreur de syntaxe)

L'interpréteur a rencontré une instruction qu'il ne comprend pas.
Après ce message, vous êtes directement sous le mode **edit**.

3 UNEXPECTED RETURN (Return inattendu)

Une commande **RETURN** a été détectée sans qu'une commande **GOSUB** n'ait auparavant lancé le sous-programme en cours.

4 DATA EXHAUSTED (Il n'y a plus de données DATA)

Une commande **READ** a essayé de lire plus de **DATA** qu'il n'y en avait.

5 IMPROPER ARGUMENT (Argument incorrect)

Erreur d'ordre général. L'argument d'une fonction ou le paramètre d'une commande n'est pas acceptable.

6 OVERFLOW (Dépassement arithmétique)

Erreur de calcul se produisant lors de la tentative d'une utilisation d'un chiffre en virgule flottante trop grand (supèrieur à 1.7'38) ou lors d'une conversion donnant un nombre entier trop grand.

7 MEMORY FULL (mémoire saturée)

Le programme ou les variables prennent trop de place en mémoire. Peut se produire aussi lors d'une utilisation de structure de boucles trop compliquée , ou trop de **GOSUB** .
A chaque ouverture de fichier une mémoire tampon est créée, ce qui peut être une cause de limitation pour la commande **MEMORY** .

8 LINE DOES NOT EXIST (ligne inexistante)

Tentative d'utiliser une ligne qui n'existe pas en mémoire.

9 SUBSCRIPT OUT OF RANGE (indice hors limite)

Un des indices de votre tableau est trop grand ou trop petit.

10 ARRAY ALREDY DIMENSIONED (Tableau déjà dimensionné)

Un tableau déjà créé a fait l'objet d'une tentative de dimensionnement par la commande DIM.

11 DIVISION BY ZERO (Division par zéro)

Il est impossible de diviser par zéro.

12 INVALID DIRECT COMMAND (Commande directe non acceptable)

Une commande non valable a été utilisée en mode direct.

13 TYPE MISMATCH (Type de variable ne correspondant pas)

On a donné une valeur numerique à une variable chaîne de caracteres ou le contraire; ou un caractère non valable a été découvert par une commande READ ou INPUT.

14 STRING SPACE FULL (Espace réservé aux chaînes saturé)

L'espace alloué au chaîne est entierement occupé.

15 STRING TOO LONG (Chaîne trop longue)

Une chaîne de plus de 255 caractères a été rencontrée.

16 STRING EXPRESSION TOO COMPLEX (Chaîne trop compliquée)

Arrive souvent si dans une même expression vous utilisez trop de commandes BASIC.

17 CANNOT CONTINUE (On ne peut pas continuer)

Il est impossible de reprendre le cours du programme avec la commande CONT. Une modification du programme a eu lieu.

18 UNKNOWN USER FONCTION (Fonction inconnue)

Un appel à une fonction FN a été fait sans que celle-ci n'ait été definie par la commande DEF FN.

19 RESUME MISSING (Commande resume absente)

Aprés un ON ERROR GOTO l'ordinateur n'a pas trouvé la commande RESUME.

20 UNEXPECTED RESUME (Resume inattendu)

Une commande **RESUME** a été trouvée sans qu'il y ait eu un appel par **ON ERROR GOTO.**

21 DIRECT COMMAND FOUND (Une commande directe trouvée)

Lors du chargement d'un programme une ligne ne comportant pas de numéro s'est présentée.

22 OPERAND MISSING (opérande absent)

L'interprétreur vient de trouver une expression incomplète.

23 LINE TOO LONG (Ligne trop longue)

La longueur maximale d'une ligne de BASIC est de 255 caractères.

24 EOF MET (Rencontre d'une fin de fichier)

Le programme a essayé de lire un fichier après une fin de fichier.
(EOF = End Of File)

25 FILE TYPE ERROR (Erreur de type de fichier)

Le fichier n'est pas de type requis.
Tentative d'ouvrir un programme BASIC.

26 NEXT MISSING (Next manquant)

Une commande FOR a été détectée sans qu'il y ait eu un NEXT équivalent.

27 FILE ALREADY OPEN (Fichier déjà ouvert)

Une tentative d'ouverture de fichier déjà ouvert a été detecté.

28 UNKNOW COMMAND (Commande inconnue)

Le BASIC ne comprend pas cette commande externe.

29 WEND MISSING (Wend manquant)

Il n'y a pas de **WEND** dans la boucle commençant par **WHILE**.

30 UNEXPECTED WEND. (Wend inattendu)

Un **WEND** ne correspondant pas au **WHILE** a été découvert.

31 FILE NOT OPEN. (Fichier non ouvert)

Tentative de lire ou d'écrire sur un fichier qui n'est pas ouvert.

32 BROKEN IN (Interrompu)

Voir le paragraphe "erreurs de disquettes".

ERREURS DE DISQUETTES :

Des erreurs peuvent se produire lors de l'utilisation des disquettes.

Le BASIC les regroupe toutes sous le numéro d'ERReur 32. Malgré cela il est possible d'obtenir plus d'information par la commande DERR.

Nous allons vous expliquer la signification des valeurs qu'elle renvoie.

Erreur AMSDOS	Valeur DERR	Source de l'erreur
0	0 ou 22	Activation de ESC.
14	142 (128+14)	Etat du canal non valable.
15	143 (128+15)	Fin de fichier matérielle.
16	144 (128+16)	Mauvaise commande généralement nom du fichier incorrect.
17	145 (128+17)	Fichier déjà existant.
18	146 (128+18)	Fichier non existant.
19	147 (128+19)	Catalogue saturé.
20	148 (128+20)	Disquette pleine.
21	149 (128+21)	Changement des disquettes avec fichiers ouverts.
22	150 (128+22)	Fichier en lecture seulement.
26	154 (128+26)	Fin de fichier logicielle.

Si AMSDOS a déjà rapporté une erreur, le bit 7 a pris la valeur 1, décalant celle de DERR de 128.

Il existe d'autres erreurs rapportées par DERR, elles proviennent du contrôleur de disquette et sont codées sur un octet.

Voici la signification des huit bits :

BIT	Signification
0	Adresse manquante.
1	Ecriture impossible. Disquette protégée.
2	Pas de données. Secteur introuvable.
3	Unité non prête. Pas de disquette dans l'unité
4	Surcharge.
5	Erreur de donnée. Erreur CRC (Cyclic Redundancy Check).
6	Toujours à 1 pour indiquer une erreur venant du contrôleur de la disquette.
7	Sur 1 si AMSDOS a déjà rapporté l'erreur.

LES COMMANDES DISQUETTES

DEFINITION D'AMSDOS:

AMSDOS est le programme système qui permet l'utilisation des disquettes. Lors de l'explication des commandes BASIC, nous avons vu un certain nombre de commandes qui utilisent ce programme.

Mais AMSDOS fournit aussi quelques commandes accessibles seulement en mode direct.

Ces commandes, appelées commandes externes sont toujours précédées du symbole | (obtenue en appuyant simultanément sur les touches SHIFT et @.

Nous allons les examiner plus en détails.

Mais avant une petite astuce:

IL arrive que l'on ait besoin, lors d'un effacement ou d'une copie, de travailler sur plusieurs fichiers en même temps. C'est possible, grâce à deux jockers: ? et *.

?: signifie ne pas tenir compte de la lettre qui se trouve à la position du ?

ex: "essai.bas" est égale à "essa?.bas"
mais "essax.bas" est aussi égale à "essai?.bas".

*:signifie ne pas tenir compte de tout ce qui suit.

ex: "essai.bas" est égale à "essai.*"
mais "essai.bak" est aussi égale à "essai.*".

| A
| B

Permet d'indiquer au système, sur quelle unité de disquette il doit diriger ses commandes.
Cette commande n'est effective que si vous possédez deux lecteurs de disquettes.
L'unité par défaut est A

| CPM

Charge et initialise le système d'exploitation CP/M. Deux systèmes vous sont fournis CP/M 2.2 et CP/M Plus.

| DIR (chaîne alphanumérique)

ex: | DIR,"*.BAS"

Donne sur l'écran le catalogue de la disquette sous forme CP/M. En l'absence de chaîne , "*.*" est pris par défaut.

| DISC

Equivaut aux commandes | DISC.IN et DISC.OUT réunies.

| DISC.IN

Oblige le système à lire des données sur la disquette.

| DISC.OUT

Oblige le système à écrire des données.

| DRIVE

ex: | DRIVE,"A"

Permet de désigner une des unités de disquettes.

| ERA

ex: | ERA "nom.*"

Efface tous les fichiers qui correspondent au nom indiqué. Possibilité d'utiliser les jockers.

| REN

ex: | REN "nouveau.bas","ancien.bas"

Permet de renommer un programme ou un fichier sur disquette. Le nouveau nom ne doit pas exister. Les jockers ne peuvent êtres utilisés.
Un numéro d'utilisateur peut être utilisé et peut donc supplanter toutes les valeurs par défaut.

ex: | REN "1:nouveau.bas","5:ancien.bas"

Rebaptise le fichier de l'utilisateur 5 en l' affectant à l'utilisateur 1.

| TAPE

A utiliser seulement si un lecteur de cassette est raccordé à l'ordinateur, équivaut aux commandes | TAPE.IN ET | TAPE.OUT.

| TAPE.IN

Lit des données sur le lecteur de cassettes.

| TAPE.OUT

Ecrit des données sur le lecteur de cassettes.

| USER

| USER <nombre entier (entre 0 et 15)>

ex: | USER,5

Permet de déterminer la section du catalogue sur laquelle il y aura des accès disque.

LE SYSTEME CP/M

Le CP/M est sans doute le système d'exploitation le plus utilisé au monde. De ce fait, il vous permet d'avoir accès à une immense blibliothèque de programmes.

Pour avoir accès aux commandes CP/M , il vous faut d'abord insérer dans le lecteur la face 1 de la disquette intitulée CP/M plus.
Et de tapez:

```
| CPM
```

Au bout de quelques secondes le message suivant apparaît sur l'écran:

```
CP/M plus Amstrad Consumer Electronics PLC
```

Le symbole A> est l'équivalent de READY en BASIC.

ATTENTION:

Une fois en mode CP/M les commandes basic ne dont plus admises.

ex: si vous tapez MODE 2

L'ordinateur vous répondra par:

```
MODE 2?
```

CP/M a donc ses commandes spécifiques:

1) LES COMMANDES EN MODE DIRECT:

Ces commandes peuvent êtres tapées après le symbole A> ou B>

DIR

Similaire à la commande du BASIC, elle affiche tous les fichiers du catalogue de la disquette avec la place disponible.

L'utilisation des jockers est possible, mais les fichiers à attribut "SYS" ne sont pas permis

ex: DIR : donne le répertoire de l'unité par défaut.
DIR b:donne le répertoire de l'unité B.
DIR *.BAK: donne tous les fichiers de type BAK.

DIRS ou DIRSYS :

Liste uniquement les fichiers ayant "SYS" comme attribut.

Cette commande fonctionne exactement comme DIR.

ERA ou ERASE :

Cette commande efface sur la disquette les fichiers spécifiés.

L'utilisation des jockers provoque l'affichage d'une demande de confirmation avant que la commande ne s'effectue vraiment.

La commande ERA ne mentionne pas les fichiers qu'elle détruit.
Cette commande ne fonctionne que si les fichiers ne sont pas de type "lecture seulement"

REN ou RENAME :

Permet de renommer un fichier.

Le nouveau doit figurer en premier suivi du signe "=", puis de l'ancien nom. Si le nouveau nom existe déjà sur la disquette un message d'erreur est généré.

TYPE et TYP.

Permet d'afficher à l'écran le fichier spécifié. Si ce fichier n'est pas de type ASCII, il risque d'y avoir à l'écran des choses imprévisibles.

USER ou USE.

Change le numéro d'utilisateur courant, ce qui permet de partager le catalogue. Au départ CP/M initialise le numéro 0.

Les fichiers qui possédent "SYS" comme attribut peuvent être accessibles par n'importe quels utilisateurs, ce qui permet de pas avoir à les copier dans chaque zone.

LES COMMANDES TRANSITOIRES:

Elles permettent de réaliser des opérations un peu plus compliquées qu'avec les commandes directes.

Leur utilisation est identique. La seule différence et qu'elles se trouvent sur la disquette au lieu d'être en mémoire centrale.

DISCKIT3

Permet de formater, copier et vérifier les disquettes, suivant l'option choisie, un menu vous indique la touche à activer.

Menu principal:

COPY	7
FORMAT	4
VERIFY	1
EXIT FROM PROGRAM	0

Pour exécuter la procédure désirée, il vous suffit d'appuyer sur la touche de fonction portant le n° correspondant.
Les touches de fonctions se trouvent sur le pavé numérique.

En appuyant sur F0 , vous revenez sous le système CP/M.

La première chose à faire dans tout système informatique est une sauvegarde, un accident est si vite arrivé et perdre vos disquettes CP/M serait une catastrophe pour vous.

Nous allons donc faire une copie de vos disquettes systèmes en appuyant sur F7.

Dans le cadre d'une utilisation à une seule unité de disquette vous aurez le message suivant:

```
Y  Copy
Any other key to exit menu
```

Vous devez alors retirer la disquette système et insérer la disquette à dupliquer. Une fois fait, vous tapez sur Y.

L'ordinateur affiche pour information le format de la disquette.
Un message apparaît ensuite:

```
Insert disc to WRITE
Press any key to continue
```

Vous devez alors retirer la disquette SOURCE qui se trouve dans le lecteur et insérer la disquette sur laquelle vous voulez copier.

TOUTES LES INFORMATIONS CONTENUES SUR LA DISQUETTE SERONT IRREMEDIABLEMENT PERDUES.

Les informations concernant la disquette OBJET apparaîtront en haut de l'écran, même si cette disquette est vierge.

Au cas où elle serait mal formatée, le formatage s'effectuera lors de la duplication et le message suivant s'affichera:

```
Disc isn't formatted (or fautly)
going to format while copying
disc will be system format
```

Une fois fait l'ordinateur vous demandera à nouveau d'insérer la disquette source:

```
Insert disc to READ
press any key to continue
```

Vous devez mettre la disquette et appuyer sur une touche.

Vous devrez donc alterner disquette SOURCE et disquette OBJET jusqu'à la duplication complète signalée par le message:

```
Copy completed
remove disc
press any key to continue
```

Vous avez deux possibilités, soit

- refaire une copie dans ce cas vous appuyer sur Y, soit
- vous retournez au menu principal.

DANS LE CAS D'UTILISATION D'UN SECOND LECTEUR DE DISQUETTE :

Un nouveau menu apparait:

Read from a: 8
Read from b: 5
Exit menu 2

Il n'est plus nécéssaire d'effectuer le changement de disquette.

Le menu ci-dessus vous demande de choisir l'unité qui contiendra la disquette source.

Un second menu s'affiche:

Write to a: 9
Write to b: 6
Exit menu 3

Vous pouvez également choisir l'unité qui contiendra la disquette objet. Rien ne vous empèche de choisir la même unité pour la lecture et l'écriture.

Le reste de la procédure demeure identique. Il vous faut cependant retirer les DEUX disquettes pour poursuivre.

FORMATAGE D'UNE DISQUETTE:

En appuyant dans le menu princial sur F4, vous lancez la procédure de formatage.

Un menu s'affiche alors:

System format	9
Data format	6
Vendor format	3
Exit menu	.

Comme dans les autres menus vous devez appuyer sur les touches F9, F6, F3 pour obtenir le format souhaité.
Mais auparavant voici quelques explications concernant ces différents formats:

Tous les formats ont quelques points communs:

-ILs ont tous 40 pistes numérotées de 0 à 39.

-Leur répertoire peut contenir au maximun 64 fichiers.

-La taille d'un secteur est toujours de 512 octets. Celle du bloc CP/M est de 1024 octets.

FORMAT SYSTEM:

Posséde 9 secteurs par piste de &41 à &49 dont deux sont résevés.

Permet le chargement du CP/M, de ce fait il est le plus couramment utilisé. Pour le réamorçage à chaud, le CP/M 2.2 demande une disquette avec un format SYSTEM.

Il existe une version VENDOR qui est en fait une version SYSTEM ne possédant aucun logiciel dans les pistes réservées. Ce format est utilisé pour les logiciels destinés à la vente.

FORMAT DATA ONLY:

Posséde 9 secteurs par piste, numérotés de &C1 à &C9.

Ce format ne permet pas l'amorçage à chaud. Limité à AMSDOS et à CP/M Plus, il permet de gagner un peu de place sur la disquette.

FORMAT IBM:

Posséde 8 secteurs par piste, numérotés de &1 à &8.
1 piste réservée.

Ne fonctionne qu'avec le CP/M 2.2. Est conforme avec le format de l'IBM PC.L'AMSTRAD 6128, ne permet que la lecture et l'écriture sur des disquettes au format IBM.

Imaginons que vous ayez choisi le format DATA, le message suivant s'affiche à l'écran:

Y Format as Data
Any other key to exit menu

Vous devez alors retirer la disquette CP/M et mettre à la place votre disquette à formater en appuyant sur Y.

LE FORMATAGE EFFACE COMPLETEMENT LA DISQUETTE !

Une fois le formatage terminé vous pouvez en refaire un autre en appuyant sur Y. Pour revenir au menu, vous tapez n'importe quelle lettre à l'exception de Y.

SI vous possédez deux unités de disquettes, vous aurez un troisième menu :

Format A: 8
Format B: 5
Exit menu 2

Nous vous conseillons de choisir l'option B, ce qui vous permet de garder votre disquette CP/M dans le lecteur A.

VERIFICATION DES DISQUETTES:

Obtenue en appuyant sur la touche de fonction F1.

Une fois votre disquette insérée, cette procédure affiche le format de la disquette et effectue une lecture de tous les fichiers présents, si une erreur est détectée, elle apparaît sur l'écran.

La fin de la vérification est signalée par:

Verify completed
Remove disc
Press any key to contenue

POUR ALLER PLUS LOIN AVEC CP/M :

Dans la première partie de ce chapitre nous avons sourtout vu dans CP/M la possibilité de faire des copies ou de formater des disquettes, mais CP/M ce n'est pas seulement cela, c'est aussi reconfigurer le clavier, changer les couleurs ou encore les imprimantes, voyons tout cela plus en détails :

Changement du jeu de caractères :

Le 6128 possède un jeu complet de caractères internationaux.

La commande **language** permet de modifier certains caractères pour pouvoir utiliser certains logiciels .

Ex: **language 3**

Installe le jeu de caractères britannique en ramplacant le signe # par £ (le jeu de caractères américain est le jeu par défaut).

Les couleurs :

Sous CP/M Plus, le 6128 (avec un moniteur couleur) affichent des caractères en blanc brillant sur fond bleu. Vous pouvez les modifier par la commande **palette**, qui possède plusieurs paramétres, un pour chaque encre: l'encre 0 inclut le fond et l'encre définit la couleur du texte.

Chaque couleur est codée par un nombre compris entre 0 et 63. Sur un moniteur monochrome ce paramètre définit l'intensité d'affichage.

Vous pouvez spécifier n'importe quel numéro d'encre compris entre 1 et 16, mais seules les deux premières encres seront visibles sous le mode 80 colonnes.

Ex: **palette 63,1**

Permet d'inverser les spécifications normales des encres 0 et 1, changeant le fond en blanc brillant (63) et le texte en bleu (1).

Pour vous aider à sélectionner vos couleurs, reportez-vous au tableau qui suit (page suivante), vous pouvez soit utiliser l'hexadécimal ou décimale.

couleur	héxa	décimal	couleur	héxa	décimal
NOIR	00	0	BLEU PASTEL	2B	43
BLEU	02	2	ORANGE	2C	44
BLEU VIF	03	3	ROSE	2E	46
ROUGE	08	8	MAGENTA PASTEL	2F	47
MAGENTA	0A	10	VERT VIF	30	48
MAUVE	0B	11	VERT MARIN	32	50
ROUGE VIF	0C	12	TURQUOISE VIF	33	51
VIOLET	0E	14	VERT CITRON	38	56
MAGENTA VIF	0F	15	VERT PASTEL	3A	58
VERT	20	32	TURQUOISE PASTEL	3B	59
TURQUOISE	22	34	JAUNE VIF	3C	60
BLEU CIEL	23	35	JAUNE PASTEL	3E	61
JAUNE	28	40	BLANC BRILLANT	3F	63
BLANC	2A	42			

MODIFICATION DU CLAVIER :

Il est possible de modifier les codes générés par le clavier à l'aide de la commande **setkeys**.

Ceci permet d'affecter des codes aux touches physiques ou logiques.

Les codes doivent être inscrits dans un fichier dont le nom est ensuite passé en paramètre dans la commande **setkeys**.

Ce fichier de commande peut être créé par un éditeur de texte **pip** ou même à partir du **basic** par exemple :

```
setkeys keys.tst
```

Où le fichier **keys.tst** contient :
E &&c <**dir** '**m**> touche logique 12
8 N S C <' **h** > espace arriere = [**control**]**h**,08 ASCII

Ce qui modifie la touche logique [**control**] [**enter**] (code &8c), qui devient **dir**, et transforme la touche de retour arrière du curseur en espace arrière.

