

# AMSTRAD CPC 464

GUIDE  
de  
L'UTILISATEUR



**NOTICE**

**LES APPAREILS AMSTRAD CPC 464/6128  
SONT CONFORMES A LA NORME  
NF C 91-022 RELATIVE AUX EMISSIONS  
RADIOELECTRIQUES.**

# **AMSTRAD CPC464 & CPC6128**

## **CLAVIER AZERTY**

### **NOTE IMPORTANTE**

**Vous possédez la version AZERTY du CPC, et un certain nombre de précisions sont nécessaires pour une meilleure lecture de votre manuel.**

#### **1 CLAVIER FRANCAIS (AZERTY) ET ANGLAIS (QWERTY)**

La différence entre les claviers AZERTY et QWERTY n'est pas qu'une question d'ordre des touches de l'Alphabet, la langue Française possède des lettres accentuées, qui pour la plupart, ont été ajoutées sur le clavier français. De ce fait un certain nombre de caractères du Basic Locomotive se sont trouvés "chassés" de leur place habituelle sur le clavier. Ces caractères qui se trouvent ajoutés créent une contrainte obligeant certains caractères utilisés sous Basic, à perdre leur représentation usuelle à l'écran. Ainsi, par exemple Le symbole de la division entière du Basic ne se trouve plus représenté comme "\\" mais comme "ç". L'effet de ce caractère reste le même, sa représentation seule est altérée ... Ce n'est qu'une question de convention.

Il est important de noter qu'un langage de programmation n'est pas nécessairement un langage d'écriture. La francisation du clavier des CPC aura tendance à faciliter la frappe et la lecture des messages d'un programme, sans toutefois vous permettre d'utiliser les caractères accentués dans les noms de variable ni d'obtenir directement une lettre avec un accent circonflexe. En effet l'accent circonflexe en Basic est le symbole de l'élevation à la puissance: Il vous faudra donc résoudre ce problème par un moyen logiciel.

Vous ne devrez jamais perdre de vue que les logiciels développés jusqu'à maintenant sur nos machines de type QWERTY peuvent utiliser un moyen d'interrogation du clavier relatif à la position physique de la touche et non pas au caractère correspondant à cette touche. Prenons l'exemple d'un jeu vous proposant à un certain stade l'option: [Quitter]. Si le test est effectué sur la position physique de la touche 'Q', le programme ne réagira pas en appuyant sur la lettre Q du clavier AZERTY. Il ne réagira en fait qu'à la frappe de la lettre A (les lettres A et Q ayant sur un QWERTY ou AZERTY la même position physique sur le clavier).

Vous pouvez trouver un clavier QWERTY représenté à la page 4 de cette note

#### **2 FRAPPE ET VISUALISATION**

Le manuel que vous avez sous les yeux contient des exemples et des explications pour un CPC QWERTY. Certaines de ces explications relatives au clavier s'en trouvent erronées. Toutefois, dans les exemples, TAPEZ toujours au CLAVIER les caractères que vous LISEZ dans le MANUEL.

Si vous lisez dans le Manuel:

PRINT 3<sup>2</sup> (3 puissance 2)

tapez au clavier sur les touches suivantes:

PRINT 3<sup>↑ 2</sup>

lisez sur l'écran:

PRINT 3 ^ 2

Ce qui est écrit à l'écran est valide quoique différent. La correspondance des symboles altérés est la suivante:

VALEUR DECIMALE	HEXA	CARACTERE	CARACTERE
		AZERTY	ASCII
64	40	à	@
92	5C	ç	\
94	5E	^	↑
123	7B	é	{
124	7C	ù	
125	7D	é	}

Familiarisez vous avec ces 6 principales différences, c'est la contrainte la plus gênante qu'apporte le clavier AZERTY.

### 3 CONTROLE DU CLAVIER

En AZERTY, pour taper un chiffre, il faut auparavant appuyer sur la touche SHIFT puis sur le chiffre correspondant. La touche CAPS LOCK ne fait que bloquer les lettres de l'Alphabet en majuscule. Pour bloquer tout le clavier en position haute (toutes les touches shiftées), vous devez appuyer simultanément sur les touches CTRL et CAPS LOCK. La même combinaison vous fera revenir à un clavier normal.

*SIVOUS UTILISEZ DES DISQUETTES ...*

### AZERTY SOUS AMSDOS

Hormis CAT, toutes les commandes AMSDOS sont précédées du symbole: !  
Lorsque vous taperez 'l' vous verrez apparaître à l'écran le symbole 'ù'. Dans la syntaxe des commandes AMSDOS, la variable (qui représente un nom de fichier) est toujours précédée du caractère @. La visualisation de ce caractère en sera évidemment altérée.

Sur le manuel vouserez par ex: ! ren,@n\$,@a\$  
vous taperez sur le clavier: ! r e n ,@ n \$ ,@ a \$  
vous verrez à l'écran: û r e n ,à n \$ ,à a \$

### SOUS CP/M

CP/M est appelé en tapant sous Basic la commande: ù c p m  
Une fois sous CP/M, aucun des caractères spéciaux n'est nécessaire dans la syntaxe des commandes.  
Si vous tapez un texte sous CP/M, tous les caractères accentués sont disponibles et seront conformes à l'édition comme à l'impression.

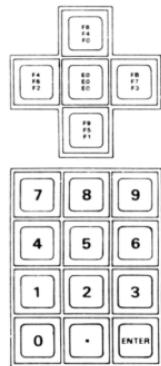
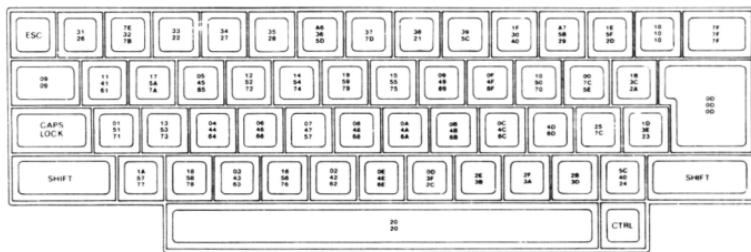
Notez que les crochets droit et gauche sont imprimés ainsi quand l'imprimante est réglée en caractères français:

écran	imprimante
[ ----->	°
]	§

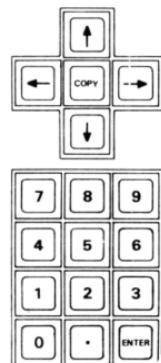
Sous CP/M+ sur le 6128, il est quelquefois utile d'exécuter le programme LANGUAGE 0 pour programmer.

# CPC464 AZERTY

Valeurs ASCII par défaut (en hexadécimal)



Caractères d'expansion, valeurs et emplacements par défaut

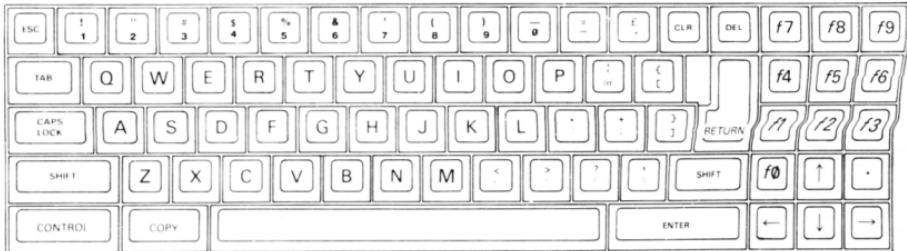


# CPC6128 AZERTY & QWERTY

#### **Valeurs ASCII par défaut ( en hexadécimal )**



### **Caractères d'expansion, valeurs et emplacements par défaut**



# L'ORDINATEUR PERSONNEL COULEUR CPC464 64K

## Comment se servir de ce guide

L'informatique a parcouru un grand chemin en un temps très court. De toutes les inventions du 20ème siècle, l'ordinateur est certainement la plus extraordinaire.

Les capacités des machines et des logiciels ont progressé si rapidement que même les utilisateurs habituels ont du mal à suivre, et pour montrer aux possesseurs d'un CPC464 tout le potentiel et la subtilité de son BASIC, de son système d'exploitation et de son architecture il faudrait des milliers de pages.

Ce guide est donc une introduction condensée au CPC464 et son logiciel. Il sera complété par beaucoup de cours et de publications spécifiques et détaillés.

Ceux qui sont déjà familiers avec d'autres dialectes BASIC deviendront très rapidement familiers avec la conception du BASIC AMSTRAD - et les débutants apprécieront vite la nature directe et claire de la terminologie utilisée - il a été écrit et conçu pour éviter les fantaisies qu'on trouve dans beaucoup de BASIC non standards, ainsi que l'introduction de capacités en temps réel qu'on ne trouvait pas auparavant sur ce genre d'ordinateur.

Ce guide est divisé en trois parties.

Le premier est le cours élémentaire pour débutants, écrit spécialement pour présenter les concepts et les termes de l'informatique aux novices. Si vous n'avez pas encore utilisé un micro-ordinateur et écrit de petits programmes, alors il vaut mieux lire ce cours introductif.

Ceux qui ont déjà essayé un micro-ordinateur peuvent commencer au chapitre 1. Nous avons répété quelques points essentiels concernant la mise en route et avons insisté sur les points importants du CPC464 tout en supposant que vous connaissiez un peu la terminologie.

Chacun des chapitres *-Eléments de-* est un guide général donnant les caractéristiques performantes du CPC464. Quelques éléments fondamentaux sont répétés pour rendre service à ceux qui veulent se lancer tout de suite dans les graphiques ou les sons, après une brève introduction au clavier et un apprentissage méthodique du BASIC.

Le cours d'initiation appelé Guide du BASIC en un livre et deux cassettes vous permet une approche en profondeur pour comprendre les nombreuses facettes de votre CPC464 et son potentiel illimité comme éducateur, console de jeux et ordinateur pur et simple, et si vous voulez apprendre d'une manière valable nous vous conseillons d'en faire l'achat si ce n'est déjà fait !

Enfin de nombreux appendices donnent une vue générale des ordinateurs ainsi que des caractéristiques techniques et des points de référence.

Bonne chance - vous n'auriez pas pu choisir meilleure qualité à ce prix là, ni un ordinateur aussi capable de développer vos connaissances sur le sujet. Il n'y a pas de moyen plus facile pour apprendre l'informatique que de se servir d'un ordinateur - et le CPC464 est certainement le bon compagnon .

## AMSOFT

Un département de



©Copyright 1984 MSOFT, AMSTRAD SARL, AMSTRAD plc et Locomotive Software Ltd

'Ni l'information contenue aux présentes, ni le produit décrit dans ce manuel,, ne peuvent être modifiés ou reproduits totalement ou partiellement, en tout ou partie, sous quelque forme que ce soit, sans l'accord écrit préalable d'Amstrad SARL.'

AMSOFT et AMSTRAD accepteront volontiers  
vos suggestions à propos de l'ordinateur ou de ce guide

Toute correspondance doit être adressée à

AMSTRAD FRANCE  
143 Grande rue  
92310 Sèvres

Toute maintenance et service après vente concernant le produit doivent être effectués obligatoirement par des revendeurs Amsoft agréés. Ni Amsoft ni Amstrad ne seront responsables, de quelque façon que ce soit, de toute perte ou dommage causé par une maintenance ou service effectué par des personnes non-agréés.

Ce guide est seulement destiné à faciliter l'utilisation du produit par le lecteur et, par conséquent, ni Amsoft ni Amstrad ne seront responsables de toute perte ou dommage quelconque qui pourrait résulter de l'utilisation de toutes informations, renseignements, erreurs ou omissions contenus dans ce guide ainsi que de toute utilisation impropre du produit.'

première édition 1984  
publié par Amstrad  
composition par Amsoft Computer Graphics

# **IMPORTANT**

En lisant ce guide, vous devez faire attention aux différents types de caractères d'imprimerie qui précisent les références faites aux programmes.**[TOUCHES]** qui existent sur le clavier,mais ne donnent pas un caractère sur l'écran,et des **'descriptions générales'** qui se rapportent aux mots dans les programmes, mais ne sont pas frappés en même temps que l'instruction.

1. Ne jamais brancher le clavier, le moniteur ou le modulateur à un appareil ou prise de courant autre que celle décrite dans ce guide. Cela pourrait causer des dommages sérieux et invalider la garantie.
2. Pas de boissons, ou vases de fleurs près du clavier, de l'ordinateur ou du modulateur : si un liquide se répandait, des ennuis s'ensuivraient. Dans tous les cas, consultez un expert.
3. Ne pas bloquer les trous de ventilation derrière ou au dessus des appareils.
4. Si vous coupez le courant, tout ce qui est dans la mémoire de l'ordinateur sera perdu. Si vous désirez sauvegarder un programme, lisez le chapitre 2 après avoir lu le cours élémentaire.
5. Il vaut mieux utiliser le type de cassettes recommandées pour les micros. Mais il est parfaitement acceptable de se servir de bonnes cassettes audio fabriquées par de bonnes marques pourvu qu'elles ne soient ni CrO<sub>2</sub>, ni 'métal' ou de durée supérieure à 90 minutes(C-90).

Pour pouvoir vous y retrouver plus facilement dans vos cassettes de programmes,nous vous conseillons les cassettes C-12 (6 minutes par face).

6. Les cassettes avec des programmes pour d'autres ordinateurs ne marcheront pas sur le CPC464.
7. Si la cassette n'a plus les deux petits taquets de protection, le bouton marqué REC (enregistrement) ne pourra pas être pressé. Ne cherchez pas à forcer car vous pourriez endommager le mécanisme. Si vous voulez néanmoins l'enregistrer il faut mettre du Scotch par dessus les deux ouvertures de protection.
8. Vérifiez que la bande de la cassette a dépassé le début de la zone magnétique.
9. Ne pas utiliser ou ranger les appareils directement au soleil, et dans les endroits trop chauds, froids, humides ou poussiéreux, ou soumis à des vibrations trop importantes. Ne jamais ranger les cassettes à proximité de champs magnétiques (haut-parleurs, gros moteurs électriques).

