PRELIMINAIRES

Nous vous remerçions d'avoir acheté ce logiciel POWERSOFT et espérons que vous en serez entierement satisfait.

Ce logiciel est composé de deux cassette :

CASSETTE 1 : HBASIC et CHARGEUR.
CASSETTE 2 : DEMONSTRATION (HBASIC en mémoire).

Le listing de cette démonstration est fournis en annexe de cette documentation pour que vous puissiez vous familiarisez avec le HBASIC.

Il est hautement recommandé de lire le mode d'emploi avant d'essayer le HBASIC, et en particulier les premiers chapitres. Nous supposons que vous possedez des connaissances en BASIC, et le cas échéant n'hésitez pas à vous reporter au manuel fournit avec votre AMSTRAD.

Nous vous souhaitons BONNE CHANCE et bien du plaisir avec HBASIC.

INTRODUCTION

Le HBASIC est une extension au BASIC 1.0 ou BASIC 1.1 de l'amstrad. C'est à dire qu'il apporte 46 nouvelles fonctions à ce dernier.

Le HBASIC se charge en RAM en début de cession, cela veut dire que lorsque l'on éteint l'appareil, le HBASI est éffacer et donc qu'il faut le recharger à chaque fois que l'on désire s'en servir ou utiliser un programme qui en utilise les fonctions.

CHAPITRE PREMIER

I.1 - CHARGEMENT DU HBASIC

A. Mettre la cassette HBASIC dans le lecteur de cassette.
B. Rembobinez la cassette si necessaire.
C. Appuyer sur la touche 'PLAY' du lecteur de cassette.
D. Tapez RUN "HBASIC.BAS" et appuyez sur la toche (ENTER).
E. Le message Press PLAY then any key : apparait
F. Appuyer sur (ENTER)

Au bout d'un moment apparait la logo de HBASIC, puis au bout de 5 minutes environ le HBASIC est prêt à etre utilisé.

I.2 - GENERALITES

A. Le HBASIC se charge en haut de votre mémoire, il vous reste donc 33987 octets de libre. Cependant, il y a certaines précautions à observer impérativement lors de l'emploi du HBASIC :

- Ne jamais tenter de modifier le contenu de la mémoire à des adresses comprises entre 34400 et 49152.
- Toujours vérifier que les parametres que l'on donne à une instruction sont bien des variables entières et déclarés.

Les deux sections suivantes vont vous expliquez plus en détail le pourquoi de ces précautions

B. HBASIC est un logiciel entièrement en language machine d'un taille d'environ 8000 octets, il est logé en haut de la mémoire afin de laisser le plus de place disponible à l'utilisateur. Donc, si l'on modifie la mémoire entre les adresses 34400 et 49152, l'on modifie par conséquent HBASIC, ce qui peut avoir des conséquences désastreuses.

De plus, il ne faut pas faire d'instruction MEMORY au dessus de 34400, ainsi MEMORY 40000 effacerait inévitablement le HBASIC et risquerait fortement de planter la machine dès que l'on utiliserait une de ses fonctions.

Précisons qu'à partir de 49152 ce trouve la mémoire écran, mais nous reviendrons sur ce sujet plus tard.

C. Une variable est déclaré à partir du moment ou elle est définie, le simple fait d'écrire A%=10, déclare la variable, c'est à dire que l'AMSTRAD sait que cette variable existe, et il sait où la trouver. Il fait donc que la variable est été déjà utilisée au moins une fois.

Une variable entière est une variable dont la valeur est comprise entre - 32767 et + 32767 (nombre entier), il existe deux façons de 'déclaré' qu'une variable est entière :

- en la faisant suivre du signe % (A%).
- en la définissant comme entière (DEFINT A).

Nous conseillons donc d'utiliser la premiere solution si il n'y a que quelques variables utilisées, sinon la deuxième solution permet une plus grande souplesse quand le nombre de variables devient grand.

En effet l'introduction DEFINT permet de déclarer comme etant entières toutes les variables commençant par une même lettre.

DEFINT A : toutes les variables commençant par la lettre A.
DEFINT Q-V : toutes les variables commençant par les lettres de Q à V.

ATTENTION : toutes les instructions du HBASIC dont le parametre n'est pas entier plantera irrémédiablement l'AMSTRAD.

De plus, le fait d'utiliser des variables entières permet de gagner en rapidité (ex : boucles FOR - NEXT). Mais attention, cela s'applique uniquement au cas du HBASIC, rien ne vous empêche d'utiliser des variables non entières avec les instructions BASIC normales.

D. Le HBASIC génère des messages d'erreur tout comme le BASIC (en fait se sont les même), il y en à trois types différent :

- OPERANDE MISSING, le nombre de paramêtres transmis à cette instruction est inexacte.
- IMPROPER ARGUMENT, un ou plusieurs des paramêtres de l'instruction est une variable non déclarée.
- UNKNOW COMMAND, le nom de l'instruction est inexacte (erreur de frappe ou de syntaxe).

De plus, il est inutile d'utiliser l'instruction SYMBOL AFTER, en effet automatiquement lors du chargement le HBASIC effectue un SYMBOL AFTER 32, tous les caractères sont donc redéfinissables.

En effet si vous tentez de faire un SYMBOL AFTER, vous aurez l'erreur IMPROPER ARGUMENT qui apparaitra.

E. Toutes les instructions doivent être précédées du caractère | (SHIFT à, après le P sur votre clavier).

Tous les paramètres doivent être précédés d'une virgule ',' et du caractère à (après le P sur votre clavier).

FORMAT : |instruction ,à variable %

Il peut bien sûr y avoir plusieurs paramètres, mais toujours avec la même syntaxe. Voici quelques exemples avec l'instruction |CIRCLE, cette instruction admet deux paramètres qui sont le rayon et la couleur du cercle.

|CIRCLE 10,1 est fausse, la bonne syntaxe est

R%=10 : C%=1 : |CIRCLE ,àR% ,àC% mais,

|CIRCLE ,àR ,àC n'est pas forcement faux :

En effet, s'il y a eu auparavant une instruction DEFINT qui incluait les variables commençant par R et C (DEFINT C-R), celles - çi seraient donc entières, mais dans le cas contraire, cette ligne n'est pas valable et peut entrainer des conséquences désagréables. D'autre part, il faut bien entendu que R et C aient été utilisés au moins une fois auparavant.

Ce chapitre d'introduction est terminé, il contient des éléments indispensable à toute utilisation préalable du HBASIC, relisez-le si nécessaire ou si vous rencontrez un problême.

RESOLUTIONS ET COULEURS

Ce chapitre à pour but d'expliquer ou de clarifier certains points en ce qui concerne l'AMSTRAD, ceci afin de permettre une meilleure compréhension et utilisation des instruction du HBASIC par la suite.

II.1 - LE TEXTE

A. Quand vous allumez votre AMSTRAD, vous pouvez taper du texte à l'écran. Ce texte s'affiche suivant une matrice de lettres, chiffres, et signes spéciaux appellés caractères, cette affichage se mesure en nombre caractères par ligne et selon le nombre de lignes. Arrivé en bas de l'écran, le texte 'monte' et la première ligne disparait.

On parle donc de caractères par ligne et de nombre de lignes pour dire la taille d'un écran, c'est ainsi qu'à l'allumage, votre AMSTRAD possède une capacité d'affichage de 40 caractères par lignes et ce sur 25 lignes.

Tapez : MODE 2 (ENTER)

L'écran s'éfface et en haut à gauche apparait le mot Ready qui est écrit plus petit que tout à l'heure. En effet, dans ce mode on peut afficher 80 caractères par ligne et ce sur 25 lignes soit 25*80.

Tapez : MODE 0 (ENTER)

L'écran s'éfface, Ready apparait en haut à gauche, mais bien plus gros. Pour etre exacte, il faudrait dire beaucoup plus large. En effet dans ce mode nous avons toujours 25 lignes mais seulement 20 caractères par ligne soit 20*25.

L'écran possède donc 3 modes d'écran differents.

B. Revenons en mode 1, tapez : MODE 1 (ENTER)

Tapez : PEN 2 (ENTER)

Le mot Ready apparait écrit en bleu (si vous ne possédez pas la couleur, le mot Ready doit apparaitre dans un vert légèrement plus clair), nous avons donc changer la couleur de l'écriture.

Tapez : PAPER 3 (ENTER)

La couleur du fond change et devient ROUGE, mais la couleur des caractères n'a pas changer. Cela veut dire que nous pouvons changer la couleur d'écriture et la couleur de fond, c'est comme si nous avions du papier de couleur (couleur de fond, instruction PAPER, papier en Anglais) et des crayons de couleur (couleur d'écriture, instruction PEN, crayon en Anglais) et que nous pouvions changer la couleur du papier et du crayon à tout moment. Suivant les mode d'écran, nous avons le choix entre un nombre différent de couleurs pour le fond ou le texte (écriture) :

MODE	COULEURS
0	16
1	4
2	2

Le nombre que nous indiquons après une instruction PAPER ou PEN indique le numéro de la couleur choisie, en MODE 1, à l'allumage, il y a la répartition suivante

NUMERO	COULEUR
0	Noir
1	Jaune D'or
2	Bleu
3	Rouge

Ainsi, si l'on veut écrire rouge sur fond noir, il faut taper : PAPER 0 : PEN 3 (ENTER)

Tapez : INK 3,26 (ENTER)

Brusquement, tout ce qui était rouge sur l'écran estr devenu blanc. Que c'est-il passé ? ... Il faut savoir qu'en réalité l'AMSTRAD possède 27 couleurs dufférentes mais qu'elles ne peuvent être affichées toutes en même temps, il est donc nécéssaire de faire un cohix, ainsi en mode 1 nous avons la possibilité d'avoir 4 couleurs parmis 27, voyons un exemple préçis.

Considérons nos quatre couleurs numérotés de 0 à 3, et qu'avec celle-çi nous voudrions la combinaison suivante :

NUMERO	COULEUR
0	Blanc
1	Vert
2	Orange
3	Rose

C'est à dire que lorsque nous ferons PAPER 3 par exemple, nous obtiendrons un fond rose.

Tapez : INK 0,26 : INK 1,9 : INK 2,15 : INK 3,16
Tapez : (ENTER), puis PAPER 3 (ENTER)

Nous obtenons bien un fond Rose, et pour avoir des caractères vert, il faut taper PEN 1. L'instruction INK permet de définir la couleur de nos crayons qui sont numérotés de 0 à 3 et ceçi avec un choix de 27 encres (INK) différentes pour chaque crayon. Donc nous n'aurons que 4 couleurs simultanément sur une palette de 27 (numérotées de 0 à 26).

Le premier nombre de l'instruction INK indique le numéro de la couleur, et le second le numéro de la couleur. C'est ainsi que nous pouvons choisir sur une gamme de 27 couleurs une de ces dernieres pour chacun de nos quatres crayons. Il est tout à fait possible de donner à deux crayons la meme couleur.

Tapez : INK 2,9 (ENTER)

Il n'y aura plus de différence entre la couleur du crayon 1 et du cranyon 2. Il en va de meme avec les 16 crayons et en mode 2 avec les 2 crayons.

Evitez de prendre comme couleur de papier et comme couleur d'écriture le même crayon car vous n'y verriez plus rien. Pour avoir la liste des 27 couleurs reportez vous aux appendices de la documentation de l'AMSTRAD.

C. Vous n'avez sans aucun doute remarquez un caree plein qui suit les caractères que vous frappez à l'écran, cela s'appelle le CURSEUR, nous l'appelerons le CURSEUR TEXT, car il suit le texte qui est frappé. Ce curseur peut être positionner à n'importe quel endroit de l'écran.

Tapez : LOCATE 5,10 : PRIT "AMSTRAD" (ENTER)

Vous verrez apparaitre le mot AMSTRAD à la 5° colonne et à la 10° ligne. Il est ainsi possible d'atteindre n'importe qu'elle position de l'écran. L'origine de celui-çi est en haut à gauche en coordonnées 1,1. Ainsi LOCATE 1,1 vous enverra dans le coin gauche suppérieur de l'écran.

II.2 - LE GRAPHISME

Mais l'AMSTRAD ne sais pas seulement écrire, il sait aussi déssiner. Il possède donc un autre crayon qui lui permet de dessiner et qui est différent de celui qui permet d'écrire.

A. Les couleurs fonctionnent comme pour le texte, mais dans le basic normal, il n'y a pas d'instruction PEN ou PAPER relative au graphisme, pour obtenir un fond d'une couleur donnée, il faut tapez : CLG (N° Couleur).

Tapez : CLG 3 (ENTER)

Vous obtiendrez alors un fond rouge mais l'écran s'éffacera. Pour le crayon graphique, il n'y à pas d'instruction spécifique, très souvent le numéro de la couleur est précisé dans l'instruction même de trace : DRAW 100,100,1 tracera une ligne jaune, car 1 est le numéro de la couleur, et à l'allumage, la couleur 1 est le jaune.

B. La résolution d'un ordinateur est le nombre de points différents que celui-çi peut mettre à un endroit précis de l'écran. Pour récupérer ces points, on utilise un repère qui possède une origine. Sur l'AMSTRAD, l'origine est en 0,0 et est situé au bas gauche de l'écran. Le point est repéré par un couple de coordonnées X et Y, La coordonée X corespond à la colonne ou se situe le point et la coordonée Y à la ligne ou il se situe. sur l'AMSTRAD, la résolution varie selon le mode :

MODE	RESOLUTION
0	160 * 200 points
1	320 * 200 points
2	640 * 200 points

Mais sur l'AMSTRAD les choses ne sont pas si simple ; en effet theoriquement le point de coordonne 159,199 en mode 0 est supposé se situé dans le coin haut à droite de l'écran, or il n'en est rien.

Tapez : MODE 0 : PLOT 159,199,1 (ENTER)

Un point s'allume, mais il n'est pas ou nous l'attendions. En effet, pour simplifier la tache au programmeur et lui éviter de recalculer tous ses points si jamais il change de mode, AMSTRAD à décider que les 3 modes auraient un même repère. ATTENTION : cela ne signifie pas qu'ils ont la même résolution, il s'agit la d'un simple artifice car en effet, le coin en haut et à droite de l'écran à pour coordonnée 638,398.

Tapez : PLOT 638,398,1

Le point dans le coin en haut à gauche s'allume bien, donc l'ordinateur effectue une conversion entre les coordonées du repère pour les transformées en coordonées physiques ainsi le point 638,398 dans le repère correspond au point 319,199 de coordonnée physique. En effet, les coordonées physique changent selon le mode tandis que le repère ne bouge pas. La différence entre les modes est la taille du point, ainsi :

PLOT 0,0,1 : PLOT 0,1,1 n'allumera qu'un point
PLOT 0,2,1 allumera le point adjacent.

En résumé, voiçi le tableau de prise en compte :

MODE	PRISE EN COMPTE
0	1 sur 4
1	1 sur 2
2	1 sur 1 soit tous les points.

Le mode 2 est donc celui qui à le plus de points différents, donc la plus grande finesse de trace, en échange, il n'à que 2 couleurs. Le mode 1 à deux fois plus de couleurs, mais deux fois moins de précision, c'est le mode qui à le meilleurs rapport couleur / précision.

Pour terminer, il existe un curseur graphique qui lui est invisible, il se déplace lorsque l'on effectue un trace (DRAW ou PLOT). on peut le grace à l'instruction MOVE X,Y. Ce déplacement est absolu, c'est à dire indépendant de la position du curseur. Il amenera le curseur graphique en X,Y et MOVER X1,Y1 l'ammenera en X+X1, Y+Y1 car c'est un déplacement relatif à partir de la derniere position du curseur.

Voilà pour la mise au point. Nous esperons qu'elle vous aura été bénéfique.

Ce chapitre à pour but d'expliquer le fonctionement des instructions du HBASIC qui permettent de tracer des figures simples. Les coordonnées qui seront indiquées éventuellement, le seront par rapport au repère standard de l'AMSTRAD dans un repère de 640*400 points soit le MODE 2 (Cf chapitre II).