GESTION DES IMPRIMANTES :

Il est possible d'initialiser des imprimantes par la commande :

setlst <nomfich>

Le <nomfich> est un fichier contenant la ou les chaînes à envoyer à l'imprimante. Comme pour le fichier de commande, pour **setkeys**, les codes de contrôle peuvent être représentés par :

'<caractère>

Ou

''<valeur du caractère>'

Ou encore

''<nom du code de controle>'

Pour avoir tous les codes possibles, vous devez vous reporter au tableau ASCII qui se trouve dans le chapitre APPROFONDISSONS.

Dans de nombreuses imprimantes la valeur 15 représente un code définissant une impression condensée.

La commande :

print #8,chr$(15)

Serait la syntaxe en **basic**.
Sous CP/M, elle devient :

setlst condense

Le fichier condense doit contenir l'une des lignes ci-dessous :

```
"si"
"o
"&f"
"15"
```

Qui représentent toutes la valeur décimale 15.

Il arrive que pour certains programmes d'application, l'écran doit être de 24 lignes * 80 colonnes.
La commande **set24*80** permet d'obtenir ce format d'écran.

Pour l'activer, il suffit de taper :

```
set24*80
```

Ou

```
set24*80 on
```

Pour le désactiver utilisez :

```
set24*80 off
```

GESTION D'UNE INTERFACE SERIE :

Il vous est possible grâce à CP/M de gérer une interface série RS232 ne possédant qu'un seul canal. Pour visualiser ces principales caractèristiques, tapez la commande **setsio**:

```
setsio
```

Ou une commande comprenant l'une (voire la totalité) de ces selections:

```
setsio,
rx 1200,
tx 75,
parity none,
stop 1,
bits 8,
handshake on,
xoff off
```

Pour une nouvelle configuration.

Le débit de l'interface et l'état **xon/xoff** peuvent également être fixés par la commande **device** qui gère les périphériques logiques et physiques. Les périphériques logiques sont indiqués par un deux points (:). Pour visualiser tous les attributs des périphériques installés, tapez:

```
device
```

Pour les modifier : un périphérique.

```
device sio[1200]   fixe le débit à 1200 bauds
device sio[xon]    active le protocole xon/off.
decice sio[noxon]  désactive le protocole xon/off.
```

Vous pouvez également modifier les affectations entre périphériques logiques et physiques

con: correspond à **crt** (clavier/écran)
aux: correspond à **sio** (interface série en option)
lst: correspond à **lpt** (interface Centronicx pour imprimante)

La commande :

device lst:sio

Envoie des données destinées à l'imprimante vers l'interface série.

IL s'agit d'un réacheminement de canal à ne pas confondre avec la commande **pip**. Les deux commandes **get** <nom de fichier> et **put** <nom de fichier> réacheminent les données à destination et en provenance de la console et les sorties vers l'imprimante en les dirigeant vers un fichier particulier et non pas vers un périphérique.

pip

L'utilitaire **pip** (Programme d'Interconnextion aux Périphériques) vous permet d'échanger des informations entre l'ordinateur et ses périphériques.

La forme de ces commandes est:

pip <destination>=<source>

La <source> et la <destination> peuvent être soit un nom de fichier (jockers acceptés) pour la source, soit le code périphérique.

Vous pouvez utiliser les codes suivants :

Pour la source

con :entrée console
aux :entrée auxiliaire
eof :marque de fin de fichier

Pour la destination

con :sortie console
aux : sortie auxiliaire
lst :imprimante
prn : imprimante avec tabulations supplémentaires, numérotation de lignes et sauts de page.

Exemples :

pip b:=a:*.Com

Permet de copier tous les fichiers avec extension **.com** de l'unité **A** sur l'unité B.

pip keyboard.cpm=keys.ccp

Réalise une copie de keys.ccp et l'appelle **keyboard.cpm.**

pip con:=keys.ccp

Envoie le fichier keys.ccp à l'écran (même effet que **type keys.ccp**).

pip lst :=keys.ccp

Envoie le fichier keys.ccp sur l'imprimante.

pip typein.txt=con:

Place des données venant du clavier dans le fichier **typein.txt.**

L'opération s'achève par un **control z**. Pour accéder à une nouvelle ligne, vous devez envoyer à chaque fois un **control j** (code ASCII du retour à la ligne) après [**return**].

Si vous tapez **PIP** sans paramètres, vous aurez alors à l'écran le symbole *: vous pourrez alors rentrer les commandes qui vous intéressent.

Ce procédé est très utile si vous voulez copier des fichiers lorsque **pip.com** ne se trouve ni sur la disquette source ni sur la disquette objet. Vous pouvez charger **pip** à partir de la face d'une disquettes systèmes.

Pour sortir de la commande **pip** appuyez sur la touche **return** à l'apparition du symbole *.

Vous remarquerez que **pip** ne peut effectuer que des copies de fichiers sur un système à une seule unité de disquettes.

GESTION DU SYSTEME :

Les programmes transitoires **dir, erase, rename,** et **type** permettent plus de possibilités que leurs homologues intégrés. Les paramètres secondaires sont indiqués entre crochets.

IL existe sur la face 3 des disquettes systèmes un programme **help** d'aide à la compréhension de toutes ces commandes.

En voici quelques exemples :

dir [full]

Affiche la taille et les natures des fichiers.

erase *.com [confirm]

Permet d'avoir un message qui vous demande une confirmation de destruction à chaque fichier rencontré.

rename

Demande que vous tapiez l'ancien et le nouveau nom du fichier que vous voulez renommer .

rename *.sav=*.bak

Permet de renommer tous les fichiers ayant un suffixe **.bak** en fichier **.sav**

type keys.wp [nopage]

Permet de supprimer la pagination de l'écran.

La commande **set** permet de modifier les attributs tels que **sys** ou **ro**. Cette commande peut être accompagnée de jockers.

```
set *.com [ro]
set keys.ccp [ro]
set a:[ro]
```

Ces commandes affectent l'attribut **ro** (lecture seulement) aux fichiers spécifiés ou au disque entier de façon à empêcher tout effacement intempestif.

```
set *.com [rw]
set keys.ccp [rw]
set a:[rw]
```

Ces commandes affectent l'attribut **rw** (lecture/écriture) aux fichiers spécifiés au disque entier.

```
set *.com [sys]
set keys.ccp [sys]
```

Permettent d'affecter l'attribut **sys** aux fichiers spécifiés. Les fichiers ayant cet attribut ne seront plus listés par la commande **dir**.
Mais ils restent utilisables et de plus ils se trouvent en zone utlisateur 0 et sont donc disponibles dans toutes les zones utilisateur.

```
set *.com [dir]
set keys.ccp [dir]
```

Ces commandes retirent l'attribut **sys**.

Vous pouvez toujours à l'aide de la commande **set**, mettre soit un mot de passe ou une étiquette sur une disquette. Mais vous pouvez aussi affecter ces attributs à un fichier individuellement.

```
set [name=chef]
set [password=bob]
set [protect=on]
```

Agissent sur la disquette de l'unité par défaut.

```
set *.*[password=bob]
set *.*[protect=read]
```

Agissent sur les fichiers de la disquette de l'unité par défaut. Ici les jockers indiquent que tous les fichiers de la disquette sont concernés par la commande.

IL vous est également possible de marquer par la commande **initdir** une disquette en vue de la prise en compte de la date et de l'heure lors d'une création ou de la mise à jour d'un fichier.

```
initdir
set [create=on]
set [upodate=on
dir [full]
```

Ces commandes permettent de marquer la disquette (**initdir**), puis de spécifier que la date et l'heure seront prises en compte pour la création et la mise à jour du fichier. La commande **dir [full]** permet de visualiser la catalogue de la disquette avec la date et l'heure de création ou de mise à jour.

Mais il ne faut pas oublier de mettre l'horloge du système à jour par la commande :

```
date set
```

Vous pourrez la vérifier en tapant.:

```
date
```

Une petite recommandation toutefois il vaut mieux sur une disquette où vous avez mis un mot de passe ou une étiquette ne plus écrire des données sous CP/M 2.2 ou AMSDOS car ces derniers ne possédent pas ces fonctions.

Vous ne pourez accéder qu'aux fichiers de l'unité par défaut.

setdef *,a:

Permet de retouver des fichiers même s'ils ne sont pas dans l'unité par défaut. Dans notre exemple, si l'unité par défaut est **b,** les fichiers seront quand même retrouvés s'ils n'existent que dans l'unité **a.**

setdef [page]
setdef [nopage]

Ces commandes activent ou désactivent la pagination automatique de la console.

La plupart des paramètres spécifiés dans les commandes **devices,set** et **setdef** doivent être initialisés tout comme la date à chaque lancement de CP/M PLUS .

Submit est nécessaire à l'exécution automatique de fichiers de commandes. Ces fichiers de commandes contiennent du texte et il est possible d'y ajouter des lignes de données pour des programmes à condition que ces derniéres commencent par <.

Il existe plusieurs versions de la commande **show** qui permettent d'afficher la taille de l'unité, l'espace disponible, le nombre d'entrées du catalogue disponible sur la disquettes avec les zones utilisateurs contenant les fichiers et s'il existe, le label de la disquette.

show b:
show b:[label]
show b:[users]
show b:[dir]
show b:[drive]

Une fois votre travail terminé sur CP/M plus vous revenez au basic par la commande :

amsdos

UTILISATION DE CP/M 2.2 :

Nous allons dans cette partie insister sur les différences de fonctionnement du système exploité par CP/M 2.2.

Le CP/M 2.2 est chargé à partir des deux premières pistes d'une des disquettes système. Le système d'amorçage diffère de CP/M Plus. Vous devez faire attention à ne pas les confondre.

Normalement, vous pouvez utiliser des disquettes commercialisées, IBM ou formatées pour des données dans n'importe quelle unité de disque. Pour des raisons de fonctionnement du système, nous vous conseillons de limiter leur utilisation à la deuxième unité.

En dehors de l'intervention d'un utilitaire CP/M (tel que **FILECOPY**), le CP/M 2.2 ne vous permet pas d'écrire des données sur une disquette sans qu'elle ait auparavant été amorcée. Le système de formatage (Système, Data, IBM) n'est pris en compte que lors de l'amorçage .

Pour l'unité par défaut A: cet amorçage se produit chaque fois que CP/M 2.2 repasse en mode direct ou que l'on envoie un (control C) après le message **A>**. La deuxième unité est amorcée la première fois que l'on y accède.

Lors d'une tentative d'écriture sur une disquette n'ayant pas été amorcée, vous aurez le message suivant :

bdos err on <drive>: **r/o**

Vous devez appuyer sur n'importe quelle touche pour continuer. Si en plus la disquette est d'un format différent, une erreur en lecture ou en écriture se produit. Tapez sur la touche C pour continuer.

Lorsque vous achetez un logiciel sur disquette vous devez le copier sur une des disquettes sytèmes CP/M 2.2 à l'aide de **FILECOPY** ou **PIP** ou encore convertir votre disquette en disquette système en y ajoutant Le CP/M à l'aide des commandes **BOOTGEN** et **SYSGEN**.

SYSGEN est un programme de copie qui vous indique par des messages l'insertion des disquettes source et cible et copie les pistes de CP/M 2.2 d'une disquette sur l'autre.

BOOTGEN permet lui de copier le secteur 1 de la piste 0 (le chargeur) et le secteur configuration .

DIR n'accepte pas de paramètre hormis l'identificateur de fichier. Les fichiers apparaissent selon leur ordre d'entrée sur le répertoire de la disquette.

STAT possède certaines fonctions de base de **SET** et **SHOW**.

```
STAT
STAT A:
STAT B:
```

Permet d'avoir l'état de la disquette ainsi que la place disponible.

```
STAT *.COM
STAT EX1.BAS
```

Affichent des compléments d'information sur un fichier particulier.

```
STAT *.COM $R/O
STAT EX1.BAS $R/O
```

Permet d'attribuer à un fichier l'option lecture seulement , évitant ainsi les accidents d'écriture.

```
STAT *.COM $R/W
STAT EX1.BAS $R/W
```

Redonne à un fichier l'option lecture/écriture.

```
STAT *.COM $SYS
STAT EX1.BAS $SYS
```

Rend invjsible sur le catalogue un fichier en lui mettant un statut système ce que lui permet d'être inaccessible aux programmes de copie de fichiers tout en restant disponible à tout autre usage.

```
STAT *.COM $DIR
STAT EX1.BAS $DIR
```

Revient à un statut précédent , redonnant au fichier sa place dans le catalogue.

Le programme **FILECOPY** permet la duplication des fichiers sur un système à une seule unité. Il vous donne toutes les indications pour le changement des disquettes, de plus, si vous utilisez des jockers il vous demande une confirmation à chaque fichier rencontré tout en affichant les noms des fichiers au fur et à mesure ede la copie.

```
FILECOPY *.COM
```

Copie tous les fichiers avce l'attribut .COM.

```
FILECOPY PIP.COM
```

Copie le fichier PIP.COM.

L'utilitaire **DISCKIT2** a les mêmes fonctions que **DISCKIT3** mais comme il dispose de moins de place en mémoire il copie les disquettes plus lentement.

Il est possible de transférer des fichiers entre la cassette et la disquette.

CLOAD (Cassette LOAD)
Comporte deux paramètres :

1 nom fichier source (cassette).
2 nom fichier destination (disquette).

Si le premier est omis CLOAD lit le premier fichier qu'il trouve sur la cassette.

Si le deuxième est omis, le fichier destination prendra le même nom que le fichier source.

Il est possible de supprimer les messages liés à l'utilisation des cassettes en mettant un ! comme premier caractère du fichier cassette.

Ex: CLOAD "ficcas" ficdisc.txt

CSAVE (Cassette Save).
Comporte trois paramètres

1: nom fichier source (disquette)
2: nom fichier destination (cassette).
3: vitesse d'enregistrement.

Si le deuxième paramètre est omis le nom sera celui du fichier source.

Il est possible de supprimer les messages liés à l'utilisation des cassettes en mettant un ! comme premier caractère du fichier cassette.

Si les deux noms sont spécifiés, il est possible d'utiliser le troisième paramètre qui définit la vitesse d'enregistrement :

0 pour 1000 bauds.
1 pour 2000 bauds.

Ex: **CLOAD** ficdisc.txt "ficass" 1

Il vous est possible de redéfinir les caractéristiques du clavier de votre AMSTRAD 6128 , du lecteur de disquette et de l'interface série par le programme **SETUP**.

Ce programme est dirigé par un menu :

Lorsqu'un écran est correct ou ne nécessite plus aucune modification, vous passez au suivant en répondant Y à la question :

IS THIS CORRECT (Y/N): _

Il vous est possible d'arrêter le programme à tout moment en tapant simultanément sur les touches CONTROL et C.

Une fois toutes les modifications faites le système vous demande :

DO YOU WANT TO UPDATE YOUR SYSTEM DISC (Y/N) : _

(Confirmez-vous cette mise à jour de la configuration de votre système)
Vous pouvez revenir à l'ancienne configuration en tapant N.

DO YOU WANT RESTART CP/M (Y/N): _

Permet d'initialiser et donc de tester votre nouvelle configuration en (tapant y).

ET ENFIN :

Les commandes :

DISCKIT2 ,
SYSGEN ,
BOOTGEN ,
FILECOPY ,
SETUP ,
CSAVE ET
CLOAD

ne fonctionnent que sous le CP/M 2.2.

Il existe d'autres utilitaires de programmation qui sont disponibles sur la face 4 d'une des disquettes système.

Leur utilisation nécessite une bonne connaissance de l'environnement CP/M.

ASM Assembleur 8080
DDT Utilitaire de mise au point des codes d'assemblage du 8080
DUMP Utilitaire de vidage de fichier sous forme hexadécimale.
ED Editeur de texte simple.
LOAD Convertit les fichiers .HEX, créés avec ASM, en fichiers .COM
MOVCPM Crée le CP/M 2.2 avec une taille TPA réduite.
SUBMIT Utilitaire d'exécution de fichier de commandes
XSUB Utilitaire de traitement par lots.

APPROFONDISSONS

SYSTEMES NUMERIQUES :

Vous avez sans doute remarqué que depuis le début de ce manuel nous avons fait appel à des termes du style:

octets ,bits ,binaire ,hexadécimal

Ces termes familiers à certains peuvent en dérouter d'autres.

Nous avons vu que le cerveau du 6128 est un microprocesseur Z80. Un microprocesseur travaille de manière digitale, ce qui veut dire qu'il ne reconnait dans toutes ses opérations internes que deux états que nous pouvons représenter par 'éteint' ou 'allumé' ou encore par 1 ou 0.

On désigne donc un commutateur sous le terme de **bit**.
En fait dans le microprocesseur, les commutateurs sont regroupés par 8 **bits**, ce que l'on appelle **octet**.

Un groupe de huit **bits** peut être représentés par:

7 6 5 4 3 2 1 0
0 1 0 1 0 1 0 1

A la ligne supérieure figure la numérotation des **bits**. Qui va de 0 à 7, nous voyons en dessous le contenu de chaque **bit** soit 0 ou 1.

Avec un **bit** nous pouvons représenté que deux états.

8 **bits** peuvent déjà en représenter beaucoup plus.

Si nous comparons avec les chifres décimaux auquels nous sommes habitués.

3 2 1 0
7 5 4 2

Nous avons ici numéroté chaque chiffre mais cette fois de 0 à 3.

Comment calculer la valeur d'un tel nombre?

Chaque chiffre a une valeur de 0 à 9 qui représente une valeur dix fois plus importante chaque fois que l'on se déplace vers la gauche.

2
+ 4*10
+ 5*10*10
+ 7*10*10*10

= 7542

Nous procédons de la même manière pour calculer le contenu d'un **octet**. Le système n'est plus ici le système décimal mais le sytème **binaire** car chaque unité au lieu de prendre dix valeurs ne peut plus en prendre que deux.

Un chiffre plus à gauche ne représente non plus une valeur dix fois supérieure mais seulement deux fois.

Ainsi l'**octet** de notre exemple représente la valeur:

0 1 0 1 0 1 0 1

1
+ 0*2
+ 1*2*2
+ 0*2*2*2
+ 1*2*2*2*2
+ 0*2*2*2*2*2
+ 1*2*2*2*2*2*2
+ 0*2*2*2*2*2*2*2

Soit :

1+0+4+0+16+0+64 = 85

Nous pouvons nous faciliter la tâche en faisant une table de conversion:

RANG	VALEUR
0	2'0= 1
1	2'1= 2
2	2'2= 4
3	2'3= 8
4	2'4= 16
5	2'5= 32
6	2'6= 64
7	2'7=128

Vous avez sans doute remarqué que la plus grande valeur que nous pouvons représenter sur un **octet** est :

1 1 1 1 1 1 1 1

soit:

```
   1
+  2
+  4
+  8
+ 16
+ 32
+ 64
+128

=255
```

En effet sur 8 **bits** nous avons 2'8 = 256 possibilités, ce qui nous permet de représenter tous les nombres de 0 à 255.

Sachant que l'ordinateur possède des cases mémoires longues d'un octet, vous voyez maintenant pourquoi l'instruction **poke** ne peut pas avoir un paramètre plus grand que 255.

La manipulation de ces nombres codés reste quand même assez délicate, c'est pourquoi a été introduit un système pour manipuler les **octets**.

En divisant la série de huit **bits** en deux, nous obtenons deux groupes de quatre **bits** qui peuvent chacun représenter 16 valeurs différentes, d'où l'idée de créer un autre système numérique possédant 16 symboles différents pouvant exprimer avec deux chiffres de ce système tout nombre binaire de 8 **bits**.

```
7654 3210

0101 0101
 5    5
```

Chaque **octet** est ainsi divisé en deux demi-**octets** ou **quartets**.

Vous voyez qu'il y a un nouveau problème qui se pose.
Notre système décimal allant de 0 à 9, comment allons-nous représenter les nombres de 10 à 15 ?.

Ce problème a été résolu en ajoutant les lettres de A à F.

Ce système nouvellement créé est appelé système **hexadécimal** .

Voici une table de conversion :

binaire	hexadécimal	décimal
0000	0	0
0001	1	1
0010	2	2
0011	3	3
0100	4	4
0101	5	5
0110	6	6
0111	7	7
1000	8	8
1001	9	9
1010	A	10
1011	B	11
1100	C	12
1101	D	13
1110	E	14
1111	F	15

Nous voyons que notre **octet** contenait la valeur hexadécimal 55.

Sur l'AMSTRAD 6128 les nombres hexadécimaux sont désignés par le symbole & et servent aux programmeurs qui utilisent l'assembleur, langage à mi-chemin entre le BASIC et le langage machine.

LE LANGAGE MACHINE:

Dans le paragraphe ci-dessus nous vous avons expliqué la signification des termes **octets et bits** en vous disant qu'ils étaient surtout utiles pour la programmation en **langage machine.**

Nous allons essayer de vous expliquer en quelques mots quels sont ses avantages et ses inconvénients.