10. Prenez soin de vos cassètes et nettoyez régulièrement votre lecteur et vous devriez sauvegarder et retrouver vos programmes sans erreurs.
11. Il n'y a pas de pièces à entretenir à l'intérieur des appareils. Ne cherchez pas à ouvrir. Pour tout problème faites appel à du personnel qualifié.
12. Il est interdit de copier ou d'adapter tout ou partie des informations, programmes ou produits décrits dans ce guide.

# Table des matières

## A propos de ce guide

### Cours Elémentaire pour débutants (élémentaire, mon cher Watson)

---

*Une gentille introduction pour les nouveaux de l'informatique*

- E1 Mise en marche
- E2 Familiarisation avec le clavier
- E3 Graphiques, modes et musique

## 1 Apéritifs

---

- Branchemet de l'ordinateur
- Mise au courant
- Faisons connaissance avec le clavier
- Affichage de l'ensemble des caractères
- Edition de l'écran

## 2 Le lecteur de cassette Datacorder

---

- Chargement et sauvegarde avec le datacorder
- La cassette 'Bienvenue'

## 3 Les éléments du BASIC

---

- Une introduction aux principes du BASIC du CPC464
- La syntaxe du BASIC AMSTRAD
- Quelques exercices simples
- PRINT et l'agencement de l'écran

## 4 Variables, opérations et données

---

- Variables et opérations
- Comment arranger l'affichage
- Données et rangements
- Prévoir les dimensions
- Position du curseur

## **5 Les éléments graphiques**

---

Les principes à la base des graphiques de l'AMSTRAD CPC464:

Couleur de l'encre (INK), du papier (PAPER)

Stylo (PEN), bordure (BORDER)

MODES, PIXELS, ORIGINS, WINDOWS (fenêtres)

Petits trucs graphiques

Comment définir son ( ou ses ) caractères

## **6 Les éléments sonores**

---

La variété du son sur le CPC464

Enveloppe de ton et de volume

Des sons à la queue leu-leu

Effets sonores

## **7 L'imprimante et les manettes de jeux**

---

Comment jouer avec le joystick

La commande JOY

Et si on imprimait

## **8 Guide précis du BASIC de l'AMSTRAD**

---

Un sommaire complet du langage BASIC et des mots réservés pour programmer le CPC464, par ordre alphabétique

## **9 Pour les programmeurs avancés**

---

L'organisation interne des programmes - logiciels intégrés

Interruuteurs : à quoi ils servent

Contrôlez votre caractère

De la relation entre les programmes de code - machine et le langage BASIC de haut niveau

## **10 De l'avantage des interruptions**

---

Travailler en temps réel

AFTER (après...) EVERY (à chaque...) et REMAIN  
(et le reste...)

## Appendices

---

- I**      Guide des débutants sur ce qu'on peut attendre et ne pas attendre d'un ordinateur.
- II**     Bits et octets - essai sur le binaire et l'hexadécimal
- III**    Codes ASCII et la police de caractères  
Définition des caractères et grilles  
Codes du clavier, jetons d'expansion
- IV**    Introduction pour les chevronnés et étude sommaire
- V**     L'interface avec l'utilisateur, le bus d'expansion  
Les prises d'entrées sorties
- VI**    Papiers quadrillés pour organiser vos textes
- VII**   Feuille de musique  
Notes, périodes et fréquences
- VIII**   Mots réservés, messages et codes d'erreurs
- IX**    Petit dictionnaire
- X**     Accents et traitement de textes.



# **AMSTRAD CPC464**

## **COURS ELEMENTAIRE POUR DEBUTANTS**

### **Eléments E1:**

## **MISE EN MARCHE**

Premières instructions pour ouvrir, brancher et mettre en route votre ordinateur CPC464.

L'ordinateur personnel couleur AMSTRAD CPC464 peut fonctionner avec les écrans suivants:

- 1.1 AMSTRAD GT64, moniteur monochrome vert**
- 1.2 AMSTRAD CTM64O, moniteur couleur**
- 1.3 AMSTRAD FMP1, boîtier d'alimentation/  
modulateur ET un poste de Télévision (prise  
péritel).**

Veuillez vous rapporter à la section appropriée pour brancher votre ordinateur correctement et commencer à vous en servir.

### **1.1 AMSTRAD GT64, moniteur monochrome vert**

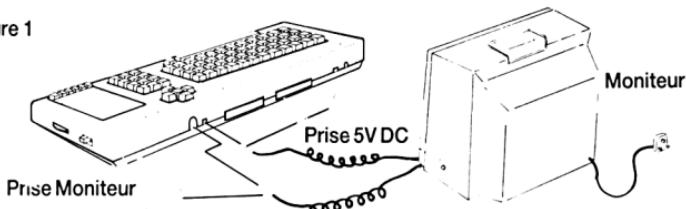
Ouvrir la boîte du poste moniteur.

## ATTENTION

Débrancher la prise mâle de la prise de courant quand l'ordinateur n'est pas utilisé  
Il n'y a pas de connections intérieures à faire, il n'est donc pas nécessaire d'ouvrir les appareils.

L'ordinateur devra être placé devant le moniteur sur une table à proximité de la prise de courant. Comme sur la figure 1, brancher la prise à 6 plots (6 pin DIN) dans la prise marquée MONITOR derrière l'ordinateur. De même pour la prise plus petite d'alimentation dans la prise marquée **5V DC** derrière l'ordinateur.

Figure 1



Vérifiez que le bouton **POWER** du moniteur est **OFF** (position sorti).

Branchez la prise courant du moniteur sur le courant 240v AC.

Maintenant pressez le bouton du moniteur et mettez l'ordinateur en marche en déplaçant l'interrupteur à glissière marqué **POWER** sur le côté droit.

La lampe rouge sur le dessus de l'ordinateur devra s'allumer, et vous verrez apparaître sur le moniteur l'image suivante:

**Amstrad 64K Microcomputer (v1)**  
**©1984 Amstrad Consumer Electronics plc**  
**and Locomotive Software Ltd.**

**BASIC 1.0**

**Ready**

Curseur

Pour éviter la fatigue des yeux, ajustez le bouton de contrôle marqué **BRIGHTNESS** à votre convenance sans éblouissement ni écriture brouillée.

Il faut aussi ajuster le contrôle de **CONTRAST** au point minimum pour une vision confortable.

Le contrôle vertical sur le GT64 est marqué **V-HOLD** et doit être ajusté de telle manière que l'image est positionnée correctement au milieu de l'écran, sans sautillement ou roulis.

## 1.2 AMSTRAD CTM64O , Moniteur Couleur

Ouvrir la boite du poste moniteur.

### ATTENTION

Débrancher la prise mâle de la prise de courant quand l'ordinateur n'est pas utilisé. Il n'y a pas de connections intérieures à faire, il n'est donc pas nécessaire d'ouvrir les appareils.

L'ordinateur devra être placé devant le moniteur sur une table à proximité de la prise de courant. Comme sur la figure 1, sur la page précédente, il faut alors brancher la prise à 6 plots (6 pin DIN) dans la prise marquée **MONITOR** derrière l'ordinateur. De même pour la prise plus petite d'alimentation dans la prise marquée **5V DC** de derrière l'ordinateur.

Vérifiez que le bouton **POWER** du moniteur est **OFF** (position sorti). Branchez la prise courant du moniteur sur le courant 240v AC.

Maintenant pressez le bouton du moniteur et mettez l'ordinateur en marche en déplaçant l'interrupteur à glissière marqué **POWER** sur le côté droit.

La lampe rouge sur le dessus de l'ordinateur devra s'allumer, et vous verrez apparaître sur le moniteur l'image suivante:



Pour éviter la fatigue des yeux, ajustez le bouton de control marqué **BRIGHTNESS** à votre convenance sans éblouissement ni écriture brouillée.

### 1.3 AMSTRAD MP1, adaptateur Couleurs Péritel, pour utiliser avec votre poste de Télé couleur.

Le boitier MP1 est un appareil optionnel que vous pouvez acheter si vous avez déjà un ordinateur CPC464 avec le moniteur vert GT64. Le MP1 vous permet d'utiliser l'ordinateur avec votre poste télé couleur et d'utiliser ainsi toutes les possibilités de votre CPC464.

Sortir le boîtier MP1 de son carton.

#### ATTENTION

Débrancher la prise mâle de la prise de courant quand l'ordinateur n'est pas utilisé

Il n'y a pas de connexions intérieures à faire, il n'est donc pas nécessaire d'ouvrir les appareils.

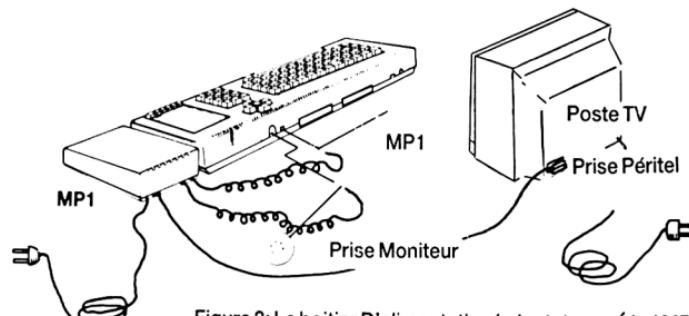


Figure 2: Le boîtier D'alimentation/adaptateur péritel MP1

Le MP1 se place sur le côté droit de l'ordinateur lequel ordinateur devra être placé devant le poste Télé sur une table à proximité de la prise de courant. Comme sur la figure 2, brancher la prise à 6 plots (6 pin DIN) du MP1 dans la prise marquée **MONITOR** derrière l'ordinateur. De même pour la prise plus petite d'alimentation dans la prise marquée **5V DC** de derrière l'ordinateur. Mettez la prise d'antenne du MP1 dans celle de votre télé couleur.

Vérifiez que le bouton **POWER** de l'ordinateur est **OFF** (sur le cote) et branchez la prise courant du MP1 sur le courant **240v AC**.

Mettez alors le volume sur votre télé au minimum,branchez votre poste télé et ensuite mettez l'ordinateur en marche avec le bouton glisseur.

La lampe rouge sur le dessus de l'ordinateur devra s'allumer, et il faut maintenant régler votre TV pour recevoir le signal du CPC464 parfaitement.

Seuls les boutons de contraste,de luminosité et de couleurs peuvent avoir besoin d'être réglés pour arriver à l'image suivante:

**Amstrad 64K Microcomputer (v1)**

**©1984 Amstrad Consumer Electronics plc  
and Locomotive Software Ltd.**

**BASIC 1.0**

**Ready**

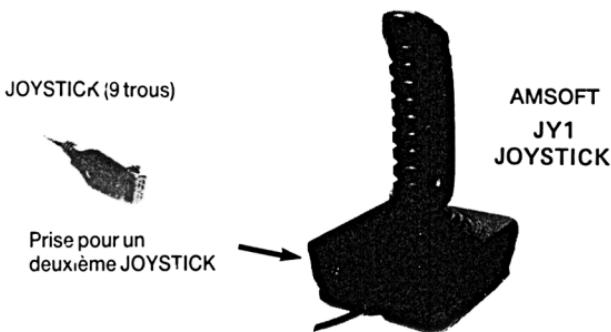


Curseur

Tournez encore jusqu'à ce que l'image soit la plus claire possible. L'écriture sera orange sur fond bleu.

## 1.4 Manette de jeux-Joystick

Le joystick AMSOFT modèle JY1 est un extra que vous pouvez acheter si vous utilisez l'ordinateur CPC464 avec des jeux qui permettent l'usage du contrôle par joystick et de faire FEU pendant le jeu. Le JY1 peut être branché à l'arrière du CPC464 avec la prise 9 trous marquée **USER PORTS**. L'AMSTRAD CPC464 peut être utilisé avec deux joysticks, le deuxième étant branché dans une prise située dans le premier.



## 1.5 Cassette de Bienvenue

Emboîtée dans une des pièces de polystyrène expansé qui maintenait votre ordinateur, vous aurez découvert la cassette de Bienvenue. Ouvrez le lecteur en appuyant sur la touche **[STOP/EJECT]** puis mettez la cassette dans le lecteur comme sur la figure 3, en vous assurant que **SIDE 1** = face 1 est sur le dessus:

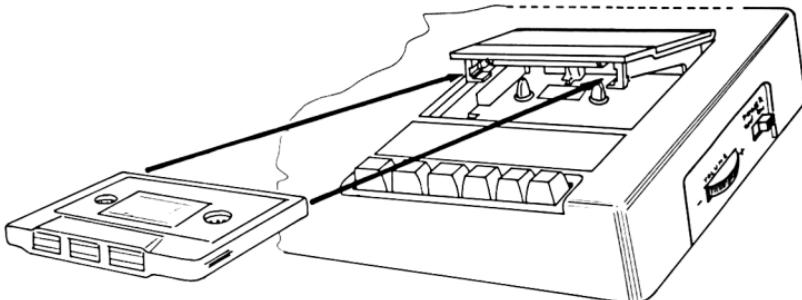


Figure 3 Comment introduire une cassette dans le Datacorder

Fermez le dessus du lecteur jusqu'au clic, puis pressez la touche **[REW]** pour être sûr que vous êtes au début de la cassette. Aussitôt que la cassette s'arrête, appuyez sur la touche **[STOP/EJECT]**. Remettez le compteur à 000 en pressant sur le bouton **[COUNTER RESET]**.