III.1 - CRAYON ET PAPIER

Comme vous l'avez vu au chapitre II, l'AMSTRAD ne possède pas d'instruction PEN et PAPER correspondantes au graphisme, c'est maintenant chose faite avec |GRAPHICPEN et |GRAHICPAPER.

A. L'instruction |GRAPHICPAPER permet de changer la couleur du crayon graphique, c'est à dire que les prochains traces s'éffectuerons dans cette couleur. Ainsi le programme suivant permet de changer la couleur du crayon graphique :

```
10 MODE 1
20 C%=3 : |GRAPHICPEN,àC%
30 DRAW 100,100
```

La ligne 20 selectionne la couleur 3 (rouge à l'allumage) donc la ligne tracé en 30 sera ROUGE.

B. L'instruction |GRAPHICPAPER permet de changer la couleur de fond de la fenêtre graphique. En effet, grace à l'instruction ORIGIN on peut réduire la taille de la fenêtre graphique :

```
10 MODE 1
20 ORIGIN 0,0,100,540,100,300
30 C%=2 : |GRAPHICPAPER,àC%
40 CLG
```

Seul une partie de l'écran sera éffacée et deviendra bleu. Bien que l'on pouvait faire la même chose en remplaçant la ligne 30 par CLG 2.

Mais là ou les choses changent, c'est quand nous voulons écrire des caractères avec TAG, la couleur de fond ne peut être fixer que par CLG qui efface l'écran, avec |GRAPHICPAPER ce problème disparaît :

```
Tapez :
40 MOVE 320,200 : TAG : PRIT "AMS";
50 C%=3 : |GRAPHICPAPER,àC%
60 PRIN "TRAD";
```

AMS sera écrit sur fond BLEU et TRAD sur fond ROUGE, avec CLG ce n'était pas possible.

III.2 - LE CERCLE

Quand nous désirons tracer un cercle, il faut connaitre trois élémemnts :

Le centre du cercle
Le rayon du cercle
La couleur du cercle

Mais nous pouvons aussi vouloir ne tracer qu'une partie de cercle c'est à dire un arc de cercle ou encore déformer notre cercle c'est à dire une ellipse, HBASIC apporte trois nouvelles instructions permettant de réaliser ces figures, |CIRCLE, |ARC, |ELLIPSE.

A. La commande |CIRCLE permet de tracer un cercle à l'écran. Il faut préciser le rayon et la couleur, le centre sera la position du curseur graphique. Ainsi si nous voulons tracer un cercle de centre 320,200, de rayon 100 et de couleur 1 nous ferons :

```
10 MOVE 320,200
20 R%=100 : C%=1 : |CIRLE,àR% ,àC%
```

Voiçi un petit programme qui dessine des cercles concentriques :

```
10 MODE 2
20 C%=1
30 FO R%=10 TO 100 STEP 10
40 MOVE 320,200
50 |CIRCLE,àR% ,àC%
60 NEXT R%
```

Regardez la ligne 40, elle est obligatoire ; détruisez là et regarder le programme, le résultat est totalement différent, en effet une fois le trace du cercle éxécuté, le curseur graphique se trouve sur le dernier point allumé à l'écran, c'est à dire dans le cas du cercle à droite du cercle à une distance égale au rayon, soit sir le cercle à un centre en 320,200 et un rayon de 100, le cuseur se trouve alors en 420,200.

B. La commande |ARC permet de tracer un arc de cercle c'est à dire une partie de cercle. Pour délimiter la partie du cercle à tracer, on utilise la graduation en angles (le cerlce fait 360° -degrés-). Il faut donc donner l'angle de départ, l'angle d'arrivée, le rayon du cercle, et la couleur, le centre etant donné par le curseur graphique. Ainsi, si l'on veut déssiner un quart de cercle (le deuxième) de rayon 150, de centre 200,200 et de couleur 1, on fera

```
10 MODE 2
20 MOVE 200,200
30 A1%=90 : A2%=180 : ' ARC DE 90° à 180°
40 R%=150:C%=1
50 |ARC,àA1%, àA2%, àR%, àC%
```

Attention, le curseur graphique à la fin du tracé sera au début de l'arc car ce dernier se trace de 180° vers 90°.

Voici un programme qui déssine un joli coeur :

```
10 MODE 2
20 MOVE 260,140
30 A1%=0: A2%=180 : R%=40 : C%=1
40 |ARC,àA1%, àA2%, àR%, àC%
```

```
 50 MOVE 340,140
 60 A1%=0 : A2%=180 : R%=40 : C%=1
 70 |ARC,àA1%, àA2%, àR%, àC%
 80 MOVE 380,140
 90 A1%=180 : A2%=240 : R%=160 : C%=1
100 |ARC,àA1%, àA2%, àR%, àC%
110 MOVE 220,140
120 A1%=300 : A2%=360 : R%=160 : C%=1
130 |ARC,àA1%, àA2%, àR%, àC%
```

C. La commande |ELLIPSE permet de tracer une ellipse cette figure se distingue du cercle par le fait qu'elle possède 2 rayons. Il faut toujours préciser la couleur, le centre étant indiqué par le cuseur graphique. Ainsi si nous, voulons tracer une ellipse de centre 320,200 de rayon en X=50 et en Y=100 de couleur 1, nous ferons :

```
10 MODE 2
20 MOVE 320,200
30 RX%=50 : RY%=100 : C%=1
40 |ELLIPSE,àRX%, àRY%, àC%
```

L'ellipse sera deux fois plus haute que large. Voci un programe qui déssine une sphère :

```
 10 MODE 2
 20 C%=1 : RY%=100
 30 FOR RX%=10 TO 100 STEP 10
 40 MOVE 320,200
 50 |ELLIPSE,àRX%, àRY%, àC%
 60 NEXT RX%
 70 RX%=100
 80 FOR RY%=10 TO 100 STEP 10
 90 MOVE 320,200
100 |ELLIPSE,àRX%, àRY%, àC%
110 NEXT RY%
```

III. 3 - AUTRE

Après les surfaces centriques, nous alons voir le tracé de rectangle et d'axes gradués. Les commandes sont |BOX et |AXES.

A. La commande |BOX permet de tracer un rectangle en connaissant sa largeur (axe des X), et sa hauteur (axe des Y), il faut préciser sa couleur. Le curseur graphique doit être le coin gauche de sa base. Ainsi, si l'on veut tracer un rectangle de 100*200 (L*H), de base 10,16 et de couleur 1 on fera :

```
10 MODE 2
20 LA%=100 : HA%=200 : C%=1
30 MOVE 10,16
40 |BOX,àLA%, àHA%, àC%
```

Ici, pas de problème avec le curseur graphique car il revient au point de départ. Voiçi un programme qui permet de tracer une pyramide :

```
10 MODE 2
20 C%=1 : LA%=10 : HA%=6
30 FO U=1 TO 80
40 MOVER -2,-2
50 |BOX,àLA%, àHA%, àC%
60 LA%=LA%+4
70 HA%=HA%+4
80 NEXT U
```

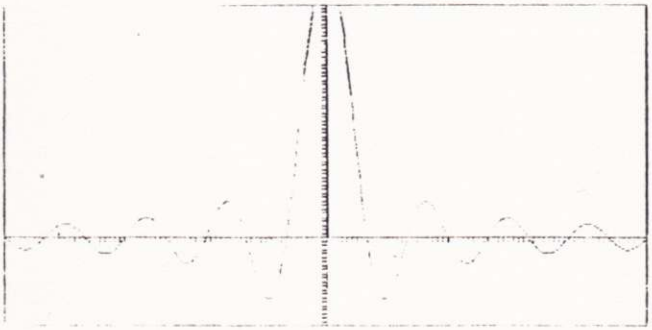
B. La commande |AXES permet de tracer un axe gradué sur l'écran, essentiellement destiné au tracé de courbes. Il faut préciser l'origine de l'axe, le pas de graduation en X, en Y et la couleur. Ainsi si l'on veut tracer un axe centré en 320,100 gradué de 10 en 10 sur X et de 6 en 6 sur Y de couleur 1 on fera :

```
10 MODE 2
20 XX%=320 : YY%=100
30 GX%=10 : GY%=6
40 C%=1
50 |AXES,àXX%, àYY%, àGX%, àGY%, àC%
```

Ici, le curseur graphique sera au sommet de l'axe des Y (en 320,398). Voici un programme qui trace la courbe d'équation Y=SIN(X) / X :

```
10 MODE 2
20 XX%=300 : YY%=110 : GX%=5 : GY%=5 : C%=1
30 |AXES ,àXX%, àYY%, àGX%, àGY%, àC%
40 XX%=638 : YY%=398 : MOVE 0,0
50 |BOX,àXX%, àYY%, àC%
60   MOVE 0,110
70 DEG
80 FOR X=-1441 TO 1441 STEP 4
90 Y=SIN(X)/X
100 MOVE X*0.22+320,Y*20000+110
110 NEXT X
```

Voiçi la recopie d'écran de cette courbe :



IV.1 - MELANGES

Quand l'AMSTRAD allume un point à l'écran, normalement celui-çi ne tient pas compte de la couleur qu'avait le point point avant d'être allumé. Ainsi :

MODE 1 : PLOT 100,100,1 allumera un point jaune à l'écran.
PLOT 100,100,2 cela mettra le point bleu.

Tapez : |GOR puis PLOT 100,100,1

Normalement, notre point aurait du redevenir jaune mais non, il est devenu rouge ? Il y a eu une addition des couleurs, notre point etait à la couleur 2, nous l'avons mis à la couleur 1, il est affiché à la couleur 3 (2+1 = 3), c'est l'instruction |GOR qui nous a permis de passer en mode mélange OR, GOR signifie GRAPHIC OR, OR est un opérateur logique (OU logique), il y à trois mode de mélanges disponibles sur l'AMSTRAD, mais pour le comprendre faisons donc un peu de BINAIRE : Votre AMSTRAD ne connait que 2 choses 0 et 1, c'est peu mais bien suffisant. En effet, quand on vous dit que votre AMSTRAD comprend 64 Ko de RAM cela veut dire qu'il contient 1024*64 octets, un octet etant un ensemble de 8 bits le bit etant l'unité de base, il peut comprendre les valeurs 0 ou 1, l'octet peut prendre des valeurs comprises entre 0 et 255 ; soit le nombre 1101 en BINAIRE, composé de 4 bits, quel est sa valeur décimal (le décimal etant la systeme à 10 chiffres dans lequel nous comptons).

Considérons le chiffre 1234 en décimal, nous pouvons le décomposer comme suit :

1000 + 200 + 30 + 4 ou
1*10^3 + 2*10^2 + 3*10^1 + 4*10^0

Nous utilisons des puissances de 10 car nous sommes en base 10, en base 2 (BINAIRE), nous utilisons des puissances de 2 ainsi 1101 se décompose comme suit :

1*2^3 + 1*2^2 + 0*2^1 + 1*2^0 soit

1*8 + 1*4 + 0*2 + 1*1 = 8 + 4 + 1 = 13

Si de la même manière, nous calculons 11111111 (8 bits soit 1 octet) nous obtiendrons 255.

Donc pour revenir à nos 64 Ko cela veut dire que nous avons 64*1024*8 bits soit 524288 bits (1Ko vaut 1024 octets -2^10-).

Mais cela nous éloigne de nos modes de mélange, nous en étions à l'opération logique OR. Cet opérateur logique éffectue une opération entre les bits de deux octets (dans notre cas l'octet contenant la couleur du point et l'octet contenant la couleur à laquelle nous désirons mettre le point), voçi ce que l'on appelle la table de vérité de l'opérateur logique OR :

1 OR 1 = 1
1 OR 0 = 0
0 OR 1 = 0
0 OR 0 = 0

Faites l'éssai avec 1 OR 2 (soit 01 or 10 en binaire) ce qui donne bien 3 (11 en binaire).

Il existe 3 opérateur logique qui sont : OR, XOR, AND. Nous allons voir les tables de vérité des deux autres opérateurs logiques XOR et AND :

1 XOR 1 = 0	1 AND 1 = 1
1 XOR 0 = 1	1 AND 0 = 0
0 XOR 1 = 1	0 AND 1 = 0
0 XOR 0 = 0	0 AND 0 = 0

l'opérateur logique XOR possède des propriétées intéressantes :

Tapez : |GXOR : MOVE 0,0 : DRAW 100,100,1

Cela trace un trait jaune, jusque là rien de spécial, maintenant tapez :

MOVE 0,0 : DRAW 100,100,1 puis

MOVE 0,0 : DRAW 100,100,1

La ligne disparait puis réapparait, notez au passage l'instruction qui active l'opérateur logique XOR (GXOR veut dire GRAPHIQUE XOR), donc en mode XOR, un trait peut être éffacer simplement en retraçant par dessus dans la même couleur. Mais le plus important est que si il y avait eu un déssin, il n'aurait pas été altéré, XOR permet de poser un déssin en transparent par dessus un autre, un premier tracé permet de poser le dessin, un deuxième l'enlever sans altérer ce qu'il y avait sur l'écran.

Nous voyons pour l'opérateur logique AND que de nouveaux mélanges sont possible, donc à chaque opérateur correspond un mélange, il en existe un quatrième qui est le mode normal, celui par défaut (à l'allumage), c'est à dire que le point est mis à la couleur spécifiée dans l'instruction quelque soit celle qui était sur l'écran.

HBASIC apporte un jeu de quatre opérateurs logiques qui permettent de fixer un des quatres mode de mélanges. Quand un mode est activé, il est valable pour tous les traces y compris ceux de figures (cercle, arc, ellipse), et meme les caractères quand ils sont écrit avec l'instruction TAG, voiçi ces quatre instructions :

IGOR	Mode OR
IGXOR	Mode XOR
IGAND	Mode AND
IGNORMAL	Mode NORMAL

IV.2 - REMPLIR

Imaginons un cercle tracé sur notre écran, nous voulons le colorier, il n'existe pas de moyen simple pour cela (sur le CPC 464), aussi, le HBASIC apporte une solution à ce problème : Le FILL (remplir en Anglais) et qui admet deux paramètres, la couleur à remplir et la couleur de remplissage, ce qui donnera :

```
10 MODE 1
20 R%=100 : C%=1
30 MOE 320,200 : |CIRCLE,àR%, àC%
40 MOVE 320,200 : CAR%=0 : CDR%=1
50 |FILL,àCAR%, àCDR%
```

CAR est la couleur à remplir et CDR la couleur de remplissage, ATTENTION : FILL ne marche que pour des surfaces entièrement fermées, même s'il manque un point pour fermer la surface, cela aura le désagréable effet de remplir le reste de l'écran (il y aura une fuite), ceci est valable même pour les bords de l'écran, ainsi si l'on trace un demi cercle en bas de l'écran et que l'on fait un FILL ensuite, cela risque de planter la machine. Il faut aupparavant tracer un cadre qui délimite les bords de l'écran (avec BOX par exemple) avant d'utiliser FILL. Nous vous conseillons de toujours tracer ce cadre avant d'utiliser FILL quitte à l'éffacer ensuite. La fonction FILL tient compte du mode de mélange (OR, XOR, AND, NORMAL), attention donc aux mauvaises surprises faites donc |GNORMAL avant votre FILL.