En vous précisant bien qu'il ne s'agit nullement d'un approfondissement mais seulement d'une prise de contact.

Si après avoir lu cette documentation, l'envie de 'bidouiller' directement en langage machine vous démange.

Nous ne pouvons que vous conseiller la lecture des livres de la collection AMSTRAD parus chez MICRO APPLICATION.

Le **langage machine** et le langage de programmation que l'ordinateur peut traiter directement.

Chaque ordinateur possède un micro-processeur qui est le cerveau de l'ordinateur.

Ce circuit intégré est appelé 'unité centrale' ou **CPU** (central processing unit); l'unité centrale commande la marche de l'ordinateur et les appareils connectés ou périphériques.

L'unité centrale est le composant le plus important de l'ordinateur.

Lorsque nous programmons en **langage machine** nous utilisons des instructions qui sont directement exploitables par cette unité centrale ce qui fait que le **langage machine** est différent sur chaque ordinateur .

Sur l'AMSTRAD 6128 l'unité centrale est un Z80 A, c'est une unité centrale très puissante qui comprend plus 600 instructions.

POURQUOI LE LANGAGE MACHINE :

Sur presque tous les ordinateurs familiaux un langage de programmation est intégré, dans la majorité des cas, il s'agit du **basic**

Comme vous l'avez vu tout au long de la documentation le **basic** n'est pas très difficile à apprendre et dans bien des cas il rend de très grands services.

Mais essayons d'aller plus loin et d'imaginer comment l'unité centrale traite les ordres venus du **basic.**

Nous avons vu que l'unité centrale ne pouvait comprendre que le langage machine, il faut donc à chaque fois que l'on veut faire exécuter une instruction **basic** faire appel à un autre programme qui va traduire ces commandes **en langage machine** .

Ce programme est plus communément appelé **interprèteur.** L'inconvénient de ce programme, c'est qu'il traduit instruction après instruction, si dans le programme **basic,** il y a plusieurs fois la même instruction l'**interprèteur** fera sa traduction à chaque fois .Vous voyez le ralentissement qui peut en découler.

Vous avez sans doute remarqué que si l'on programmait en langage machine, nous pourrions économisé le temps pris par l'**interprèteur** .

Malheureusement, le langage machine présente l'inconvénient d'être très abstrait.

Il est en effet plus dur de représenter les nombres, c'est pourquoi des langages de programmation dits évolués tels que le **basic** ou le **logo** ont été développés.

Ces langages travaillent en effet sur des concepts au lieu de travailler sur des chiffres.

Ces langages représentent un bon compromis dans la communication entre l'homme et la machine.

Malheureusement nous avons vu que ces langages ont de gros inconvénients en ce qui concerne la rapidité d'exécution, la place occupée en mémoire voire aussi les possibilités de programmation .

Concernant la rapidité d'exécution certains langages comme le **pascal** utilise ce que l'on appelle un **compilateur** . Ce programme remplit la même fonction que l'interpréteur, avec cette différence qu'au lieu de traduire instruction par instruction le **compilateur** commence d'abord à tout traduire le programme pour générer un programme objet qui sera exécutable directement par l'unité centrale.

Si le programme est modifié, il faut à nouveau faire appel au compilateur ce qui rend la mise au point assez longue.

L'idéal serait d'avoir un **interpréteur** pour la mise au point et un **compilateur** pour l'exécution.

Voici donc bien un premier avantage essentiel du **langage machine** : les programmes machine peuvent être exécutés jusqu'à 1000 fois plus vite que des programmes **basic** .

L'instruction return du **basic** est exécutée en 0,6 millisecondes alors que l'instruction équivalent en machine ne dure que 2.5 microsecondes.

Le **langage machine** est donc 240 fois plus rapide pour l'instruction ret et l'équivalent en **langage machine** de l'instruction **poke** est même 1000 fois plus rapide que celle-ci.

Ces différences sont importantes, si l'on veut effectuer un tri sur un grand nombre de données ou décaler le contenu des sections de la mémoire comme cela est nécessaire pour les programmes de traitement de texte.

De même, pour les programmes de jeux ces programmes ont souvent besoin d'une grande vitesse pour être aussi spectaculaires et passionants et sans le langage machine ils n'y aurait sans doute pas autant de fans qu'actuellement.

Autres avantages du langage machine :

Les programmes machine sont souvent plus courts que les programmes **basic** ce qui permet d'économiser de la place en mémoire.

Quand vous commencerez à programmer en langage machine, vous vous apercevrez qu'un programme de 500 octets est déjà important et qu'il permet de faire un grand nombre de choses.

Par contre il faudrait beaucoup plus de place pour stocker l'équivalent en **basic**.

Pour connaitre la longueur d'un programme en **basic** en octets, appliquez la formule:

print himem-fre(0)-370.

Un autre avantage des programmes en langage machine est qu'il permet seul d'utiliser pleinement toutes les possibilités d'un ordinateur, en **langage machine**, on est par exemple à même de programmer des entrées/sorties. On peut donc communiquer beaucoup plus facilement avec les périphériques.

Ces exemples vous ont montré assez brièvement les énormes avantages du langage machine.

Le langage machine étant simplement constitué d'une suite de nombres, fait qu'il est presque impossible de développer des programmes d'envergure, c'est pourquoi les pionniers de l'informatique ont inventé un langage intermédiaire qui rend le programme machine beaucoup plus clair et compréhensible, l'**assembleur**.

Le langage **assembleur** affecte à chaque code machine une série de symboles.

Ces symboles se composent de:

1 mot d'instruction, la plupart du temps l'abréviation du nom anglais de l'instruction, aussi appelée mnémonique

2 un opérande qui permet d'indiquer des adresses ou des constantes.

La création d'un programme en **langage machine** peut ainsi se faire en écrivant en langage **assembleur**.
Le langage **assembleur** est ensuite traduit en code machine.

NOTRE PREMIER PROGRAMME EN LANGAGE MACHINE :

Pour vous montrer l'intérêt qu'il y a à apprendre le langage machine, rien ne vaut une comparaison entre un programme **basic** et un programme **machine.**

Vous allez taper le programme **basic** suivant :

```
10 HL=&C000
20 POKE HL,&CC
30 HL=HL+1
40 IF HL<=&FFFF THEN 20
50 RETURN
```

Vous tapez ensuite MODE 2 et GOSUB 10 et regardez de qui arrive

Le programme **basic** suivant charge un programme en **langage machine** qui exécute la même chose que le précédent programme.

```
10 MEMORY &9FFF
20 FOR I=&A000 TO &A009
30 READ A
40 POKE I,A
50 NEXT I
60 END
70 DATA &21,&00,&C0,&36,&CC,&23,&BC,&20,&FA,&C9
```

Tapez ensuite MODE 2 chargez le programme en faisant RUN et lancez le avec CALL &A000

Vous avez remarquez que le programme **basic** dure environ 1 minute
Le programme **machine** environ de 1/10 de seconde
La longueur des programmes est de :

basic : 88 octets
machine : 10 octets

Il est évident que certains termes vous paraissent assez abstraits, nous vous rappelons que nous avons juste voulu que vous preniez conscience des avantages du **langage machine**.

Mais il ne faut pas non plus oublier son pricipal inconvénient : sa difficulté.

LES CARACTERES ASCII ET GRAPHIQUES :

Lorsque vous tapez une lettre au clavier de votre ordinateur, vous faites intervenir toute une série de combinaison de contacts électriques.

Les signaux électriques déclenchés par cette touche sont traduit dans les circuits internes de l'ordinateur afin de produire à l'écran certains motifs de points. Ces motifs forment le jeu de caractères du 6128.

Nous avons vu plus haut que sur un octet nous pouvions codifié 256 combinaisons de caractères.

Pour de raisons d'évidence techniques le clavier ne peut avoir au plus qu'une centaine de touches.

Ces 256 possibilités dépassent largement les 96 caractères dits d'une machine à écrire.

C'est pourquoi certains caractères du 6128 ne sont accessibles uniquement à l'aide de la commande **chr$(n)**.

Le jeu de caractères standards s'appelle un 'sous jeu'. Dans le monde de l'informatique ces caractères sont regroupès sous le code ASCII, acronyme d' AMERICAN STANDARD CODE FOR IMFORMATION INTERCHANGE, à l'origine , ce système assurait la liaison entre différents ordinateurs.

Nous allons vous fournir la liste complète de tous les caractères ASCII disponibles sur le 6128.

décimal	octal	hexadécimal	caractères ascii
0	000	00	NUL [ctrl @]
1	001	01	SOH [ctrl a]
2	002	02	STX [ctrl b]
3	003	03	ETX [ctrl C]
4	004	04	EOT[ctrl d]
5	005	05	ENQ[ctrl e]
6	006	06	ACK[ctrl f]
7	007	07	BEL[ctrl g]
8	010	08	BS [ctrl h]
9	011	09	HT [ctrl i]
10	012	0A	LF [ctrl j]
11	013	0B	VT [ctrl k]
12	014	0C	FF [ctrl l]
13	015	0D	CR [ctrl m]
14	016	0E	SO [ctrl n]
15	017	0F	SI [ctrl o]
16	020	10	DLE [ctrl p]
17	021	11	DC1 [ctrl q]
18	022	12	DC2 [ctrl r]
19	023	13	DC3 [ctrl s]
20	024	14	DC4 [ctrl t]
21	025	15	NAK [ctrl u]
22	026	16	SYN [ctrl v]
23	027	17	ETB [ctrl w]
24	030	18	CAN [ctrl x]
25	031	19	EM [ctrl y]
26	032	1A	SUB [ctrl z]
27	033	1B	ESC
28	034	1C	FS
29	035	1D	GS

décimal	octal	hexadécimal	caractères ascii
30	036	1E	RS
31	037	1F	US
32	040	20	SP
33	041	21	!
34	042	22	"
35	043	23	#
36	044	24	$
37	045	25	%
38	046	26	&
39	047	27	'
40	050	28	(
41	051	29	)
42	052	2A	*
43	053	2B	+
44	054	2C	,
45	055	2D	-
46	056	2E	.
47	057	2F	/
48	060	30	0
49	061	31	1
50	062	32	2
51	063	33	3
52	064	34	4
53	065	35	5
54	066	36	6
55	067	37	7
56	070	38	8
57	071	39	9
58	072	3A	:
59	073	3B	;

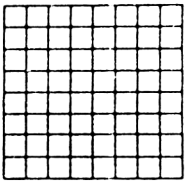
décimal	octal	hexadécimal	caractères ascii
60	074	3C	<
61	075	3D	=
62	076	3E	>
63	077	3F	?
64	100	40	@
65	101	41	A
66	102	42	B
67	103	43	C
68	104	44	D
69	105	45	E
70	106	46	F
71	107	47	G
72	110	48	H
73	111	49	I
74	112	4A	J
75	113	4B	K
76	114	4C	L
77	115	4D	M
78	116	4E	N
79	117	4F	O
80	120	50	P
81	121	51	Q
82	122	52	R
83	123	53	S
84	124	54	T
85	125	55	U
86	126	56	V
87	127	57	W
88	130	58	X
89	131	59	Y

décimal	octal	hexadécimal	caractères ascii
90	132	5A	Z
91	133	5B	[
92	134	5C	\
93	135	5D	]
94	136	5E	^
95	137	5F	_
96	140	60	`
97	141	61	a
98	142	62	b
99	143	63	c
100	144	64	d
101	145	65	e
102	146	66	f
103	147	67	g
104	150	68	h
105	151	69	i
106	152	6A	j
107	153	6B	k
108	154	6C	l
109	155	6D	m
110	156	6E	n
111	157	6F	o
112	160	70	p
113	161	71	q
114	162	72	r
115	163	73	s
116	164	74	t
117	165	75	u
118	166	76	v
119	167	77	w
120	170	78	x
121	171	79	y
122	172	7A	z
123	173	7B	{
124	174	7C	|
125	175	7D	}
126	176	7E	~

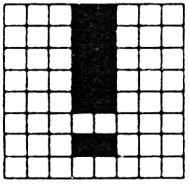
JEU DE CARACTERES.

Les caractères décrits dans les pages suivantes, sont contenus dans des matrices standards de 8*8 utilisées pour l'affichage du 6128.

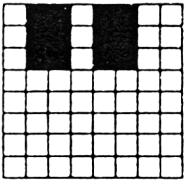
L'utilisateur a la possibilité de définir des caractères pour créer des effets spéciaux en les regroupant ou en les alignant.