Pressez la touche **[CTRL]** (=contrôle), et EN MEME TEMPS, appuyez sur la petite touche **[ENTER]** au bas du pavé numérique à côté du Datacorder. L'écran répondra avec l'instruction:

**RUN"**  
**Press PLAY then any key**

Ce qui vous demande d'appuyer sur la touche **[PLAY]** du lecteur de cassette jusqu'à ce qu'elle soit bien enclenchée, puis pressez n'importe quelle autre touche du clavier, une lettre, un chiffre ou **[ENTER]** ou encore la barre d'espacement.

La bande va tourner, et après un petit moment, vous verrez le message suivant apparaître sur l'écran:

**Loading BIENVENUE 1 block 1**

Ce qui veut dire que le programme BIENVENUE va se mettre dans la mémoire de l'ordinateur pendant environ 5 minutes et vous voyez sur l'écran le numéro du block se changer en 2,3 etc jusqu'à l'arrêt de la cassette. Le programme Bienvenue va alors commencer.

Il suffit de le regarder et d'admirer. Comme c'est un programme continu, quand vous avez fini de le regarder, appuyez sur la touche **[ESC]** deux fois de suite. Cela arrête le programme et vous pouvez pressez sur **[STOP]** et éjecter la cassette, puis la tourner sur la face 2.

N'oubliez pas d'appuyer sur **[REW]** pour être sûr que la bande est revenue à son départ.

Pressez la touche **[CTRL]** (=contrôle), et EN MEME TEMPS, appuyez sur la petite touche **[ENTER]** au bas du pavé numérique à côté du Datacorder. L'écran répondra avec l'instruction:

**RUN"**  
**Press PLAY then any key**

Ce qui vous demande d'appuyer sur la touche **[PLAY]** du lecteur de cassette jusqu'à ce qu'elle soit bien enclenchée, puis pressez n'importe quelle autre touche du clavier, une lettre, un chiffre ou **[ENTER]** ou encore la barre d'espacement.

La bande va tourner, et après un petit moment, vous verrez le message suivant apparaître sur l'écran:

**Loading BIENVENUE 2 block 1**

Suivez les instructions qui apparaissent sur l'écran, et le programme vous invitera à participer en tapant les instructions sur le clavier à mesure que le programme se déroule.

## 1.6 COMMENT CHARGER LES AUTRES CASSETTES

LA CASSETTE DE BIENVENUE PEUT SEULEMENT ETRE CHARGEÉE ET EXECUTÉE comme on l'a vu dans le paragraphe précédent (1.5). Des programmes BASIC non protégés peuvent être chargés par la méthode alternative suivante. Réenroulez le programme que vous avez utilisé en pressant la touche **[REW]** du datacorder jusqu'à l'arrêt puis appuyer sur la touche **[STOP/EJECT]**.

Remettez l'ordinateur et la mémoire à zéro en appuyant sur les touches **[CTRL].[SHIFT]** et **[ESC]** dans cet ordre - et en les gardant appuyées ensemble une demi-seconde- l'écran redevient uni avec le message habituel. Quand vous voyez **[ENTER]** cela veut dire que vous devez presser sur une des deux touches **[ENTER]-ne tapez pas le mot Enter sur le clavier! Les guillements " sont obtenus en appuyant en même temps sur les touches **[SHIFT]** et le 2 en haut du clavier.**

Tapez:

**Load ""**

puis appuyez sur **[ENTER]**

L'ordinateur vous répond...

**Press PLAY then any key:**

Ce qui signifie d'appuyer sur la touche **[PLAY]** du datacorder jusqu'à ce qu'elle soit bien enclenchée, puis appuyer sur une touche quelconque du clavier, lettre ou **[ENTER]** ou la barre centrale.

La bande va marcher et après un petit moment vous verrez le message suivant apparaître sur l'écran:

**Loading .nom du programme. block 1**

Les numéros de blocks vont augmenter jusqu'à la fin du chargement, et le message:

**Ready**

...qui veut dire Prêt, va apparaître sur l'écran.

Ou bien, vous pouvez préciser le nom du programme que vous voulez charger. Pour faire cela,tapez:

**Load ".titre." [ENTER]**

L'ordinateur vous répond...

**Press PLAY then any key:**

Ce qui signifie d'appuyer sur la touche **[PLAY]** du datacorder jusqu'à ce qu'elle soit bien enclenchée, puis appuyer sur une touche quelconque du clavier, lettre ou **[ENTER]** ou la barre centrale.

La cassette va tourner. Si le programme que vous voulez n'est pas au début de la cassette, l'ordinateur va chercher le titre exact du programme que vous avez demandé. Faites attention à taper le titre correctement.

Si, pendant qu'il cherche votre titre, l'ordinateur trouve un autre programme, le message suivant apparaît:

**Found .un autre titre. block 1**

Autrement dit, l'ordinateur ne va pas charger ce programme, mais continuer à chercher votre titre jusqu'à ce qu'il ait trouvé votre programme, ou que vous appuyez sur la touche **[ESC]** pour arrêter la recherche.

Quand le programme a été trouvé, ce message apparaîtra sur l'écran:

**Loading .titre. block 1**

Les numéros vont augmenter jusqu'à la fin du chargement, et le message:

**Ready**

...vous informe que c'est prêt.

Vous tapez alors:

**run [ENTER]**

...et le programme que vous venez de charger va marcher. S'il y avait déjà un programme en mémoire, il sera effacé et le programme chargé prendra sa place.

Pour exécuter un programme directement, sans le charger d'abord, il suffit de taper:

**RUN "" [ENTER]**

...L'ordinateur vous répond...

**Press PLAY then any key:**

Ce qui signifie d'appuyer sur la touche **[PLAY]** du datacorder jusqu'à ce qu'elle soit bien enclenchée, puis appuyer sur une touche quelconque du clavier, lettre ou **[ENTER]** ou la barre centrale. L'ordinateur cherche, charge et exécute le programme sans autre instruction.



## Eléments E2:

# FAMILIARISATION AVEC LE CLAVIER

Nous allons maintenant vous expliquer la signification de quelques touches du clavier de l'ordinateur. Ceux qui ont l'habitude des micro-ordinateurs peuvent sauter ce chapitre.

### [ENTER]

Il y a deux touches bleues [ENTER]. L'une et l'autre font entrer dans l'ordinateur l'information que vous venez de taper. Après avoir appuyé sur la touche [ENTER], une ligne nouvelle commence sur l'écran. Chaque instruction que vous tapez sur le clavier de l'ordinateur doit être suivie de la touche [ENTER].

A partir de maintenant, quand vous verrez [ENTER], cela voudra dire appuyez sur la touche [ENTER] après chaque instruction ou ligne de programme.

### [DEL]

Cette touche est utilisée pour effacer un caractère à gauche du curseur (par exemple une lettre ou un nombre) que vous ne voulez pas conserver.

Taper abcd et vous verrez la lettre d à gauche du curseur. Si vous décidez que vous ne voulez pas cette lettre d'appuyez sur la touche [DEL] une fois et la lettre d disparaît. Si vous continuez à appuyer sur cette touche [DEL], les lettres abc disparaissent aussi.

### [SHIFT] (passage en majuscules)

Il y a deux touches vertes [SHIFT]. Si vous appuyez sur une des deux, et la maintenez appuyée pendant que vous tapez un autre caractère, une majuscule ou le symbole sur le haut de la touche va s'afficher sur l'écran.

Tapez la lettre a, puis appuyez sur [SHIFT] et tapez sur la lettre a de nouveau. Sur l'écran vous verrez:

aA

Maintenant tapez quelques espaces en maintenant la barre d'espacement appuyée. Essayez ceci avec les touches du haut du clavier, au dessus des lettres. Tapez le chiffre 2, puis appuyez sur [SHIFT] et tapez le chiffre 2 de nouveau. Vous verrez sur l'écran:

2"

Vous pouvez maintenant voir ce qui arrive quand la touche **[SHIFT]** est pressée et qu'on tape un caractère en même temps. Essayez en tapant chacune des touches, d'abord toutes seules, puis en appuyant sur la touche **[SHIFT]** en même temps.

#### **[CAPS LOCK]** (verrouillage des majuscules)

Cette touche a le même effet que la touche **[SHIFT]**, excepté qu'on doit la presser une seule fois. Toutes les lettres seront affichées en capitales, sauf les touches numériques qui resteront des chiffres. Appuyez sur **[CAPS LOCK]** puis tapez:

abcdef123456

Sur l'écran vous voyez:

ABCDEF123456

Et les chiffres ont été tapés tels quels. Maintenant si vous appuyez sur **[SHIFT]** en même temps, cela devient:

ABCDEF!"#\$%&

Si vous voulez revenir aux minuscules, pressez **[CAPS LOCK]** de nouveau. Si vous voulez taper les majuscules et les symboles du dessus des touches sans avoir à presser tout le temps sur **[SHIFT]**, vous pouvez le faire en appuyant sur **[CTRL]**, puis sur **[CAPS LOCK]**. Maintenant tapez:

abcdef123456

Vous verrez apparaître

ABCDEF!"#\$%&

On peut quand même taper des chiffres en utilisant ceux du pavé numérique sur la droite du clavier.

Si vous maintenez la touche **[CTRL]** et appuyez sur **[CAPS LOCK]** une fois, vous êtes ramenés où vous étiez auparavant, majuscules ou minuscules. Si vous êtes toujours en majuscules, appuyez sur **[CAPS LOCK]** une fois pour revenir en minuscules.

#### **[CLR]**

Cette touche sert à effacer un caractère qui est sous le curseur. Tapez: ABCDEFGH. Le curseur est à droite du (H). Puis appuyez quatre fois sur le curseur gauche [→]. Le curseur se superpose à la lettre E. Notez que la lettre E est toujours visible sous le curseur. Appuyez sur la touche **[CLR]** une fois et vous verrez que la lettre E a été effacée et que les lettres FGH se sont déplacées d'une case vers la gauche et que la lettre F est sous le curseur. Appuyez sur la touche **[CLR]** sans la lâcher. Les lettres F, puis G et H disparaissent.

## [ESC]

Cette touche rouge en haut du clavier à gauche, permet de prendre la poudre d'**ESCampette**, autrement dit d'interrompre un programme momentanément si on presse la touche une seule fois et de reprendre le programme si on tape une autre touche ou d'arrêter complètement si on appuie encore une fois sur [ESC].

Bon. Maintenant appuyez deux fois de suite sur la touche [ESC].

## IMPORTANT

Quand vous arrivez au bord droit de l'écran en tapant plus de 40 caractères, le caractère suivant apparaîtra automatiquement à la ligne suivante. Cela veut dire qu'il NE FAUT PAS presser [**ENTER**] comme ceux qui ont l'habitude des machines à écrire tapent RETOUR ou ENVOI à la fin d'une ligne.

L'ordinateur fait cela pour vous automatiquement, et réagira à un [**ENTER**] non prévu en vous donnant un message d'erreur - (d'habitude Syntax Error) - soit immédiatement, soit quand vous faites marcher le programme.

## Syntax Error (ou erreur de grammaire)

Si le message: **Syntax Error** apparaît sur l'écran, l'ordinateur vous dit qu'il ne comprend pas une des instructions que vous avez tapées. Par exemple si vous tapez:

**printt [ENTER]**

Le message suivant apparaît:

**Syntax Error**

Car l'ordinateur ne connaît pas la commande **printt**.

Si vous vous trompez dans une ligne de programme comme:

**10 printt "abc" [ENTER]**

Le message **Syntax Error** n'apparaîtra que lorsque vous faites marcher le programme.

Tapez:

**run [ENTER]**

Sur l'écran vous voyez:

**Syntax error in 10**

**10 printt "abc"**

Ce message vous dit dans quelle ligne se trouve l'erreur et vous présente la ligne du programme avec le curseur- éditeur pour que vous puissiez corriger l'erreur.

Appuyez sur le curseur droit [→] jusqu'à ce que le curseur soit au dessus du **t** de **printt**. Puis pressez la touche [**CLR**] pour enlever le **t** superflu, puis pressez [**ENTER**] pour introduire la ligne correcte dans l'ordinateur.

Tapez: **run [ENTER]**... et vous verrez que l'ordinateur a accepté l'instruction et affiché abc

# Une introduction aux mots réservés du BASIC AMSTRAD

Dans le chapitre 8, vous trouverez une description illustrée de tous les mots réservés du BASIC d'AMSTRAD. Nous allons maintenant présenter quelques uns des plus employés.

## CLS

Tapez: cls (cela remet l'écran tout propre...). Vous pouvez le taper en majuscules ou minuscules. Puis frappez la touche [ENTER]. L'écran s'efface complètement et le mot Ready (=prêt) avec le curseur se voit en haut à gauche de l'écran.

## PRINT

On s'en sert chaque fois qu'on veut afficher des caractères, des mots ou des chiffres sur l'écran. Si vous tapez l'instruction suivante:

```
print "Bonjour" [ENTER]
```

Sur l'écran on voit:

Bonjour

Les guillemets " " sont utilisés pour dire ce qu'on veut afficher-Bonjour-qui est apparu sur l'écran aussitôt que [ENTER] a été frappé. Tapez c l s [ENTER] pour dégager l'écran.

## RUN

L'exemple précédent montrait un programme simple d'une ligne. La plupart des programmes ont plusieurs lignes. Devant chaque ligne, un numéro est d'abord placé: ces numéros disent à l'ordinateur dans quel ordre il faut exécuter le programme.

### Remarque IMPORTANTE:

Il faut bien faire la différence entre le zéro -0- et le O.Pour vous cela semble naturel,mais on doit bien faire la différence quand on tape; c'est pour cela que le zéro est écrit et affiché avec une barre en travers.