IV.3 - AGRANDIR

HBASIC possède une fonction puissante qui vous permet d'agrandir une partie de ce qui se trouve à l'écran, cette fonction possède deux modes de fonctionement et chacun de ces modes un option. Mais voyons d'abord l'option :

Quand nous voulons agrandir un dessin, nous prenons chaque point du dessin et nous dessinons à la place un rectangle dont l'échelle d'agrandissement sera égale hauteur choisie en X et à la largeur choisie en Y, par exemple prenons un agrandissement de 3 fois X et de 4 fois Y soit 3 points sur 4, et ce pour chaque points, ainsi le programme suivant agrandit le mot AMSTRAD :

```
10 DEFINT A-Z
20 CLS : PRINT "AMSTRAD"
30 DX=7*16 : DY=16 : SX=3 : SY=4 : CA=0
40 MOVE 0,400-16 : |ZOOM,àDX, àDY, àSX, àSY, àCA
50 LOCATE 1,10
```

Les deux premiers paramètres indiquent la taille de la zone à agrandir, et le curseur est positionner dans le coin bas et à droite de cette zone (un peu comme avec l'instruction BOX), les deux paramètres suivants indiquent les échelles d'agrandissement pour l'axe des X et des Y, le dernier paramètre représente notre option, au lieu de mettre un rectangle plein, nous nous sommes dit pourquoi ne pas mettre un caractère graphique ? Remplacez la ligne 30 par celle-çi :

```
30 DX=7*16 : DY=16 : SX=3 : SY=8 : CA=206
```

Au lieu d'un rectangle plein, il y a le caractère ASCII dont le code est 206. Cela veut dire que si le dernier caractère n'est pas null, c'est le caractère de code ASCII qui est mis à la place. Mais si nous avons le caractère en entier, c'est que nous avons changer les échelles de façon à ce que notre rectangle soit de la taille d'un carractère, maintenat remplacez la ligne 30 par celle-çi :

```
30 DX=7*16 : DY=16 : SX=3 : SY=4 : CA=206
```

Nous avons garder l'option caractère (CA≠0) mais redonnons aux echelles leurs valeurs initiales, le résultat est bien différent !. De plus l'instruction ZOOM prend en compte le mode de mélange, il est donc possible d'obtenir des effets à l'infini et de plus les caractères sont redéfinissables, c'est maintenant à vous de jouer et de laisser libre cours à votre imagination. Vous avez sans doute remarquer que l'agrandissement se produisait en effaçant la zone originale. Il est possible d'avoir un agrandissement qui se produit en un autre endroit de l'écran, pour cela il suffit de préciser les coordonnées de l'agrandissement après les echelles et avant le caractère de remplissage. Remplacez la ligne 40 par celle-çi :

```
40 MOVE 0,400-16 : X=10 : Y=15 : |ZOOM,àDX, àDY, àSX,
   àSY, àX, àY, àCA
```

L'agrandissement se fait en bas de l'écran, conservant l'original intact, d'ailleurs si vous prenez comme echelle 1 et 1 cela aura commme effet de dupliquer la zone à un autre endroit de l'écran, mais nous verrons plus loin un autre moyen d'y arriver.

IV.4 - JOYSTICK

L'instruction |JOYMOVE permet de déplacer un caractère à l'écran à l'aide du JOYSTICK (si vous en possedez un, sinon ne l'utilisez pas, vous ne pourriez que planter votre AMSTRAD).

Il faut deux parametres, le caractère à déplacer et le pas de déplacement, plus le pas est grand, plus le caractère se déplace vite, plus il est petit, plus il est précis mais moins vite il se déplace. De plus en mode XOR, lie caractère se déplace en transparence, pour les autres modes, cela dépend du caractère et du pas de déplacement, c'est à vous de voir. Pour sortir de ce mode, il suffit de presser le bouton de feu qui vous rend la main, sans quoi il ne vous reste plus qu'à eteindre votre AMSTRAD. ATTENTION au crayon, s'il est de la couleur du fond, vous ne verrez rien , utilisez GRAPHICPEN si néscéssaire. Voçi un exemple de programme qui commence par déplacer le caractère en transparence puis qui laisse une trainée derrière lui :

```
10 MODE 2
20 DEFINT A-Z
30 C=1 : |GRAPHICPEN,àC
40 |GXOR
50 MOVE 320,200
40  A=ASC("+") : V=4 : |JOYMOVE,àA, àV
50 FOR U=1 TO 100 : NEXT U
60 |GOR : A=129 : V=2 : |JOYMOVE,àA, àV
```

Plusieurs remarques s'imposent :

- Il n'y à pas de caractère % à aucun paramètre d'instruction car ils sont inutiles vue qu'en ligne 20 nous avons définie toutes les variables comme etant entière, mais attention il ne faut pas oublier cela sous peine de facheuses consequences.

- De plus, on pourrait se poser des questions quant à la nescessité de la ligne 70, qui est en réalité un boucle d'attente, cela est rendu nescessaire. En effet, quand vous appuyez sur le bouton de feu le programme continue tout de suite et enchaine ; seulement, à moins de relacher le bouton de feu extrement rapidement, la préssion sur le bouton est détecté par le JOYMOVE de la ligne 80 et la main vous est rendu immédiatement sans que vous ayez le temps de déplacer votre caractère.

La position à laquelle se positionera le caractère est celle du curseur graphique. De plus le curseur graphique est positioné par le caractère à l'endroit ou ce dernier est affiché, ces coordonées peuvent etres récupérées par les instruction XPOS et YPOS avec :

PRINT XPOS, YPOS

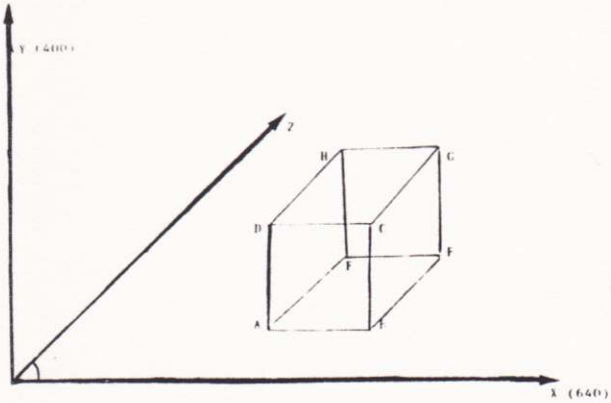
Ce chapitre ce termine ici, mais ce n'est pas tout, il vous reste bien des choses à découvrir, a voir, et à savoir sur le HBASIC.

LA TROISIEME DIMENSION

Ce chapitre à pour but d'expliquer le fonctionement des instructions du HBASIC permettant de dessiner en trois dimensions.

V.1 - LE PROBLEME

Comment dessiner en trois dimensions quand on a un écran plat soit deux dimensions ? Il faut trouver une astuce ou plutôt une formule. Qui n'a pas dessiné un cube vu en perspective dans sa jeunesse ? En deux dimensions, nous avons deux axes perpendiculaires X et Y, en trois dimensions, un autre axe Z vient s'ajouter, toujours perpendiculaire aux deux premiers, mais afin qu'il soit visible sur notre feuille de dessin, nous sommes obliger de l'incliner (vers la droite en general), l'inclinaison peut être variable et est donnée par un angle, le schéma ci-dessous nous montre ce qu'il en est avec un cube dessiné dans notre repère en trois dimensions :



Regardons la façe arrière du cube (face EFGH), la coordonée en Z à eu pour effet de la décaler par rapport à la face avabt (face ABCD), cela correspond donc à un changement de repère (la façe avant a toutes ses coordonnées en Z nules), or ce changement de repère fait intervenir un angle A, il faut donc avant de dessiner en trois dimensions connaitre cet angle. Le HBASIC possède un jeu de 9 instructions qui permettent de dessiner en trois dimensions et de faire varier cet angle de représentation. On considère que l'on est dans un repère X,Y,Z identique au précédent. Ainsi, les instructions néscéssitent un paramètre de plus pour indiquer les coordonées par rapport aux coordonnées du BASIC.

Elles sont identiques au niveau de fonctionement à celles du BASIC de l'amstrad et tiennent compte du mode de mélange, des changements d'origine et de la taille de la fenêtre graphique. Le repère considéré a une taille de 640*400 en X,Y, la taille du repère en Z dépend de l'angle de représentation. Les coordonnées doivent êtres entieres (entre -32767 et + 32767). Cela n'a pas d'importance si l'on sort de l'écran, celui-çi stopera à la limite.

V.2 - LES INSTRUCTIONS

La première que nous alons voir est |TDA (cela veut dire Tri Dimensionnal Angle), elle permet de fixer l'angle de représentation dont nous venons de parler, elle admet donc un paramètre, ainsi si nous voulons un angle de 45°, nous ferons :

```
A%=45 : |TDA,àA%
```

Le fait de changer l'angle de représentation du dessin change également le dessin, vous pourrez le constater dans le programme de démontration donné en fin de chapitre.

Les autres instructions sont au nombre de 8 et sont :

|TDRAW
|TDRAWR
|TPLOT
|TPLOTR
|TMOVE
|TMOVER
|TTEST
|TTESTR

Les six premières instructions ont le même éffet que leur homologue en BASIC (sauf le T ainsi TDRAW correspond à DRAW), pour chacune d'elle il faut donc préciser les trois coordonées et la couleur (sauf pour |TMOVE, |TMOVER, |TTEST et |TTESTR bien sur), ainsi pour tracer une ligne en 100,50,75 en couleur 1 on fera :

X%=100 : Y%=50 : Z%=75 : C%=1 : |TDRAW,àX%, àY%, àZ%, àC%

Pour les instructions de TEST (|TTEST et |TTESTR), les trois premiers paramètres donnent bien la couleur du point, mais il faut apporter certaines précisions, dans le cas d'un test, il y a passage du HBASIC vers l'utilisateur, il faut donc bien que HBASIC mette la couleur du point quelquep part, en basic in écrit :

A%= TEXT X%,Y%

En HBASIC cela n'est pas possible, il faut écrire :

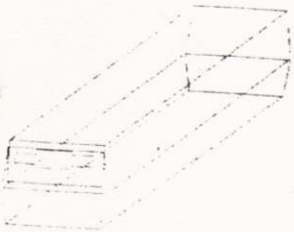
A%=0 : |TTEST,àX%, àY%, àZ%, àA%

La couleur du point se trouvera alors transférée dans la variable A%, (on ecrit A%=0 pour déclaré la variable, sinon une erreur apparaîtrait). Il en va de même pour |TTESTR. Si vous désirez de plus amples informations sur ces instructions, reportez-vous au chapitre 10 : LE JEU D'INSTRUCTION.

LA TROISIEME DIMENSION

Voiçi maintenant un exemple de programme qui dessine une unité de disquette en trois dimensions (il n'y a pas le controleur !).

```
10 FOR I=0 TO 360 STEP 40:!TDA.@I:MOVE 220,100:GOSUB 20:CLG
:NEXT I
20 RESTORE 30
30 DATA 999,40,80,0,0,0,250,95,0,0,0,0,-250,-95,0,0,-5,30,0,
0,0,250,5,-30,0,-5,30,0,105,0,0,-5,-30,0,5,30,0,0,0,-250,-5,
30,0,5,30,0,-105,0,0
40 DATA 5,40,0,0,0,250,-5,-40,0,5,40,0,95,0,0,5,-40,0,-5,40,
0,0,0,-250,5,-40,0,-5,40,0,-95,0,0
50 DATA 0,-35,0,95,0,0,0,30,0,-95,0,0
60 DATA 999,7,-7,-2,22,0,0,0,3,0,35,0,0,0,-2,0,22,0,0,0,-10,
0,-22,0,0,0,-2,0,-35,0,0,0,2,0,-22,0,0,0,10,0
70 DATA 999,-6,-35,2,900,0,0
80 DEFINT a-z:c=1
90 READ x,y,z:IF x=999 THEN 130
100 IF x=900 THEN 140
110 !TDRAWR,@x,@y,@z,@c
120 GOTO 90
130 x=y:y=z:READ z:!TMOVER,@x,@y,@z:GOTO 90
140 END
```



VI. 1 - DEFINITION

Une zone, sous entendu de l'écran , est une partie de celui-ci. Donc tous ce chapitre va être axé sur une zone. Nous allons pouvoir manipuler notre zone un peu dans tous les sens, mais cependant il convient de faire très attention dans l'utilisation de cette zone. Aussi allons nous exposer quelques notions qui vous permettrons de mieux comprendre :

A. Comme nous l'avons dit au chapitre I, la mémoire écran commence en 49152. C'est l'endroit de la mémoire ou est stocké le contenu de l'écran. Elle à une taille de 16Ko (de 49152 à 65536, fin de la mémoire). Chaque octet de cette mémoire représente un certain nombre de points à l'écran :

MODE	Nb DE POINTS PAR OCTET
0	2
1	4
2	8

B. Toutefois, quelques précautions sont à prendre, cette mémoire écran bouge, quand vous voyez l'écran monter ou descendre, la mémoire écran bouge de 8 lignes soit 640 octets. Le début de la mémoire écran a donc bougé. Mais quand nous définissons une fenêtre (WINDOW), le mouvement de l'écran est alors obtenu par un transfert d'octets, cela veut dire que le début de la mémoire écran n'a pas bougé, ainsi la plus grande fenêtre que nous pouvons définir pour que le début de la mémoire écran ne bouge pas est :

```
WINDOW §0, 1, 80, 1, 24
```

Ici, nous somme en mode 2, en mode 1 il aurait fallu écrire : WINDOW §0, 1, 40, 1, 24 ; pour tenir compte de la largeur de l'écran.

Ces deux notions sont très importantes dans l'utilisation de notre zone, en effet dans toutes les manipulations (transfert, remplissage, scrolling) le HBASIC suppose que la mémoire écran commence en 49152, si l'écran à bougé (monté ou descendu) cela n'est plus vrai.

L'instruction WINDOW permet d'éviter cela. Mais avant de faire WINDOW il faut être sur que la mémoire écran commence en 49152. Le seul moyen est d'utiliser l'instruction MODE qui réinitialise la mémoire écran, la procédure est donc la suivante :

Instruction MODE
Instruction WINDOW
Définition de la zone
Utilisation de la zone

ATTENTION : Cette procédure doit impérativement être respectée avant utilisation de la zone.

Ceci doit être fait pour chaque chqngement de mode, bien etendu ceçi ne conçerne que les instructions relatives à la zone. Voyon un exemple de chaque procédure pour chaque mode :

MODE	PROCEDURE
0	MODE 0: WINDOW §0, 1, 20, 1, 24
1	MODE 1: WINDOW §0, 1, 40, 1, 24
2	MODE 2: WINDOW §0, 1, 80, 1, 24

Puis définition et utilisation de la zone. Mais voyons maintenant comment définir et utiliser une zone.

VI.2 - LES INSTRUCTIONS

A. L'instruction |DEFZONE permet de définir une zone, cette définition s'effectue de façon un peu similaire à l'instruction WINDOW, |DEFZONE admet quatre paramètres A, B, C ,D :

A : Position la plus à gauche de la zone
B : Positon la plus à droite de la zone
C : Position la plus en haut de la zone
D : Position la plus en bas de la zone

Les positions en largeur (gauche et droite) sont donées en octets et donc varient selon le mode, en effet selon le mode un octet ne représente pas toujours le même nombre de points (CF page 29).

Nous attirons votre attention sur le fait qu'il s'agit de points physique directement liés à la résolution du mode (CF chapitre 2), donc quelque soit le mode, la largeur d'une ligne sera de 80 octets. Mais si tous cela vous semble compliqué faites des éssais, en respectant bien les précautions citées plus haut. Les positions en hauteur (haut et bas) sont donées en caractères (et varient donc de 1 à 25). Voiçi un exemple :

```
10 MODE 2
20 WINDOW §0, 1, 80, 1, 25
30 A%=20 : B%=60 : C%=50 : D%=20
40 |DEFZONE,àA%, àB%, àC%, àD%
50 E%=255 : |FILLZONE,àE%
```

Faites le marcher, puis essayez en MODE 1 ou 0 sans oublier de changer le WINDOW en ligne 20. Vous verrez que la largeur de votre ligne varie selon le mode.