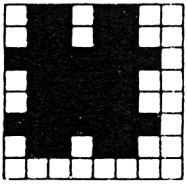
32 &H20
&X00100000



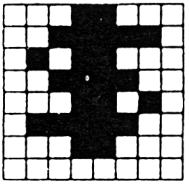
33 &H21
&X00100001



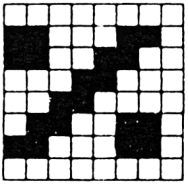
34 &H22
&00100010



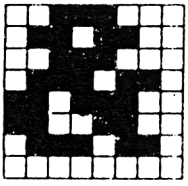
35 &H23
&X00100011



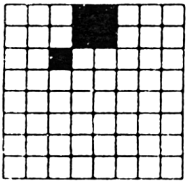
36 &H24
&X00100100



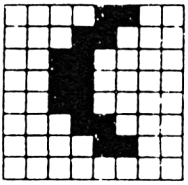
37 &H25
&X00100101



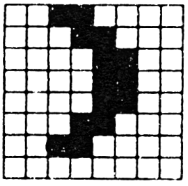
38 &H26
&X00100110



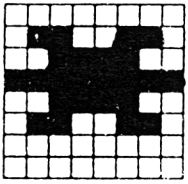
39 &H27
&X00100111



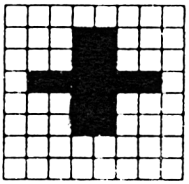
40 &H28
&X00101000



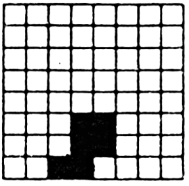
41 &H29
&X00101001



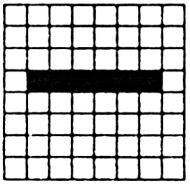
42 &H2A
&X00101010



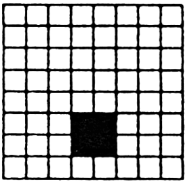
43 &H2B
&X00101011



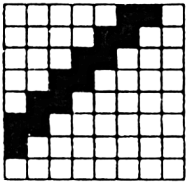
44 &H2C
&X00101100



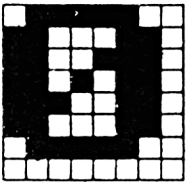
45 &H2D
&X00101101



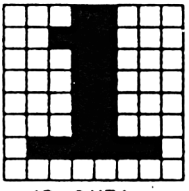
46 &H2E
&X00101110



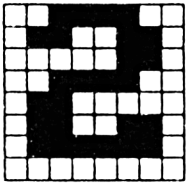
47 &H2F
&X00101111



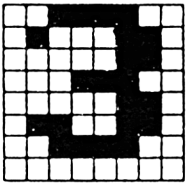
48 &H30
&X00110000



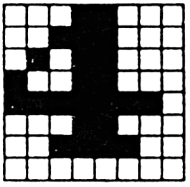
49 &H31
&X00110001



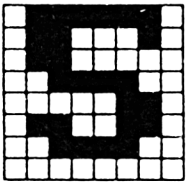
50 &H32
&X00110010



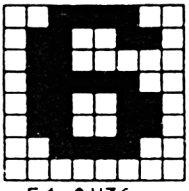
51 &H33
&X00110011



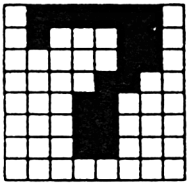
52 &H34
&X00110100



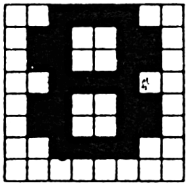
53 &H35
&X00110101



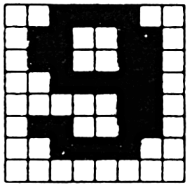
54 &H36
&X00110110



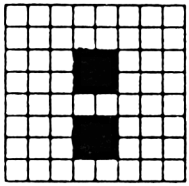
55 &H37
&X00110111



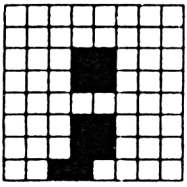
56 &H38
&X00111000



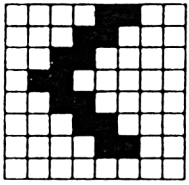
57 &H39
&X00111001



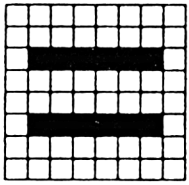
58 &H3A
&X00111010



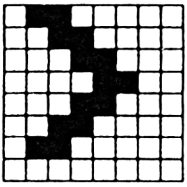
59 &H3B
&X00111011



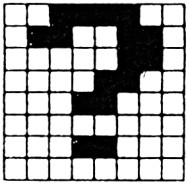
60 &H3C
&X00111100



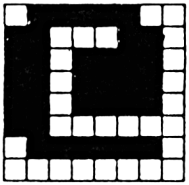
61 &H3D
&X00111101



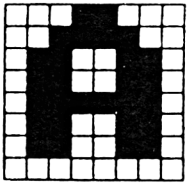
62 &H3E
&X00111110



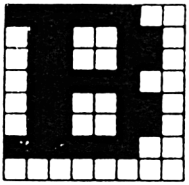
63 &H3F
&X00111111



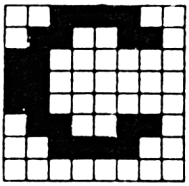
64 &H40
&X01000000



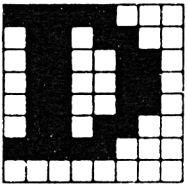
65 &H41
&X01000001



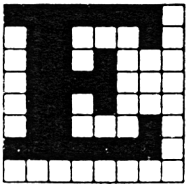
66 &H42
&X01000010



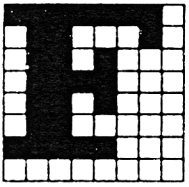
67 &H43
&X01000011



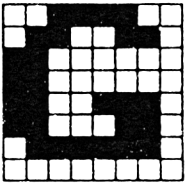
68 &H44
&X01000100



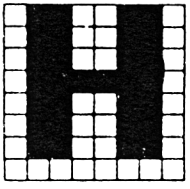
69 &H45
&X01000101



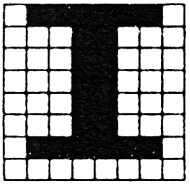
70 &H46
&X01000110



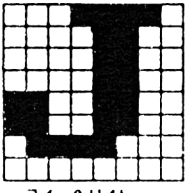
71 &H47
&X01000111



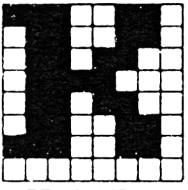
72 &H48
&X01001000



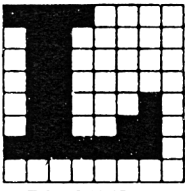
73 &H49
&X01001001



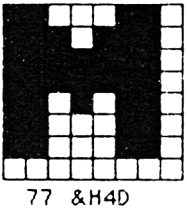
74 &H4A
&X01001010



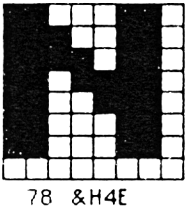
75 &H4B
&X01001011



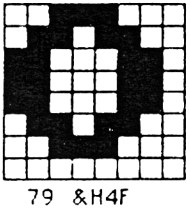
76 &H4C
&X01001100



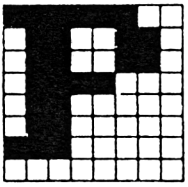
77 &H4D
&X01001101



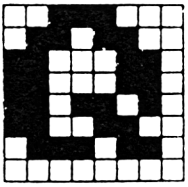
78 &H4E
&X01001110



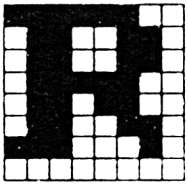
79 &H4F
&X01001111



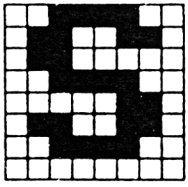
80 &H50
&X01010000



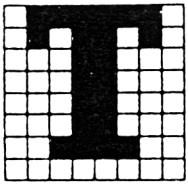
81 &H51
&X01010001



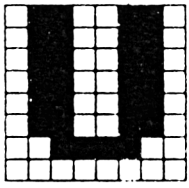
82 &H52
&X01010010



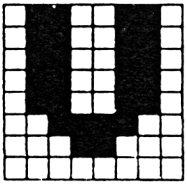
83 &H53
&X01010011



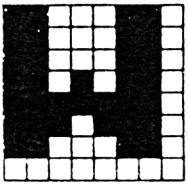
84 &H54
&X01010100



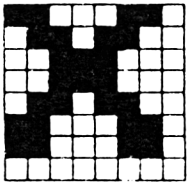
85 &H55
&X01010101



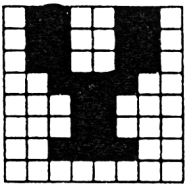
86 &H56
&X01010110



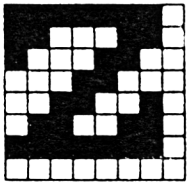
87 &H57
&X01010111



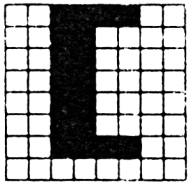
88 &H58
&X01011000



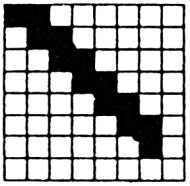
89 &H59
&X01011001



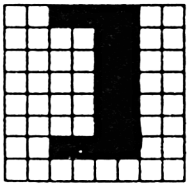
90 &H5A
&X01011010



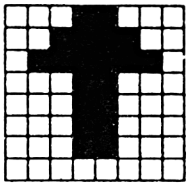
91 &H5B
&X01011011



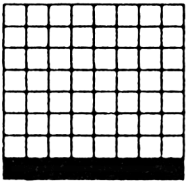
92 &H5C
&X01011100



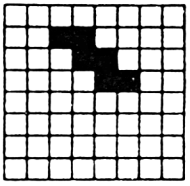
93 &H5D
&X01011101



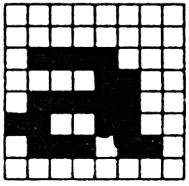
94 &H5E
&X01011110



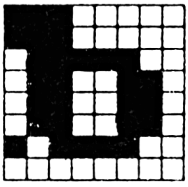
95 &H5F
&X01011111



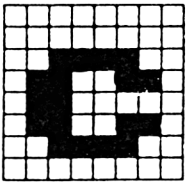
96 &H60
&X01100000



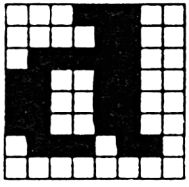
97 &H61
&X01100001



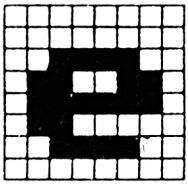
98 &H62
&X01100010



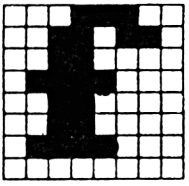
99 &H63
&X01100011



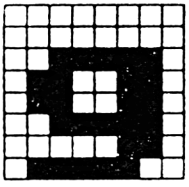
100 &H64
&X01100100



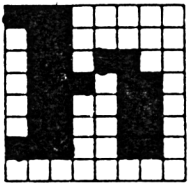
101 &H65
&X01100101



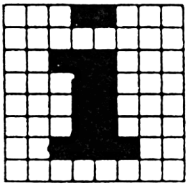
102 &H66
&X01100110



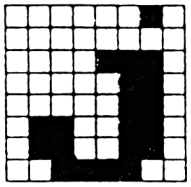
103 &H67
&X01100111



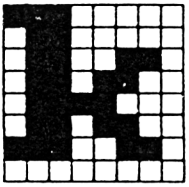
104 &H68
&X01101000



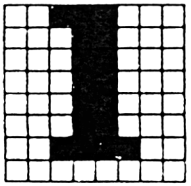
105 &H69
&X01101001



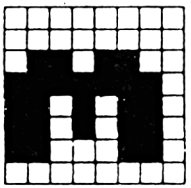
106 &H6A
&X01101010



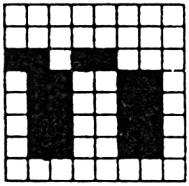
107 &H6B
&X01101011



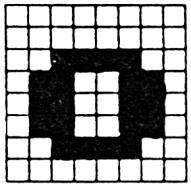
108 &H6C
&X01101100



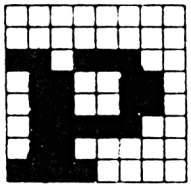
109 &H6D
&X01101101



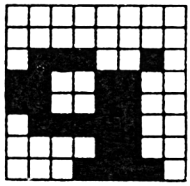
110 &H6E
&X01101110



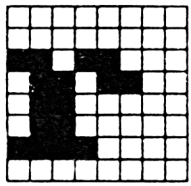
111 &H6F
&X01101111



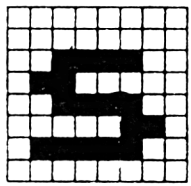
112 &H70
&X01110000



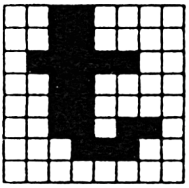
113 &H71
&X01110001



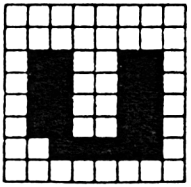
114 &H72
&X01110010



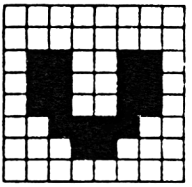
115 &H73
&X01110011



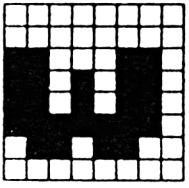
116 &H74
&X01110100



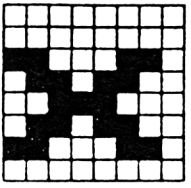
117 &H75
&X01110101



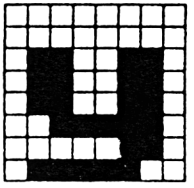
118 &H76
&X01110110



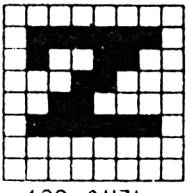
119 &H77
&X01110111



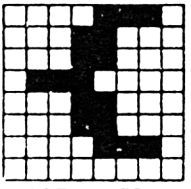
120 &H78
&X01111000



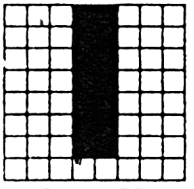
121 &H79
&X01111001



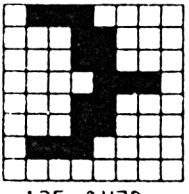
122 &H7A
&X01111010



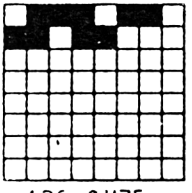
123 &H7B
&X01111011



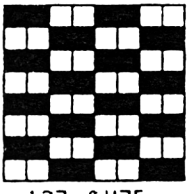
124 &H7C
&X01111100



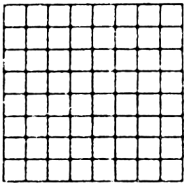
125 &H7D
&X01111101



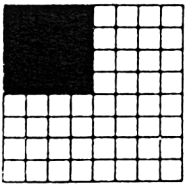
126 &H7E
&X01111110



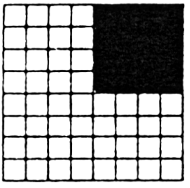
127 &H7F
&X01111111



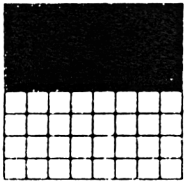
128 &H80
&X10000000



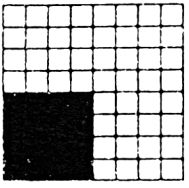
129 &H81
&X10000001



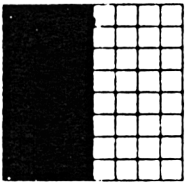
130 &H82
&X10000010



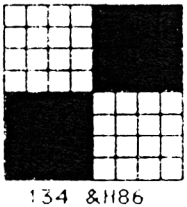
131 &H83
&X10000011



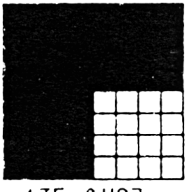
132 &H84
&X10000100



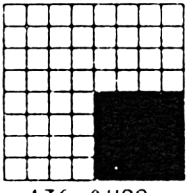
133 &H85
&X10000101



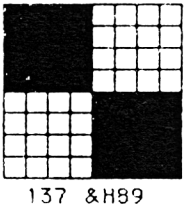
134 &H86
&X10000110



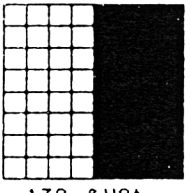
135 &H87
&X10000111



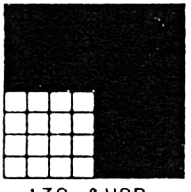
136 &H88
&X10001000



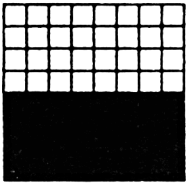
137 &H89
&X10001001



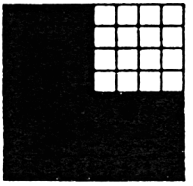
138 &H8A
&X10001010



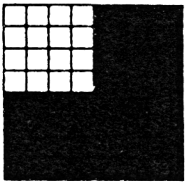
139 &H8B
&X10001011



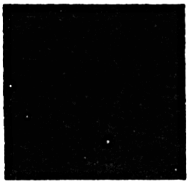
140 &H8C
&X10001100



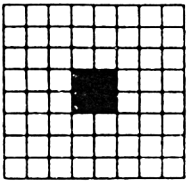
141 &H8D
&X10001101



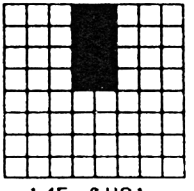
142 &H8E
&X10001110



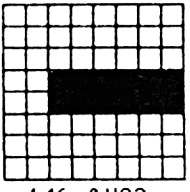
143 &H8F
&X10001111



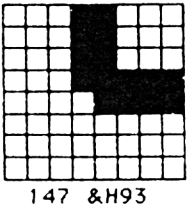
144 &H90
&X10010000



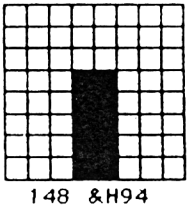
145 &H91
&X10010001



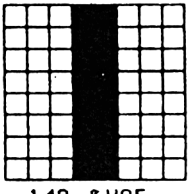
146 &H92
&X10010010



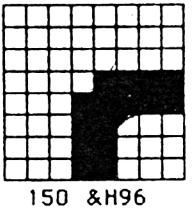
147 &H93
&X10010011



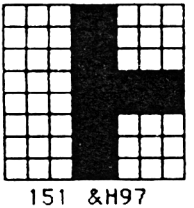
148 &H94
&X10010100



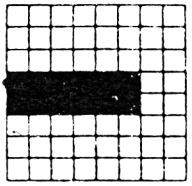
149 &H95
&X10010101



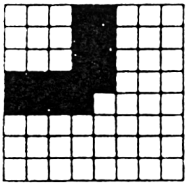
150 &H96
&X10010110



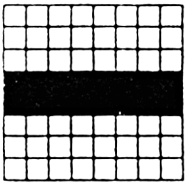
151 &H97
&X10010111



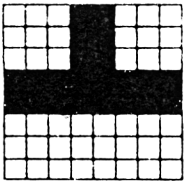
152 &H98
&X10011000



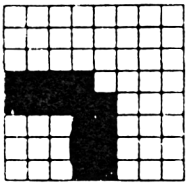
153 &H99
&X10011001



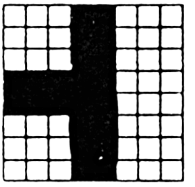
154 &H9A
&X10011010



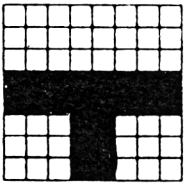
155 &H9B
&X10011011



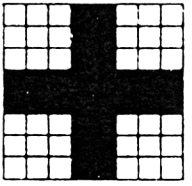
156 &H9C
&X10011100



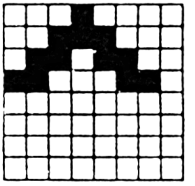
157 &H9D
&X10011101



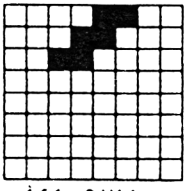
158 &H9E
&X10011110



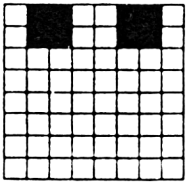
159 &H9F
&X10011111



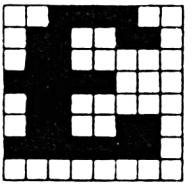
160 &HA0
&X10100000



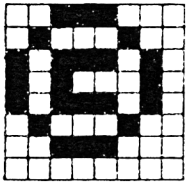
161 &HA1
&X10100001



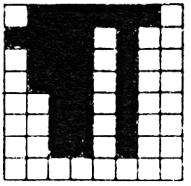
162 &HA2
&X10100010



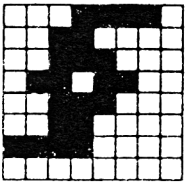
163 &HA3
&X10100011



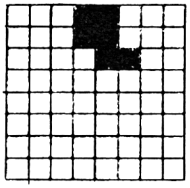
164 &HA4
&X10100100



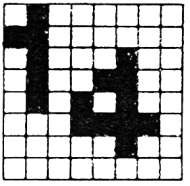
165 &HA5
&X10100101



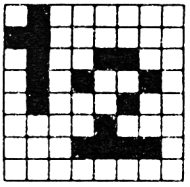
166 &HA6
&X10100110



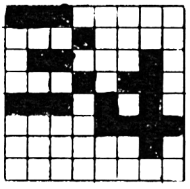
167 &HA7
&X10100111



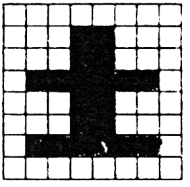
168 &HA8
&X10101000



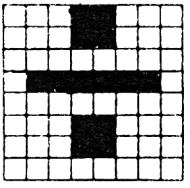
169 &HA9
&X10101001



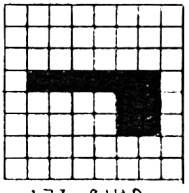
170 &HAA
&X10101010



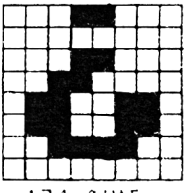
171 &HAB
&X10101011



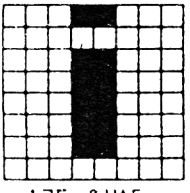
172 &HAC
&X10101100



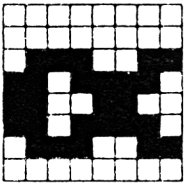
173 &HAD
&X10101101



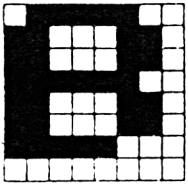
174 &HAE
&X10101110



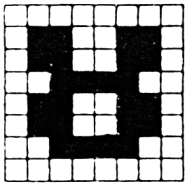
175 &HAF
&X10101111



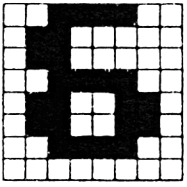
176 &HB0
&X10110000



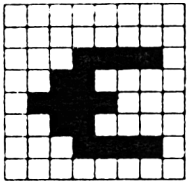
177 &HB1
&X10110001



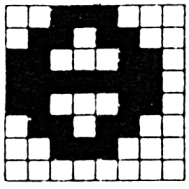
178 &HB2
&X10110010



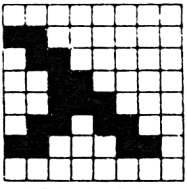
179 &HB3
&X10110011



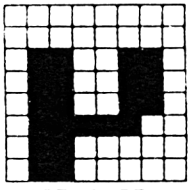
180 &HB4
&X10110100



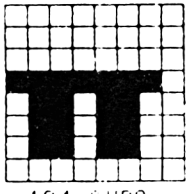
181 &HB5
&X10110101



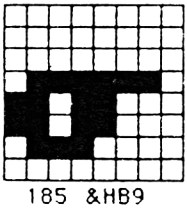
182 &HB6
&X10110110



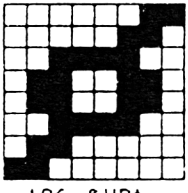
183 &HB7
&X10110111



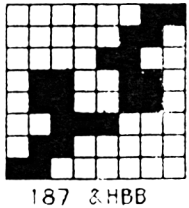
184 &HB8
&X10111000



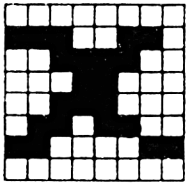
185 &HB9
&X10111001



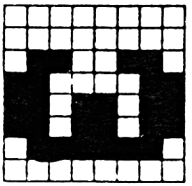
186 &HBA
&X10111010



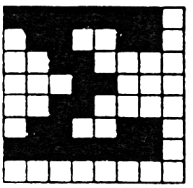
187 &HBB
&X10111011



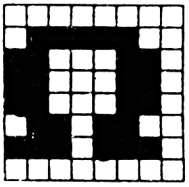
188 &HBC
&X10111100



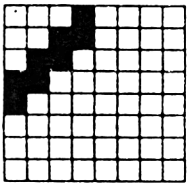
189 &HBD
&X10111101



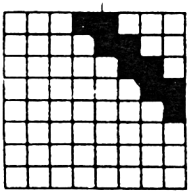
190 &HBE
&X10111110



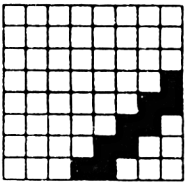
191 &HBF
&X10111111



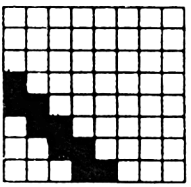
192 &HC0
&X11000000



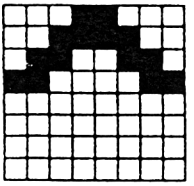
193 &HC1
&X11000001



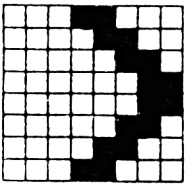
194 &HC2
&X11000010



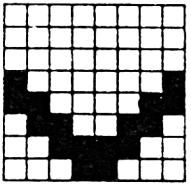
195 &HC3
&X11000011



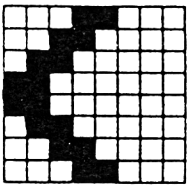
196 &HC4
&X11000100



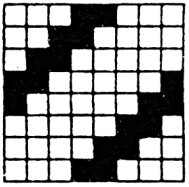
197 &HC5
&X11000101



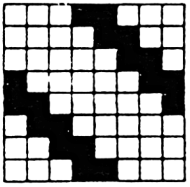
198 &HC6
&X11000110



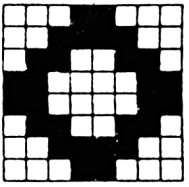
199 &HC7
&X11000111



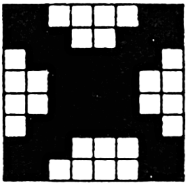
200 &HC8
&X11001000



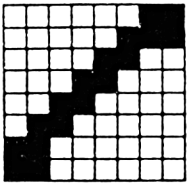
201 &HC9
&X11001001



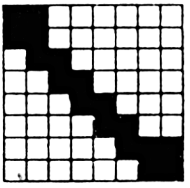
202 &HCA
&X11001010



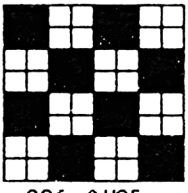
203 &HCB
&X11001011



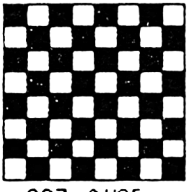
204 &HCC
&X11001100



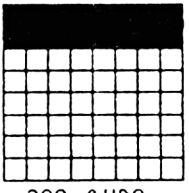
205 &HCD
&X11001101



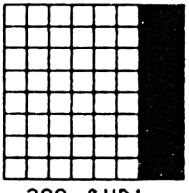
206 &HCE
&X11001110



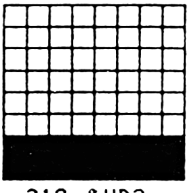
207 &HCF
&X11001111



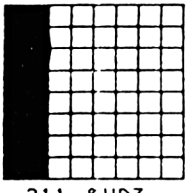
208 &HD0
&X11010000



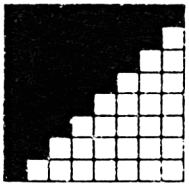
209 &HD1
&X11010001



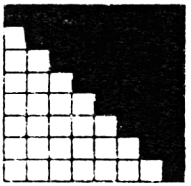
210 &HD2
&X11010010



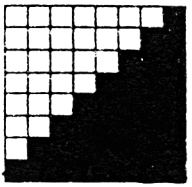
211 &HD3
&X11010011



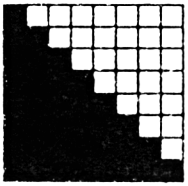
212 &HD4
&X11010100



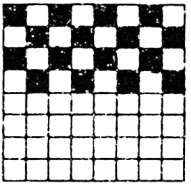
213 &HD5
&X11010101



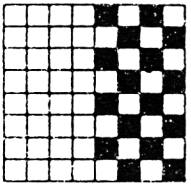
214 &HD6
&X11010110



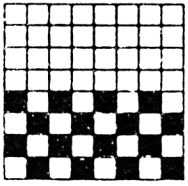
215 &HD7
&X11010111



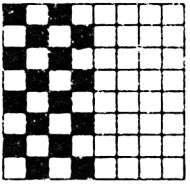
216 &HD8
&X11011000



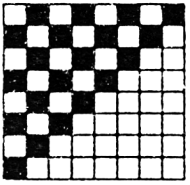
217 &HD9
&X11011001



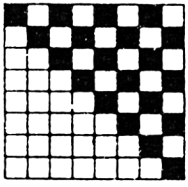
218 &HDA
&X11011010



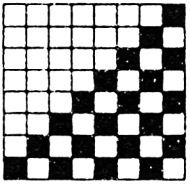
219 &HDB
&X11011011



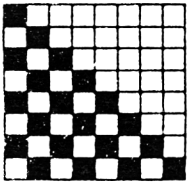
220 &HDC
&X11011100



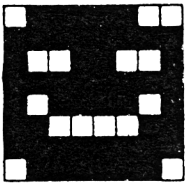
221 &HDD
&X11011101



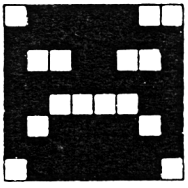
222 &HDE
&X11011110



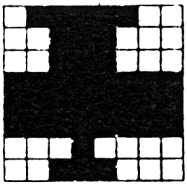
223 &HDF
&X11011111



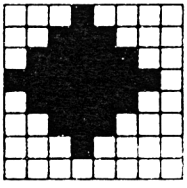
224 &HE0
&X11100000



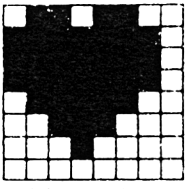
225 &HE1
&X11100001



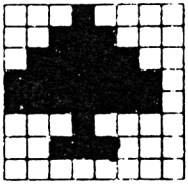
226 &HE2
&X11100010



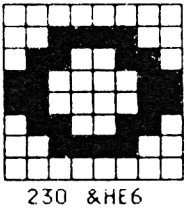
227 &HE3
&X11100011



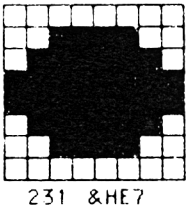
228 &HE4
&X11100100



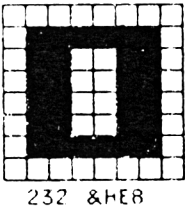
229 &HE5
&X11100101



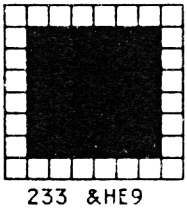
230 &HE6
&X11100110



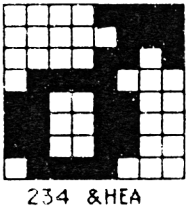
231 &HE7
&X11100111



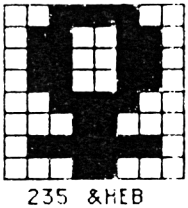
232 &HE8
&X11101000



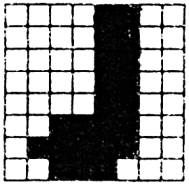
233 &HE9
&X11101001



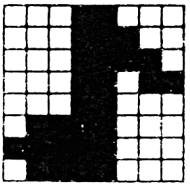
234 &HEA
&X11101010



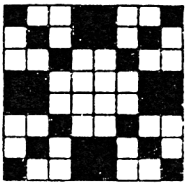
235 &HEB
&X11101011



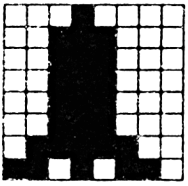
236 &HEC
&X11101100



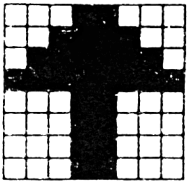
237 &HED
&X11101101



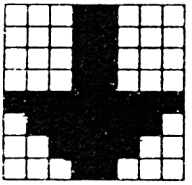
238 &HEE
&X11101110



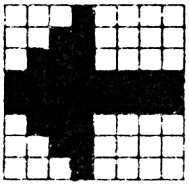
239 &HEF
&X11101111



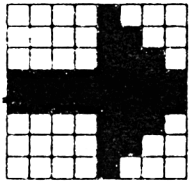
240 &HF0
&X11110000



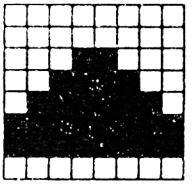
241 &HF1
&X11110001



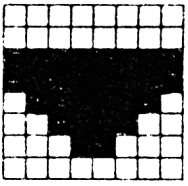
242 &HF2
&X11110010



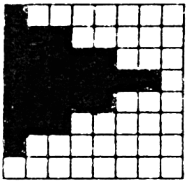
243 &HF3
&X11110011



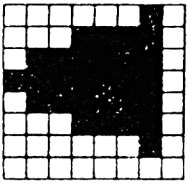
244 &HF4
&X11110100



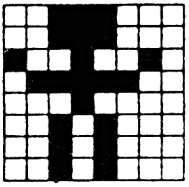
245 &HF5
&X11110101



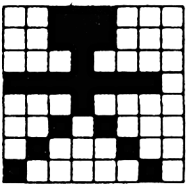
246 &HF6
&X11110110



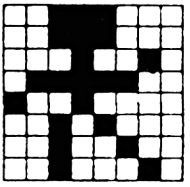
247 &HF7
&X11110111



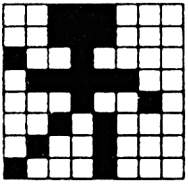
248 &HF8
&X11111000



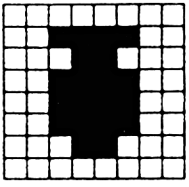
249 &HF9
&X11111001



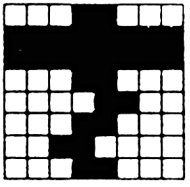
250 &HFA
&X11111010



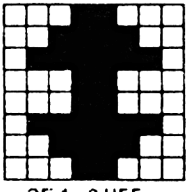
251 &HFB
&X11111011



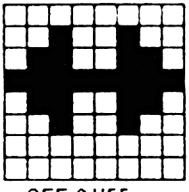
252 &HFC
&X11111100



253 &HFD
&X11111101



254 &HFE
&X11111110



255 &HFF
&X11111111

Dans ce chapitre nous allons vous initier à la programmation du LOGO sur AMSTRAD 6128 :

LE LANGAGE LOGO.

Le LOGO est un langage assez récent (des années 70), il a été conçu par une équipe de chercheurs en informatique et d'éducateurs menés par Seymour Papert .

La raison d'être de ce langage est de permettre à quiconque de faire des programmes sans avoir des notions d'informatique.

Le LOGO est très populaire auprès de jeunes enfants grâce notamment à sa tortue graphique, qui peut être déplacée à l'aide de commandes simples.

La version que nous allons découvrir s'appelle DR. LOGO, elle a été développée spécialement pour les AMSTRAD. afin de tirer partie au maximun des possibilités de l'ordinateur.

Pour lancer DR LOGO , vouz devez inserer la face 1 d'une des disquettes système dans l'unité de disque et tapez :

```