Quand on appuie sur [ENTER], la ligne est stockée en mémoire jusqu'à l'exécution du programme. Tapez:

```
10 print "Bonjour" [ENTER]
```

Remarquez que quand vous pressez [ENTER], Bonjour n'est pas affiché sur l'écran. Pour afficher, il faut taper run:

```
run [ENTER]
```

Vous voyez maintenant Bonjour sur l'écran. Un petit truc: au lieu de taper tout le temps print vous pouvez utiliser le symbole ? comme par exemple

**10 ? "Bonjour" [ENTER]**

## LIST

Lorsqu'un programme a été mis en mémoire, on peut vérifier ce qu'on a tapé en demandant la liste du programme. Tapez

**List [ENTER]**

et sur l'écran on voit

**10 PRINT "Bonjour"**

qui est le programme stocké dans la mémoire. Avez vous remarqué que le mot PRINT est maintenant en majuscule? Ce qui signifie que l'ordinateur a accepté le mot PRINT comme un mot réservé du BASIC.

Tapez **cls [ENTER]** pour éclaircir l'écran, ce qui n'enlève pas votre programme de la mémoire.

## GOTO

Le mot GOTO dit à l'ordinateur d'aller d'une ligne à l'autre pour sauter une ligne ou fabriquer un circuit interne. Tapez cela:

**10 print "Bonjour" [ENTER]**

**20 goto 10 [ENTER]**

puis run [ENTER]

et vous voyez Bonjour s'afficher sans interruption une ligne après l'autre. Pour arrêter, appuyez sur **[ESC]**. Pour recommencer, une autre touche. Pour arrêter complètement, tapez **[ESC]** deux fois de suite.

Faites

**cls [ENTER]**

Pour avoir le mot Bonjour affiché sur toute la ligne, il suffit de mettre un point-virgule à la fin de la ligne 10, après les guillemets.

**10 Print "Bonjour" ; [ENTER]**

**20 goto 10 [ENTER]**

**run [ENTER]**

Le point-virgule commande à l'ordinateur d'afficher le caractère suivant après l'autre, 13 colonnes après le premier caractère. C'est intéressant pour afficher en colonnes distinctes, mais si le nombre de caractères dépasse 12, le caractère suivant est décalé de 13 colonnes.

De nouveau, pour sortir du programme, appuyez, sur **[ESC]** deux fois de suite. Pour rafraîchir complètement la mémoire de l'ordinateur, appuyez sur **[SHIFT]**, **[CTRL]** et **[ESC]** dans cet ordre, et l'ordinateur se remet à zéro.

## INPUT

Cette commande sert à dire que l'ordinateur attend que l'on ait tapé quelque chose avant de continuer. Par exemple:

```
10 input "Quel est votre age"; age [ENTER]
20 print "Vous paraissez nettement moins que vos
      "; age; "ans" [ENTER]
```

Sur l'écran on voit

**Quel est votre age?**

Si vous donnez votre age, mettons 18, puis **[ENTER]**, on voit alors s'afficher:

**Vous paraissez nettement moins que vos 18 ans.**

Cet exemple montre l'utilisation de la commande INPUT et d'un nombre variable. Le mot **age** est mis en mémoire à la fin de la ligne 10 pour que l'ordinateur associe le mot **age** à tout nombre tapé après le point d'interrogation, et affiche ces nombres à la place du mot **age** tapé à la ligne 20. Et bien que nous ayons utilisé le mot **age** pour la variable **age**, on aurait pu aussi bien utiliser une lettre **a**, ou **b**, ou **z**.

Remettez l'ordinateur à zéro avec les touches (**[CTRL][SHIFT]** et **[ESC]**). Si vous voulez un INPUT fabriqué à partir de caractères (lettres ou lettres et chiffres), le signe du dollar \$ doit être placé à la fin de la variable.

Tapez le programme suivant (en faisant attention à mettre un espace après le **r** de Bonjour et avant le **m** de mon).

```
10 input "Quel est ton nom"; nom$ [ENTER]
20 print "Bonjour "; nom$, "mon nom est Arnold"
run[ENTER]
```

Sur l'écran on voit

**Quel est ton nom?**

Tapez votre nom puis **[ENTER]**

si votre nom est Daniel, vous verrez sur l'écran

**Bonjour Daniel, mon nom est Arnold**

(petite remarque, Arnold était le nom de code de l'AMSTRAD CPC464 pendant son développement)

Nous avons utilisé **nom\$** comme variable (= ensemble de lettres pouvant changer), on aurait pu utiliser, **a\$**. Nous allons maintenant combiner les deux exemples précédents en un seul programme.

Faisons à nouveau (**[CTRL][SHIFT]** et **[ESC]**). Puis tapons le programme suivant:

```
5 cls [ENTER]
10 input "Quel est ton nom"; a$[ENTER]
20 input "Quel est ton age"; b [ENTER]
30 print "Je dois dire " ; a$ " que tu ne fais pas
      tes" ; b"ans" [ENTER]
      run[ENTER]
```

Dans le programme nous avons utilisé 2 variables, a\$ pour le nom et b pour l'age.  
Sur l'écran on voit:

**Quel est ton nom?**

Tapez votre nom (Daniel) puis [**ENTER**]. La question suivante apparaît:

**Quel est ton age?**

Maintenant tapez votre age (18) puis [**ENTER**]. Ceci apparaît:

**Je dois dire Daniel que tu ne fais pas tes 18 ans**

## EDITER UN PROGRAMME

Si une des lignes du programme a été tapée incorrectement, donnant un message -syntax error - ou un autre message, il est possible d'éditer (autrement dit modifier) cette ligne sans avoir à la retaper. Supposons que le programme précédent ait été mal tapé, comme suit:

```
5 clss
10 input "Quel est ton nom" ; a$ [ENTER]
20 input "Quel est ton age" ; b [ENTER]
30 print "Je dois dire" ; a$ "que tu ne fais
pas tes" ; b"ans" [ENTER]
```

Il y a trois erreurs dans le programme ci-dessus

dans la ligne 5 on a tapé **clss** au lieu de **cls**

dans la ligne 10 on a tapé **to** au lieu de **ton**

dans la ligne 30 on a oublié l'espacement entre **dire** et les guillemets "

Il y a trois méthodes pour éditer un programme. La première est de retaper, entièrement la ligne. Quand une ligne est retapée et entrée en mémoire, elle remplace la ligne qui avait le même numéro. La deuxième méthode est d'éditer et la troisième est appelée Copy Cursor autrement dit Copie avec l'aide du Curseur.

### Méthode par Edition

Pour corriger la ligne 5, tapez:

**edit 5 [ENTER]**

La ligne 5 est imprimée avec le curseur surimposé sur le 5. Pour enlever le s en trop dans **clss**, appuyer sur le curseur droit [→] jusqu'à ce qu'il soit sur le dernier s, puis appuyer sur la touche [**CLR**]. Le s a disparu. Maintenant appuyez [**ENTER**] et la ligne 5 est correcte dans la mémoire. Il suffit de taper:

**List [ENTER]**

Pour vérifier que la ligne 5 est correcte.

## Méthode par copie avec le curseur.

Pour corriger les fautes des lignes 10 et 30, appuyez et maintenez la touche [SHIFT] en pressant sur le curseur [↑]; lorsque le curseur est au début de la ligne 10, vous voyez que le curseur principal en bas n'a pas bougé. Puis pressez la touche verte [COPY] jusqu'à ce que le curseur soit placé dans l'espace entre to et nom. La ligne 10 est réécrite en même temps en bas et le curseur principal stoppe à la même place que le curseur de copie. Puis tapez la lettre n, qui apparaît sur la ligne du bas seulement.

Le curseur principal a bougé mais le curseur de copie est resté à sa place. Maintenant appuyez la touche [COPY] pour afficher la totalité de la ligne 10. Pressez [ENTER] et cette nouvelle ligne 10 est mise en mémoire. Le curseur de copie disparait et le curseur principal se met en dessous de la ligne 10. Pour corriger la deuxième faute, maintenez la touche [SHIFT] et appuyez sur le curseur [↑] pour faire apparaître le curseur de copie au début de la ligne 30.

Appuyez [COPY] jusqu'à ce que le curseur de copie soit superposé sur les guillemets juste après dire. Appuyez une seule fois sur la barre d'espacement. Un espace va s'introduire sur la ligne du bas. Restez appuyé sur la touche [COPY] jusqu'à la fin de la ligne. Puis [ENTER] et vous pouvez demander la liste du programme en tapant: list [ENTER].

Puis remise à zéro avec les touches [CTRL], [SHIFT] et [ESC].

## IF THEN

Nous allons améliorer le programme précédent en se servant des commandes IF et THEN. (Si .. Alors).

Tapez le programme suivant, en remarquant que nous introduisons deux symboles nouveaux. < signifie moins grand que et se situe à côté de la lettre M, > signifie plus grand que et est à côté du signe < moins grand que.

```
5 cls [ENTER]
10 input "Quel est ton nom" ; a$ [ENTER]
20 input "Quel age as-tu" ; age [ENTER]
30 if age <13 then 60 [ENTER]
40 if age <20 then 70 [ENTER]
50 if age >19 then 80 [ENTER]
60 print "Donc " a$ " Tu n'est pas encore un
adolescent avec tes" age "ans" : end [ENTER]
70 print "Donc " a$ " Tu es en pleine adolescence avec
tes" age "ans" : end [ENTER]
80 print "Eh bien "; a$ " Vous êtes supposé
être un adulte avec vos" age "ans" [ENTER]
```

Pour vérifier que le programme est correct, faites

list [ENTER]

Puis

**run [ENTER]**

Et vous pouvez répondre aux questions de l'ordinateur et voir ce qui se passe.

Vous pouvez constater les effets du **IF** (=si) et du **THEN** (=alors) comme commandes dans un programme. Nous avions aussi ajouté le mot **end** (=fin) à la fin des lignes 60 et 70. Ce mot réservé **END** est utilisé pour finir un programme. S'il n'était pas là le programme continuerait à marcher et afficherait aussi les lignes 70 et 80. De même pour le **END** à la fin de la ligne 70.

Les deux points: avant le mot **END** le séparent de l'instruction précédente. Les deux points: sont utilisés pour séparer des instructions et en mettre plusieurs sur une même ligne. Nous avons aussi ajouté la ligne 5 pour remettre un écran net et nous le ferons au début de chaque programme, pour rendre les choses plus claires.

Les autres mots réservés associés à **IF** et à **THEN** comprennent **ELSE**, **OR** et **GOTO**. Ces mots seront expliqués avec la commande **IF** au cours des pages et chapitres suivants.

Remise à zéro avec **[CTRL] [SHIFT]** et **[ESC]**.

## FOR TO NEXT

Nous allons utiliser maintenant les commandes **FOR... TO ... NEXT**. Dans cet exemple, nous allons montrer comment l'ordinateur peut afficher la table de  $12(1 \times 12, 2 \times 12, 3 \times 12, \text{etc}...)$

Tapez ce qui suit, en notant que le symbole \* signifie multiplier (on ne peut pas utiliser x, car pour l'ordinateur c'est seulement la lettre x)

```
5 cls [ENTER]
10 for a = 1 to 20 [ENTER]
20 print a"*12="a*12 [ENTER]
30 next a [ENTER]
run [ENTER]
```

Les colonnes ne sont pas très nettes, on va donc taper le programme suivant:

```
5 cls
10 for a = 1 to 9
20 print a"*12="a*12
30 next a
40 for a= 10 to 20
50 print a"*12="a*12
60 next a
run
```

Essayez ce programme pour les tables de multiplication pour d'autres nombres: si vous voulez voir la table de 17, il suffit de remplacer le nombre 12 dans les lignes 20 et 50 par le nombre 17. Retour à la case départ en appuyant sur les touches **[CTRL] [SHIFT]** et **[ESC]**.

Il est possible de préciser des escaliers de valeurs en utilisant la commande **STEP** avec le **FOR TO NEXT**. Pour plus de renseignements, étudiez la description de **FOR** dans le chapitre 8.

## ARITHMETIQUE SIMPLE

Votre CPC464 peut vous servir de calculatrice très facilement.

Pour mieux le comprendre, faites donc les exemples suivants, en utilisant ? au lieu de la commande PRINT ; la réponse sera affichée aussitôt que la touche [ENTER] est frappée.

### ADDITION

(le signe plus + se fait en pressant [SHIFT] et ;)

Tapez

? 3 + 3 [ENTER]  
6

(Vous n'avez pas à taper le signe = car il est sous-entendu lorsque vous faites [ENTER])

Tapez

? 8 + 4 [ENTER]  
12

### SOUSTRACTION

(le signe moins – se fait avec la touche =, sans [SHIFT])

Faites

? 4 – 3 [ENTER]  
1

Puis

? 8 – 4 [ENTER]  
4

### MULTIPLICATION

(Le signe de la multiplication est \* et s'obtient en pressant [SHIFT] et :)

Faites:

? 3 \* 3 [ENTER]  
9

Puis:

? 8 \* 4 [ENTER]  
32

## DIVISION

(on utilise la barre / en dessous du point d'interrogation ? pour la division)

Faites:

?3 / 3 [ENTER]  
1

Puis:

?8 / 4 [ENTER]  
2

## RACINE CARREE

Pour trouver la racine carrée d'un nombre vous utilisez **sqr ( )**. Le nombre dont on veut extraire la racine doit être entre les parenthèses.