B. Vous avez remarquez dans le programme précédent la ligne 50 qui a pour effet de remplir la zone dans une couleur qui varie selon le mode de l'écran. Cette instruction |FILLZONE permet donc de remplir la zone avec un octet bien précis, suivant la valeur de l'octet et du mode le résultat peut etre bien différent. Pour avoir une meilleure idée de ce que cela peut donner voyons le codage des couleurs sur l'AMSTRAD :

En mode 2 chacun des octets représente 8 points, soit un point par BIT ainsi 255 représente une ligne de 8 points allumés, la valeur 1 représente le point le plus à droite de notre ligne (essayez avec 1 puis 128 pour bien saisir la différence).

En mode 1 comme en mode 0, les choses se corsent, en effet en mode 1 un octet représente 4 points soit 2 bits pour un point, nous obtenons la répartition suivante :

POINT	BIT 1	BIT 2
1	1	16
2	2	32
3	4	64
4	8	128

Examinons ce tableau, cela veut dire que chaque couleur est codé sur 2 BITS (donc 4 valeurs différentes pour chaque couleur de 0 à 3), ainsi si nous voulons que le point 1 soit à la couleur 3 et les autres à la couleur 0 cela donnera 17 comme valeur à notre octet. (le point 1 est le point le plus à droite de nos quatre points).

Si nous voulons maintenant avoir le point 3 à la couleur 2 et le point 1 à la cour 3 cela nous donnera :

1 + 16 + 64 soit 81

Si nous voulons remplir toute notre zone en couleur 2 cela donnera :

16 + 32 + 64 + 128 Soit 240

240 sera la valeur que nous donnerons à notre octet de remplissage.

En mode 0 nous obtenons un tableau différent (seulement deux points par octets), chaque couleur est codée sur 4 BITS :

POINT	BIT 4	BIT 3	BIT 2	BIT 1
1	1	8	4	64
2	2	32	16	128

La méthode de calcul est la même qu'en mode 1. tout ceci est loin d'être simple, mais rassurez-vous même AMSTRAD reconnaît que ce n'est pas 'TRIVIAL' !!

C. L'instruction |LOGZONE permet de remplir la zone avec un octet, mais elle permet de mélanger cet octet avec ceux de la zone. Le mélange s'éffectue à l'aide d'opérateurs logique (encore eux !), il faut préciser l'opérateur logique numéroté de 1 à 3 :

```
1          XOR
2          AND
3          OR
```

Puis l'octet soit :

```
10 MODE 2
20 WINDOW §0, 1, 80, 1, 24
30 A%=20 : B%=60 : C%=5 : D%=20
40 |DEFZONE ,àA%, àB%, àC%, àD%
50 LOCATE 30,7 : PRINT "POWER SOFT"
60 A%=1 : B%=255 : |LOGZONE,àA%, àB%
```

Dans ce cas, cela a pour éffet d'inverser le contenu de la zone.

D. Nous allons voir un ensemble de 4 instructions qui permettent de 'SCROLLER' la zone dans les quatre directions (haut, bas, droite, gauche). Le meilleur moyen de comprendre le scrolling est d'éssayer :

```
10 MODE 2
20 WINDOW §0, 1, 80, 1, 25
30 LOCATE 20,5 : PRINT "       HBASIC"
40 LOCATE 20,6 : PRINT "-- THE AMSTRAD POWER --"
50 A%=20 : B%=40 : C%=5 : D%=7
60 |DEFZONE,àA%, àB%, àC%, àD%
70 |UMOVE
```

Essauyer en remplaçant |UMOVE à la ligne 70 par :

|DMOVE (scrolling vers le bas)
|LMOVE (scrolling vers la gauche)
|RMOVE (scrolling vers la droite)

E. Il serait intéressant de stocker la zone dans un endroit de la mémoire afin de pouvoir la rappeler ultérieurement. Aussi HBASIC apporte deux instructions qui permettent ceci :

|STOREZONE permet de stocker la zone à une adresse en mémoire, il fait préciser cette adresse. Mais avant tout il faut connaitre la taille de la zone en octets

Soit L la largeur de la zone en octets
Soit H la hauteur de la zone en caractères

La zone prendra : L*H*8 octets à stocker. Or ces octets doivent etres mis à un emplacement ou il ne risque pas d'être modifiés et ou ils ne risquent pas de gener le bon fonctionement de l'AMSTRAD, il y a pour cela une solution simple : Supposons que notre zone fasse 1000 octets, nous commencos par déclarer un tableau dimensionné à 500 éléments, le tableau étant composé de variables entieres :

```
DIM A%(500)
```

L'AMSTRAD aura automatiquement reservé une zone de 1000 octets en mémoire, il fait maintenant savoir ou commence la zone, tapez :

```
PRINT àA%(0)
```

Voiçi l'adresse de début de cette zone de 1000 octets, il faudra pour stocker notre zone faire :

```
10 MOE 2
20 WINDOW §0, 1, 80, 1, 24
30 A%=20 : B%=60 : C%=5 : D%=20
30 |DEFZONE,àA%, àB%, àC%, àD%
40 TZ=40*5*Z : DIM A%(TZ/2)
50 ADS=àA%(0)
60 |STOREZONE,àADS
```

Attention à ne pas faire de CLEAR par la suite, cela effacerait cette partie de la mémoire et perdrait notre zone. De plus il est hautement conseillé avant de rapeller la zone de redemander l'adresse de début de zone comme à la ligne 50 :

TZ représente la taille de la zone
ADS représente l'adresse de début de stockage de la zone.

MAis maintenant que notre zone est stocké, nous désirons peut-être rappeler cette zone, pour cela il suffit d'utiliser |RECALLZONE, il faut simplement préciser l'adresse à laquelle est stocker la zone :

Ici nous ferions : 70 M%=0 : |RECALLZONE,àM%, àADS

L'interèt d'une telle manoeuvre b'est peut être pas évidente à première vue, mais rien ne vous empêche de définir par exemple une zone, y déssiner les différentes phases d'un mouvement, de stocker chaque phase à une adresse différente, et de rappeler ensuite chaque phase de ce mouvement donnant ainsi une impression d'animation. ATTENTION : avant de faire |RECALLZONE, n'oubliez pas de réactualiser l'adresse de début de la zone qui peut avoir changer, il suffit de faire ADS%=àA%(0) (dans notre exemple précis) avec RECALLZONE.

Voyons maintenant le paramètre M, il indique le mode de mélange, il y en à 4 comme en BASIC :

0	NORMAL
1	XOR
2	AND
3	OR

Mais il est aussi possible de changer l'origine de la zone avec ORGZONE, la zone conserve la même taille seul son origine change :

```
10 MODE 2
20 WINDOZ §0, 1, 80, 1, 24
30 A%=20 : B%=30 : C%=5 : D%=10
40|DEFZONE,àA%, àB%, àC%, àD%
50 A%=255 : |FILLZONE,àA%
60 A%=40 : B%=12 : |ORGZONE,àA%, àB%
70 a%=255 : |FILLZONE,àA%
```

Les coordonées de la zone sont indiques en octets pour la première et en caractères pour la seconde, l'origine étant le coin haut à gauche de la zone. L'origine est donné par rapport au coin haut à gauche de l'écran. Ainsi, par exemple il est possible de dupliquer une zone en plusieurs endroits de l'écran.

Le HBASIC n'est pas seulement un outil graphique comme pourrait le laisser croire le chapitre précédent, il permet aussi de mieux connaitre l'AMSTRAD. Il apporte un ensemble de cinq fonctions dont la puissante fonction EXEC que nous verrons en dernier, mais voyons ces instruction.

VII.1 -LIRE

L'AMSTRAD dispose à l'origine d'une instruction permettant de lire le contenu de sa mémoire : PEEK, cette instruction permet de connaitre le contenu d'un octet à une adresse en mémoire. Mais bien souvent nous avons besoin de connaitre la valeur sur 16 bits que représente un groupe de deux octets en mémoire, aussi avons nous avec le HBASIC l'instruction |DEEK qui permet de connaitre la valeur du mot (2 bits) situé à une adresse en mémoire :

```
10 AD%=30000 : A%=0 : |DEEK,àAD%, àA%
```

La valeur du mot sera dans la variable A, il faut donc préciser dans quelle variable l'on veut récuperer la valeur et il faut que cette variable soit déclarée, avec PEEK cela aurait donné :

```
AD%=30000 : A% = PEEK(AD%) + PEEK(AD%+1)*256
```

Cela néscéssite une addition et une multiplication pour arriver au même résultat. Mais des fois, nous voulons visualiser le contenu de la mémoire en faisant ce que l'on appelle un DUMP ASCII. Le DUMP ASCII consiste à mettre dans une variable alphanumérique les caractères dont le code correspond à une valeur de la mémoire, ainsi si nous voulons voir le contenu de la mémoire comprise entre 30000 et 30020 nous ferons :

```
10 AD%=30000 : A$=STRING$(20,32)
20 |DUMP,àAD%, àA§
```

A$ contiendra les valeurs de la mémoire de 30000 à 30020 sous la forme de caractères ASCII. La longueur sur laquelle s'éffectue le DUMP est donnée par la longueùr de cette chaine qui doit être non nul, AD% etant l'adresse du DUMP. Cette instruction peut ainsi être utilisée pour chercher une suite d'octets dans la mémoire.

VII.2 - ECRIRE

Mais lire n'est pas tout, il faut aussi pouvoir modifier le contenu de la mémoire, le BASIC dispose de l'instruction POKE, le HBASIC dispose de DOKE et MOKE qui sont à DEEK et DUMP ce que POKE est à PEEK. Tout comme il est utile de connaitre la valeur d'un mot de 16 bits, il peut être utile d'écrire un mot à une adresse en mémoire :

```
10 AD%=30000 : A%=1234 : IDOKE,àAD%, àA%
```

AD% étant l'adresse ou l'on désire écrire et A% la variable contenant le mot, en BASIC cela aurait donné :

```
10 AD%=30000 : A%=1234
20 H%=INT(A %/256) : L%=A% MOD 256
30 POKE AD%,L% : POKE AD%,H%
```

Voyez la différence !!. MOKE permet d'écrire plus, et fonctionne sur le même principe que DUMP, ainsi si l'on veut écrire 'HBASIC : La puissance de l'AMSTRAD' dans la mémoire, on fera :

```
10 MEMORY 34000
20 A$= "HBASIC : La puissance de l'AMSTRAD" : AD%=34001
20 IMOKE,àAD%, àA%
```

VII.3 - EXEC

Votre AMSTRAD est géré essentiellement par un micropocésseur Z80, qui possède un language interne : le language machine. Ainsi le BASIC tout comme le HBASIC sont écrit en ASSEMBLEUR (language machine codifié). Quand vous faites MODE 2 sur votre ordinateur l'AMSTRAD recherche dans une table s'il trouve le mot MODE et si OUI, il va à une adresse mémoire correspondant à l'instruction MODE en language machine.

Ainsi, le BASIC et surtout le système d'exploitation (il gère le clavier, lécran, le son...) contiennent un tas de sous programmes appelés plus communément routines. Seulement il y à un problème : la plupart de ces routines néscéssitent des paramètres ou même retournent des valeurs, or ces paramêtres ou ces valeurs sont passées à travers les registres du Z80 (les registres sont des sortes de variables internes au Z80), il existe un ensemble de 8 registres principaux dans le Z80 :

H	L
B	C
D	E
A	F

Chaque registre représente un octet, si l'on à besoin d'une valeur sur 16 bits l'on peut associer ces registres par paire (HL, BC, DE). Le registre A est l'accumulateur qui sert de fourre tout (très souvent utilisé). Le registre F est le rigistre d'indicateurs, ces indicateurs sont les suivants :

F :	S	Z		H		P	N	C
BITS :	7	6	5	4	3	2	1	0

Et voiçi leur signification :

- S signe
- Z zéro
- H demi report
- P parité
- N non utilisé
- C retenue (CARRY)

EXEC permet d'appeler une de ces routines en communiquant directement les paramêtres dans les registres, au retour au BASIC, les variables contenant les valeurs de ces registres reviennent avec les nouvelles valeurs après execution de la routine. Exec admet 5 paramêtres qui sont l'adresse et 4 registres. Prenons un exemple concret : La routine SCR_SET_MODE, qui permet de fixer le mode de l'écran, se situe en &BCOE, le mode d'écran doit être dans le registre A, donc si nous voulons passer en mode 2 nous ferons :

```
AD%=&BCOE : HL%=0 : BC%=0 : DE%=0 : AF%=&0200
|EXEC,àAD%, àHL%, àBC%, àDE%, àAF%
```

Voilà qui est fait, plusieurs remarques s'imposent :

- EXEC admet 5 paramêtres, leur ordre est extremement important et ne saurait etre changer, de plus tous les paramêtres doivent êtres des variables entieres définies, ici nous n'avons pas besoin de HL, BC, DE, c'est pourqoui nous les mettons à zéro.

- Mais ceci n'est pas le seul usage d'EXEC, vous pouvez très bien vous en servir pour des routines que vous avez vous même programmer.

- Toutes les instructions de ce chapitres sont à utiliser avec précaution, ne modifier pas la mémoire à tort et à travers, vous risqueriez de mauvaises surprises et attention avec EXEC à ce que vous appelez en mémoire.

- Si vous désirez mieux connaitre l'AMSTRAD, nous vous conseillons de vous reporter au : CPC 464 FORMWARE. Mais malheureusement, ce livre est en Anglais (il ne tient qu'à vous de le traduire -300 PAGES-).

Ce chapitre est terminé, et maintenat MUSIQUE MAESTRO.

Quand nous voulons jouer un morceau de musique sur l'AMSTRAD, nous devons utiliser l'instruction SOUND et nous occuper de ses 7 paramêtres. Si nous voulons jouer un simple DO sur la voie A d'une longueur équivalente à celle d'une noire nous devons faire :

```
SOUND 1,478,50
```

Ne serait-il pas plus simple de faire :

```
A$="DA4":|PLAY,àA$
```

En effet, avec PLAY les instructions de ce macro-language musical sont dans une chaine de caractères alphanumérique, qui ensuite est éxécutée.

Mais revenons à notre sujet, c'est tout de même plus parlant. C'est le but de l'instruction PLAY qui en fait constitue un macro-language musical.

VIII.1 - LA NOTE

Pour jouer une note sur l'AMSTRAD, il faut connaitre trois paramêtres au minimum :

- La note
- La voie sur laquelle elle sera jouée
- La durée de cette dernière.

Avec PLAY, les notes sont désignée comme suit :

DO	D
RE	R
MI	M
FA	F
SOL	S
LA	L
SI	I

Il ne faut pas oublier le silence :

^n (n longueur du silence)

Ensuite vient le mode :

Voie A A
Voie B B
Voie C C

Puis la longueur. Ici il y à deux façons de préciser celle-çi, soit deux modes :

MODE 0

Il est activé en faisant : à0

Voiçi la correspondance entre la longueur des notes et leur valeurs :

Ronde	16
Blanche	8
Noire	4
Croche	2
Db croche	1

MODE 1

Il est activé en faisant : à1

Dans ce mode, la longueur indiquée comme longueur correspond en fait à l'envellope de volume et la durée de la note est donc celle de l'envellope.

Résumons nous : DA4

D Note
A Voie sur laquelle est jouée la note
4 Longueur de la note

La voie et la longueur sont optionelles, si elles ne sont pas précisées, HBASIC prendra la dernière voie et la dernière longueur utilisée.

De plus des formes comme :

DA4RMFSLI, DARAMA, D4R3M6 sont parfaitement autorisées

- La première joue l'octave entière sur la voie A avec une longueur de 4.
- La seconde ne précise que la voie.
- La troisième ne précise que la longueur qui change à chaque fois.

VIII.2 - AUTRE

A. Il faut pouvoir préciser l'octave sur laquelle se joue la note, sans quoi nous n'aurions une gamme composée que d'une octave.