|CPM
```

Une fois le curseur à l'écran vous prenez la face n°3 et vous tapez :

```
SUBMIT LOGO3
```

Vous devez ensuite avoir à l'écran un message d'acceuil ainsi qu'un point d'interrogation. Ce qui signifie que l'ordinateur est prêt à recevoir vos instructions .

ESSAYONS !

Tapez au clavier la commande ci-contre :

fd 50 <return>

Vous verrez alors apparaître à l'écran une flèche grand format (la célèbre tortue du LOGO); la flèche ou tortue a avancé de 50 unités en tracant une ligne derrière elle.

L'écran s'est divisé en deux laissant une grande place pour le graphique et au bas de l'écran un espace pour vos commandes.

Vous voyez qu'en faisant fd (ForwarD) la tortue avance du nombre d'unités spécifié dans la commande.

Vous pouvez aussi la faire reculer : par bk (BacK), mais aussi aller à droite rt (RighT) ou à gauche lt (LefT) ou encore effacer l'écran par cs (Clear Screen).

Vous voyez que la tortue est vraiment un animal docile qui vous obéit au doigt et au clavier...

POUR ALLER PLUS LOIN.

Jusqu'à présent nous n'avons tapé que des ordres simples ou des primitives un peu comme les commandes directes en basic, il serait intéressant de pouvoir exécuter tout un groupe de commandes, ce qui en LOGO se nomme PROCEDURE.

Une procédure est tout simplement un groupe de primitives regroupées sous un nom .

Une procédure simple :

Nous voulons dessiner un carré de 50 unités de côté , vous savez qu'il nous suffit de taper 4 fois l'ordre ci-dessous:

```
fd 50 rt 90
```

C'est un peu long à taper nous pouvons alors employer l'odre REPEAT en écrivant:

```
repeat 4 [fd 50 rt 90]
```

C'est bien, mais il y a un problème , à chaque fois que nous voudrons dessiner un carré il faudra taper la ligne, il nous faut donc créer une procédure qui dessinera un carré :

```
to carre
repeat 4 [fd 50 rt 90]
end
```

C'est tout : notre procédure est créée, pour l'exécuter il suffit de taper son nom. Nous avons pris comme nom carré parce que c'est le plus facile à retenir mais nous aurions pu prendre n'importe quel autre nom.

Nous avons maintenant une procédure pour dessiner un carré de 50 unités de coté.

Il est possible d'introduire des paramètres dans une procédure, dans notre exemple le carre ne pouvez avoir que 50 unités de côté.

```
to carre :cote
repeat 4 [fd :cote rt 90]
end
```

Nous avons introduit dans cette procédure une variable :cote permettant de définir notre carré. Vous remarquerez les deux points précédent le mot "cote", ils signalent au LOGO que le mot qui va suivre devra être considéré comme une variable.

Ce langage DR LOGO permet aussi de stocker des valeurs et de les transmettre à une procédure.

Nous allons modifier notre procédure CARRE:

```
to carre
repeat 4 [fd :cote rt 90]
make "cote :cote +4
carre
end
```

Pour interrompre le programme, tapez sur ESC.

Vous pouvez connaître la valeur du côté par la commande :

```
:cote
```

Le LOGO vous donne alors sa valeur.

CORRECTION DES PROGRAMMES :

Dans tout langage informatique, vous devez avoir la possibilité de modifier et corriger vos procédures. Le DR LOGO est particulièrement bien fourni sur ce point-là.

Vous pouvez modifier les procédures ou les lignes de commandes directes à l'aide des flèches de déplacement du curseur.

En combinant la touche CONTROL avec les flèches de déplacement du curseur, vous vous déplacez d'un bord à l'autre de l'écran.

Quelques autres touches peuvent aussi être utilisées:

CLR : efface le caractère qui se trouve sous le curseur.
DEL : efface le caractère à gauche du curseur.
RETURN : passe à la ligne suivante.
ESC : indique la fin de la correction.
COPY : indique que la procédure est corrigée.

Pour corriger une procédure déjà existante, il faut taper la commande ed:

ex: ed "carre

Le DR LOGO affiche alors le contenu de la procédure carré.

PLACE MEMOIRE:

Le DR LOGO utilise un espace de travail qui se compose de noeuds, vous pouvez à tout moment connaître le nombre de noeuds en tapant :

```
nodes
```

Il se peut que parfois DR LOGO se bloque quelques instants, ceci est dû au fait qu'il n'y a plus de place en mémoire et qu'il est obligé de faire le ménage. On peut cependant le devancer en demandant l'exécution de se nettoyage par la commande :

```
recycle
```

Nous avons terminé notre petit tour d'horizon du DR LOGO, nous allons maintenant examiner plus en détail la liste des primitives.

Pour sortir du langage vous tapez:

```
bye
```

LISTE DES PRIMITIVES PAR ODRE ALPHABETIQUE:

TRAITEMENT DES MOTS:

ascii

Retourne le code ASCII du caractère entré.

ex: ?**ascii** "a
65

bf (but first= sauf le premier)

Donne le mot entré sans son premier élément.

?**bf** "essai
ssai
?**bf** [4 5 6]
[5 6]

bl (but last = sauf le dernier)

Donne le mot entré sans son dernier élément .

?**bl** "essai"
essa
?bl [4 5 6]
[4 5]

char

Retourne le caractère qui correspond au code ASCII

```
?char 65
a
```

count

Donne le nombre d'éléments du mot entré

```
?count "annee
5
?count [ 4 5 6]
3
```

emptyp

Retourne la valeur booléenne vrai (true) ou faux (false) selon que l'objet entré est vide ou non

```
?emptyp "
true
?emptyp []
true
?emptyp "tt
false
```

first

Donne le premier élément de l'objet en supprimant les crochets de la ligne

```
?first "essai
e
?first [1 2 3]
1
```

fput

Retourne un nouvel objet en concaténant le premier élément au deuxième

```
?fput "e "ssai
essai
?fput 4 [5 6]
[4 5 6]
```

item

Donne l'élément se trouvant à la position spécifiée.

```
?item 2 "essai"
s
```

last

Retourne le dernier élément entré

```
? last "essai
i
```

lc

Convertit tous les caractères alphabétiques en minuscules

```
?lc "ESSAI
essai
```

list

Retourne la liste des éléments entrés en les mettant entre crochets

```
?list 1 2 3 4
[1 2 3 4]
```

listp

Donne TRUE (vrai) ou FALSE (faux) selon que l'objet entré est une liste ou pas.

```
?listp "essai
false
?listp [ ceci est un essai]
true
```

lput

Forme un nouvel objet en mettant le premier élément à la suite du deuxième

```
?lput "s "essai
essais
```

memberp

Retourne la valeur TRUE (vrai) si le premier élément fait partie du deuxième

```
?memberp "i "essai
true
?memberp "z "essai
false
```

numberp

Donne la valeur TRUE (vrai) si l'objet est un nombre.

```
?numberp 45.56
true
?numberp "essai
false
```

piece

Retourne un objet composé des éléments de l'objet initial déterminés par les deux premiers paramètres.

```
?piece 2 4 "exemple
xem
?piece 2 4 [voila un bon exemple]
[un bon exemple]
```

se

Donne la liste des éléments entrés en supprimant les crochets

```
?make "instr_list rl
repeat 4 [fd 50 rt 90]
?run (se "cs :instr_list "ht)
```

(Vous obtenez le tiret en actionnant [shift]0)

shuffle

Retourne une liste dans un ordre aléatoire.

```
?shuffle [q s d f]
[f d s q]
```

uc

Convertit les caractères alphabétiques en majuscules

```
?uc "essai
ESSAI
```

where

Donne un nombre calculé découlant de l'expression memberp vraie

```
?memberp "s "essai
true
?show where
2
```

word

Retourne un seul mot à partir des mots rentrés

```
?word "ams "trad
amstrad
```

wordp

Donne une réponse TRUE (vrai) si l'objet est un nombre ou un mot.

```
?wordp "essai
true
?wordp []
false
?wordp [1 2 3]
false
```

OPERATIONS ARITHMETIQUES :

arctan

Donne l'arc-tangente (en degrès) du nombre entré.

```
?arctan 0
0
?arctan 1
45
```

cos

Retourne le cosinus du nombre entré en degrés

```
?cos 90
0
?cos 180
-1
```

int

Donne la partie entière du nombre entrè.

```
?int 2/3
0
?int 6.78
6
```

quotient

Retourne le résultat de la division entière des deux nombres

```
?quotient 15 8
1
?15/8
1.875
```

random

Retourne un nombre entier inférieur au nombre entré et pris au hasard

```
?random 30
```

remainder

Donne le reste entier du quotient du premier nombre entré par le second.

```
?remainder 15 2
1
?remainder 10 5
0
```

rerandom

Provoque la répétition aléatoire d'une expression

```
?repeat 10 [(type random 10 char 9)]
9 2 0 5 1 4 7 6 8
?repeat 10 [(type random 10 char 9)]
?rerandom
?repeat 10 [(type random 10 char 9)]
3 7 5 3 2 0 4 2 6
?rerandom
?repeat 10 [(type random 10 char 9)]
3 7 5 3 2 0 4 2 6
```

round

Retourne le nombre arrondi à l'entier supérieur du nombre entré.

```
?round 4.89809809
5
?round 4.222222
4
```

sin

Donne le sinus du nombre entré en degré.

```
?sin 50
0.642787575721741
```

+

Retourne la somme des nombres entrès

```
? + 4 2
6
? 8 + 8
16
```

—

Retourne la différence des deux nombres

```
? - 7 5
2
? 10 - 5
5
```

*

Retourne le produit des deux nombres

```
? * 4 5
20
? 4 * 5
20
```

/

Retourne le quotient décimal des deux nombres

```
? / 6 3
2
? 8 / 4
2
```

OPERATIONS LOGIQUES:

and

Répond TRUE (vrai) si le résultat des expressions entrées est vrai.

```
?and (9<10) (4<5)
true
?and (9>10) (4>5)
false
```

not

Répond TRUE (vrai) ou FALSE (faux) selon que le résultat de l'expression est vrai ou pas.

```
?not (4>5)
true
?not (4<5)
false
```

OR

Répond TRUE (vrai) si toutes les expressions sont vraies

```
?or (1=2) (2=3)
false
?or (1<2) (3>2)
true
```

=

Répond TRUE (vrai) si les deux expressions sont égales dans le cas contraire répond FALSE (faux).

```
?= "6128 "6128
true
? 4 = 5
false
```

>

Répond TRUE (vrai) si le premier élément est effectivement plus grand que le second, FALSE (faux) dans le cas contraire.

```
?> 4 5
false
?> 5 4
true
```

<

Répond TRUE (vrai) si le premier élément est effectivement plus petit que le second, FALSE (faux) dans le cas contraire.

```
?< 4 5
true
?< 5 4
false
```

VARIABLES :

Comme nous l'avons vu dans certains exemples le DR LOGO autorise la manipulation des variables.

Elles peuvent être locales, c'est à dire uniquement accessibles à la procédure en cours, par la commande **local**:

```
>( local "a "b "c
```

Pour donner une valeur à une variable:

make:

```
 make "cote 40
?:cote
40
```

Pour savoir si une variable est déjà utilisée, vous devez utiliser la commande **namep**

```
? make "nouvel "an
?:nouvel
an
?namep "nouvel
true
?namep "an
false
```

thing

Retourne la valeur de la variable

```
?make "langage "logo
?thing "langage
logo
```

En plus des primitives le DR LOGO possède également des procédures toutes faites qui facilitent grandement la programmation. Nous allons les examiner plus en détail.

define

Construit la procédure définie dans la liste et portant le nom spécifié

```
?define "dire.coucou [[] [pr "coucou"]]
?po "dire.coucou
to dire.coucou
pr "coucou
.text "dire.coucou
[[] [pr "coucou]]
end
```

end

indique la fin de la procédure, doit se trouver seulement sur la denière ligne de la procédure.

```
?to dessin
>repeat 3 [fd 50 rt 120]
>end
dessin defined
```

po

Sert à afficher sur l'écran la procédure ou la variable mise après le guillemet.

```
?po "dessin
?to dessin
>repeat 3 [fd 50 rt 120]
>end
?po "cote
cote is 3    (la valeur du coté est 3)
```

pots

Permet d'avoir à l'écran le titre de toutes les procédures présentes dans l'espace de travail.

```
?pots
```

text

Donne la liste de la procédure spécifiée

```
?text "dessin
[[] [repeat 3 [fd 50 rt 120]]]
```

to

Indique au DR LOGO que nous allons commencer la définition d'une procédure.

PROCEDURES POUR LA CORRECTION:

ed

Permet de charger à l'écran la procédure et/ou les variables désignées dans la mémoire d'édition de l'écran.

```
?ed "dessin
```

edall

Charge sur l'écran toutes les variables et procédures de l'espace de travail désignées dans la mémoire de l'écran et entre l'éditeur d'écran.

```
?edall
```

edf

Charge directement le fichier spécifié dans la mémoire de l'écran ou crée un nouveau fichier en entrant l'éditeur d'écran avec une mémoire vide.

```
?edf "dessin
```

PROCEDURES POUR LA GESTION DE L'IMPRIMANTE :

copyon

Edite le texte par écho.

```
?copyon
```

copyoff

Arrête l'édition du texte par écho.

```
?copyoff
```

PROCEDURES POUR LA GESTION DE L'ECRAN :

ct

Efface totalement l'écran et ramène le curseur en haut à gauche de celui-ci.

```
?ct
```

cursor

Donne la position actuelle du curseur (numéro de colonne et numéro de ligne).

```
?ct
?cursor
[0 1]
```

pr

Affiche sur l'écran les objets au clavier sur l'écran texte, enlève les crochets de la liste et va à la ligne.

```
?pr [1 2 3]
1 2 3
?pr [x y z ]
x y z
```

setcursor

Détermine la position du curseur

```
?ct
to curseur
>make "x random 20
>make "y random 12
>setcursor  list :x :y pr "*
>end
```

setsplit

Permet de définir le nombre de l'écran texte.

```
?setsplit 12
```

show

Affiche les objets entrés au clavier sur la fenêtre de l'écran texte, en gardant les crochets et en passant à la ligne suivante.

ts

Permet de réserver tout l'écran pour le texte.

```
?ts
```

type

Affiche les objets entrés au clavier dans la fenêtre de l'écran texte en enlevant les crochets mais sans passer à la ligne suivante.

```
?type [x y z]
[x y z ]
```

PROCEDURES PERMETTANT DE GERER L'ECRAN GRAPHIQUE :

clean

Efface tout l'écran graphique sans toucher a la tortue.

```
?fd 50
?clean
?rt 90
?fd 50
```

cs

Efface l'écran graphique et ramène la tortue en position 0,0 tournée vers le haut avec le stylo baissé.

```
?fd 50 rt 90
?cs
```

dot

Affiche un point sur l'écran graphique à la position spécifiée et selon la couleur du stylo en cours.

```
?dot [50 10]
```

dotc

Retourne le numéro de la couleur du point qui se trouve aux cooordonnées spécifiées, donne -1 si le point n'existe pas.

```
?cs
?setpc 2
?dot -70,70
?setpc 2
?dot 70 70
?setpc 3
?dot 70 -70
?dotc 70 70
?dotc -70 -70
0
?dotc 1500 2500
```

fence

Permet de définir les limites de l'écran pour la tortue, l'empéchant d'en sortir.

```
?fence
?fd 1000
Turtle out of bounds (la tortue est hors des limites)
```

fill

Peint une zone dans la couleur du stylo en cours en changeant tous les points contigus horizontaux ou verticaux .

```
?make "x 5
?cs
?st
?pd
?repeat 30 [fd :x rt 90 make "x :x + 5]
?fd 20 rt 90
?fd 10
?pu
?home
?bk 2
?home
?bk 2
?pd
?setpc 2
?fill
```

fs

Permet de réserver la totalité de l'écran au graphique.

```
?fs
```

pal

Retourne les trois nombres représentant les quantités de rouge, vert, bleu qui composent le stylo.

```
?pal 2
[0 2 2]
```

setbg

Permet de changer la couleur du fond de l'écran en fonction de la couleur spécifiée.

```
?setbg 2
```

setpal

Assigne pour le stylo les quantités de rouge, de bleu et de vert. Voir la commande **pal**.

```
?setpal 3 [1 1 1]
?pal 3
[1 1 1]
```

setcrunch

Donne au rapport de l'écran graphique la valeur du nombre spécifié.

```
?sf
[0 ss 5 fence 1]
?to carre
>repeat 4 [fd 50 rt 90]
>end
carre defined
?setscrunch 2
?fs
[0 ss 5 fence 2]
?carre
?setsrunch 2.5
?carre
```

sf

Permet d'avoir des informations sur l'état de l'écran graphique sous la forme:
[<couleur du fond> <état de l'écran> <longueur de l'écran texte> <fenêtre> <scrunch>

Où :

Couleur du fond:
Est la couleur du stylo de fond, cette couleur est toujours zéro sous CP/M 2.2

Etat de l'écran:
ss: signifie que l'écran est partagé.
fs: indique que l'écran est entièrement réservé au graphique.
ts: l'écran est totalement réservé au texte.