Faites:

?sqr (16) [ENTER] (C'est la même chose que  $\sqrt{16}$ )  
4

Puis:

?sqr (100) [ENTER]  
10

## PUISSEANCES

(C'est le symbole  $\uparrow$ , en dessous de f, donc pas de [SHIFT])

exemples de puissances:  $3^2$  (3 au carré),  $3^3$  (3 au cube),  $7^5$  (7 puissance 5, c'est à dire  $7 \times 7 \times 7 \times 7 \times 7$ )

Faites

?3 ↑ 2 [ENTER] (c'est la même chose que  $3^2$ )  
9

Puis

?8 ↑ 4 [ENTER] (c'est pareil que  $8^4$ )  
4096

## RACINES CUBIQUES

On peut facilement calculer les racines cubiques en se servant d'un exemple semblable aux racines carrées (racine carrée = puissance -2 racine cubique = puissance -3)

Pour trouver la racine cubique de 27 (qu'on écrit  $\sqrt[3]{27}$ )

Faites:

?  $27 \uparrow (1/3)$  [ENTER]  
3

Puis pour la racine cubique de 125

?  $125 \uparrow (1/3)$  [ENTER]  
5

## CALCULS COMPOSÉS

Les calculs avec addition, multiplication, etc sont connus de l'ordinateur, mais il faut faire attention à la priorité.

La priorité va à la multiplication, puis division, puis addition, enfin soustraction.

Ces priorités interviennent quand il y a ces quatre opérations et nous les expliquerons un peu plus loin en y ajoutant les puissances.

Soit le calcul suivant

$3+7-2*7/4$

On pourrait penser que le calcul est fait de la manière suivante:

$$\begin{aligned} & 3+7-2*7/4 \\ &= 8*7/4 \\ &= 56/4 \\ &= 14 \end{aligned}$$

En réalité la calcul est fait comme suit:

$$\begin{aligned} & 3+7-2*7/4 \\ &= 3+7-14/4 \\ &= 3+7-3,5 \\ &= 10-3,5 \\ &= 6,5 \end{aligned}$$

Pour le prouver, il suffit de le taper:

?  $3+7-2*7/4$  [ENTER]  
6.5

On peut changer la manière dont l'ordinateur calcule en ajoutant des parenthèses, l'ordinateur effectuant d'abord les opérations entre parenthèses.

Faisons donc, pour voir

?  $(3+7-2)*7/4$  [ENTER]  
14

## ENCORE DES PUISSANCES

Quand on veut se servir de très grands ou très petits nombres dans les calculs, il est quelquefois utile de se servir de la notation scientifique. La lettre E est utilisée en informatique pour les exposants des nombres en base 10. (Vous pouvez utiliser E ou e, sans oublier que pour certains mathématiciens e peut avoir un sens différent avec les logarithmes népériens)

Par exemple 300 est pareil que  $3 \times 10^2$ . En notation scientifique, on écrit 3E2. De même, 0.03 est pareil que  $3 \times 10^{-2}$ . On l'écrit 3E-2. Essayez les exemples suivants:

**1.  $30 \times 10$**

Frapper

?30\*10 [ENTER]  
300

Ou bien

?3e1\*1e1

**2.  $3000 \times 1000$**

Faites

?3e3\*1e3 [ENTER]  
3000000

**3.  $3000 \times 0.001$**

Tapez:

?3e3\*1e-3 [ENTER]  
3



# Eléments E3: Graphiques, modes et sons

L'ordinateur personnel en Couleurs Amstrad CPC464 a trois modes d'affichage pour l'écran: Mode 0, Mode 1 et Mode 2.

Quand on met l'ordinateur en marche, il est automatiquement en mode 1.

Pour comprendre le système des modes, après avoir mis en route, appuyez sur le chiffre 1 et maintenez appuyé jusqu'à ce qu'il y ait 2 lignes pleines de 1. Maintenant, si vous comptez le nombre de 1 sur une ligne, vous verrez qu'il y en a 40: c'est à dire qu'en Mode 1, il y a 40 colonnes. Appuyez sur [ENTER], vous aurez le message '**Syntax Error**', mais n'en tenez pas compte, c'est pour mieux montrer rapidement le principe. Faites ensuite:

## mode 0 [ENTER]

Vous voyez que les caractères sur l'écran sont plus larges. Appuyez sur le chiffre 1 pour avoir 2 lignes pleines de 1 et si vous les comptez, vous voyez qu'il y en a 20 par ligne. Donc il y a 20 colonnes en Mode 0. Appuyez sur [ENTER] de nouveau. Et tapez:

## mode 2 [ENTER]

Vous verrez que l'écriture est plus petite dans ce Mode, et que si vous tapez une ligne de 1, il y en a 80 sur une ligne. En résumé.

Mode 0 = 20 colonnes

Mode 1 = 40 colonnes

Mode 2 = 80 colonnes

Tapez [ENTER] une fois de plus.

## COULEURS

Il y a un choix de 27 couleurs. Sur un moniteur vert, on les voit comme des tons de vert. Si vous avez acheté le moniteur vert GT64, vous pouvez acheter l'Adaptateur Péritel pour vous servir de votre télé couleurs.

En Mode 0, on peut avoir 16 des 27 couleurs en même temps sur l'écran, en Mode 1, on peut avoir 4 sur 27, et en Mode 2, 2 couleurs sur 27.

On peut changer la couleur de la bordure du cadre extérieur appelé **BORDER**, la couleur du papier, ou l'endroit où on écrit appelé **PAPER** et la couleur du stylo, **PEN**, c'est à dire la couleur des caractères, indépendamment les unes des autres.

Les 27 couleurs disponibles sont indiquées dans le tableau 1 chacun avec le numéro de référence de l'encre, **INK**, ou pour simplifier de la couleur.

INK No.	Couleur de l'encre	INK No.	Couleur de l'encre INK
0	Noir	14	Bleu Pastel
1	Bleu	15	Orange
2	Bleu Vif	16	Rose
3	Rouge	17	Magenta Pastel
4	Magenta	18	Vert Vif
5	Mauve	19	Vert marin
6	Rouge Vif	20	Turquoise Vif
7	Pourpre	21	Vert citron
8	Magenta Vif	22	Vert Pastel
9	Vert	23	Turquoise Pastel
10	Turquoise	24	Jaune Vif
11	Bleu ciel	25	Jaune Pastel
12	Jaune	26	Blanc brillant
13	Blanc		

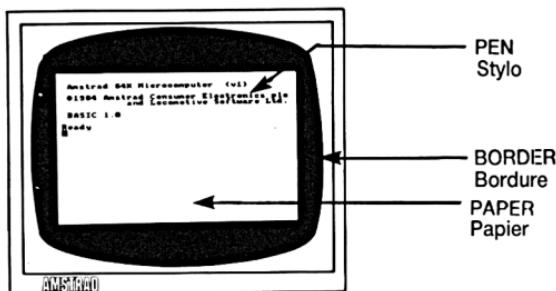
**Tableau 1:** les numéros d'encre (INK) et les couleurs.

Comme nous le disions auparavant, l'ordinateur est en mode 1 à l'origine. Pour retourner au mode 1, tapez:

**mode 1 [ENTER]**

lorsqu'on commence les couleurs sont comme suit

BORDER	(cadre)	= couleur No. 1 (bleu)
PAPER	(écran)	= couleur No. 1 (bleu)
PEN	(caractères)	= couleur No. 24 (jaune Vif)



**BORDER** est le cadre entourant le **PAPER**. (Au départ, **BORDER** et **PAPER** sont bleus tous les deux). **PEN** (stylo) donne la couleur des caractères.

Pour expliquer un peu mieux, on peut associer **PEN** et **PAPER** à une feuille de papier et un stylo. On peut changer l'encre (**INK**) dans un stylo, comme la couleur des caractères. Comme la couleur de la feuille de papier peut changer, ainsi peut changer la couleur du **PAPER** sur l'écran.

Pour changer la couleur du cadre, **BORDER**, tapez:

**border 0 [ENTER]**

La couleur du cadre passe du bleu au noir. Dans le tableau 1 vous voyez que 0 est le noir. On peut changer sa couleur en tapant: border X, X étant la couleur désignée par un nombre de 0 à 26.

Puis faites:

**cls [ENTER]**

pour un écran neuf. Pour voir changer la couleur de **PAPER**:

**paper 2 [ENTER]**

la couleur du fond se change en turquoise vif. Faites ensuite

**cls [ENTER]**

Pour voir changer la couleur des caractères, faites

**pen 3 [ENTER]**

la couleur d'écriture a changé et le mot Ready est affiché en Rouge vif. De nouveau, faisons

**cls [ENTER]**

C'est ici qu'il faut regarder le tableau 2 et garder un oeil sur le tableau 1. (facile à dire...). Au départ, le numéro du **PAPER** est 0: Sur le tableau 2, Paper 0 en mode 1 a pour couleur 1 = bleu, si on regarde le tableau 1.

En changeant **PAPER** en **PAPER 2**: sur la colonne **PAPER 2**, en Mode 1, vous voyez le nombre 20. Si vous regardez maintenant tableau 1, vous voyez que 20 est la couleur turquoise vif!

No.de Papier/stylo	Couleur d'Encre (INK)		
	Mode 0	Mode 1	Mode 2
0	1	1	1
1	24	24	24
2	20	20	1
3	6	6	24
4	26	1	1
5	0	24	24
6	2	20	1
7	8	6	24
8	10	1	1
9	12	24	24
10	14	20	1
11	16	6	24
12	18	1	1
13	22	24	24
14	Flashing 1,24	20	1
15	Flashing 16,11	6	24

tableau 2. PAPER/ PEN/MODE/INK reference



Un peu compliqué, mais tout deviendra plus facile avec la pratique, et croyez moi, c'est plus facile à faire qu'à expliquer.

Au départ, on a PAPER 0 couleur no. 1 (bleu) et également PEN 1 couleur no. 24 (jaune vif), cela en mode 1.

Quand on écrit PEN 3, on change de stylo et on prend le stylo No. 3 du tableau 2 dont la couleur est 6: allons voir au tableau 1 et la couleur 6 est rouge vif.

Vous me suivez jusque là?

Nous utilisons donc PAPER 2 (couleur 20 = turquoise vif) et PEN 3 (couleur 6 = rouge vif).

On va maintenant changer la couleur de l'encre dans le stylo et la couleur du fond (donc du PAPER). Comme exemple, nous allons changer PAPER 2 en noir (couleur = 0) et PEN 3 en blanc brillant (couleur = 26)

Pour cela on tape

ink 2,0 [ENTER]  
ink 3,26 [ENTER]

Bon. Revenons à la case départ et appuyons sur les touches **[CTRL]**, **[SHIFT]** et **[ESC]**: on revient à PAPER 0 de couleur 1 = bleu et PEN 1 de couleur 24 = jaune vif.

Si on avait à taper les couleurs au départ, on ferait ink 0,1 et ink 1,24.

Faisons un petit changement

ink 0,0 [ENTER]

puis

ink 1,26 [ENTER]

on doit voir des lettres blanches sur fond noir.

## COULEURS INTERMITTENTES (FLASH)

On peut faire varier les caractères d'une couleur à l'autre en utilisant un autre numéro de couleur pour la définition INK de la commande PEN.

Tapez donc:

INK 1,26,6 [ENTER]

la couleur des caractères va changer rapidement entre le blanc brillant et le rouge vif.

1 est le numéro du stylo (PEN), 26 et 6 les numéros des couleurs. On peut faire la même chose avec la couleur du fond.

ink 0,9,24 [ENTER]

Attention les yeux ! Arrêtons vite ces effets: **[CTRL]**, **[SHIFT]** et **[ESC]**.

Notez en regardant le tableau 2, qu'en mode 0, deux des numéros de PEN ou de PAPER sont intermittents d'origine, autrement dit, il n'est pas nécessaire d'ajouter un autre chiffre à la commande INK.

On le voit en faisant:

```
mode 0 [ENTER]  
pen 15 [ENTER]
```

et vous voyez sur l'écran, le mot Ready va alterner entre bleu ciel et rose! Si vous faites ensuite:

```
paper 14 [ENTER]  
cls [ENTER]
```

maintenant, en plus de Ready qui flash, le fond de PAPER change entre jaune et bleu.

On peut changer les couleurs du flash d'origine en donnant un nouvelle commande PEN ou PAPER. Par exemple:

```
ink 15,0,26 [ENTER]
```

la couleur doit varier du blanc au noir, et le numéro du stylo est 15.

Enfin on peut faire alterner la couleur du cadre en ajoutant un autre numéro de couleur à la commande BORDER.

Par exemple:

```
border 6,9 [ENTER]
```

la couleur du cadre passe du rouge vif au vert.

Nous revenons à notre écran bleu et or : **[CTRL]**, **[SHIFT]** et **[ESC]**.

## PROGRAMME DE DÉMONSTRATION

Pour voir ce qu'on peut faire avec les couleurs et comprendre un peu mieux le système **PEN**, **BORDER**, **INK** et **PAPER**, tapez ce programme et faites le marcher.(run).

Nous avons ajouté quelque commandes de son, qui seront expliquées un peu plus tard.

```
10 mode 0: ink 0,2:ink 1,24: paper 0[ENTER]
20 pen 1: for b=0 to 26: border b[ENTER]
30 Locate 3,12:print"COULEUR du CADRE";b ENTER]
40 Sound 4,(40-b)[ENTER]
50 for t=1 to 600:next t:next b:cls [ENTER]
60 for p=0 to 15:paper p:pen 5:print "papier";
   p:print [ENTER]
70 for n=0 to 15:pen n:print "STYLO";n[ENTER]
80 sound 1,(n*20+p)[ENTER]
90 for t=1 to 100:next t:next n[ENTER]
100 for t=1 to 1000:next t:cls:next p[ENTER]
110 cls:paper 0:pen 1:locate 7,12:print
   "FIN":for t=1 to 2000: next t [ENTER]
120 mode 1: border 1:ink 0,1:ink 1,24:
   paper 0:pen 1[ENTER]
run[ENTER]
```

## GRAPHIQUES

A partir de maintenant, nous ne préciserons plus de taper **[ENTER]** à la fin de chaque ligne. Nous supposerons que vous le faites automatiquement.