On (n numéro de l'octave -0 à 7-)

Voilà ce qui nous fait 8 octaves, mais ce n'est pas tout :

>Permet de passer à l'octave supérieure
<Permet de passer à l'octave inférieure

B. Mais un morceau de musique peut se jouer plus ou moins fort :

Pn (n volume -0 à 15-)

Il reste le tempo et le rythme pour être vraiment complet :

C. Le tempo est la vitesse à laquelle est jouée la partition, il se règle comme suit :

Vn (n vitesse -20 normale-)

ATTENTION : ceçi n'est pas valable en MODE 1, où l'on ne peut avoir de tempo.

D. Une partition est jouée de différentes manières, coulée ou saccade, cela est réglé par le temps que l'on laisse entre chaque note, plus le temps est grand, plus la partition est jouée saccade.

En (N temps entre les notes en 100° de secondes)

E. An (n envellope de son -0 à 15, Mode 0 uniquement-)
Tn (n envellope de ton -0 à 15-)

F. Toutes ces instructions (A, T, E, V, P) admettent les espaces entre elles et leur paramêtres :

A$="A 1T 2E 3V 20P 15":|PLAY,àA$ est admis

Mais : A$="A 1 T2 E 3":|PLAY,àA$ ne l'est pas

Il ne doit pas y avoir d'espace après le paramêtre, seulement avant.

Voiçi pour finir un morçeau du BEAU DANUBE BLEU, qui vous permettra de vous familiarisez avec PLAY. Et surtout n'hésitez pas à vous lancer et expérimentez, c'est la meilleur façon d'apprendre.

```
10 DATA "@0P15E2V21A1T1C3D4DMSSB S4SBM4MB"
20 DATA "<D4DMSSB>S4SBF4FB<R4RFLLB>L4LBF4"
30 DATA "FB<R4RFLLB>L4LBM4MB<D4DMS>DB>D4"
40 DATA "DB<L4LB<R4RFLL16F+4S>M16<D4MMBR4"
50 DATA "LBS4D20"
60 FOR U=1 TO 5
70 READ A$
80 |PLAY,@A$
90 NEXT U
100 END
```

DIVERS

Ce chapitre à pour but d'expliquer les diverses instructinos du HBASIC qui ne rentraient dans aucune des sections précédentes.

IX.1 - LE CADRE

Le BASIC de l'AMSTRAD permet de faire une gestion éfficace des fenêtres, HBASIC y apporte une touche d'esthétique, en permettant de tracer des cadres à ces fenêtres ou de les enlever :

|WINBORDER permet de tracer un cadre à une fenêtre, il faut donc préciser la fenêtre. Si elle n'à pas été définie, cela tracerait un cadre dans le portour de l'écran. Voyons un exemple :

```
10 MODE 2
20 WINDOW §1, 20, 60, 5, 10
30 A%=1 : |WINBODER,àA%
```

Pour enlever le cadre tapez :

```
40 |WINNAKE,àA%
```

Signalons que si vous désirez changer la forme du cadre, il vous suffit de redéfinir les caractères employés pour tracer ce cadre :

ENDROIT	CARACTERE ASCII
Coin en haut à gauche	150
Coin en haut à droite	156
Coin en bas à gauche	147
Cotes verticales	149
Cotes horizontales	154
Coin en bas à droite	153

IX.2 - LA COPIE

Le HBASI apporte deux instructions qui permettent de garder une trace écrite de vos oeuvres :

|TEXT, envoie sur une imprimante le contenu de l'écran (uniquement les caractères). L'imprimante n'à pas d'importance, mais attention aux codes graphiques ou de contrôles (code ASCII inférieur à 32 ou supérieur à 127) qui risquent de vous créer de mauvaises surprises.

|GCOPY, execute une recopie graphique de l'écran, mais attention votre imprimante doit être un compatible EPSON en mode graphique, sans quoi vous n'obtiendrez rien de valable. Cette recopie marche quelque soit le mode. Vous en avez vu des exemples au cours de ce manuel.

ATTENTION : Votre imprimante doit être correctement branchée, en mode ON LINE pour utiliser ces deux instructions sans quoi il ne vous reste plus qu'à réinitialiser votre AMSTRAD.

IX.3 - AUTRE

Il reste encore à voir deux instruction :

|COPYCHR, permet de connaitre le code ASCII du caractère à la position courante du curseur, voiçi un exemple :

```
10 LOCATE 5,5 : PRIN "HBASIC"
20 LOCATE 5,5 : A%=0 : |COPYCHR,àA%
30 END
```

A% contiendra la valeur du code ASCII de H c'est à dire le nombre 72.

|COLSET, permet de fixer les couleurs de base, le nombre de paramêtres est indeterminé (à votre gout), ainsi :

```
10 A%=0 : B%=2 : C%=3 : D%=26
20 |COLSET,àA%, àB%, àC%, àD%
```

Ce qui donnera comme couleurs en MODE 0 ou 1

N° CRAYON	COULEUR
0	Noir
1	Bleu
2	Rouge
3	Blanc

Ce chapitre est maintenant terminé, avec lui se terminent les explications des instruction HBASIC.

LE JEU D'INSTRUCTION

Ce chapitre est un chapitre de référence ou chaque instruction du HBASIC est décrite. Ces instructions sont classées par ordre alphabétique afin de permettre une recherche plus facile et plus rapide.

Si l'emploi d'une des instructions vous pose un problème, référez vous à un des chapitres précédent ou elle sera expliquée dans son contexte.

Certaines instructions portent en bas de page :

INSTRUCTIONS ASSOCIEES :

Cela veut dire que l'emploi de ces instructions est hautement recommandé avant d'utiliser l'instruction citée.

NOTE :

Les paramêtres de toutes les instructions doivent être des variables entières.

ARC

FONCTION :

Permet de tracer un arc de cercle sur l'écran. Le curseur graphique doit être positionner sur le centre du cercle. Il faut indiquer l'angle de départ (A1), l'angle d'arrivée (A2), le rayon (R) et la couleur (C) du cercle.

ATTENTION : le rayon doit être suppérieur à 1, et il faut que l'angle d'arrivée soit suppérieur à l'angle de départ.

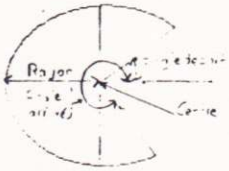
SYNTAXE :

!ARC,àA1, àA2, àR, àC

A1	Angle de départ	000 - 359
A2	Angle d'arrivée	A1 - 359
R	Rayon	000 - 640
C	Couleur	000 - 016 selon le mode

EXEMPLE :

```
10 Angledepart%=45                          'Angle de depart
20 Anglearrive%=300                         'Angle d'arrivee
30 Rayon%=100                               'Rayon du cercle
40 Couleur%=1                               'Couleur du trace
50 Xcentre%=320:Ycentre%=100                'Centre du cercle
60 MOVE Xcentre%,Ycentre%                   'Curseur graphique sur le centre
70 '
90 !ARC,@Angledepart%,@anglearrive%,@rayon%,@couleur%   'Trace de l'arc
```



AXES

FONCTION :

Permet de tracer un axe gradué sur l'écran. Il faut préciser l'origine de l'axe, la graduation en X, en Y et la coueleur de trace de ce dernier.

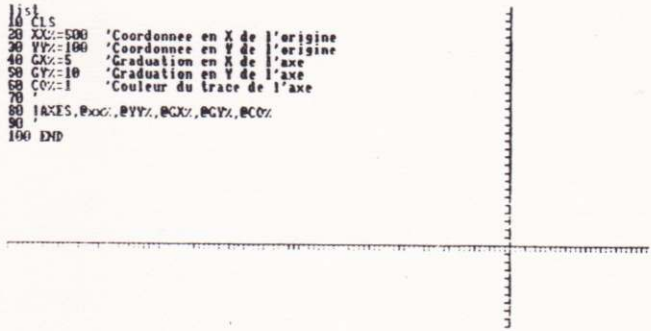
SYNTAXE :

!AXES,àXX, àYY, àGX, àGY, àCO

XX	Coordonnée en X de l'origne de l'axe	
YY	Coordonnée en Y de l'origine de l'axe	
GX	Graduation de l'axe en X	001 - 255
GY	Graduation de l'axe en Y	001 - 255
CO	Couleur de l'axe	

EXEMPLE :

```
list
10 CLS
20 XX%=500  'Coordonnee en X de l'origine
30 YY%=100  'Coordonnee en Y de l'origine
40 GX%=5    'Graduation en X de l'axe
50 GY%=10   'Graduation en Y de l'axe
60 CO%=1    'Couleur du trace de l'axe
70 '
80 !AXES,@XX%,@YY%,@GX%,@GY%,@CO%
90 '
100 END
```



BOX

FONCTION :

Permet de tracer un rectangle sur l'écran. Le curseur graphique doit être positionnr sur la base du rectangle. Il faut indiquer la largeur (LA), la hauteur (HA) et la couleur (C) du rectangle.

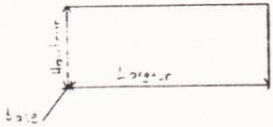
SYNTAXE :

!BOX,àLA, àHA, àC

LA	Largeur du rectangle	001 - 640
HA	Hauteur du rectangle	001 - 400
C	Couleur du rectangle	000 - 016 dépend du mode

EXEMPLE :

```
10 Largeur%=200                          'Largeur du rectangle
20 Hauteur%=100                          'Hauteur du rectangle
30 Couleur%=1                            'Couleur du trace
40 Xbase%=320:Ybase%=100                 'Base du rectangle
50 MOVE Xbase%,ybase%                    'Curseur graphique a la base
60 !BOX,@Largeur%,@Hauteur%,@Couleur%    'Trace du rectangle
```



CIRCLE

FONCTION :

Permet de tracer un cercle sur l'écran. Le cuseur graphique doit être positionner sur le centre du cercle. Il faut indiquer le rayon (R) et la couleur (C) du cercle.

ATTENTION : Le rayon doit être suppérieur à 1.

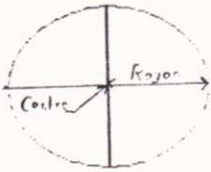
SYNTAXE :

|CIRCLE,àR, àC

R	Rayon du cercle	001 - 640
C	Couleur du cercle	000 - 016 dépend du mode

EXEMPLE :

```
10 RayonX=100                          'Rayon du cercle
20 CouleurX=1                          'Couleur du cercle
30 XcentreX=320:YcentreX=100           'Centre du cercle
40 MOVE XcentreX,YcentreX              'Curseur graphique sur le centre
50 |CIRCLE,@RayonX,@CouleurX           'Trace du cercle
```



COLSET

FONCTION :

Permet de fixer le jeu de couleur de base en mode 2 ou en mode 1 (en mode 0 permet de fixer les deux ou quatres premières couleurs). Les couleurs données correspondent aux INK(S) dans l'ordre des paramêtres.

SYNTAXE :

|COLSET,àC0, àC1 (,àC2, àC3)

C0	Couleur encre 0	INK 0,C0
C1	Couleur encre 1	INK 1,C1
C2	Couleur encre 2	INK 2,C2 mode 1
C3	Couleur encre 3	INK 3,C3 mode 1

EXEMPLE :

```
10 CFOND%=2      'Couleur de fond bleu foncé
20 CPAPIER%=26   'Couleur d'écriture blanche
30 |COLSET,àCFOND%, àCPAPIER%
40 END
```

COPYCHR

FONCTION :

Permet de connaitre le code ASCII du caractère qui à la position actuelle du curseur texte. Il faut indiquer la variable qui contiendra le code ASCII.

SYNTAXE :

|COPYCHR,àVA

VA Variable qui contiendra le code ASCII

EXEMPLE :

```
10 VA%=0                    'Initialisation variable
20 |COPYCHR,àVA%
30 END
```

DEEK

FONCTION :

Permet un accès direct à la mémoire. Permet de connaitre la valeur 16 bits (-32767, 32767) à une adresse précise. Il faut donner l'adresse et la variable qui recevra la valeur.

SYNTAXE :

```
IDEEK,àAD, àVA
```

AD Adresse à lire
VA Variable qui recevra la valeur

EXEMPLE :

```
10 AD%=30000            'Adresse
20 VA%=0                'Variable
30 IDEEK,àAD%, àVA%
40 END
```

DEFZONE

FONCTION :

Permet de définir une zone sur l'écran pour un usage ultérieur. Il faut préciser les limites de la zone. Elles se définissent de la même façon que pour les fenêtres.

ATTENTION : Pour le mode 1, il faut multiplier la largeur par 2 et par 4 pour le mode 0.

SYNTAXE :

```
|DEFZONE,àPG, àPD, àPH, àPB
```

PG	Point le plus à gauche de la zone
PD	Point le plus à droite de la zone
PH	Point le plus en haut de la zone
PB	Point le plus bas de la zone

EXEMPLE :

```
10 PG%=5                    'Point gauche
20 PD%=20                   'Point droite
30 PH%=5                    'Point haut
40 PB%=10                   'Point bas
50 |DEFZONE,àPG%, àPD%, àPH%, àPD%
60 END
```

DMOVE

FONCTION :

Permet de scroller vers le bas une zone précédemment définié par DEFZONE.

ATTENTION : La zone doit avoir été définie par DEFZONE.

SYNTAXE :

|DMOVE

EXEMPLE :

```
10 PG%=5                    'Point gauche
20 PD%=20                   'Point droite
30 PH%=5                    'Point haut
40 PB%=10                   'Point bas
50 |DEFZONE,àPG%, àPD%, àPH%, àPB%
60 |DMOVE
70 END
```

INSTRUCTIONS ASSOCIEES :

DEFZONE (obligatoire)

DOKE

FONCTION :

Permet un accès direct à la mémoire. Permet de poker (d'écrire) une valeur de 16 bits (-32767, 32767) à une adresse précise. Il faut donner l'adresse et la valeur.

SYNTAXE :

|DOKE,àAD, àVA

AD Adresse à doker
VA Valeur à doker

EXEMPLE :

```
10 AD%=30000            'Adresse
20 VA%=-10000           'Valeur
30 |DOKE,àAD%, àVA%
40 END
```

DOUBLEZONE

FONCTION :

Permet de doubler en hauteur la zone précédemment définie par DEFZONE.

ATTENTION : La zone doit avoir été définie par DEFZONE.

SYNTAXE :

|DOUBLEZONE

EXEMPLE :

```
10 PG%=5                    'Point gauche
20 PD%=20                   'Point droite
30 PH%=5                    'Point haut
40 PB%=10                   'Point bas
50 |DEFZONE,àPG%, àPD%, àPH%, àPB%
60 |DOUBLEZONE
70 END
```

INSTRUCTIONS ASSOCIEES :

DEFZONE (obligatoire)

DUMP

FONCTION :

Permet de faire un DUMP ASCII à une adresse en mémoire. Le dump sera mis dans une variable alphanumérique. La longueur du dump est celle de la variable alphanumérique.

SYNTAXE :

```
|DUMP,àAD, àVA$
```

AD	Adresse à regarder (lire)
VA$	Variable qui contiendra le DUMP

EXEMPLE :

```
10 AD%=30000                    'Adresse
20 DUMP$=STRING$(100,32)        'Dump sur 100 octets
30 |DUMP,àAD%, àDUMP$
40 END
```

ELLIPSE

FONCTION :

Permet de tracer une ellipse sur l'écran. Le curseur graphique doit être positionner sur le centre de l'éllipse. Il faut indiquer le rayon suivant l'axe des X (RX), le rayon suivant l'axe des Y (RY) et la couleur de l'ellipse (C).

ATTENTION : Les rayons RX et RY doivent être supérieurs à 1.

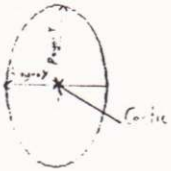
SYNTAXE :

|ELLIPSE,àRX, àRY, àC

RX	Rayon selon l'axe des X	001 - 640
RY	Rayon selon l'axe des Y	001 - 640
C	Couleur de trace	001 - 016 dépend du mode

EXEMPLE :

```
10 RayonX%=50                          'Rayon en X de l'ellipse
20 RayonY%=100                         'Rayon en Y de l'ellipse
30 Couleur%=1                          'Couleur du trace
40 Xcentre%=320:Ycentre%=100           'Centre de l'ellipse
50 MOVE Xcentre%,Ycentre%              'Curseur graphique sur le centre
60 |ELLIPSE,@RayonX%,@RayonY%,@Couleur% 'Trace de l'ellipse
```



EXEC

FONCTION :

Permet d'appeler un programme en language machine et d'y passer la valeur des différents registres du Z80 lors de l'execution. Au retour au BASIC les variables contiendront les valeurs des registres du Z80.