Longueur de l'écran texte:
Donne le nombre de lignes qui sont réservées au texte sur l'écran graphique.

Fenêtre:
window: indique qu'il n'y a pas de limite, la tortue peut sortir.
wrap: si la tortue sort de l'écran, elle apparaît de l'autre côté.
fence: une barrière est mise, la sortie ne peut sortir.

Srunch:
Le rapport d'écran (non disponible sur CP/M 2.2), est toujours égal à 1 et peut être modifié par **setsrcunch**.

```
?sf
[0 ss fence 1]
```

ss

Permet de réserver une fenêtre de texte sur l'écran graphique.

```
?ss
```

wrap

Fait apparaître la tortue du côté opposé où elle sort.

```
?cs wrap
?rt 5 fd 1000
?cs window
?rt 5 fd 1000
```

PROCEDURES POUR GERER LE GRAPHISME DE LA TORTUE:

bk

Fait reculer la tortue du nombre d'unités indiqué.

```
?cs fd 100
?bk 50
```

fd

Avance la tortue dans le sens de la flèche, du nombre indiqué de pas.

home

Fait retourner la tortue à sa position initiale au point 0,0, tournée vers le haut.

```
?fd 50
?rt 90
?fd 50
?home
```

ht

Permet de rendre la tortue invisible, de ce fait clarifie et accélère le dessin.

```
?ht
?cs fd 70
?st
```

lf

Fait tourner la tortue sur la gauche du nombre de degrés indiqués.

```
?cs fd 50
    ?lf 10
    ?fd 40
```

pd

Permet de poser le stylo après qu'il ait été levé par la procédure **pu.**

```
    ?fd 30 pu 30
    ?pd
    ?fd 30
```

pe

Cette instruction rend la couleur du stylo à 0, permettant ainsi d'effacer les traits sur l'écran.

```
    ?fd 30
    ?pe
    ?bk 15
    ?fd 50
    ?pd fd 25
```

pu

Lève le stylo, permettant ainsi de ne plus laisser de trace sur l'écran.

px

Permet de changer la couleur de tout ce qui a été tracé auparavant dans la couleur opposée ou logiquement complémentaire.

```
?fd 20 pu fd 20
?pd stepc 3 fd 20
?px
?bk 80
?fd 80
?pd bk 100
```

rt

Fait tourner la tortue vers la droite du nombre de degrés spécifiés.

```
?cs fd 50 rt 10 fd 20
```

seth

Permet de faire tourner la tortue dans la direction spécifiée en degrés. Si le nombre est positif, la tortue tourne dans le sens des aiguilles d'une montre, sinon elle tourne dans le sens inverse.

```
?seth 45
```

setpc

Le stylo prend la couleur indiqué par le nombre donné, 0 est la couleur du fond.

```
?setpc 2
```

setpos

Permet de placer la tortue à la position indiquée par les coordonnées.

```
?setpos [10 10]
```

setx

Fait varier la tortue sur l'axe horizontale en fonction de la coordonnée x spécifiée.

```
?setx 30
?fd 50
?setx -30
?fd 50
```

sety

Fait varier la tortue sur l'axe vertical en fonction de la coordonnée y spécifiée.

```
?sety 40
?fd 20
?sety -20
?fd 30
```

st

Rend à nouveau visible la tortue.

```
?ht
?fd 30
?st
```

tf

Permet d'avoir un certain nombre d'informations concernant la tortue.
[<coordx> <coordy> <direction> <état du stylo> <couleur du stylo> <visibilité>

Où:
coordx: est la coordonnée x de la tortue.

coordy: est la coordonnée y de la tortue.

direction: indique la direction degrés vers laquelle se tourne la tortue.

Etat du stylo:
pd: stylo baissé.
pe: stylo qui efface.
px: stylo inversé.
pu: stylo levé.

Couleur du stylo: donne le numéro de la couleur du stylo n.

Visibilité: TRUE (vrai) si le stylo est visible , FALSE (faux) dans l'autre cas.

```
ex: ?setpos [10 20]
    ?rt 30
    ?setpc 3
    ?pe
    ?ht
    ?pf
    [10 20 30 pe 3 false]
```

towards:

Donne une direction qui pointe la tortue vers les coodonnées spécifiées.

```
?seth towards list x y
```

PROCEDURES GERANT L'ESPACE DE TRAVAIL :

er

Efface la procédure spécifiée.

```
?er "dessin
```

erall

Efface toutes les procédures et les variables de l'espace de travail.

```
?erall
```

ern

Permet d'effacer la ou les variables définies.

```
?make "rayon [10]
?make "longeur [50]
?:rayon ?:longeur
[10]
[50]
?ern [rayon longeur]
?:rayon
rayon has no value (rayon n'a plus de valeur)
```

nodes

Donne le nombre de noeuds disponibles dans l'espace de travail.

```
?nodes
```

noformat

Permet de supprimer dans l'espace de travail le formatage de la procédure et les commentaires afin de libérer des noeuds.

```
?noformat
```

poall

Affiche les définitions de toutes les procédures et variables de l'espace de travail.

pons

Affiche les noms et les valeurs de toutes les variables globales de l'espace de travail.

```
?pons
longeur is 50
cote is 10
angle is 45
```

pops

Affiche les noms et les définitions de toutes les procédures de l'espace de travail.

```
?pops
```

recycle

Permet de libérer le plus de noeuds possibles et réorganise l'espace de travail.

```
?recycle
?nodes
```

LISTES DE PROPRIETE :

glist

Donne une liste de tous les objets de l'espace de travail auxquels on a attaché une des propriétes.

```
?glist "def
```

gprop

Permet d'obtenir la valeur de la propriété du nom désigné.

```
?make "cote 40
?grop "cote ".apv
40
```

plist

Donne la liste des propriétés attachées à un mot.

```
?plist "cote
?[.apv 40]
```

pprop

Permet de créer un lien entre un mot et une propriété.

```
?pprop "cercle ".apv "rayon
?.cercle
rayon
```

pps

Permet d'afficher toutes les paires de propriétés non standard de tous les objets de l'espace de travail.

```
?pprop "marignan "date 1515
?pps
marignan date is 1515
?plist "marignan
[date 1515]
```

remprop

Permet d'éliminer la propriété spécifiée de la liste de propriété du mot.

```
?remprop "cercle ".apv
```

UTILISATION DES DISQUETTES :

changdef

Permet de renommer un fichier dans le catalogue de la disquette.

```
?dir
[cercle dessin]
?chandef "carre "dessin
?dir
[cercle carre]
```

defaultd

Affiche le numéro de l'unité de disquette par défaut en cours.

```
?defaultd
?A:
```

dir

Permet d'afficher la liste complète de tous les fichiers présents sur la disquette. L'emploi du jocker ? est autorisé comme pour le basic. DR LOGO ne tient pas compte du jocker *.

```
?dir "a:?????
```

dirpic

Donne la liste des fichiers graphiques de l'unité par défaut ou spécifiée.

```
?dirpic "b:
[dessin etoiles figures]
```

load

Permet de charger le fichier spécifié dans l'espace de travail de l'ordinateur.

```
?load "dessin
?load "b:dessin
```

loapic

Charge sur l'écran graphique le dessin sauvegardé dans le fichier graphique spécifié.

```
?loadpic "cercle
?loadpic "b:cercle
```

save

Permet d'écrire le contenu de l'espace travail dans un fichier sur la disquette.

```
?save "dessin
```

savepic

Permet d'écrire le contenu de l'écran graphique dans un fichier graphique sur la disquette.

```
?savepic "cercle
?savepic "b:cercle
```

Remarque sur l'utilisation des disquettes.

Il est préférable pour un maximun de sécurité de ne pas sauvegarder sur les disquettes systémes.

Si vous utilisez la version CP/M 2.2 du DR LOGO, vous devez veiller à ce qu'il y ait assez de place sur la disquette travail car vous ne pourrez plus changer de disquette en cours de procédure.

setd

Permet de définir l'unité spécifiée comme unité de disque par défaut.

```
?defaultd
?a:
?setd b:
?dir
[triangle boite]
```

GERER LE CLAVIER ET LES MANETTES DE JEU:

buttonp

Retourne TRUE (vrai) si le bouton de la manette de jeu est appuyé. Les deux manettes sont différenciées par les numéros 0 et 1.

```
?to pan
>label "boucle
>if (buttonp 0) [pr [pr [pan 0]]
>if (buttonp 1) [pr [pr [pan 1]]
>go "boucle
>end
```

Voir la commande **paddle** pour tester la position des manettes.

keyp

Donne TRUE (vrai) si un caractère vient d'être tapé au clavier.

```
?to testtouch
>if keyp [op rc ] [op "]
>end
```

paddle

Permet de savoir dans quelle position se trouvent les manettes 0 ou 1 par différents codes:

255: rien
0: en haut
1: en haut à droite
2: a droite
3: en bas et à droite
4: en bas
5: en bas et à gauche
6: a gauche
7: en haut et à gauche.

```
?paddle 0
255
```

Voir la commande **buttonp** pour tester les boutons de tir.

rc

Affiche le premier caractère tapé au clavier.

```
?make "touche rc
appuyez sur la touche A:
?:touche
a
```

rl

Donne une liste contenant une entrée au clavier qui doit être suivie d'un retour chariot.

```
?make "lisr rl
repeat 3 [fd 40 rt 120]
?:list
repeat 3 [fd 40 rt 120]
```

rq

Donne un mot contenant une ligne entrée au clavier, qui doit être suivi d'un retour chariot.

```
?make "commande rq
repeat 4 [fd 40 rt 90]
?:commande
repeat 4 [fd 40 rt 90]
```

GESTION DU SON:

Seule le version du LOGO sur les ordinateurs AMSTRAD posséde des commandes pour la gestion du son. Elles sont identiques à leurs équivalents en BASIC, pour plus d'informations concernant en particulier les paramètres, nous vous conseillons de vous reporter à la section BASIC de ce manuel.

sound

Permet de stocker un son dans une queue. Dans le format suivant: [<état de canal> <période sonore> <durée> <volume> <enveloppe de volume> <enveloppe de tonalité> <bruit>.

Tous le paramètres après la durée peuvent être omis.

```
? sound [1 10 25]
```

env

Permet de construire l'enveloppe de volume, la forme de la note. Avec comme format: [<numéro de l'enveloppe> <section de l'enveloppe>]

```
?env [1 50 1 10]
?sound [1 100 150 5 1]
```

ent

Construit l'enveloppe de tonalité avec comme format: [<numéro de l'enveloppe > <section de l'enveloppe>]

```
?ent [1 50 1 10]
?sound [1 100 150 5 11]
```

release

Permet de libérer les canaux suspendus par la commande sound. Les canaux à libérer sont indiqués de la façon suivante:

0: Aucun
1: A
2: B
3: A et B
4: C
5: A et C
6: B et C
7: A et B et C.

?release 7

POUR CONTROLER LE DEROULEMENT DU PROGRAMME:

bye

Permet de sortir du DR. LOGO

```
?bye
```

co

Pour continuer le cours de la procédure après un arrêt provoqué par CTRL Z, pause ou erract.

```
?co
```

go

Fait exécuter la ligne de commande ayant l'étiquette spécifiée par la commande **go**.

```
?go "test
```

if

Permet de faire exécuter une ou deux listes d'instructions suivant la valeur de l'expression rentrée, elles doivent impérativement être entourées de crochets.

```
?if (:x > :Y) [pr [x plus grand que y]]
?if (:x < :Y) [pr [y plus grand que x]]
```

label

Pose une étiquette sur une ligne de la procédure qui pourra être appelée à l'aide de la commande go.

```
?label "test
```

op

Fait afficher l'objet entré et interrompt le cours de la procédure.

```
?op [total]
```

repeat

Permet de répéter la liste d'instructions autant de fois que le nombre l'indique.

```
?repeat 3 [fd 40 rt 120]
```

run

Fait exécuter la ligne d'instructions

```
?make "instr_list repeat 3 [fd 40 rt 120]
?run [instr_list
```

stop

Interrompt l'exécution de la procédure en cours et fait revenir l'ordinateur au niveau supérieur (symbole ?) ou à la procédure d'appel.

```
?stop
```

wait

Permet d'arrêter le cours de le procédure pendant le temps spécifié par le nombdre donné. Pour calculer le temps de pause il faut appliquer la formule suivante:

longueur du temps= nombre entré * 1/60 seconde.

```
?wait 20
```

AIDES A LA PROGRAMMATION :

catch

Permet de traiter une erreur ou un cas particulier survenu pendant l'exécution d'une série d'instructions.

```
?catch "error [+ [] []]
?pr [coucou je suis la]
coucou je suis la
```

error

Permet d'avoir la liste des instructions ayant causé la dernière erreur.

```
?catch "error
?show error
```

notrace

Permet de désactiver la commande **trace**.

```
?notrace
```

nowatch

Permet de désactiver la commande **watch**.

```
?nowatch
```

pause

Suspend l'exécution de la procédure en cours, permettant ainsi de visualiser une ou plusieurs variables.

```
?if :test <5 [pause ]
```

throw

Permet de faire exécuter la ligne d'instruction identifiée par le nom entré sous **catch**.

```
?throw "test
```

trace

Fait afficher le nom de chaque procédure en cours d'exécution.

```
?trace
```

watch

Fait afficher le nom de chaque expression en cours d'exécution.

```
?watch
```

PRIMITIVES SYSTEMES :

.contents

Affiche le contenu de l'emplacement des symboles du DR LOGO.

.deposit

Place le deuxième nombre entré à l'emplacement mémoire donné par le premier nombre.

.examine

Fait afficher le contenu de l'emplacement mémoire spécifié.

.in

Recherche la valeur en cours de l'entrée spécifiée.

.out

Envoie la valeur introduite vers l'entrée spécifiée.

VARIABLES SYSTEME :

erract

Renvoie au niveau supèrieur TOPLEVEL, lorsque TRUE (vrai) provoque une pause au moment d'une erreur.

false

Valeur système.

redefp

Quand TRUE (vrai) permet la redéfinition des primitives.

toplevel

throw "**toplevel** clos toutes les procédures en attente.

true

Valeur système.

Propriétés du système :

.apv

Valeur de la variable associée à la propriété, valeur de la variable globale.

.def

Définition d'une procédure.

.enl

Fin d'une ligne de procédure interrompue par un retour chariot ou des espaces.

.emt

Début d'une ligne de procédure interrompue par un retour chariot ou des espaces.

.prm

Identifie une primitive.

.rem ou ;

Remarques ou commentaires.

BANK MANAGER_____

1) EXAMEN DE LA MEMOIRE DU 6128.

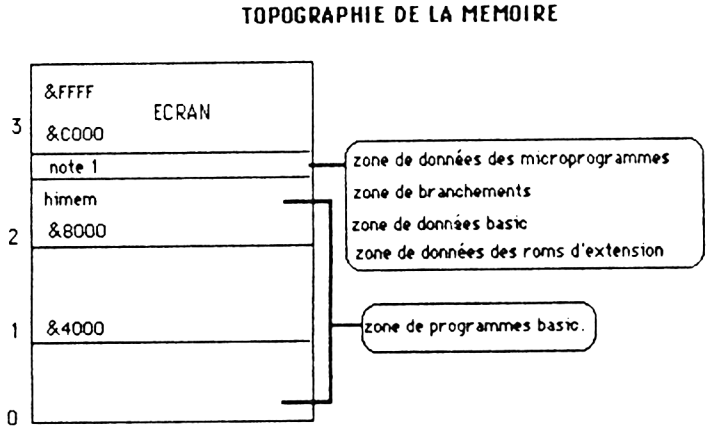


Fig 1.1 schéma de la mémoire sous le basic 1.1

Le 6128 possède :

- une mémoire RAM de 128 Ko
- et une ROM de 48 Ko.

Les premiers 64 Ko de la RAM sont divisés en quatre blocs de 16 KO chacun, numérotés de 0 à 3.

La figure 1.1 représente ces 64 Ko, nous voyons que :

- l'écran occupe le bloc 3,
- que la partie supérieure du bloc 2 est utilisée par les variables système.

Vous remarquerez que sur les 128 Ko de RAM, 64 Ko restent inutilisés.
En fait, l'ordinateur n'utilise au plus que 112 Ko de la mémoire:
64 Ko de RAM + 48 Ko de ROM.

Vous savez sans doute que le "cerveau" d'un ordinateur est son microprocesseur.

Pour le 6128 celui-ci est un Z80, qui est un micropocesseur 8 bits.

Le Z80 ne peut gérer que 64 ko à la fois, aussi le système d'exploitation contient-il des instructions qui sollicitent la ROM des microprogrammes, plutôt que le bloc 0 de la RAM, et la ROM du basic ou de la disquette, plutôt que le bloc 3.

Ce changement de bloc intervient automatiquement lorsque le basic ou les microprogrammes sont requis.

Ce procédé est également appliqué à la RAM pour favoriser le recouvrement de la RAM plutôt que celui de la ROM. Le changement est assuré par un programme rédigé en assembleur.

Chaque lot de 16 Ko s'appelle un 'bloc', quatre blocs (64 Ko) forment un banc. La technique de sélection de bloc se nomme donc 'changement de banc'.

A l'aide du programme BANKMAN.BAS résidant sur la face 1 des disquettes sytèmes, nous pouvons avoir accès au 64 Ko de mémoire supplémentaire. Ce qui peut servir d'espace provisoire pour le stockage d'écran graphique.

EX: un jeu vidéo contenant plusieurs écrans pourra être accéléré en stockant à l'avance les dessins déjà prêts.

Cette nouvelle partie de la mémoire peut également servir à étendre l'espace de travail pour les variables, ce qui permet d'avoir une extension de l'espace réservé aux chaînes ou encore tout simplement comme disque virtuel (en RAM).

PREMIERE UTILISATION:

STOCKAGE DES ECRANS.

BANK MANAGER permet de mettre le bloc 1 de côté et de sélectionner à sa place l'un des quatre autres blocs.

Le contenu du bloc 1 est préservé et restitué lorsque BANK MANAGER a terminé.

Deux seules commandes gérent le déplacement d'un bloc à l'autre:

|screenswap

Intervertit le contenu des deux blocs.

|screenswap ,[<zone écran>,]<numéro d'écran>,<numéro d'écran>

|screencopy

Copie le contenu d'un bloc dans un autre bloc.

|screencopy , [<zone écran>,]<numéro d'écran cible>,<numéro d'écran d'écran source>

<zone écran>

Le premier paramètre <zone écran> peut être omis, il provoque la copie d'un 1/64 du bloc soit 256 octets.

Il peut varier de 0 à 63. Ce mode permet d'effectuer une autre tâche lors de l'opération de copie. L'échange total du contenu de l'écran dure 150 unités TIME, soit environ 150/300 secondes.

<numéro d'écran cible>

Le premier <numéro d'écran> doit être 1 (écran standard), le second peut prendre les valeurs 2 3 4 ou 5.

Il est à signaler que les copies faisant intervenir l'écran 1 se déroulent beaucoup plus rapidement que les autres.