Il y a un certain nombre de caractères symboliques dans la mémoire de l'ordinateur. Pour afficher un de ces symboles, on utilise le mot réservé **CHR\$( )**. Entre les parenthèses, on met le numéro du symbole, entre 32 et 255.

Faites **[CTRL]**, **[SHIFT]** et **[ESC]** pour la remise à zéro, puis tapez:

```
print CHR$(250)
```

Sur l'écran, vous verrez le caractère No. 250, qui représente un homme marchant vers la droite. Pour voir tous les caractères et symboles sur l'écran avec leur numéro, tapez le programme suivant, en n'oubliant pas de presser **[ENTER]** après chaque ligne.

```
10 for n=32 to 255: print n; "="; chr$(n);
20 next n
run
```

Vous trouverez en Appendice III, tous les caractères avec leurs références. (Vers la fin du guide)

## LOCATE (=placer le curseur)

On se sert de cette commande pour mettre le curseur à un certain endroit de l'écran. En dehors de la commande LOCATE, le curseur se trouve en haut à gauche de l'écran, ce qui correspond en coordonnées x,y à 1,1 (x est la position horizontale, y la position verticale). En Mode 1, il y a 40 colonnes et 25 lignes. Pour mettre un caractère en haut au milieu de l'écran, on fait x=20 et y=1. Essayons:

**mode 1** ... l'écran se vide, le curseur est en haut, à gauche

```
10 locate 20,1  
20 print chr$(250)  
run
```

Vous n'avez pas oublié [ENTER] après chaque ligne? Bravo! Pour vérifier que c'est le haut de l'écran, tapez:

**border 0**

Le cadre sera noir et vous verrez le petit bonhomme au milieu de la ligne supérieure.

En Mode 0, il n'y a que 20 colonnes, mais toujours 25 lignes. Si on tape:

```
mode 0  
run
```

le bonhomme apparaît à droite en haut de l'écran, car si c'est la 20ème colonne, c'est la dernière en mode 0. En mode 2 il y a 80 colonnes. Essayez de devinez où se trouve le petit homme.

```
mode 2  
run
```

revenez au Mode 1, en faisant

**mode 1** (n'oubliez pas [ENTER])

Vous pouvez essayer tout seul de varier les nombres après locate et dans CHR\$(1); Pour l'exemple, essayez

**Locate 20,12: print CHR\$(224)**

Vous voyez une figure souriante sur l'écran.

Pour avoir le symbole 250 répété sur l'écran, il suffit de taper:

```
5 cls  
10 for x=1 to 39  
20 locate x,20  
40 print chr$(250)  
50 next x  
60 goto 5  
run
```

Faites [ESC] deux fois pour arrêter.

Pour effacer le caractère précédent avant d'afficher le suivant:

**40 print " "; chr\$(250)**

(la nouvelle ligne 40 remplace automatiquement la précédente ligne 40)

Tapez

**run**

pour que le mouvement soit plus joli, ajoutez la ligne suivant:

**30 call &bd19**

On peut encore améliorer ce programme en ajoutant des petites pauses et en utilisant d'autres symboles en plus. Pour ce faire, vous allez taper:

**list**

Puis ajouter les lignes suivantes au programme:

```
60 for n=1 to 300: next n
65 for x= 39 to 1 step-1
70 locate x,20
75 call &bd19
80 print chr$(251); ""
85 next x
90 for n=1 to 300: next n
95 goto 10
run
```

Essayez maintenant ce petit programme. Nous avons ajouté quelques petites commandes et mots réservés qui seront expliqués dans les chapitres suivants. Pour le moment, il suffit d'écrire le programme suivant:

```
new
10 mode 1
20 locate 21,14:print chr$(244)
30 tag
40 for x=0 to 624 step 2
50 mover-16,0
60 if x<308 or x>340 then y=196:goto 90
70 if x<324 then y=x-104:goto 85
80 y=536-x
85 sound 1,0,20,7
90 move ox, oy:print " ";:ox=x:oy=y
100 move x,y
110 if (x mod 4)=0 then print chr$(250);
    else print chr$(251);
120 for n=1 to 4: call &bd19:next n
130 next x
140 tagoff
150 goto 20
run
```

## PLOT (détermine un point sur l'écran)

Contrairement à la commande LOCATE, PLOT sert à fixer la position du curseur graphique, en utilisant les coordonnées des pixels (un pixel est un petit élément de l'écran).

REMARQUE: le curseur graphique n'est pas visible et il est différent du curseur des caractères. Il y a 640 pixels horizontaux sur 400 verticaux. Les coordonnées x et y sont définies par rapport au coin à gauche et en bas de l'écran, qui a pour coordonnées 0,0. Ces coordonnées sont les mêmes en Mode 0, en Mode 1 et en Mode 2.

L'ordinateur ayant été remis à zéro en faisant [CTRL] [SHIFT] et [ESC], essayez (sans oublier [ENTER]):

```
plot 320,200
```

Un petit point apparaît au centre de l'écran. Changeons de Mode en tapant:

```
mode 0  
plot 320,200
```

Le point est toujours au milieu de l'écran, mais plus grand. De même si on passe en Mode 2:

```
mode 2  
plot 320,200
```

Le point est toujours au centre, mais il est beaucoup plus petit. Essayez de dessiner des points sur l'écran avec les différents modes pour s'habituer à la commande PLOT. Quand vous avez terminé, revenez en Mode 1 avec un écran net en tapant:

```
mode 1
```

## DRAW (Dessine une ligne)

Remise à zéro: [CTRL] [SHIFT] et [ESC]. La command DRAW dessine une ligne à partir de la position du curseur graphique. Pour voir avec plus de précision ce qui se passe, dessinons un rectangle avec le programme suivant. On commence à fixer le curseur avec la commande. Puis on dessine une ligne à partir de ce point, vers le haut, puis vers la droite, etc...

Tapez:

```
5 cls  
10 plot 10,10  
20 draw 10,390  
30 draw 630,390  
40 draw 630,10  
50 draw 10,10  
60 goto 10  
run
```

Puis appuyez [ESC] deux fois pour sortir du programme. Il faut alors ajouter les lignes suivantes pour dessiner un deuxième rectangle à l'intérieur du premier. Tapez les lignes suivantes:

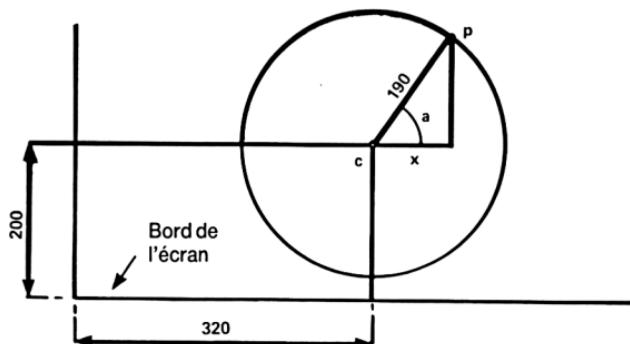
```
60 plot 20,20
70 draw 20,380
80 draw 620,380
90 draw 620,20
100 draw 20,20
200 goto 10
run
```

Appuez sur [ESC] deux fois pour sortir du programme.

## CERCLES

On peut dessiner des cercles en utilisant les commandes **PLOT** ou **DRAW**. Sur la figure suivante un point p de coordonnées x et y peut être placé avec la commande **PLOT**, en utilisant les formules

$$x=90 \cdot \cos(a) \text{ et } y=190 \cdot \sin(a)$$



## NEW (nouveau)

Nous avons déjà utilisé la commande **NEW** avant même de faire un programme. Elle indique à l'ordinateur d'effacer ce qu'il y a en mémoire, d'une manière semblable à **[CTRL][SHIFT][ESC]**. Cependant, elle diffère dans la mesure où l'écran n'est pas effacé. Ce peut être utile quand on veut garder un programme périmé sur l'écran pendant qu'on tape un nouveau programme.

Revenons à notre cercle: dans le programme précédent, on avait fixé des points par rapport au coin en bas à gauche de l'écran (coordonnées 0,0). Si on veut un cercle au milieu de l'écran, il faut dessiner par rapport à un centre 320,200 qui est le milieu de l'écran.

Un programme pour avoir un cercle suit:

```
new
5 cls
10 for a=1 to 360
15 deg
20 plot 320, 200
30 plot 320+190*cos(a),200+190*sin(a)
40 next
run
```

Le rayon du cercle peut être plus petit, il suffit de changer le nombre 190 (190 est le nombre de pixels, disons que ce cercle a un rayon de 190 pixels)

Pour voir un cercle déterminé d'une manière différente (en radians), enlever la ligne 15 en tapant:

15

Pour faire un cercle plein (un disque) avec des lignes tracées à partir du centre, éditez (=modifiez) la ligne 30 en remplaçant le mot **PLOT** par le mot **DRAW**. La ligne 30 devient

```
30 draw 320+190*cos(a),200+190*sin(a)
```

Essayez cela avec et sans la ligne 15.

Notez que dans la ligne 40, il y a **next** tout seul et pas **next a**. On peut mettre **NEXT** tout seul dans un programme. L'ordinateur déterminera à quel **FOR** il se rapporte. Dans les programmes où il y a beaucoup de **FOR** et **NEXT**, il vaut mieux ajouter la variable après **NEXT** afin de mieux s'y retrouver.

## ORIGIN

Dans le programme précédent, on s'est servi de la commande PLOT pour placer le centre du cercle, puis on a ajouté les coordonnées x,y de la circonference. Nous pouvons utiliser au lieu de cela la commande ORIGIN (attention, pas de E à ORIGIN, c'est un mot RÉSERVÉ), placer le centre d'abord puis représenter la circonference. Vérifions avec le programme suivant:

```
new
 5 cls
10 for a=1 to 360
15 deg
20 origin 320,200
30 plot 190*cos(a),190*sin(a)
40 next
run
```

De nouveau, on peut modifier les lignes 15 et 30 pour enlever deg (pour degrés) ou dessiner (DRAW) le disque à partir du centre.

Pour avoir quatre petits cercles sur l'écran, faisons le programme suivant:

```
new
 5 cls
10 for a=1 to 360
15 deg
20 origin 196,282
30 plot 50*cos(a),50*sin(a)
40 origin 442,282
50 plot 50*cos(a),50*sin(a)
60 origin 196,116
70 plot 50*cos(a),50*sin(a)
80 origin 442,116
90 plot 50*cos(a),50*sin(a)
100 next
run
```

Une fois de plus, on peut supprimer la ligne 15 et modifier les lignes 30, 50, 70 et 90 pour utiliser la commande DRAW.

## GOSUB RETURN

S'il y a dans un programme des instructions qui doivent être exécutées plusieurs fois, on peut utiliser un sous-programme qui est appelé par la commande GOSUB suivi du numéro de la ligne où commence le sous-programme.

La fin d'un sous-programme est indiquée par la commande RETURN. A ce moment, l'ordinateur revient à la ligne qui était juste après la ligne du GOSUB en question.

Dans le programme précédent, l'instruction plot 50\*cos(a). 50\*sin(a) était répétée 4 fois. Cette instruction peut être donnée comme un sous programme et utilisée à chaque fois qu'on en a besoin en utilisant la commande GOSUB.

Tapons le programme suivant.

```
new
 5 cls
10 for a=1 to 360
15 deg
20 origin 196,282
30 gosub 120
40 origin 442,282
50 gosub 120
60 origin 196,116
70 gosub 120
80 origin 442,116
90 gosub 120
100 next
110 end
120 plot 50*cos(a),50*sin(a)
130 return
run
```

REMARQUE: Nous avons mis END (= fin du programme) à la ligne 110; autrement le programme continuerait après la ligne 100 et referait l'instruction 120, qu'on utilise seulement quand la commande GOSUB appelle cette ligne. D'une manière générale, les sous-programmes appelés par des GOSUB sont placés après la fin du programme principal.

Pour finir, essayez donc le programme suivant qui contient pas mal de commandes et mots réservés que vous devriez maintenant comprendre.

```
new
10 mode 0: border 6:paper 0:ink0,0
20 gosub 160:for x=1 to 19:locate x,3
30 pen 15:print"';chr$(238)
40 for t=1 to 50:next t:sound2,(x+100)
50 next x:gosub 160:for b=3 to 22
60 locate 20,b:pen 7:print chr$(252)
70 cls:gosub 160:next b
80 sound 2,0,100,15,0,0,1
90 gosub 160:border 16,24:locate 20,25
100 pen 14:print chr$(253);
110 for t=1 to 1000:next t
120 border 6:gosub 160:for f=3 to 24
130 locate 10,(25-f):pen 2
140 print chr$(144):cls:gosub 160
150 sound 7,(100-f),5:next f:goto 10
160 locate 10,25:pen 12
170 print chr$(239):return
run
```

## LE SON ET LES EFFETS SONORES

Les effets sonores sont produits par un haut-parleur dans l'ordinateur lui-même. Si vous utilisez l'adaptateur Péritel et un poste Télé, mettez le volume au minimum.

Le niveau du son peut être ajusté au moyen du bouton de contrôle de **VOLUME** sur le côté droit de l'ordinateur. Le son peut aussi passer par la fiche d'entrée auxiliaire de votre système stéréo, en utilisant la fiche (I/O) à gauche derrière l'ordinateur. Ceci vous permet d'écouter le son produit par l'ordinateur sur vos haut-parleurs ou votre casque d'écoute.