SYNTAXE :

```
|EXEC,àAD, àHL, àBC, àDE, àAF
```

AD	Adresse à appeler
HL	Valeur du registre HL
BC	Valeur du registre BC
DE	Valeur du registre DE
AF	Valeur du registre AF

EXEMPLE :

```
10 SCRSETMODE%=&HBC0E          'Adresse à appeler
20 HL%=0                       'Registre HL
30 BC%=0                       'Registre BC
40 DE%=0                       'Registre DE
50 AF%=&H0100                  'Registres A et F
60 |EXEC,àSCRSETMODE%, àHL%, àBC%, àDE%, àAF%
70 END
```

FILL

FONCTION :

Permet de remplir une zone graphique avec une couleur précise. Il faut préciser la couleur de remplissage et la couleur à remplir.

ATTENTION : La zone doit être entièrement délimité par un trace d'une couleur différente de la couleur à remplir.

SYNTAXE :

|FILL,àCF, àCR

CF Couleur à remplir
CR Couleur de remplissage

EXEMPLE :

```
10 LA%=50                        'Largeur
20 HA%=100                       'Longueur
30 CO%=1                         'Couleur
40 MOVE 320,200                  'Curseur au centre
50 |BOX,àLA%, àHA%, àCO%         'Rectangle
60 COLAREMPLIR%=0                'Couleur à remplir
70 COLREMPLISSAGE%=1             'Couleur de remplissage
80 |FILL,àCOLAREMPLIR%, àCOLREMPLISSAGE%
90 END
```

FILLZONE

FONCTION :

Permet de remplir un zone précédemment définie par DEFZONE avec un octet. Il faut préciser ce dernier.

ATTENTION : La zone doit avoir été définie par DEFZONE.

SYNTAXE :

|FILLZONE,àMO

MO Octet de remplissage

EXEMPLE :

```
10 PG%=5                          'Point gauche
20 PD%=20                         'Point droit
30 PH%=5                          'Point haut
40 PB%=10                         'Point bas
50 |DEFZONE,àPG%, àPD%, àPH%, àPB%
60 OCTET%=255                     'Octet de remplissage
70 |FILLZONE,àOCTET%
80 END
```

GCOPY

FONCTION :

Permet de faire une recopie d'écran haute résolution sur une imprimante EPSON (ou compatible). La copie marche quelque soit le mode.

ATTENTION : Cela ne marche en aucun cas avec l'imprimante AMSTRAD DMP-1.

SYNTAXE :

|GCOPY

EXEMPLE :

```
10 |GCOPY
20 END
```

GAND GNORMAL GOR GXOR

FONCTION :

Met en mode d'encre graphique AND, NORMAL ou OR, ou XOR.

SYNTAXE :

|GAND

|GNORMAL

|GOR

|GXOR

EXEMPLE :

```
10 |GAND
20 END

10 |GNORMAL
20 END

10 |GOR
20 END

10 |GXOR
20 END
```

GRAPHICPAPER

FONCTION :

Met la couleur de fond pour la fenêtre grahique.

SYNTAXE :

|GRAPHICPAPER,àCO

CO Couleur de fond de la fenêtre grahique

EXEMPLE :

```
10 COULEUR%=1                    'Couleur de fond
20 |GRAPHICPAPER,àCOULEUR%
30 END
```

GRAPHICPEN

FONCTION :

Met le crayon graphique (couleur des traces) à une couleur précise.

SYNTAXE :

|GRAPHICPEN,àCO

CO Couleur de traçe

EXEMPLE :

```
10 COULEUR%=1                    'Couleur de traçe
20 |GRAPHICPEN,àCOULEUR%
30 END
```

JOYMOVE

FONCTION :

Permet de déplacer un caractère à l'écran à l'aide du JOYSTICK (manche à balai). Le déplacement s'éffectue avec un pas donné. Quand on appuie sur le bouton feu, le déplacement s'arrête. Il faut préciser le caractêre de déplacement. La position du curseur graphique indiquera ou le caractère devra être en premier.

SYNTAXE :

|JOYMOVE,àCA, àDE

CA	Caractère à déplacer	
DE	Pas de déplacement	001 - 255

EXEMPLE :

```
10 CA%=ASC("+")                 'Caractère à déplacer
20 PAS%=8                       'Pas du déplacement
30 MOVE 320,200                 'Init. du déplacement
40 |JOYMOVE,àCA%, àPAS%
50 END
```

LMOVE

FONCTION :

Permet de scroller vers la gauche un zone précédemment définie par DEFZONE.

ATTENTION : La zone doit avoir été définie par DEFZONE

SYNTAXE :

```
|LMOVE
```

EXEMPLE :

```
10 PG%=5                          'Point gauche
20 PD%=20                         'Point droite
30 PH%=5                          'Point haut
40 PB%=10                         'Point bas
50 |DEFZONE,àPG%, àPD%, àPH%, àPB%
60 |LMOVE
70 END
```

INSTRUCTIONS ASSOCIEES :

DEFZONE (obligatoire)

LOGZONE

FONCTION :

Permet de remplir une zone précédemment définie par DEFZONE avec un octet conjointement avec un opérateur logique. Il faut préciser l'opérateur (OP) et l'octet de remplissage (MO).

Opérateurs logiques : 1 - XOR 2 - OR 3 - AND

ATTENTION : La zone doit avoir été définie par DEFZONE

SYNTAXE :

|LOGZONE,àOP, àMO

OP	Numéro de l'opérateur logique	001 - 003
MO	Octet de remplissage	000 - 255

EXEMPLE :

```
10 PG%=5·                          'Point gauche
20 PD%=20                          'Point droite
30 PH%=5                           'Point haut
40 PB%=10                          'Point bas
50 |DEFZONE,àPG%, àPD%, àPH%, àPB%
60 OCTET%=255                      'Octet de remplissage
70 OPERATEUR%=1                    'Opérateur logique XOR
80 |LOGZONE,àOPERATEUR%, àOCTET%
90 END
```

INSTRUCTIONS ASSOCIEES :

DEFZONE (obligatoire)

MOKE

FONCTION :

Permet de POKER (écrire) une chaine ASCII à un emplacement de la mémoire. Il faut préciser l'adresse (AD) et la chaine (A$).

SYNTAXE :

!MOKE,àAD, àA$

AD Adresse ou l'on va moker
A$ Variable que l'on va moker

EXEMPLE :

```
10 ADRESSE%=30000              'Adresse à moker
20 CHAINE$="POWERSOFT"         'Chaine à moker
30 !MOKE,àADRESSE%, àCHAINE$
40 END
```

ORGZONE

FONCTION :

Permet de changer l'origine de la zone précédemment définie par DEFZONE pour un usage ultérieur. Il faut la nouvelle origine de la zone.

ATTENTION : La zone doit avoir été définie par DEFZONE

SYNTAXE :

|ORGZONE,àXX, àYY

XX	Coordonnée en X de la zone	001 - 080 dépend du mode
YY	Coordonnée en Y de la zone	001 - 025

EXEMPLE :

```
10 PG%=5                          'Point gauche
20 PD%=20                         'Point droite
30 PH%=5                          'Point haut
40 PB%=10                         'Point bas
50 |DEFZONE,àPG%, àPD%, àPH%, àPB%
60 XZONE%=10                      'Colonne 10
70 YZONE%=5                       'Rangée 5
80 |ORGZONE,àXZONE%, àYZONE%
90 END
```

INSTRUCTIONS ASSOCIEES :

DEFZONE (obligatoire)

PLAY

FONCTION :

Permet de jouer de la musique avec les trois voies de l'AMSTRAD. Le morceau de musique est introduit dans une chaine de caractères et est executé ensuite. Les notations utilisés sont les suivantes :

Notes :	Voies :	Longueur :
Do D Ré R Mi M Fa F Sol S La L Si I	Voie 1 A Voie 2 B Voie 3 C	Ronde 16 Blanche 8 Noire 4 Croche 2 DD croche 1
Alterations :	**Octaves :**	**Volume :**
Dièse + Bémol -	On (n octave 0-7) > octave supérieure < octave inférieure	Vn (n volume 1-15)
Ecart note :	**Envellopes :**	**Soupirs :**
En (durée en 100° de secondes)	Amplitude An (n env. ampli.) Ton Tn (n env. ton)	Sn (n durée du soupir)

SYNTAXE :

|PLAY,&A$

A$ Variable contenant la musique

PLAY

EXEMPLE :

```
 10 A$="P15"                    'Volume maximum
 20 A$=A$+"O3"                  '3° octave
 30 A$=A$+"E2"                  '2/100 entre chaque notes
 40 A$=A$+"V20"                 'Vitesse normale
 50 A$=A$+"DA4"                 'Do noir sur voie 1
 60 A$=A$+"R+"                  'Ré Dièse
 70 A$=A$+"MFSLI"               'Octave
 80 A$=A$+" DRMFSLI"            'Octave supérieure
 90 !PLAY,àA$                   'Musique maestro
100 END
```

RECALLZONE

FONCTION :

Permet de rappeler une zone précédemment définie par DEFZONE. La zone est mélangé à l'écran selon 4 modes différents. il faut préciser l'adresse ou à été stocké la zone (AD) et le mode de mélange (MM).

Modes de mélange : 0 - Normal 1- Xor 2 - Or 3 - And

ATTENTION : La zone doit avoir été définie par DEFZONE

SYNTAXE :

|RECALLZONE,àMM, àAD

MM	Mode de mélange	000 - 003
AD	Adresese ou est stockée la zone.	

EXEMPLE :

```
10 PG%=5                          'Point gauche
20 PD%=20                         'Point droit
30 PH%=5                          'Point haut
40 PB%=10                         'Point bas
50 |DEFZONE,àPG%, àPD%, àPH%, àPB%
60 MODE%=1                        'Mode XOR
70 ADRESSE%=30000                 'Adresse de lecture
80 |RECALLZONE,àMODE%, àADRESSE%
90 END
```

INSTRUCTIONS ASSOCIEES :

DEFZONE (obligatoire)
STOREZONE

RMOVE

FONCTION :

Permet de scroller vers la droite une zone précédemment définie par DEFZONE.

ATTENTION : La zone doit avoir été définie par DEFZONE

SYNTAXE :

|RMOVE

EXEMPLE :

```
10 PG%=5                          'Point gauche
20 PD%=20                         'Point droite
30 PH%=5                          'Point haut
40 PB%=10                         'Point bas
50 |DEFZONE,àPG%, àPD%, àPH%, àPB%
60 |RMOVE
70 END
```

INSTRUCTIONS ASSOCIEES :

DEFZONE (obligatoire)

STOREZONE

FONCTION :

Permet de stocker une zone précédemment définie par DEFZONE à un emplacement mémoire. Il faut indiquer cet emplacement mémoire.

ATTENTION : La zone doit avoir été définie par DEFZONE.

SYNTAXE :

```
|STOREZONE,àAD
```

AD Emplacement mémoire ou sera stocker la zone

EXEMPLE :

```
10 PG%=5                          'Point gauche
20 PD%=20                         'Point droite
30 PH%=5                          'Point haut
40 PB%=10                         'Point bas
50 |DEFZONE,àPG%, àPD%, àPH%, àPB%
60 MEMORY 30501
70 EMPLACEMENT%=30501
80 |STOREZONE,àEMPLACEMENT%
90 END
```

INSTRUCTIONS ASSOCIEES :

STOREZONE (obligatoire)

TDA

FONCTION :

Fixe l'angle de représentation pour les traces en trois dimensions.

SYNTAXE :

|TDA,àAN

AN	Angle de représentation	000 - 360

EXEMPLE :

```
10 ANGLE%=45                    'Angle de représentation
20 |TDA,àANGLE%                 'Mémorisation de l'angle
30 END
```

TEXT

FONCTION :

Permet de faire une recopie d'écran TEXT. C'est à dire que seul les caractères reconnus seront imprimés. Cela fonctionne quelque soit l'imprimante.

ATTENTION : Il ne faut pas qu'il y ait de caractères de code ASCII supérieur à 127 sur l'écran.

SYNTAXE :

```
!TEXT
```

EXEMPLE :

```
10 ! TEXT
20 END
```

TDRAW

FONCTION :

Permet de tracer une ligne en trois dimensions. Il faut préciser les coordonnées d'arrivée (X, Y, Z) et la couleur du trace (C).

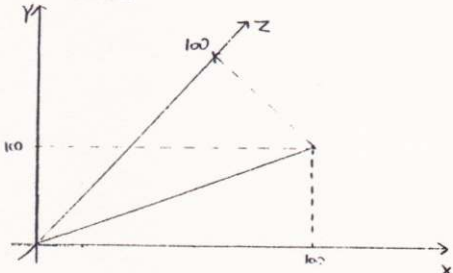
SYNTAXE :

|TDRAW,àX, àY, àZ, àC

X	Coordonnée selon l'axe des X
Y	Coordonnée selon l'axe des Y
Z	Coordonnée selon l'axe des Z
C	Couleur du trace

EXEMPLE :

```
10 MOVE 0,0                    'Retour a l'origine
20 X%=100                      'Coordonnee en X
30 Y%=100                      'Coordonnee en Y
40 Z%=100                      'Coordonnee en Z
50 Couleur=1                   'Couleur du trace
60 '
70 |TDRAW,@X%,@Y%,@Z%,Couleur%      'Trace
80 '
90 END
```



TDRAWR

FONCTION :

Permet de tracer une ligne dont les coordonnées sont relatives au dernier point allumé sur l'écran en trois dimensions. Il faut préciser les coordonnées relatives (RX, RY, RZ) et la couleur de trace (C).

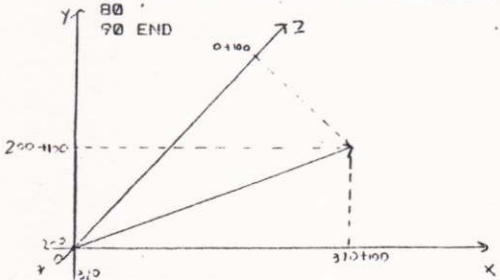
SYNTAXE :

|TDRAWR,àRX, àRY, àRZ, àC

RX	Coordonnée relative selon l'axe des X
RY	Coordonnée relative selon l'axe des Y
RZ	Coordonnée relative selon l'axe des Z
C	Couleur de trace

EXEMPLE :

```
10 MOVE 320.200          'Deplacement au centre de
l'ecran
20 RX%=100               'Coordonnee en X
30 RY%=100               'Coordonnee en Y
40 RZ%=100               'Coordonnee en Z
50 Couleur=1             'Couleur du trace
60 '
70 |TDRAWR.@RX%.RY%.@RZ%.Couleur%      'Trace
80 '
90 END
```



TMOVE

FONCTION :

Permet de déplacer le curseur graphique à une position en trois dimensions, il faut préciser les coordonnées d'arrivée (X, Y, Z).

SYNTAXE :

```
|TMOVE,àX, àY, àZ
```

X	Coordonnée selon l'axe des X
Y	Coordonnée selon l'axe des Y
Z	Coordonnée selon l'axe des Z

EXEMPLE :

```
10 X%=100                    'Coordonnée en X
20 Y%=100                    'Coordonnée en Y
30 Z%=100                    'Coordonnée en Z
40 |TMOVE,àX, àY, àZ
50 END
```

TMOVER

FONCTION :

Permet de déplacer le curseur graphique à une position dont les coordonnées sont relatives au dernier point allumé sur l'écran en trois dimensions. Il faut préciser les coordonnées relatives (RX, RY, RZ).