Il vous faut également veiller au mode de l'écran sous lequel vous travaillez, concrétement vous ne pouvez pas changer un écran de 80 colonnes par un écran de 40 ou 20 colonnes.

MISE EN PRATIQUE:

Pour commencer, vous devez installer le controleur de blocs, autrement dit BANK MANAGER.

Vous devez insérer la face 1 de la disquette système et tapez au clavier :

```
run "bankman"
```

Tapez ensuite:

```
mode 1
```

Vous tapez alors:

```
' voici l'ecran original
|screencopy ,3,1 'stocke l'écran en no 3
cls
```

L'écran s'efface de nouveau.

Vous pouvez tapez:

```
'ceci est l'écran numéro 2
|screencopy ,2,1  'envoi l'écran en mémoire 2
|screenswap ,2,3 'echange l'écran 2 et 3
|screencopy ,1,3  'affiche le contenu de l'écran 3
|screencopy ,1,2  'affiche le contenu de l'écran 2
```

Vous voyez : la manipulation est très simple et peut rendre de grands services.

DEUXIEME UTILISATION:

LES FICHIERS VIRTUELS :

Les derniers 64 Ko peuvent être utilisés comme fichier en mémoire vive (RAM). ILs ne peuvent contenir que des données, rien n'étant prévu pour qu'ils contiennent des lignes de programme directement exécutables.

Le fichier sera constitué d'un ou de plusieurs enregistrements de longueur fixe comprise entre 0 et 255 octets. Deux octets étant le minimum.

Une fois la longueur fixée, vous pouvez accéder à chaque enregistrement grâce à son numéro.

Il vous est permis d'écrire une donnée avec une longueur définie et de la relire avec une longueur différente.

Identique au fichier à accès sélectif, le fichier virtuel reprend la notion de numéro d'enregistrement courant, permettant ainsi l'utilisation d'un numéro par défaut, très utile pour les recherches à l'intérieur du fichier.

MISE EN PRATIQUE :

|bankopen

Permet de fixer la longueur de tous les enregistrements,donne la valeur zéro pour l'enregistrement courant. Cette commánde n'affecte aucunement le contenu de la mémoire.

|bankopen ,<longueur d'enregistrement virtuel>

|bankwrite

Ecrit la chaîne alphanumérique à l'intérieur du fichier virtuel.

|bankwrite ,@ <code retour>,<chaîne alphanumérique> [,<numéro d'enregistrement virtuel>]

OU:

<numéro d'enregistrement virtuel>

Permet de définir le numéro de l'enregistrement en cours d'écriture. Si ce paramètre est omis le numéro courant est appliqué. Le pointeur d'enregistrement passe alors à l'enregistrement suivant.

<chaîne alphanumérique>

Contient l'enregistrement, si cette chaîne est inférieure à la longueur standard, les anciens caractères restent à la fin de l'enregistrement. Par contre si la chaîne est plus longue, elle est tronquée évitant ainsi de déborder sur l'enregistrement suivant.

<code retour>

Permet de vérifier si l'écriture s'est bien effectuée, retourne un nombre correspondant au numéro de l'enregistrement qui vient d'être écrit. Si une erreur a été détectée le nombre est négatif.

-1 : indique que l'adresse du numéro d'enregistrement dépasse 64 Ko.

-2 : signifie qu'il y a eu une erreur lors d'un changement de bloc, ne devrait jamais se produire.

Ex: |bankopen,7
|bankwrite ,@r%,"test",0
|bankwrite ,@r%,a$

|bankread

Permet de lire un enregistrement du fichier virtuel et le copie dans la chaîne alphanumérique.

|
bankread ,@ <code retour>,@<chaîne alphanumérique> [,<numéro d'enregistrement virtuel>]

OU:

<numéro d'enregistrement virtuel>

Permet de définir le numéro de l'enregistrement en cours d'écriture. Si ce paramètre est omis le numéro courant est appliqué. Le pointeur d'enregistrement passe alors à l'enregistrement suivant.

<chaîne alphanumérique>

Va recevoir l'enregistrement lu. Si cet enregistrement ne remplit pas entièrement la chaîne, les anciens caractères restent à la fin. Si l'enregistrement dépasse le contenu de la chaîne, les caractères exédents sont supprimés car la longueur d'une chaîne de caractères ne peut être augmentée lors d'une commande externe.

<code retour>

Permet de vérifier si la lecture s'est bien effectuée, retourne un nombre correspondant au numéro de l'enregistrement qui vient d'être lu. Si une erreur a été détectée le nombre est négatif.

-1 : indique que l'adresse du numéro d'enregistrement dépasse 64 Ko.

-2 : signifie qu'il y a eu une erreur lors d'un changement de bloc, ne devrait jamais se produire.

Ex: |bankread,@r%,b$,0

|bankfind

Passe tous les enregistrements du fichier virtuel en revue.

|bankfind ,@,<code retour>,<chaîne recherchée> [,<numéro d'enregistrement de départ>[,<numéro d'enregistrement de fin>]]

<numéro d'enregistrement de départ>

Le <numéro d'enregistrement de départ> permet de définir le début de la recherche, s'il est omis la recherche commence au numéro courant.

<numéro d'enregistrement de fin>

Si le <numéro d'enregistrement de fin> est indiqué, la recherche s'arrête après examen de cet enregistrement, à moins bien sûr que la chaîne n'ait été trouvée avant d'atteindre cet enregistrement.

Si la recherche aboutit, le pointeur prend la valeur du numéro d'enregistrement contenant la chaîne recherchée, sinon il ne change pas.

<code retour>

Permet de vérifier si la recherhe s'est bien effectuée, retourne un nombre correspondant au numéro d'enregistrement de la chaîne recherchée. Si une erreur a été détectée le nombre est négatif.

-1 : indique que l'adresse du numéro d'enregistrement dépasse 64 Ko.

- 2 : signifie qu'il y a eu une erreur lors d'un changement de bloc, ne devrait jamais se produire.

-3 : La chaîne recherchée n'a pas été retrouvée.

Une facilité pour bien utiliser cette commande, la chaîne recherchée peut contenir des jockers indiqués par des caractères nuls: CHR$(0),et la comparaison s'effectue par rapport à la <longueur d'enregistrement virtuel> ou à la longueur de la <chaîne recherchée>, si celle-ci est plus courte.

Ex: |bankfind, @r%,"test",0
|bankfind ,@r%,f$,10,20

Nous avons vu dans le chapitre BASIC toute une panoplie de commandes de son et d'enveloppe. Ces commandes vous ont sans doute paru assez compliquées, c'est pourquoi nous allons y revenir plus en détails tout au long de ce chapitre.

Nous allons commencer par la commande SOUND.
Vous vous demandez peut être quelles valeurs peuvent prendre ses quatre premiers paramètres

Numéro de canal.
Période sonore.
Durée de la note.
Volume.

La période sonore peut prendre les valeurs de 0 à 4095, bien évidemment, seules quelques-unes de ces valeurs correspondent a des notes de musique. Le nombre 239 correspond ainsi à la note DO médium. Aucune note n'est jouée lorsque la période est fixée à zéro.

Avec le troisième paramètre, il est possible de fixer la durée de la note en centièmes de seconde. Les valeurs correctes se situent entre 1 et 32767 (inclus). Pour une valeur nulle (zéro) la durée de la note est déterminée par l'enveloppe.
Une valeur négative indique qu'il faut répeter l'enveloppe le nombre de fois spécifié.
Ainsi -2 permet de répéter l'enveloppe de volume 2 fois.

Le volume est le quatrième paramètre. Il a pour valeur par defaut 12 mais il peut prendre toutes les valeurs de 0 à 15. Jusqu'à présent nous n'avons entendu que des sons où le volume était constant. Lorsque la note est modulée par une enveloppe de volume, la valeur du volume de la commande SOUND ne concerne que le départ de la note.

Nous avons volontairement laissé le plus compliqué pour la fin. Le premier paramètre est le numero de canal. Il correspond a un nombre binaire et nous vous conseillons, si ce n'est déja fait, de lire avant de continuer le chapitre APPROFONDISSONS.

Sur le 6128, vous avez le choix entre trois canaux pour jouer un son. Si vous couplez votre ordinateur sur votre chaîne stéréophonique, un canal va sortir sur l'enceinte gauche, l'autre sur la droite et le troisième sur les deux à la fois. Chacun des trois canaux porte un numéro:

1 canal A
2 canal B
4 canal C

Pour jouer sur plusieurs canaux, il suffit d'additionner les numéros de canaux désirés. Par exemple pour jouer sur le A et le B

```
1+2=3
SOUND 3,239
```

Vous vous demandez pourquoi le canal c porte le numéro 4 au lieu de porter tout simplement le 3. En fait il s'agit tout simplement de l'utilisation des puissances de 2.

```
1=2'0
2=2'1
4=2'2
```

Dans un nombre binaire de trois chiffres, chacun des chiffres peut prendre les valeurs 0 ou 1 déterminant ainsi l'état en service ou hors service du canal correspondant.

Dans notre exemple 3 équivaut à 0*4+1*2+1*1 donc 11 en notation binaire. En attribuant chaque canal à une colonne binaire, nous obtenons :

```
0 1 1
C B A
```

Nous voyons bien que le canal C est hors service, les canaux B et A sont eux en service.

Si nous voulons jouer une note sur les canaux C et A

```
4+1=5
5=101 en binaire
1 0 1
C B A
```

Seulement 1 et 2 et 4 ne sont pas les seuls paramètres de numéro de canal. Les valeurs 8, 16 et 32 peuvent être également utilisées, afin d'indiquer qu'un son produit sur un canal a rendez-vous avec un autre canal (A, B et C respectivement).

Le rendez-vous permet de synchroniser les trois canaux sonores et donc de faire exécuter une mélodie avec des pauses sans que cela occasionne un craquement désagréable du haut-parleur.

Voici comment se présente un rendez-vous.

```
          1  2  3  4  5
canal A : no no no rb rb
canal b: no rc no no ra
canql c: no rb ra no no
```

No signifie qu'une note est jouée.
Rb correspond à un rendez-vous avec le canal B
Ra correspond à un rendez-vous avec le canal A
Rc correspond à un rendez-vous avec le canal C.

Une autre possibilité de synchroniser les canaux consiste à mettre un arrêt. Un arrêt permet d'arrêter l'émission de son d'un canal déjà plein puis de le relancer avec l'instruction RELEASE.

Un avantage par rapport au rendez-vous est que le canal sonore peut continuer à être rempli pendant un arrêt contrairement au rendez-vous qui n'admet aucun son avant qu'il ne soit executé.

LES COURBES D'ENVELOPPE DU VOLUME :

L'instruction **ENV** modifie le volume d'un son de façon à imiter le jeu des instruments de musique.

Le principe d'une note est le suivant, il y a d'abord une attaque pendant laquelle le volume augmente très rapidement après quoi il descend à un niveau inférieur auquel il se maintient un certain temps avant de s'éteindre progressivement.

En utilisant l'instruction **ENV**, il faut en principe que le volume ait été mis à zéro par l'instruction **SOUND** pour que l'enveloppe fonctionne pleinement

Les courbes d'enveloppe sont affectées à des symboles ou numéros d'enveloppe qui peuvent ensuite être employés dans l'instruction **SOUND** pour appeler une courbe d'enveloppe .

Ce symbole est le premier paramètre de l'instruction **ENV.**

Les trois paramètres suivants peuvent être employés 5 fois les uns à la suite des autres.

Le nombre de pas indique en combien de pas le volume doit augmenter ou diminuer.

Le pas, multiplié par le temps de pause donne la durée du son. Si cette valeur est negative le volume diminue.

Le temps de pause est la durée en centièmes de seconde pendant laquelle doit être maintenu le volume actuel. Il peut également être appelé durée de pas.

LA COURBE DE L'ENVELOPPE DU SON :

L'enveloppe du son est dans son principe analogue à l'enveloppe du volume. Les possibilités ne sont pas aussi grandes que pour l'enveloppe de volume.

Les modifications de l'enveloppe du son permettent notamment de réaliser un vibrato (une modification minime de la hauteur donnant une impression de temblement).

La structure de cette instruction et de ces paramètres est indentique à celle **ENV**. Mais alors qu'**ENV** ne nécessite en principe que trois parties, attaque, soutien et relâche. **ENT** en nécessite le plus souvent 5.

La méthode la plus efficace pour bien comprendre le principe des enveloppes consiste à se livrer au plus grand nombre d'expériences possibles.

Nous allons vous présenter la table complète des périodes sonores pour les notes de la gamme de huit octaves.

La fréquence produite n'est exacte à 100 %. L'erreur relative est le pourcentage de la différence entre la fréquence théorique et la fréquence réelle.

NOTE	FREQUENCE	PERIODE	ERREUR RELATIVE	
C	16.352	3822	-0.007%	
C#	17.324	3608	+0.007%	
D	18.354	3405	-0.007%	
D#	19.445	3214	-0.004%	
E	20.602	3034	+0.009%	
F	21.827	2863	-0.016%	
F#	23.125	2703	+0.009%	OCTAVE -4
G	24.500	2551	-0.002%	
G#	25.957	2408	+0.005%	
A	27.500	2273	+0.012%	
A#	29.135	2145	-0.008%	
B	30.868	2025	+0.011%	

NOTE	FREQUENCE	PERIODE	ERREUR RELATIVE	
C	32.703	1911	-0.007%	
C#	34.648	1804	+0.007%	
D	36.708	1703	-0.022%	
D#	38.891	1607	-0.004%	
E	41.203	1517	+0.009%	
F	43.654	1432	+0.019%	
F#	46.249	1351	-0.028%	OCTAVE -3
G	48.999	1276	-0.037%	
G#	51.913	1204	+0.005%	
A	55.000	1136	+0.032%	
A#	58.270	1073	+0.039%	
B	61.735	1012	+0.038%	

NOTE	FREQUENCE	PERIODE	ERREUR RELATIVE	
C	65.406	956	-0.046%	
C#	69.296	902	+0.007%	
D	73.416	851	-0.037%	
D#	77.782	804	-0.058%	
E	82.407	758	+0.057%	
F	87.307	716	-0.019%	
F#	92.499	676	+0.046%	OCTAVE -2
G	97.999	638	-0.037%	
G#	103.826	602	+0.005%	
A	110.00	568	-0.032%	
A#	116.541	536	-0.055%	
B	123.471	506	-0.038%	

NOTE	FREQUENCE	PERIODE	ERREUR RELATIVE	
C	130.813	478	+0.046%	
C#	138.591	451	+0.007%	
D	146.832	426	+0.080%	
D#	155.564	402	+0.058%	
E	164.814	379	-0.057%	
F	174.614	358	+0.019%	
F#	184.997	338	+0.046%	OCTAVE -1
G	195.998	319	+0.037%	
G#	207.652	301	+0.005%	
A	220.000	284	-0.032%	
A#	233.082	268	-0.055%	
B	246.942	253	-0.038%	

NOTE	FREQUENCE	PERIODE	ERREUR RELATIVE	
C	261.626	239	+0.046%	
C#	277.183	225	-0.215%	
D	293.665	213	+0.081%	
D#	311.127	201	+0.058%	
E	329.628	190	+0.206%	
F	349.228	179	+0.019%	
F#	369.994	169	+0.046%	OCTAVE 0
G	391.995	159	-0.277%	
G#	415.305	150	-0.328%	
A	440.000	142	-0.032%	
A#	466.164	134	-0.055%	
B	493.883	127	+0.356%	

NOTE	FREQUENCE	PERIODE	ERREUR RELATIVE	
C	523.251	119	-0.374%	
C#	554.365	113	+0.229%	
D	587.330	106	-0.390%	
D#	622.254	100	-0.441%	
E	659.255	95	+0.206%	
F	698.457	89	-0.0543%	
F#	739.989	84	-0.548%	OCTAVE 1
G	783.991	81	+0.350%	
G#	830.609	75	-0.328%	
A	880.000	71	-0.032%	
A#	932.328	67	-0.055%	
B	987.767	63	-0.435%	

NOTE	FREQUENCE	PERIODE	ERREUR RELATIVE	
C	1046.502	60	+0.462%	
C#	1108.731	56	-0.662%	
D	1174.659	53	-0.390%	
D#	1244.508	50	-0.441%	
E	1318.510	47	-0.855%	
F	1396.913	45	+0.574%	
F#	1479.978	42	-0.548%	OCTAVE 2
G	1567.982	40	+0.350%	
G#	1661.219	38	+0.992%	
A	1760.000	36	+1.357%	
A#	1864.655	34	+1.417%	
B	1975.533	32	+1.134%	

NOTE	FREQUENCE	PERIODE	ERREUR RELATIVE	
C	2093.004	30	+0.462%	
C#	2217.461	28	-0.662%	
D	2349.318	27	+1.469%	
D#	2489.016	25	-0.441%	
E	2637.021	24	+1.246%	
F	2793.826	22	-1.685%	
F#	2959.955	21	-0.548%	OCTAVE 3
G	3135.963	20	+0.350%	
G#	3322.438	19	+0.992%	
A	3520.000	18	+1.357%	
A#	3729.310	17	+1.417%	
B	3951.066	16	+1.134%	

TROISIEME PARTIE :

Glossaire

Ce **glossaire** donne une brève définition des termes informatiques les plus souvent utilisés dans cette documentation.

ACCES DIRECT : Mode d'accès permettant de lire ou d'écrire des informations dans la mémoire ou dans la disquette dans n'importe quel ordre.

ADRESSE : L'étiquette ou le numéro identifiant le registre ou la case mémoire, où une unité d'information est stockée.

ADRESSE MEMOIRE : Une adresse mémoire spécifique dans l'ordinateur il y a 65536 adresses mémoire (0-65535) sur l'AMSTRAD 6128.

ALPHANUMERIQUE : Des lettres, nombres et symboles spéciaux trouvés sur le clavier, non compris les caractères graphiques.

ALU : (Arithmetic Logic Unit, Unité arithmétique et logique, UAL).
La partie du CPU où les données binaires sont traitées.

AMSDOS : (Amstrad Disk Operating System)
Système d'exploitation de l'AMSTRAD gérant les accès disques.

ANIMATION : L'utilisation des instructions de l'ordinateur pour simuler le mouvement d'un objet sur l'écran par des mouvements graduels et progressifs.

APPEL SELECTIF : (Polling)
Une méthode de contrôle des communications utilisée par des systèmes informatique/terminaux par laquelle un poste 'principal' demande aux périphériques connectés, l'un après l'autre s'ils ont des informations à envoyer.

ASCII : (American Standard Code for Information Interchange)
Un code de sept bits utilisé pour représenter des caractères alphanumériques. C'est utile pour envoyer des informations du clavier à l'ordinateur et d'un ordinateur à un autre.
Voir le code chaîne de caractères.

ASSEMBLEUR : Un programme qui traduit des instructions en assembleur en des instructions en langage machine.

ATTAQUE : La vitesse à laquelle le volume d'une note de musique s'élève de zéro à son volume maximum.

BACKUP : Une copie de disquette assurant une sauvegarde, au cas où l'original serait perdu ou endommagé.

BASE DE DONNEES : Une grande quantité de données stockées d'une manière bien organisée.
Un système de gestion de base de données est un programme qui permet d'accéder à ces informations.

BASIC : (Beginner's All-purpose Symbolic Instruction Code)
Langage de programmation évolué puissant et facile à apprendre.

BAUD : Vitesse de transmission des données en série .A l'origine un terme de télégraphe; une vitesse de 300 bauds est égale approximativement à une vitesse de transmission de 30 octets ou caractères par secondes.

BINAIRE : Un système de numérotation en base 2. Tous les nombres sont représentés par une suite de zéro ou de 1.

BIT: (Binary digIT)
La plus petite unité dans un ordinateur. Chaque chiffre du système binaire peut avoir une des deux valeurs 0 ou 1. Un bit est en fonction (on) s'il est égal à 1, un bit est hors-fonction (off) s'il est égal à 0.

BIT DE DEPART : Un bit ou groupe de bits qui indentifie le début d'un mot de données.

BIT DE FIN : Un bit ou groupe de bits qui indentifie la fin d'un mot de données et définit l'espace entre les mots de données.

BIT DE PARITE : Un 1 ou un 0 ajouté à un groupe de bits qui indentifie les bits comme étant pairs ou impairs.

BOUCLE : Une partie de programme exécutée d'une manière répétitive un nombre déterminé de fois.

BOUCLE DELAI : Un boucle FOR ... NEXT vide qui ralentit l'exécution d'un programme.

BRANCHEMENT : Pour sauter vers une section de programme et l'exécuter. GOTO et GOSUB sont des exemples d'instructions de branchement de BASIC.

BUFFER : Une partie de la mémoire réservée pour un stockage temporel, généralement utilisé pour un transfert d'informations (par exemple : transfert du clavier à l'unité centrale, ou de l'unité centrale vers l'imprimante ...)

BUS : Des lignes parallèles ou en série pour transférer des données entre les périphériques. Les ordinateurs sont souvent décrits par la structure de leurs bus (par exemple, ordinateur à bus S-100, etc ....)