La commande du son, **SOUND**, a sept paramètres. Les deux premiers doivent être précisés, les autres sont facultatifs. La commande est écrite de la manière suivante:  
**SOUND** type de canal, période du ton, durée, volume, enveloppe de volume, enveloppe de ton, période du bruit.

Dans les exemples qui suivent, nous taperons 1 comme type de canal, autrement dit son numéro de référence.

### PERIODE DU TON (ou période de la note)

Si vous consultez l'appendice VII, vous verrez que la note do normal à une période de ton de 478. Faisons

```
new  
10 sound 1,478  
run
```

Vous entendrez une note courte de do, pendant 0,2 seconde.

### DUREE

Quand il n'y a pas de durée précisée, la durée d'un son est de 0,2 seconde. L'unité de durée est 0,01; pour faire durer la note une seconde, il faudra mettre 100. Pour 2 secondes, il faudra mettre 200; par exemple:

```
10 sound 1,478,200
```

la même note do dure 2 secondes

### VOLUME

Ce nombre détermine le volume de départ d'une note. Ce nombre varie de 0 à 7 normalement, de 0 à 15 si on a précisée une enveloppe de volume. Si rien n'est précisément, 4 est utilisé automatiquement.

```
10 sound 1,478,200,3  
20 for t=1 to 500:next t  
30 sound 1,478,200,7  
run
```

le do de la ligne 10 est au volume 3, puis à la ligne 30, le volume passe à 7.  
Vous entendez la différence.

## **ENVELOPPE DE VOLUME**

La commande d'enveloppe de volume est ENV. Elle a normalement 4 paramètres: les trois derniers paramètres peuvent apparaître pour chacune des 5 sections d'enveloppe disponibles. Nous en utiliserons seulement une pour nos exemples. (Pour plus d'explication, voir chapitre 6)

**ENV** numéro d'enveloppe, nombre de pas, amplitude du pas, durée du pas

### **NUMERO D'ENVELOPPE**

C'est un numéro de référence pour qu'on l'utilise avec la commande SOUND. Il peut varier de 0 à 15.

### **NOMBRE DE PAS**

A utiliser de pair avec la durée du pas, pouvant varier de 0 à 127

### **AMPLITUDE DU PAS**

Peut varier de 0 à 15 normalement, et de -128 à +127 pour des effets spéciaux ou étranges.

### **DUREE DU PAS**

L'unité est en centième de seconde et la valeur peut varier de 0 à 255. La durée la plus longue entre deux pas est donc de 2,55 secondes.

Pour faire des essais avec l'enveloppe de volume faisons le petit essai suivant:

```
5 env 1,10,1,100  
10 sound 1,284,1000,1,1  
run
```

La ligne 10 précise la note avec période de 284, le LA international, durant 10 secondes avec un volume de départ de 1, utilisant l'enveloppe de volume numéro 1, définie à la ligne 5 comme ayant 10 pas, élévant le volume à chaque pas de 1, chaque seconde ( $100 \times 0,01$  sec).

Changez la ligne 5 de la manière suivante et faites run chaque fois pour entendre la différence:

```
5 env 1,100,1,10  
5 env 1,100,2,10  
5 env 1,100,4,10  
5 env 1,50,20,20  
5 env 1,50,2,20  
5 env 1,50,15,30
```

et essayez finalement

```
5 env 1,50,2,10
```

A mi-chemin, le niveau du son reste constant car l'amplitude a varié pendant seulement 5 secondes et le son dure 10 secondes vous pouvez essayer différentes combinaisons pour créer des sons vous-mêmes.

## **ENVELOPPE DE TON**

La commande d'enveloppe de ton est ENT.

Elle a normalement 4 paramètres.

ENT numéro d'enveloppe, nombre de pas, période du pas, durée du pas.

## **NUMERO D'ENVELOPPE**

C'est celui qu'on utilise dans la commande sound, et il peut avoir pour valeur de 0 à 15.

## **NOMBRE de PAS**

Varie de 0 à 239 et s'utilise avec la durée du pas

## **PERIODE de TON du PAS.**

Varie de -128 à +127. Les nombres négatifs donnent une note plus haute (la fréquence augmente). La période la plus courte est 0.

## **DUREE DU PAS**

Exprimée en centième de seconde elle peut varier de 0 à 255 (maximum 2,55 secondes).

Premier exemple pour comprendre l'enveloppe de ton

```
5 ent 1,100,2,2  
10 sound 1,284,200,15,0,1  
run
```

puis variez la ligne 5

```
5 ent 1,100,-2,2  
5 ent 1,100,4,20  
5 ent 1,10,-4,20
```

Puis le programme suivant

```
5 ent 1,2,17,70  
10 sound 1,142,940,15,0,1  
15 goto 5
```

Pressez [ESC] 2 Fois pour vous arrêter.

Maintenant introduisons ensemble les commandes ENT, ENV, SOUND

```
5 env 1,100,1,3  
10 ent 1,100,5,3  
20 sound 1,284,300,1,1,1  
run
```

Puis changez la ligne 10

```
10 ent 1,100,-2,3
```

Enfin

```
5 env 1,100,2,2  
10 ent 1,100,-2,2  
20 sound 1,284,200,1,1,1
```

### **BRUIT**

On peut ajouter un bruit à la fin d'une commande son, en précisant un nombre de 0 à 31.

```
5 env 1,100,3,1  
10 ent 1,100,2,2  
20 sound 1,200,100,1,1,1,5
```

# 1 Apéritifs

Pour ceux qui ont sauté le Cours Elémentaire pour débutants, l'introduction, la mise en route et la familiarisation avec le Clavier. Si vous avez des problèmes et que vous trouvez les mots trop difficiles, il vaut mieux revenir à l'introduction.

Nous allons voir dans ce chapitre

- \* Les conventions utilisées dans ce guide
- \* La mise en marche
- \* Le système du clavier

Même si vous connaissez bien la programmation et les ordinateurs, il vaut mieux s'assurer que vous connaissez le mode d'emploi. Si vous venez d'ouvrir la boîte et êtes impatient de commencer, ce chapitre vous dira tout ce qu'il faut pour satisfaire votre curiosité et commencer à utiliser le BASIC. Ceci est pour ceux qui ont déjà manipulé des ordinateurs. Les débutants devraient commencer par l'introduction et le Cours Elémentaire.

## **IMPORTANT - à LIRE ABSOLUMENT:**

### **TERMES et SYMBOLES UTILISÉS**

Pour rendre plus claires les références faites dans le texte aux touches du clavier, les conventions suivantes seront utilisées dorénavant dans ce guide:

**[ENTER]:** les touches, qui n'affichent pas un caractère sur l'écran, sont imprimées dans le texte de cette manière, entre 2 parenthèses droites [ ].

**QWERTYUIOP:** les touches qui affichent un caractère sont ainsi dans le texte, sans parenthèses droites.

**10 for n=1 to 1000:** du texte qui apparaît à l'écran ou doit être tapé sur le clavier sous cette forme.

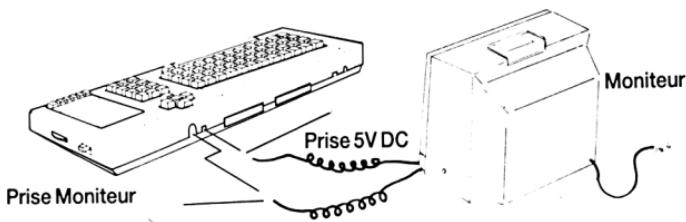
**ATTENTION** à la différence entre le 0 et le Ø (zéro). Nous supposerons que vous appuyez sur **[ENTER]** à la fin de chaque programme et de chaque commande directe.

De plus, nous supposons que vous tapez RUN après avoir écrit chaque programme. Le BASIC transforme tous les mots réservés tapés en minuscules en mots en majuscules, quand le programme est LISTé. A partir de maintenant, les mots réservés seront imprimés en majuscules, puisqu'ils apparaîtront ainsi après la commande LIST. Si vous tapez le programme en minuscules, vous pourrez plus facilement voir vos erreurs de frappe, puisque les mots réservés avec une faute seront affichés en minuscules dans la LISTe du programme.

# 1.1 Ouvrons la boîte!

## 1.1.1 Le moniteur couleur

Quand l'ordinateur est branché comme sur la figure 1, ci-dessous, mettez le moniteur en marche, puis l'ordinateur avec l'interrupteur à glissière.



Après 30 secondes, le moniteur affichera l'image suivante

```
Amstrad 64K Microcomputer (v1)
©1984 Amstrad Consumer Electronics plc
and Locomotive Software Ltd.
BASIC 1.0
Ready
```

Ceci est appelé de plusieurs manières différentes, le "Bonjour" "Hello", "Réveil", après la remise à zéro et ce message indique que l'ordinateur a remis sa mémoire à zéro, ce qui arrive quand on branche l'ordinateur et quand on presse les trois touches [CTRL] [SHIFT] et [ESC] en même temps. Essayez donc pour voir.

Ajustez le bouton "**BRIGHTNESS**" (luminosité) à droite du moniteur. La couleur est définie à la fabrication, et si vous voulez changer la couleur de l'affichage (lettres d'or sur fond bleu) il faut le programmer et vous devez vous reporter aux éléments graphiques et au guide du BASIC du chapitre 8. Si ça ne vous fait rien d'aller un peu vite, voilà un petit programme qui vous donnera une des meilleures et plus lisibles combinaisons de couleurs pour le traitement de textes en mode 2 de 80 colonnes:

```
10 mode 2  
20 ink 1,0  
30 ink 0,13  
40 border 13  
run
```

ou vous pouvez taper cela en mode direct, sans les numéros de ligne.

Si le programme ci-dessus n'a pas de sens pour vous, ou si vous faites des erreurs de frappe, il vaut mieux revenir en arrière au Cours élémentaire au début du guide. Vous y verrez que le mode 2 en 80 colonnes est le plus utile pour développer des programmes - vous pourrez le sauvegarder (après avoir lu jusqu'au chapitre 2) au début d'une cassette vierge, pour ne pas avoir à le taper à tout instant.

**Attention:** la haute luminosité du moniteur couleur signifie qu'il faut éviter la fatigue des yeux et ne pas être trop près du moniteur, ou mettre trop de luminosité. Si vous sentez vos yeux se fatiguer, arrêtez et faites autre chose, mais pour l'éviter il faut prendre les précautions suivantes.

1. Toujours avoir une autre source de lumière que le moniteur, de façon à pouvoir lire un livre facilement. Si vous lisez celui-ci à la lumière du moniteur, ce n'est pas du tout conseillé.
2. Se servir de la luminosité minimum pour voir facilement.
3. S'asseoir aussi loin de l'écran que possible.

Une lampe de bureau à côté du moniteur réduira la fatigue pourvu qu'elle soit placée sur le côté de l'écran pour éviter les réflections.

### 1.1.2 Le Moniteur Monochrome Vert

Le moniteur GT64 a trois contrôles sur le devant. Ils servent à régler la luminosité (**BRIGHTNESS**), le contraste (**CONTRAST**) et l'ajustement Vertical (**V-HOLD**), qui permet de verrouiller l'affichage et évite le roulis de l'image.

L'ajustement vertical **V-HOLD** n'a pas besoin d'ajustement fréquent. Une fois réglé, on peut l'oublier. La luminosité et le contraste doivent être ajustés en fonction des conditions de lumière de la pièce où vous êtes.

Avec un moniteur monochrome (avec une seule couleur qui varie l'intensité des caractères pour donner le contraste entre les différents éléments de l'image), le message sera le même qu'avec le moniteur couleur, sauf que le texte sera vert vif sur vert foncé.

Bien qu'une utilisation forcenée du moniteur GT64 et de votre ordinateur puisse fatiguer les yeux, l'image plus douce devrait vous permettre de travailler plus longtemps. En particulier, vous devriez pouvoir utiliser à plein les possibilités du système de 80 colonnes (80 lettres ou chiffres sur une ligne de l'écran) pour le traitement de textes, vu que la définition de la résolution (un bien grand mot pour dire qu'on peut voir les petits détails plus ou moins bien) d'un écran monochrome est toujours supérieure à celle des moniteurs couleurs (excepté les très chers).

Réglez le contrôle de luminosité [**BRIGHTNESS**] pour avoir une image nette sans que les petits points qui forment les caractères deviennent trop flous.

Pour avoir un système en 80 colonnes, faites le programme suivant pour choisir le type d'affichage qui vous convient, il suffit de taper la touche [**ESC**] deux fois de suite (un message de \*Break\* apparaîtra, il signifie seulement qu'on a interrompu le programme, mais il est toujours en mémoire.)

```
10 REM pour choisir votre type d'image
20 FOR n=0 TO 26
30 MODE 2
40 INK 1,n
50 INK 0,(26-n)
55 BORDER n
60 LOCATE 15,12: PRINT"tapez sur une
      lettre quelconque pour avoir un autre écran"
70 a$=INKEY$
80 IF a$="" GOTO 70
90 NEXT
100 GOTO 20
```

### 1.1.3 Le boitier d'alimentation adaptateur Péritel MP1

Le MP1 est un appareil en option que vous pouvez acheter si vous avez un moniteur vert GT64. Le MP1 vous permet d'utiliser l'ordinateur avec votre poste télé couleurs et ainsi de profiter des capacités couleurs de votre ordinateur CPC464.