SYNTAXE :

```
!TMOVER,àRX, àRY, àRZ
```

RX	Coordonnée relative selon l'axe des X
RY	Coordonnée relative selon l'axe des Y
RZ	Coordonnée relative selon l'axe des Z

EXEMPLE :

```
10 RX%=100                          'Coordonnée en X
20 RY%=100                          'Coordonnée en Y
30 RZ%=100                          'Coordonnée en Z
40 !TMOVER,àRX%, àRY%, àRZ%
50 END
```

TPLOT

FONCTION :

Permet d'allumer un point dont les coordonnées sont en trois dimensions. Il faut préciser les coordonnées relatives (X,Y,Z) et la couleur du point (C).

SYNTAXE :

```
|TPLOT,àX, àY, àZ, àC
```

X	Coordonnée selon l'axe des X
Y	Coordonnée selon l'axe des Y
Z	Coordonnée selon l'axe des Y
C	Couleur du point

EXEMPLE :

```
10 MOVE 0,0                    'Déplacement à l'origine
20 X%=100                      'Coordonnée en X
30 Y%=100                      'Coordonnée en Y
40 Z%=100                      'Coordonnée en Y
50 COLEUR%=1                   'Couleur du point
60 |TPLOT,àX%, àY%, àZ%, àCOULEUR%
70 END
```

TPLOTR

FONCTION :

Permet d'allumer un point dont les coordonnées sont relatives au dernier point allumé sur l'écran en trois dimension. Il faut préciser les coordonnées relatives (RX, RY, TZ) et la couleur du point (C).

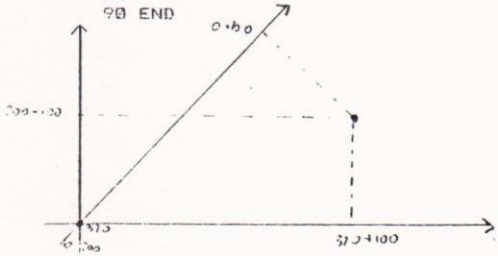
SYNTAXE :

```
|TPLOTR,àRX, àRY, àRZ, àC
```

RX	Coordonnée relative selon l'axe des X
RY	Coordonnée relative selon l'axe des Y
RZ	Coordonnée relative selon l'axe des Z
C	COuleur du point

EXEMPLE :

```
10 MOVE 320.200          'Deplacement au centre de
l'ecran
20 RX%=100               'Coordonnee en X
30 RY%=100               'Coordonnee en Y
40 RZ%=100               'Coordonnee en Z
50 Couleur=1             'Couleur du point
60 '
70 |TPLOTR,@RX%,RY%,@RZ%,Couleur%      'Allumage
80 '
90 END
```



TTEST

FONCTION :

Permet de tester la couleur d'un point dont les coordonnées sont en trois dimensions. Il faut préciser les coordonnées (X, Y, Z) et la variable qui contiendra la couleur du point (C).

SYNTAXE :

!TTEST,àX, àY, àZ, àC

X Coordonnée selon l'axe des X
Y Coordonnée selon l'axe des Y
Z Coordonnée selon l'axe des Z
C Variable contenant la couleur du point

EXEMPLE :

```
10 MOVE 0,0                    'Déplacement à l'origine
20 X%=100                      'Coordonnée en X
30 Y%=100                      'Coordonnée en Y
40 Z%=100                      'Coordonnéé en Z
50 VARIABLE%=0                 'Initialisation de la
variable
60 !TTEST,àX, àY, àZ, àVARIABLE%      'Test
70 END
```

TTESTR

FONCTION :

Permet de tester un point dont les coordonnées sont relatives au dernier point allumé sur l'écran en trois dimensions. Il faut préciser les coordonnées relatives (RX, RY, RZ) et la variable dans laquelle sera mise la couleur du point (C).

SYNTAXE :

```
!TTESTR,àRX, àRY, àRZ, àC
```

RX	Coordonnée relative selon l'axe des X
RY	Coordonnée relative selon l'axe des Y
RZ	Coordonnée relative selon l'axe des Z
C	Variable qui contiendra la couleur du point

EXEMPLE :

```
10 MOVE 320;200                    'Déplacement au centre
20 RX%=100                         'Coordonnée en X
30 RY%=100                         'Coordonnée en Y
40 RZ%=100                         'Coordonnée en Z
50 VARIABLE%=0                     'Initialisation de la
variable
60 !TTESTR,àRX%, àRY%, àRZ%, àVARIABLE%    'Test
70 END
```

UMOVE

FONCTION :

Permet de scroller vers le haut une zone précédemment définie par DEFZONE.

ATTENTION : La zone doit avoir été définie par DEFZONE

SYNTAXE :

|UMOVE

EXEMPLE :

```
10 PG%=5                          'Point gauche
20 PD%=20                         'Point droite
30 PH%=5                          'Point haut
40 PD%=10                         'Point bas
50 |DEFZONE,àPG%, àPD%, àPH%, àPB%
60 |UMOVE
70 END
```

INSTRUCTIONS ASSOCIEES :

DEFZONE (obligatoire)

WINBORDER

FONCTION :

Permet de tracer un cadre à une fenêtre. Il suffit d'indiquer le numéro de la fenêtre.

SYNTAXE :

!WINBORDER,àFE

FE N° de la fenêtre à encadrer

EXEMPLE :

```
10 WINDOW §1, 10, 20, 5 ,15    'Création fenêtre
20 FENETRE%=1                  'N° fenêtre à encadrer
30 !WINBORDER,àFENETRE%
40 END
```

WINNAKE

FONCTION :

Permet d'enlever un cadre à une fenêtre. Il suffit d'indiquer le numéro de fenêtre. Si la fenêtre n'avait pas de cadre, la fonction n'à aucun effet.

SYNTAXE :

!WINNAKE,àFE

FE N° fenêtre à désencadrer

EXEMPLE :

```
10 WINDOW §1, 10, 20, 5, 15    'Création fenêtre
20 FENETRE%=1                  'N° fenêtre à désencadrer
30 !WINNAKE,àFENETRE%
40 END
```

INSTRUCTIONS ASSOCIEES :

WINBORDER

ZOOM

FONCTION :

Permet d'agrandir une zone de l'écran. L'endroit de l'agrandissement peut être sur la zoneà agrandir ou à un autre endroit de l'écran. De plus, il peut y avoir un caractère de remplissage. Il faut préciser la taille de la zone à agrandir. les facteurs d'agrandissement en X et en Y, eventuellement les coordonnées de l'agrandissement et le caractère de remplissage (s'il est nul, il n'est pas pris en compte). Le curseur graphique doit être positionné à la base de la zone à agrandir (coin en bas à gauche).

SYNTAXE :

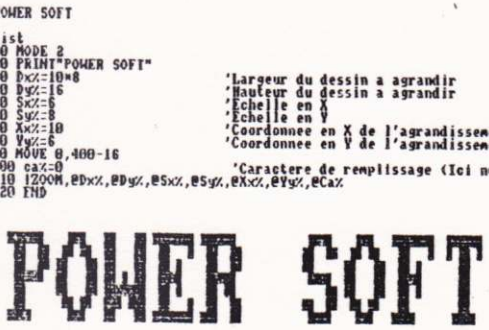
!ZOOM,àDX, àDY, àSX, àSY (,àXX, àYY), àCA

DX	Largeur de la zone à agrandir	001 - 255
DY	Hauteur de la zone à agrandir	001 - 255
SX	Facteur d'agrandissement en X	001 - 255
SY	Facteur d'agrandissement en Y	001 - 255
XX	Coordonnée en X de l'agrandissement (Facultatif)	
YY	Coordonnéé en Y de l'agrandissement (Facultatif)	
CA	Caractère de remplissage (Nom considéré si nul)	

EXEMPLE :

```
POWER SOFT

list
10 MODE 2
20 PRINT"POWER SOFT"
30 Dx%=10*8                    'Largeur du dessin a agrandir
40 Dy%=16                      'Hauteur du dessin a agrandir
50 Sx%=6                       'Echelle en X
60 Sy%=8                       'Echelle en Y
70 Xx%=10                      'Coordonnee en X de l'agrandissement
80 Yy%=6                       'Coordonnee en Y de l'agrandissement
90 MOVE 0,400-16
100 ca%=0                       'Caractere de remplissage (Ici nul)
110 !ZOOM,@Dx%,@Dy%,@Sx%,@Sy%,@Xx%,@Yy%,@Ca%
120 END
```



ANNEXE A : LISTE DES INSTRUCTIONS

ARC, AXES

BOX

CIRCLE, COLSET, COPYCHR

DEEK, DEFZONE, DMOVE, DOKE, DOUBLEZONE, DUMP

ELLIPSE, EXEC

FILL, FILLZONE

GAND, GCOPY, GNORMAL, GOR, GXOR, GRAPHICPAPER, GRAPHICPEN

JOYMOVE

LMOVE, LOGZONE

MOKE

ORGZONE

PLAY

RECALLZONE, RMOVE

STOREZONE

TDA, TEXT, TDRAW, TDRAWR, TMOVE, TMOVER, TPLOT, TPLOTR, TTEST, TTESTR

UMOVE

WINBORDER, WINNAKE

ZOOM

ANNEXE B : PROGRAMME DE DEMONSTRATION

```
10 MEMORY 31000:LOAD "basic.bin":CALL &9864
15 GOTO 870
20 DEFINT a-z
30 CLS:MODE 1:INK 0,2:BORDER 2:INK 1,26:INK 2,3:INK 3,3,15:SPEED INK 2
,1
40 !GNORMAL
50 a=1:b=13:c=2:!DEFZONE,@a,@b,@a,@c:PEN 2:PRINT"HBASIC":!DOUBLEZONE

60 LOCATE 7,2:PEN 1:PRINT" est une extension au BASIC de";"
l'AMSTRAD"
70 PRINT"

              Il apporte ":PEN 3:PRINT"


       48 nouvelles fonctions":PEN 1
80 a=15:b=59:c=13:d=15:!DEFZONE,@a,@b,@c,@d:!DOUBLEZONE:!DOUBLEZONE:LO
CATE 30,24:PRINT"======>";:GOSUB 10050
90 GOSUB 10000:PRINT"

L'on peut tracer des cercles:":b=1:FOR r=10 TO 80 STEP 10:MOVE 100,100
:!CIRCLE,@r,@b:NEXT r:PRINT"                    des portions de cercle"
100 MOVE 260,140:r=40:a=0:b=180:c=1:!ARC,@a,@b,@r,@c:MOVE 340,140:!ARC
,@a,@b,@r,@c:MOVE 380,140:r=160:a=180:b=240:!ARC,@a,@b,@r,@c:MOVE 220,
140:a=300:b=360:!ARC,@a,@b,@r,@c
110 PRINT"                    des ellipses"
120 INK 2,3:INK 3,12:e=0:r=100:FOR u=1 TO 10:MOVE 520,100:r2=u*10:c=e+
2:!ELLIPSE,@r,@r2,@c:e=e XOR 1:NEXT u
130 e=0:r=190:FOR u=1 TO 10:MOVE 520,100:r2=u*10:c=e+2:!ELLIPSE,@r2,@r
,@c:e=e XOR 1:NEXT u
140 FOR u=1 TO 10:INK 2,12:INK 3,3:FOR x=1 TO 200:NEXT x:INK 2,3:INK 3
,12:FOR x=1 TO 200:NEXT x:NEXT u
150 GOSUB 10000:PRINT"

Ou des rectangles:"
160 MOVE 300,200:e=0:b=10:d=6:FOR u=1 TO 80:MOVER -2,-2:c=e+2:!BOX,@b,
@d,@c:e=e XOR 1:b=b+4:d=d+4:NEXT u
170 FOR u=1 TO 15:INK 2,12:INK 3,3:FOR x=1 TO 100:NEXT x:INK 2,3:INK 3
,12:FOR x=1 TO 100:NEXT x:NEXT u
180 MODE 2:PRINT"L'on a aussi acces a la 3eme dimension":PRINT"Une sim
ple commande (TDA) permet de changer l'angle de representation ...":OR
IGIN 0,0,0,638,0,364
190 FOR k=0 TO 360 STEP 40:!TDA,k:MOVE 220,100:GOSUB 210:CLG:NEXT k
```

ANNEXE B : PROGRAMME DE DEMONSTRATION

```
200 GOTO 340
210 RESTORE 220
220 DATA 999,40,80,0,0,0,250,95,0,0,0,0,-250,-95,0,0,-5,30,0,0,0,250,5
,-30,0,-5,30,0,105,0,0,-5,-30,0,5,30,0,0,0,-250,-5,-30,0,5,30,0,-105,0
,0
230 DATA 5,40,0,0,0,250,-5,-40,0,5,40,0,95,0,0,5,-40,0,-5,40,0,0,0,-25
0,5,-40,0,-5,40,0,-95,0,0
240 DATA 0,-35,0,95,0,0,0,30,0,-95,0,0
250 DATA 999,7,-7,-2,22,0,0,0,3,0,35,0,0,0,-2,0,22,0,0,0,-10,0,-22,0,0
,0,-2,0,-35,0,0,0,2,0,-22,0,0,0,10,0
260 DATA 999,-6,-35,2,900,0,0
270 DEFINT a-z:c=1
280 READ x,y,z:IF x=999 THEN 330
290 IF x=900 THEN 320
300 !TDRAWR,@x,@y,@z,@c
310 GOTO 290
320 RETURN
330 x=y:y=z:READ z:!TMOVER,@x,@y,@z:GOTO 290
340 CLS:INK 0,10:INK 1,0:BORDER 10:LOCATE 2,2:PRINT"Mais HBASIC permet
 aussi de s'occuper des fenetres":e=0:!WINBORDER,@e
350 FOR u=1 TO 9:WINDOW#1,u+3,u+50,u+3,u+13:e=1:!WINBORDER,@e:NEXT u
355 WINDOW#1,13,60,13,23:CLS#1:e=1:!WINBORDER,@e
360 LOCATE#1,2,2:PRINT#1,"Ainsi on peut leur tracer un cadre":GOSUB 10
050:PRINT#1,"          Ou leur enlever":FOR u=10 TO 1 STEP -1:WINDOW#1,u+3,u+50,
u+3,u+13:e=1:!WINNAKE,@e:NEXT u:e=0:!WINNAKE,@e
370 MODE 2:PRINT"Reprenons  nos fenetres":WINDOW#1,1,39,2,25:e=1:!WIN
BORDER,@e:WINDOW#2,40,80,2,25:e=2:!WINBORDER,@e:a=21:b=59:c=4:d=14:!DE
FZONE,@a,@b,@c,d
380 WINDOW#3,2,38,3,24:WINDOW#4,41,79,3,24:LOCATE 1,1:PRINT"Ecrivons q
uelques ligne celebres:"
390 PRINT#3,"L'AVARE ":PRINT#3,"
          de Moliere
":a=2:b=9:c=3:d=4:!DEFZONE,@a,@b,@c,@d:!DOUBLEZONE:PRINT#4,"L'AVARE ":
PRINT#4,"
e


":a=41:b=48:c=3:d=4:!DEFZONE,@a,@b,@c,@d:!DOUBLEZONE
400 PRINT#3,"
":' HARPAGON. - Au voleur! au voleur! "
410 DATA " HARPAGON. -  Au voleur! au voleur!"
420 DATA " a l'assasin! au meutrier! Justice,"
430 DATA " juste ciel! Je suis perdu,je suis "
440 DATA " assassine !  On m'a coupe la gorge"
450 DATA " on m'a dercbe mon AMSTRAD!Qui cela"
```

```
460 DATA " peut ce etre ?  Qu'est-il devenu ?"
470 DATA " ou est-il ? ou se cache-t-il ? Que"
480 DATA " ferai-je pour le trouver?Ou courir"
490 DATA " ou ne pas courir?N'est-il point la"
500 DATA " n'est-t-il point ici ?Qui-est-ce ?"
510 DATA " Arrete!Rends-moi mon argent,coquin"
520 DATA " Ah!c'est moi.Mon esprit es trouble"
530 DATA " et j'ignore ou je suis,qui je suis"
540 DATA " et ce que je fais..."
550 RESTORE 410:FOR u=1 TO 14:READ a$:PRINT#3,a$:PRINT#4,a$:NEXT u
560 a=2:b=38:c=3:d=24:e=41:f=78:LOCATE 1,1:PRINT"Faisons les defiler
                    "
570 FOR u=1 TO 66:!DEFZONE,@a,@b,@c,@d:!UMOVE:!DEFZONE,@e,@f,@c,@d:!DM
OVE:NEXT u
575 FOR u=1 TO 72:!DEFZONE,@a,@b,@c,@d:!RMOVE:!DEFZONE,@e,@f,@c,@d:!LM
OVE:NEXT u
580 GOSUB 10050
590 MODE 2:WINDOW#1,1,80,2,25:e=1:!WINBORDER,@e:PEN 1:PAPER 0:INK 0,10
:INK 1,0
600 PRINT"Imaginons un logiciel graphique:"
601 PEN 0:PAPER 1:LOCATE 53,3:PRINT"  MOVE      INVERT     ZOOM ":PEN 1:
PAPER 0
605 DEFINT a-z:a=2:b=79:c=4:d=24:!DEFZONE,@a,@b,@c,@d
606 GOSUB 610:GOTO 745
610 DATA 15,20,15,20,20,25,22,26,24,30,29,32,60,43,62,50,72,54,75,51,7
4,50,77,51,78,50,82,50,83,54,85,57,88,58,102,78,105,81,112,82,102,60,1
04,59,106,56,108,55,110,53,111,52,110,50,108,48,112,45,113,43,105,41,1
10,38,106,35,105,34
611 DATA 110,30,105,29,111,29,118,22,109,18,105,18,75,21,46,13,29,9,22
,7,13,7,10,7,8,7,7,9,8,12,15,20
612 DATA 18,19,20,16,21,12,22,7,21,12,37,20,37,12,37,20,37,24,36,26,35
,29,31,32
613 DATA 900,23,27,23,26,25,25,36,26,36,24,35,22,35,26,35,22,25,22,26,
25,25,22,21,23,24,25,21,23,21,26
614 DATA 900,9,13,10,12,11,11,11,9,10,7,900,12,17,15,15,13,13,11,15,12
,17,900,29,31,30,30,26,28,25,29,29,31,900,30,29,31,28,28,27,27,28,30,2
9
615 DATA 900,37,22,90,42,91,32,95,32,95,43,98,43,101,33,106,35,104,45,
98,43,90,42,900,101,33,95,32,91,32,71,26,51,17,36,11,42,12,75,24,105,2
1,119,22,115,22,98,33,95,32,110,22
616 DATA 900,42,37,46,35,50,30,51,27,64,32,63,35,59,40,55,42,70,46,74,
45,78,40,79,38,89,42,88,45,85,48,82,50,69,46,900,32,31,94,53,102,60,90
0,94,53,109,80,104,78,83,50,83,53,89,60
617 DATA 900,95,54,100,55,103,55,106,56,103,55,104,52,108,55,104,52,10
5,46,109,48,105,46,104,45,900,86,47,95,54,900,63,44,64,49,69,51,73,47,
75,48,70,53
```