CABLE COAXIAL : Un moyen de communication, employé d'habitude dans les réseaux locaux.

CABLE PLAT : Un groupe de fils attachés en parallèle.

CARACTERE : Tout symbole sur le clavier de l'ordinateur qui est affiché à l'écran.
Les caractères comprennent des nombres, lettres, symboles de ponctuation et graphiques.

CARACTERES GRAPHIQUES : Des caractères non-alphanumériques sur le clavier de l'ordinateur.

CARTE MERE : (Motherboard)
Dans un système orienté bus, la carte qui contient les lignes du bus et les connecteurs latéraux pour prendre en charge les autres cartes du système.

CHAINE DE CARACTERES : Un caractère ou suite de caractères alphanumériques entre guillemets.

CHAINE DE CARACTERES NULLE : Un caractère vide (""). Un caractère auquel on n'a pas encore affecté un code de chaîne de caractères.

CLAVIER : Composant du processus d'entrée d'un système informatique.

CODE DE CHAINE DE CARACTERES : La valeur numérique affectée pour représenter un caractère de L'AMSTRAD 6128 en mémoire.

CODE ECRAN : Le nombre affecté pour représenter un caractère dans la mémoire écran. Lorsque vous appuyez sur une touche du clavier, le code écran pour ce caractère est transmis dans la mémoire écran automatiquement. Vous pouvez aussi afficher un caractère en stockant directement dans la mémoire écran avec la commande POKE.

CODE SOURCE : Un programme non exécutable écrit dans un langage évolué.
Un compilateur ou un assembleur doit transformer le code source en code objet (langage machine) que l'ordinateur peut comprendre.

COMMANDE : Une instruction BASIC utilisée en mode direct pour exécuter une action.
Voir mode direct.

COMPILATEUR : Un programme qui traduit un langage évolué tel que le PASCAL en un programme en langage machine.

COMPTEUR : Une variable utilisée pour compter le nombre de fois qu'un événement se produit dans un programme.

CONDITION : Expressions entre les mots IF et THEN , évaluées comme vraies ou fausses dans une intruction IF ... THEN L'instruction de condition IF... THEN donne à l'ordinateur la possibilité de faire un choix.

CONTROLE BINAIRE: Transmission des données en série dans laquelle bit est significatif. Un caractère seul est entouré de bits de départ et d'arrêt.

CONTROLE OU DETECTIONS D'ERREURS : Routines de logiciels qui identifient et corrigent souvent, des données erronées.

COORDONNEES : Un point de la grille ayant des valeurs verticales (Y) et horizontales (X).

COUCHE DE TRANSMISSION DE DONNEES : Une portion logique du contrôle des communications de données qui garantit qu'une communication entre des périphériques contigus soit sans erreur.

COULEUR DU FOND : La couleur de la zone de l'écran sue laquelle les caractères sont placés.

COULEUR DU BORD : La couleur des bords tout autour de l'écran.

COUPLEUR OU MODEM ACOUSTIQUE : Un appareil qui convertit les signaux digitaux en sons audibles pour la transmission par les ligne du téléphone. La vitesse est limitée à 1200 bauds ou bits par seconde (bps). Comparez avec modem à connection directe.

COURRIER ELECTRONIQUE : (E-Mail)
Un service de communications pour les utilisateurs d'ordinateurs où des messages avec texte sont envoyés à l'ordinateur central ou à une 'boîte aux lettres' électronique et sont récupérés plus tard par le destinataire.

CP/M : (Control Program for Microcomputers)
Système d'exploitation créé par Digital Research qui est devenu un système standard.

CPU : (Central Processing Unit-Unité centrale)
La partie de l'ordinateur contenant les circuits qui contrôlent et exécutent le déroulement des instructions d'un programme.

CRUNCH : Permet de minimiser la zone de la mémoire de l'ordinateur utilisée pour stocker un programme.

CURSEUR : Le carré qui indique la position courante à l'écran.

DATASETTE : Un périphérique utilisé pour stocker des fichiers programmes et données, séquentiellement sur une cassette.

DECLIN : La vitesse à laquelle le volume d'une note de musique décroît de sa valeur maximale à son volume moyen appelé niveau de soutien.
Voir soutien.

DECREMENT : Pour diminuer une variable d'index ou un compteur d'une valeur spécifique.

DETECTION DE CONFLIT : Une tâche réalisée dans un réseau à accès multiples pour empêcher deux ordinateurs de transmettre en même temps.

DIMENSION : La propriété d'un tableau qui détermine la direction le long d'un axe dans lequel les éléments du tableau sont stockés. Par exemple, un tableau à deux dimensions a un axe X pour les lignes et un axe Y pour les colonnes.
Voir tableau.

DONNEES : Nombres, lettres ou symboles qui sont entrés dans l'ordinateur pour être traités.

Dr. LOGO : Version de LOGO développée par Digital Research, qui est un langage de programmation avec une tortue graphique.

DUREE : Le temps pendant lequel une note de musique est jouée.

ECRAN : Unité de visualisation vidéo qui peut être une télévision ou un moniteur vidéo.

ENTIER : Un nombre entier (par exemple, un nombre avec aucun chiffre après la virgule), tel que 0,1,2,3, etc...

ENTREE : (Input)
Données mises dans l'ordinateur pour être traitées. Les différentes sources d'entrées sont le clavier, le lecteur de disquettes, le lecteur de cassettes ou le modem.

EPROM : (Electrically Programmable Read-Only Memory - Mémoire morte programmable électriquement). C'est une PROM qui peut être effacée par l'utilisateur, habituellement en l'exposant aux ultra-violets.
Voir PROM.

E/S : (I/O = Input/Output. Entrée/sortie)
Processus d'entrée des données dans l'ordinateur ou de transfert des données d'un ordinateur vers un lecteur de disquettes, une imprimante ou une mémoire.

EXECUTION : Pour exécuter les instructions spécifiées dans une commande ou une instruction d'un programme.

EXPRESSION : Une combinaison de constantes, de variables ou d'éléments d'un tableau avec des opérateurs logiques, mathématiques ou relationnels qui calculent une valeur numérique.

FICHIER : Un programme ou un ensemble de données regroupé en une seule entité. Ils sont sauvegardés sur disquette ou cassette.

FONCTION : Une opération prédéfinie qui donne une seule valeur.

FREQUENCE : Le nombre d'ondes sonores par seconde dans un son.
La fréquence correspond au son le plus haut que l'on peut entendre.

FREQUENCE PORTEUSE : Un signal constant transmis entre les périphériques de communication, ce signal est modulé pour coder des informations binaires.

GENERATEUR D'ENVELOPPE : Partie de l'AMSTRAD 6128 qui produit des formes d'ondes déterminées (en dents de scie, triangulaires, à impulsions variables et bruit) pour des notes de musique.
Voir la forme d'onde.

GRAPHISME : Visualisation d'images à l'écran, représentant des données en mémoire de l'ordinateur (par exemple , caractères symboles et dessins).

GRILLE : Une matrice à deux dimensions divisée en lignes et colonnes. Les grilles sont utilisées pour dessiner des caractères programmables.

HAUTEUR D'UN SON : La fréquence de l'onde sonore qui détermine si le son est haut ou bas.
Voir fréquence.

HARD COPY : Impression sur papier de l'écran.

HEXADECIMAL : Système de numération en base 16. Les prgrammes en langage machine sont souvent écrits en notation hexadécimal.

HOME : Le coin supèrieur gauche de l'écran.

HORLOGE : (Clock)
Le circuit de synchronisation du microprocesseur.

IC : (Integrated Circuit)
Circuit intégré. Une puce en silicium contenant un circuit électrique fait de composants tels que transistors, diodes, résistances et condensateurs. Les circuits intégrés sont plus petits, plus rapides et plus efficaces que les circuits individuels utilisés dans les anciens ordinateurs.

IMPRIMANTE : Un périphérique qui imprime le contenu de la mémoire de l'ordinateur sur une feuille de papier.

INCREMENT : Pour augmenter la variable d'un index ou le compteur d'une valeur déterminée.

INDEX : La variable du compteur dans une boucle FOR ... NEXT.

INDICE : Une variable ou une constance qui fait référence à un élément spécifique d'un tableau par sa position dans le tableau.

INSTRUCTION : Une instruction BASIC contenue dans une ligne de programme.

INSTRUCTION D'AFFECTATION : Une instruction BASIC qui donne à une variable, contance ou élément d'un tableau, une valeur numérique ou une valeur chaîne de caractère spécifique.

INTERFACE : Le point de rencontre d'un ordinateur et d'une entité externe. Soit un opérateur, un périphérique ou un moyen de communications. Une interface peut être physique, comportant des connecteurs, ou logique comportant un logiciel.

JEU DE CARACTERE : Un groupe de caractères apparentés. Les jeux de caractères de l'AMSTRAD 6128 sont : lettres majuscules, lettres minuscules et caractères graphiques.

JOYSTICK : Un périphérique qui généralement remplace la fonction des touches gérant le déplacement du curseur, très utilisé pour les jeux.

KILO-OCTET : (Kilobyte= KO)
Représente 1024 octets.

LANGAGE D'ASSEMBLAGE : Un langage orienté machine dans lequel des mnémoniques sont utilisés pour représenter chaque instruction en langage machine. Chaque CPU a son propre langage d'assemblage spécifique.

LANGAGE MACHINE : Le langage le plus bas que l'ordinateur puisse comprendre. L'ordinateur traduit tous les langages de haut niveau comme le BASIC en langage machine avant d'exécuter les instructions. Le langage machine est écrit en binaire, ce qui permet à l'ordinateur une exécution immédiate. Aussi appelé code machine ou code objet.

LECTEUR DE DISQUETTES : Une mémoire de masse à accès direct, qui sauvegarde et charge des fichiers vers et à partir d'une disquette.

LIGNE COMMUTEE : La ligne téléphonique normale commutée qui peut être utilisée comme un moyen de transmission de données.

LIGNE DE PROGRAMME : Une instruction ou une série d'instructions précédées par un numéro de ligne dans un programme.

LIGNE LOUEE SPECIALISEE : Un agencement de lignes téléphoniques spéciales fourni par les PTT. C'est indispensable pour certains ordinateurs ou terminaux pour obtenir une connection en permanence.

LOGICIEL : (Software)
Des programmes informatiques (série d'instructions) sauvegardés sur disquette, cassette ou cartouche qui peuvent être chargés dans la RAM. Le logiciel indique à l'ordinateur ce qu'il doit faire.

MATERIEL : (Harware)
Composants physiques d'un système informatique tels que le clavier, le lecteur de disquette et l'imprimante.

MATRICE : Un rectangle à deux dimensions avec des valeurs en lignes et en colonnes.

MEMOIRE A BULLES : Un type relativement nouveau de mémoire d'ordinateur, il utilise des 'case' ou 'bulles' minuscules et magnétiques pour stocker des données.

MEMOIRE CARACTERES : La zone de mémoire de l'AMSTRAD 6128 qui stocke la structure des caractères codés qui sont affichés à l'écran.

MICROPROCESSEUR : Une CPU qui est contenue dans un seul circuit intégré. Le microprocesseur utilisé dans l'AMSTRAD 6128 est le Z80.

MICROPROGRAMME : (Firmware) Des instructions de l'ordinateur stockées dans la ROM, comme des cartouches de jeux.

MISE AU POINT : Pour corriger les erreurs dans un programme.

MISE EN SERVICE : Mettre en fonction un bit un octet ou une opération de l'ordinateur.

MODE : Un état de fonctionnement.

MODE CARACTERES MULTICOLORES : Un mode graphique qui vous permet d'afficher quatre couleurs différentes dans une grille de 8*8 caractères.

MODE CARACTERES STANDARD : Le mode de l'AMSTRAD 6128 à sa mise sous tension ou lorsque vous écrivez des programmes.

MODE CONTINU : Un mode de communication à grande vitesse entre un lecteur de disquettes et un ordinateur, dans lequel l'information est transmise plusieurs fois à la vitesse normale.

MODE DIRECT : Le mode qui exécute les commandes BASIC immédiatement aprés avoir appuyé sur la touche RETURN. Aussi appelé mode immédiat .
Voir commande.

MODE DUPLEX : Permet à deux ordinateurs de transmettre et de recevoir des données en même temps.

MODE HAUTE RESOLUTION : Un mode graphique avancé pour l'AMSTRAD 6128 dans lequel vous pouvez contrôler chaque point de l'écran.

MODE HAUTE RESOLUTION MULTICOLORE : Un mode graphique qui vous permet d'afficher une des quatre couleurs pour chaque pixel dans une grille 8 * 8.
Voir PIXEL.

MODE SEMI-DUPLEX : Permet de transmettre dans une seule direction à la fois; si un appareil envoie, l'autre doit seulement recevoir les données juqu'à que ce soit à son tour de transmettre.

MODEM : (MOdulateur/DEModulateur)
Un appareil qui transforme les signaux de l'ordinateur en impulsions électriques pour la transmission par lignes téléphoniques et fait l'inverse pour la reception.

MODEM A CONNEXTION DIRECTE : Un appareil qui convertit les signaux digitaux d'un ordinateur en impulsions électroniques pour les transmettre par les lignes téléphoniques.
Comparez avec le coupleur acoustique.

MONITEUR : Un appareil qui ressemble à un appareil de télévision mais avec une image haute-résolution (plus nette) sur l'écran vidéo.

MONITEUR COMPOSITE : Un moniteur vidéo utilisé pour afficher 40 colonnes.

MONITEUR RGBI : (Red,Green,Blue,Intensity)
Un moniteur à affichage haute résolution nécessaire pour les écrans à 80 colonnes.

MOT : Nombre de BITS traités comme une seule unités par le CPU. Avec une machine 8 BITS , la longueur du mot est de 8 BITS. Avec une machine 16 BITS, la longueur du mot est 16 BITS.

NOMBRE ALEATOIRE : Un nombre décimal à neuf chiffres compris entre 0.000000001 et 0.999999999 généré par la fonction RND.

NUMERIQUE : (Digital)
Ayant rapport avec la technologie informatique et les communications de données où chaque information est codée en BITS 0 et 1 qui représentent les états en et hors fonction.

OCTAVE : Une suite de huit notes de la gamme musicale.

OCTETS : (BYTE)
Un groupe de huit bits qui constitue l'unité de stockage la plus petite dans un ordinateur. Chaque adresse mémoire de l'AMSTRAD 6128 contient un octet d'information. Un octet est l'unité de mémoire nécessaire pour représenter un caractère en mémoire.
Voir BIT.

OPERATEUR : Un symbole qui indique à l'ordinateur d'exécuter une opération mathématique, logique ou de relation sur des variables, contantes et éléménts de tabieau déterminé dans l'expression. Les opérateurs mathématiques sont +,-,*,/ et ^. Les opérateurs de relation sont <,=,>,<=,=> et <>. Les opérateurs logiques sont AND,OR,NOT,XOR.

ORDRE DES OPERATIONS : Ordre dans lequel les opérations sont calculées dans une expression mathématiques. Aussi appelé Hiérarchie des opérations.

ORDINATEUR : Un appareil éléctronique, numérique qui stocke et traite l'information. Raison d'être de cette documentation.

PAQUET DE DONNEES : Un moyen de transmettre des données en série sous forme de paquets performant qui comprend une série de contrôles d'erreurs.

PERIPHERIQUE : Tout appareil accessoire connecté à l'ordinateur tel qu'un lecteur de disquettes, une imprimante, un modem ou un joystick.

PIXEL : Terme informatique pour un élément de l'image. Chaque point de l'écran qui constitue une image est appelé un pixel. Chaque caractère sur l'écran est composé par une grille de 8*8 pixels.

POINTEUR : Un registre utilisé pour indiquer une adresse en mémoire.

PORT : Un canal par lequel les données sont transmises vers et à partir de la CPU. Un CPU de 8 BITS peut adresser 256 ports.

PORT PARALLELE : Un port utilisé pour la transmission des données avec un octet par fil multiple.

PORT SERIE : Un port utilisé pour la transmission série des données ; les bits sont transmis un BIT après l'autre par le même cable

PROGRAMME : Une série d'instructions qui indique à l'ordinateur d'exécuter une tâche spéciale. Les programmes peuvent être sauvegardés sur disquettes ou sur cassettes, résider en mémoire ou être imprimer.

PROGRAMMABLE : Capable d'être traité avec les instructions de l'ordinateur.

PROM : (Programmable Read Only Memory)
Mémoire en lecture seulement. Mémoire semi-conducteur dont le contenu ne peut pas être changé.

PROTOCOLE : Les régles que les ordinateurs suivent pour échanger des informations.Y compris l'organisation des données à transférer.

PUCE : (Chip)
Circuit électronique miniature qui exécute les opérations telles que le graphisme, le son ou les entrées/sorties.

RAM : (Read Acces Memory)
Mémoire vive. La zone programmable de la mémoire de l'ordinateur qui peut être lue et sur laquelle on peut écrire (modifier). Toutes les adresses RAM sont accessibles de la même manière à tout moment dans n'importe quel ordre.

REGISTRE : Toute adresse mémoire en RAM; chaque registre stocke un octet. Un registre peut stocker toutes les valeurs de 0 à 255 en notation binaire.

RELACHEMENT : La vitesse à laquelle le volume d'une note de musique baisse du niveau de soutien jusqu'à zéro.

REMARQUE : Commentaires utilisés pour documenter un programme. Les remarques ne sont pas exécutées par l'ordinateur, mais sont affichés dans la liste du programme.

RESEAU A ACCES MULTIPLE : Un système flexible par lequel chaque station peut avoir accès au réseau à tous moments. Des dispositions sont prises pour gérer la transmission simultanée de deux ordinateurs.

RESEAU DE BUS : Un système dans lequel tous les postes ou périphériques communiquent en utilisant un BUS ou un canal de distribution commun.

RESEAU EN ANNEAU : Un système dans lequel tous les postes sont reliés pour former une boucle ou un cercle en continu.

RESEAU LOCAL : Un des systèmes de communications de données de courte distance, caractérisé par l'usage commun d'un moyen de transmission par beaucoup de périphériques et la grande vitesse de données. Aussi appelé Local Area Network ou LAN.

RESOLUTION : La densité de pixels sur l'écran, cette densité détermine la finesse en détail de l'image affichée.

ROM : (Read Only Memory)
Mémoire morte. La zone permanente de la mémoire de l'ordinateur. Le contenu des adresses de la ROM peut être lu, mais pas modifié. La ROM de l'AMSTRAD 6128 contient le traducteur de langage BASIC et le système d'exploitation AMSDOS.

ROUTINE : Partie d'un programme auquel, on fait fréquemment appel.

RS-232 : Un standard recommandé pour les spécifications éléctroniques et mécaniques des ports de transmission en série.

SIMULATION : Technique d'émulation d'un processus réél par ordinateur, tel qu'une simulation de vol ...

SOUS-PROGRAMME : Un segment de programme indépendant séparé du programme principal qui exécute une tâche spécifique. Les sous-programmes sont appelés dans le programme principal par l'instruction GOSUB et doivent se terminer par une instruction RETURN.

SOUTIEN : Le volume moyen d'une note de musique.

SYNCHRONISATION : Une technique utilisée pour synchroniser des périphériques de communication d'envoi et réception des données. Cette technique est modulée pour coder des informations binaires.

SYNTAXE : Les règles grammaticales d'un langage de programmation.

SYSTEME D'EXPLOITATION : Un programme résidant qui contrôle tout ce que fait l'ordinateur.

SYSTEME D'EXPLOITATION DE DISQUETTES : (Disk Opérating System- DOS) Programme utilisé pour transférer des informations de et vers la disquette.

TABLEAU : Une structure de stockage de données dans laquelle une série de constantes ou de variables apparentées sont stockées dans des adresses mémoires consécutives. On se réfère à un élément pour chaque constante ou variable contenue dans un tableau. On accède à un élément par un indice.
Voir Indice.

TONALITE : Un son perceptible d'une hauteur et d'une forme d'onde spécifiques.

TORTUE : Symbole graphique qui fonctionne sous Docteur LOGO comme un curseur graphique.

TRANSMISSION ASYNCHRONE : Un système dans lequel les caractères de données sont envoyés à intervalles de temps aléatoires.
Limite la transmission par ligne téléphonique à peu près 2400 bauds.
Voir transmission synchrone.

TRANSMISSION SERIE : L'envoi de BITS de données rangés séquentiellement.

TRANSMISSION SYNCHRONE : Communications de données qui utilisent un signal de synchronisation entre les postes de réception et les postes émetteurs.

TRANSPARENT : Décrit un fonctionnement de l'ordinateur qui ne demande pas l'intervention de l'utilisateur.

VARIABLE : Une unité de stockage qui représente une chaîne de caractères ou des valeurs numériques variables. Le premier caractère doit toujours être une lettre.

VITESSE DE TRANSFERT DE DONNEES : La vitesse à laquelle les données sont envoyées à un ordinateur récepteur- donnée en BAUD, ou en BITS par seconde (bps)

Imp. Delta Graphique Service

Document numérisé
avec amour par :

AMSTRAD
CPC

MÉMOIRE ÉCRITE

http://amstradcpc.fredisland.net/