ANNEXE B : PROGRAMME DE DEMONSTRATION

```
610 DATA 900,9,9,901,0,900,75,23,901,0,900,102,21,901,0,900,63,46,901,
0,900,103,65,901,0,900,108,24,901,0,999,0,0
700 RESTORE 610
705 READ a,b:MOVE a*4+20,b*4+20
710 READ a,b
711 IF a=900 THEN a=b:READ b:MOVE a*4+20,b*4+20:GOTO 710
712 IF a=901 THEN a%=0:b%=1:!FILL,@a%,@b%:GOTO 710
713 IF a=999 THEN 740
719 DRAW a*4+20,b*4+20,1
720 GOTO 710
740 RETURN
745 DEFINT A-Z:PEN 1:PAPER 0
755 LOCATE 1,1:PRINT"Selectionnons l'option MOVE:        "
756 GOTO 800
765 LOCATE 53,4:PRINT"O DOWN"
775 LOCATE 53,5:PRINT"O UP  "
785 LOCATE 53,6:PRINT"O LEFT"
795 LOCATE 53,7:PRINT"O RIGHT":RETURN
796 GOTO 765
799 a=2:b=79:c=4:d=24
800 choix=0:GOSUB 9005:GOSUB 796:FOR u=1 TO 90:LOCATE 53,4:PRINT"0";:
NEXT u:!RECALLZONE,@choix,@sto:GOSUB 9000:!DMOVE:!DMOVE:GOSUB 10050
805 GOSUB 9005:GOSUB 796:FOR u=1 TO 90:LOCATE 53,5:PRINT"0";:NEXT u:!
RECALLZONE,@choix,@sto:GOSUB 9000:!UMOVE:GOSUB 10050
810 GOSUB 9005:GOSUB 796:FOR u=1 TO 90:LOCATE 53,6:PRINT"0";:NEXT u:!
RECALLZONE,@choix,@sto:GOSUB 9000:!LMOVE:GOSUB 10050
815 GOSUB 9005:GOSUB 796:FOR u=1 TO 90:LOCATE 53,7:PRINT"0";:NEXT u:!
RECALLZONE,@choix,@sto:GOSUB 9000:!RMOVE:GOSUB 10050
820 LOCATE 1,1:PRINT"Voyons INVERT:                          ":GOSUB 90
00:a%=255:b%=1:!LOGZONE,@b%,@a%:GOSUB 10050
830 LOCATE 1,1:PRINT"Et pour finir: ZOOM:":a%=90:b%=40:MOVE 90,90:c%=0
:!BOX,@a%,@b%,@c%:a=91:b=42:x=10:y=200:sx=3:sy=2:ca=0:MOVE 90,90:!ZOOM
,@a,@b,@sx,@sy,@x,@y,@ca
835 GOSUB 10050
836 GOTO 2000
840 RT$="@0P15E2V2!A1T1O3D4DMSS8>S4S8M4M8<D4DMSS8>S4S8F4F8<R4RFLL8>L4L
8F4F8<R4RFLL8>L4L8M4M8<D4DMS>D8>D4D8<S4S8<D4DMF>D8>D4D8<L4L8<R4RFLL16F
*4S>M16<D4MM8R4L8S4D20"
850 MODE 1:GOSUB 10000:PRINT"
```

Mais il y a aussi la musique:

"

ANNEXE B : PROGRAMME DE DEMONSTRATION

```
852 PRINT"Ecoutez......"
855 !PLAY,@rt$
865 GOTO 910
870 GOSUB 1000:CLS:MODE 2:DEFINT a-z:LOCATE 19,6:PRINT"HBASIC":dx=6*9:
dy=16:sx=8:sy=8:ca=255
875 SYMBOL 255,206,206,206,206,206,206,206,206
880 MOVE 144,304:!ZOOM,@dx,@dy,@sx,@sy,@ca
885 LOCATE 14,15:POKE &B1C8,1:PRINT"THE AMSTRAD POWER":POKE &B1C8,2
890 LOCATE 45,18:PRINT"BY":LOCATE 41,21:PRINT"POWER SOFT"
895 a=41:b=51:c=21:d=22:!DEFZONE,@a,@b,@c,@d:!DOUBLEZONE
900 GOSUB 10050:GOTO 20
910 MODE 1:GOSUB 10000
915 LOCATE 2,4:PRINT"Tout ce que vous venez de voir a ete    ecrit en
";:PEN 2:PRINT"HBASIC":PEN 1
917 PRINT"

HBASIC c'est aussi le language machine  avec par exemple EXEC qui vous
 permet   d'explorer les recoins caches de votre  AMSTRAD,plus rien ne
 vous resiste !!"
920 PRINT"
Hbasic vous reserve bien des surprises  et vous serez en mesure de pro
grammer ce que vous venez de voir sans probleme   apres avoir lu le mo
de d'emploi !!!"
925 PRINT"

     Bonne chance ...."
930 FOR U=1 TO 10:FOR z=1 TO 4000:NEXT z:NEXT u:GOTO 870
1000 MODE 0:DEFINT a-z:PRINT"HBASIC":A=1:b=26:c=1:d=2:!DEFZONE,@a,@b,@
c,@d
1010 !DOUBLEZONE:sto=30001:!STOREZONE,@sto:CLS:x=2:y=2
1020 !ORGZONE,@x,@y:ch=1:!RECALLZONE,@ch,@sto:A=1:b=27:c=1:d=4
1021 LOCATE 1,10
1030 !DEFZONE,@a,@b,@c,@d:ch=1:x=192:!STOREZONE,@sto:!LOGZONE,@ch,@x
1040 stp=sto+1000:!STOREZONE,@stp
1050 DATA 1,1,53,1,27,5,1,9,53,9,27,13,1,17,53,17,27,21
1052 DATA 27,1,1,5,53,5,27,9,1,13,53,13,27,17,1,21,53,21
1055 RESTORE 1050
1100 ch=0:FOR u=1 TO 9:READ a,b:!ORGZONE,@a,@b:!RECALLZONE,@ch,@stp:NE
XT u
1101 RESTORE 1052
1110 ch=0:FOR u=1 TO 9:READ a,b:!ORGZONE,@a,@b:!RECALLZONE,@ch,@sto:NE
XT u
1130 GOSUB 10050:RETURN
2000 x=1:c=255:!LOGZONE,@x,@c
2004 WINDOW#2,1,80,1,1:PRINT#2,"Ecrivons un menu:   "
2005 A=13:B=34:C=13:D=19:WINDOW#1,A,B,C,D:E=1:b=b+1
```

ANNEXE B : PROGRAMME DE DEMONSTRATION

```
2000 !DEFZONE,@a,@b,@c,@d:sto=3000!:!STOREZONE,@sto:CLS#1:!WINBORDER,
@E
2010 PRINT#1,"      AMSPAINT":PRINT#1,"     ---------"
2015 PRINT#1,"      F1: Aide   F4: Menu"
2020 PRINT#1,"      F2: Copie  F5: Colle"
2025 PRINT#1,"      F3: Coupe  F6: Loupe"
2030 std=sto+1500:!STORE
2040 PRINT#2,"Deplacons-le un peu:"
2050 ch=0:a=13:b=13:FOR u=1 TO 10:!RECALLZONE,@ch,@sto:a=a+3:b=b-1:!OR
GZONE,@a,@b:!STOREZONE,@sto:!RECALLZONE,@ch,@std:FOR y=1 TO 400:NEXT y
:NEXT u
2060 ch=0:FOR u=1 TO 20:!RECALLZONE,@ch,@sto:FOR y=1 TO 200:NEXT y:!RE
CALLZONE,@ch,@std:FOR y=1 TO 200:NEXT y:NEXT u:GOTO 840
2999 END
3000 a=1:b=2:c=1:d=3:!DEFZONE,@a,@b,@c,@d:GOTO 2000
8999 END
9000 a=2:b=79:c=4:d=24:!DEFZONE,@a,@b,@c,@d:RETURN
9005 A=53:B=60:C=4:D=8:!DEFZONE,@A,@B,@C,@D:sto=30501:!STOREZONE,@sto:
RETURN
9999 GOTO  9999
10000 CLS:a=1:b=13:c=2:!DEFZONE,@a,@b,@a,@c:PEN 2:PRINT"HBASIC":!DOUBL
EZONE:PEN 1:RETURN
10010 !STOREZONE,@fod%:!RECALLZONE,@cal%:RETURN
10020 !RECALLZONE,@fod%:RETURN
10040 END
10050 FOR u=1 TO 3000:NEXT u:RETURN
```

POWER SOFT (R)

BON DE GARANTIE

Nous vous prions de nous renvoyer ce certificat qu garantie le logiciel pendant un délai de 2 ans (à renvoyer dans un déla de 15 jours après l'achat).

NOM DU PROGRAMME : H B A S I C _ _ _ _ _ _ _ _ _ _ _ _ _ _ _

DATE DE L'ACHAT : _ _ _ _ _ _ _ _ _ _ _ _ _ _ _

NOM DE L'ACHETEUR: _ _ _ _ _ _ _ _ _ _ _ _ _ _ _

ADRESSE : _ _ _ _ _ _ _ _ _ _ _ _ _ _ _

_ _

_ _

CODE POSTAL VILLE: _ _ _ _ _ _ _ _ _ _ _ _ _ _ _

NOM REVENDEUR : _ _ _ _ _ _ _ _ _ _ _ _ _ _ _

CONFIGURATION SYSTEME :

ORDINATEUR	CPC 464	CPC 664	CPC 6128
IMRIMANTE	EPSON	DMP-1	AUTRE
JOYSTICK	OUI	NON	

A RENVOYER A

POWER SOFT

SERVICE APRES VENTE

7 à 9 Rue des petites écuries

75010 PARIS

FRANCE

HBASIC

TABLE DES MATIERES

Désignation	Page
Préliminaire	1
Introduction	2
Chargement de HBASIC	2
Généralités	2
Résolution et couleurs	6
Le texte	6
Le graphisme	9
Graphisme simple	12
Crayon et papier	12
Le cercle	13
Autre	16
Graphisme haut niveau	18
Mélanges	18
Remplir	20
Agrandir	21
Joystick	23
La trosième dimension	25
Le problème	25
Les instructions	26
La zone	29
Définition	29
Les instructions	30
Le language machine	37
Lire	37
Ecrire	38
Exec	38
Musique maestro	41
La note	41
Autre	43
Divers	45
Le cadre	45
La copie	45
Autre	46

TABLE DES MATIERES

Désignation	Page
Le jeu d'instruction	48
ARC	49
AXES	50
BOX	51
CIRCLE	52
COLSET	53
COPYCHR	54
DEEK	55
DEFZONE	56
DMOVE	57
DOKE	58
DOUBLEZONE	59
DUMP	60
ELLIPSE	61
EXEC	62
FILL	63
FILLZONE	64
GCOPY	65
GAND	66
GNORMAL	66
GOR	66
GXOR	66
GRAPHICPAPER	67
GRAPHICPEN	68
JOYMOVE	69
LMOVE	70
LOGZONE	71
MOKE	72
ORGZONE	73
PLAY	74
RECALLZONE	76
RMOVE	77

TABLE DES MATIERES

Désignation	Page
STOREZONE	78
TDA	79
TEXT	80
TDRAW	81
TDRAWR	82
TMOVE	83
TMOVER	84
TPLOT	85
TPLOTR	86
TTEST	87
TTESTR	88
UMVE	89
WINBORDER	90
WINNAKE	91
ZOOM	92
Liste des instructions	93
Programme de démonstration	94
Bon de garantie	100
Table des matières	101

Rectifications version disquette

La disquette contient trois programmes :

HBASIC.BAS	Implante le HBASIC en mémoire
HBASIC.BIN	Corps du HBASIC
HDEMO.BAS	Démonstration du HBASIC

De plus, les paramètres des instructions du HBASIC, contrairement à ce qui est indiqué au CHAPITRE 1 admettent tout genre de variable et non seulement des variables entières.

```
!CIRCLE,A*3+X/4,TEST(X,Y) et
!CIRCLE,100,1 sont parfaitement admis.
```

ATTENTION : quand le HBASIC doit passer des valeurs au programme basic, cela doit se faire au travaers de variables entières et définies. Ainsi l'instruction !DEEK qui donne la valeur 16 bits d'une adresse doit être écrite comme suit :

```
!DEEK,8000,@A%        'On lit la valeur avec !DEEK en 8000
PRINT A%              'On récupère la valeur qui est dans A%
```

Le meilleur moyen de comprendre est d'essayer, n'hésitez donc pas et bonne chance.