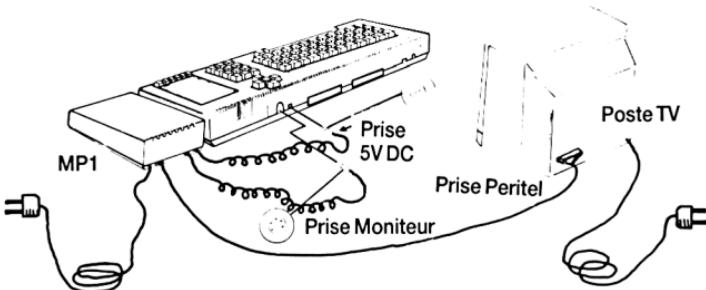


Figure 2: Branchement de l'adaptateur péritel MP1 à l'ordinateur et à votre poste Télé couleurs.

L'adaptateur MP1 doit être placé à droite de l'ordinateur sur une table à proximité de la télé et d'une prise de courant. Réglez le volume de la télé au minimum. Branchez le MP1, puis mettez le poste télé en marche, enfin l'ordinateur avec l'interrupteur à glissière sur le côté droit.

La lampe rouge sur le dessus de l'ordinateur s'allume et vous devez régler les contrôles de luminosité, de contraste et de couleurs pour obtenir l'image suivante:

```
Amstrad 64K Microcomputer (v1)
©1984 Amstrad Consumer Electronics plc
and Locomotive Software Ltd.
BASIC 1.0
Ready █
```

Figure 3: message d'accueil

Ajustez les réglages de façon à avoir une bonne image avec des caractères jaune d'or sur fond bleu Roi.

# 1.2 Premiers pas

Le message que vous voyez sur l'écran est le message d'accueil, qui revient, soit quand on met l'ordinateur en marche soit quand on presse les touches **[CTRL]**, **[SHIFT]** et **[ESC]**. L'ordinateur attend maintenant que vous lui donnez du travail à faire.

Si vous avez déjà programmé dans le langage BASIC, il est fort probable que vous avez déjà tapé un petit programme pour faire connaissance. Le BASIC d'AMSTRAD vous sera familier dans beaucoup de domaines et pour commencer nous allons faire un court programme qui va afficher tous les caractères disponibles dans l'ordinateur. C'est ce qu'on appelle la Police des caractères (mais il y en a plus de 22, exactement 224 pour être précis.), qui est le terme utilisé pour décrire l'ensemble complet de chiffres, lettres et symboles que l'on peut afficher à partir du clavier.

Certains caractères ne sont pas directement accessibles par une touche du clavier, mais on peut seulement les afficher en utilisant l'instruction **PRINT CHR\$ (·nombre·)**, que nous décrirons un peu plus loin dans ce guide.

En effet, chaque élément en mémoire dans l'ordinateur est rangé par "octet" et vous verrez si vous consultez l'Appendice II, doit utiliser un octet complet par caractère en mémoire, il vaut mieux utiliser les 256 combinaisons plutôt que les 96 à 100 caractères qu'on trouve sur la plupart des machines à écrire et gaspiller plus de 150 possibilités.

Le sous-ensemble des caractères "standard" est appelé dans le monde de l'informatique le système d'affichage ASCII qui vient de l'Amérique.

American  
Standard  
Code For  
Information  
Interchange

C'est avant tout un système qui permet à un ordinateur d'envoyer des données à un autre ordinateur sous une forme reconnaissable. C'est peut-être la seule chose universelle dans l'informatique et il est fort conseillé de consulter fréquemment l'appendice III pour vous familiariser avec le système ASCII, et les caractères additionnels du CPC464 avec leur code numérique.

Certains autres caractères qu'on ne peut pas imprimer peuvent être affichés en utilisant une combinaison de la touche **[CTRL]** (contrôle) et d'une autre touche, mais ne vous inquiétez pas, vous pourrez mieux vous y retrouver quand vous aurez compris le fonctionnement de la touche contrôle **[CTRL]**.

## 1.2.2

Pour voir exactement à quoi ressemblent ces caractères sur l'écran, tapez le programme suivant, en tenant compte de l'organisation du clavier 'physique' (nous l'appelons physique car on peut redéfinir les touches et leur faire taper bien d'autres choses en utilisant ce qu'on appelle des jetons d'expansion).

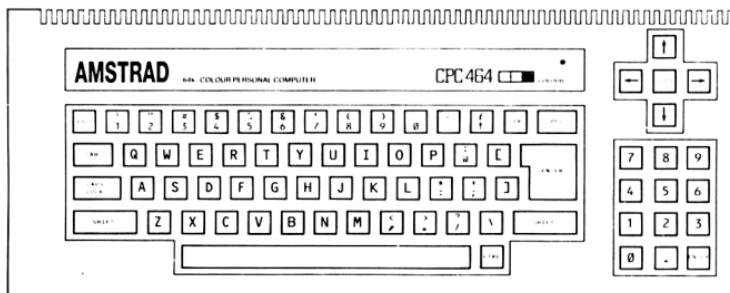


Figure 4: Le clavier du CPC464

Quelques remarques avant de taper le programme.

- Majuscules ou minuscules: pour le programme suivant, vous pouvez taper en minuscules ou majuscules (en appuyant sur la touche [SHIFT] en même temps que la lettre). Mais vous DEVEZ séparer les mots avec des espaces (ou des virgules, points virgules, deux points) aux positions indiquées car l'AMSTRAD BASIC permet l'usage des mots réservés (liste complète à l'appendice VII) à l'intérieur des noms de variables.
- Attention au zéro 0 et à la lettre O; si vous tapez l'un à la place de l'autre, vous aurez des erreurs de Syntaxe (Syntax Error sur l'écran) ou des choses incompréhensibles pour l'ordinateur.
- Chaque fois que vous tapez [**ENTER**] à la fin d'une ligne de commande ou de programme que vous venez d'écrire, vous "Faites entrer" dans l'ordinateur cette ligne, et l'ordinateur l'exécute ou la met en mémoire si la ligne commençait par un nombre.

```
10 FOR N=32 to 255
20 PRINT CHR$(N);
30 NEXT N
RUN
```

1,2,3

Voyons maintenant ce que ce programme donne (si ça ne marche pas, faites **LIST** et **[ENTER]**, vous verrez ce qui est différent dans votre **LISTing** et dans le programme ci-dessus):

Nous avons dit à l'ordinateur de montrer tout son jeu de caractères en se servant du programme que nous venons d'écrire. Si le programme ne marche pas, c'est qu'il y a une erreur de frappe que vous n'avez pas remarquée et il faut aller voir au paragraphe 1.2.7 le moyen d'y remédier.

Première chose à remarquer: on n'a pas donné la commande **PRINT "a b c d e f g h i j k l m n . . . etc"** On lui a dit **PRINT CHR\$(N)**.

N est juste un moyen de nommer une variable, ce pourrait être une autre lettre, mais puisque cette variable représente un nombre, autant prendre N. Lorsqu'on écrit:

**FOR N=32 TO 255**, nous disons à l'ordinateur que N est une variable et qu'elle va prendre les valeurs 32, puis 33, 34... etc. jusqu'à 255.

Ayant déclaré cette variable, nous disons à l'ordinateur ce qu'il doit faire avec cette variable:

20 PRINT CHR\$(N)

Ce qui veut dire: Imprime sur l'écran (= affiche) ce **CHR\$(N)**, qui est une fonction du langage BASIC qui convertit un nombre en un caractère, après avoir regardé dans sa mémoire quel est ce caractère.

Le point virgule signifie qu'il ne faut pas aller à la ligne suivante mais laisser le curseur juste après le caractère qu'on vient d'afficher.

La ligne suivante (**30 NEXT N**) commande à l'ordinateur, après avoir affiché le caractère qui correspond au nombre 32, de retourner à la ligne où est placé le **FOR** et de faire la même opération avec la valeur suivante 33 (next = suivant = à côté, en Anglais). Ce procédé s'appelle une boucle (un loop) et c'est un des éléments les plus importants de la programmation et du fonctionnement des ordinateurs.

On retrouve ces boucles dans tous les langages sous une forme ou sous une autre, car il permet d'éviter d'avoir à taper plusieurs fois la même chose d'une manière répétitive.

Quand la boucle avec le **FOR** arrive à la dernière valeur (255 pour notre programme), il exécute la ligne après la ligne où il y a le **NEXT**. Comme il n'y en a pas dans notre programme, il s'arrête et donne le message Ready, c'est à dire prêt à recevoir d'autres commandes - ou on peut faire marcher (**RUN**) le programme encore une fois. Le programme est gardé en mémoire et y reste tant que vous ne demandez pas à l'ordinateur autre chose - ou coupez le courant - auquel cas toutes les données (programmes, variables etc.) sont effacées à moins que vous ayez sauvegardé (enregistré) le programme.

Ce programme montre l'élément fondamental de l'informatique, tout ce que fait l'ordinateur se rapporte aux nombres. L'ordinateur a affiché l'alphabet et bien d'autres caractères en utilisant un nombre comme référence. Quand vous frappez la touche A, vous ne demandez pas à l'ordinateur de taper un A sur l'écran, mais de regarder dans sa mémoire l'information numérique nécessaire à l'affichage de la lettre A. L'endroit où se trouve cette information est définie par le code numérique qui est activé par l'action de frapper la touche sur le clavier.

Chaque caractère correspond à un nombre et vous pouvez les trouver dans l'Appendice III de ce guide.

De même, le caractère affiché n'a rien à voir avec l'écriture d'une lettre sur l'écran, tout se passe avec des nombres.

## 1.2.4

Ne vous inquiétez pas si vous ne comprenez pas tout le jargon et les termes techniques utilisés dans ce paragraphe. Il est important d'expliquer en détail comment l'ordinateur se débrouille avec vos commandes et arrive au résultat demandé, mais il est probable que pour apprécier pleinement ces explications, il faut avoir l'esprit technique. Si vous trouvez cela trop difficile, passez au paragraphe suivant 1.2.5.

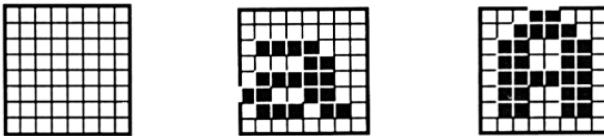
Le code pour la lettre A est 97. L'ordinateur "ne comprend pas" 97, et ce nombre doit être transformé du système décimal en un système que l'ordinateur comprend, généralement appelé code machine et des explications sur ce sujet sont rassemblées dans l'appendice II.

De prime abord, convertir du système décimal au système en base 16 (hexadécimal, ou **HEX** en abrégé) peut sembler difficile, comme d'essayer de manger en changeant la fourchette et le couteau de main.

La dextérité mentale nécessaire pour manipuler les nombres en hexadécimal (base 16) peut être longue à venir, mais une fois acquise, bien des éléments de l'informatique deviendront plus cohérents et l'architecture élégante du système numérique deviendra apparente.

Dès que l'ordinateur a traduit la frappe de la lettre A en un nombre qu'il peut comprendre, il va chercher dans sa mémoire et trouve encore une autre série de nombres qui définit le caractère. Ce caractère que vous voyez sur l'écran est construit à partir d'un bloc de renseignements, stockés dans la mémoire sous la forme d'une matrice numérique:

figure 5



une grille vierge, la grille à minuscule, la grille A majuscule.

Les éléments de la matrice sont des lignes et des colonnes de points. Le caractère est affiché en allumant ou éteignant les points qu'il faut et chaque point est une information en mémoire.

Il y a 8 lignes et 8 colonnes pour chaque caractère affiché sur l'écran du CPC464 (chacun étant défini par un octet) et si vous voulez définir un caractère qui n'est pas dans les 255 existants, vous pouvez le faire en utilisant les commandes **SYMBOL** et **SYMBOL AFTER** au chapitre 8.

Les caractères définis par l'utilisateur peuvent avoir n'importe quelle combinaison de 0 à 64 points, arrangés dans n'importe quel ordre, et donc un ensemble de caractères qui utiliserait toutes les combinaisons possibles de cette matrice comprendrait beaucoup plus de caractères. Comme on peut en plus grouper ces caractères par blocs, les possibilités ne sont limitées que par votre astuce et votre temps.

## 1.2.5 Revenons à notre programme

Le résultat du programme affiché sur l'écran est un peu désordonné. Il reste un petit bout du message Ready en haut de l'écran. Il vaudrait mieux faire place nette avant d'exécuter le programme. ajoutons donc une ligne à notre programme.

Tapez ce qui suit sur la ligne où il y a le curseur. (Le curseur est ce petit carré sous la lettre R de Ready et si vous ne le saviez pas, vous devriez repartir vite dans le Cours Elémentaire du début du Guide??)

```
5 CLS  
RUN
```

Voyez comme l'écran s'efface avant de commencer à écrire la police des caractères en haut à gauche.

Ceci aussi démontre un des aspects les plus agréables du langage BASIC, à savoir que peu importe dans quel ordre vous écrivez les numéros des lignes d'un programme, vous n'avez pas besoin d'avoir le programme affiché sur l'écran pour lui ajouter des lignes une fois qu'il est "entré" dans la mémoire.

L'ordinateur classe toujours les lignes par ordre numérique avant d'exécuter un programme. Vérifions cela en utilisant la commande LIST.

## 1.2.6 LISTing

Vous pouvez facilement vérifier ce que l'ordinateur a rangé dans sa mémoire à programmes en tapant:

```
LIST
```

et le résultat sur l'écran est:

```
5 CLS  
10 FOR N = 32 TO 255  
20 PRINT CHR$(N)  
30 NEXT N  
Ready
```

Ce programme restera dans la mémoire du CPC646 à moins que vous ne fassiez une des choses suivantes:

- \* Eteindre l'ordinateur
- \* RESET (= remise à zéro) en appuyant [CTRL] [SHIFT] [ESC] dans cet ordre et en gardant les trois touches appuyées jusqu'au message Ready
- \* LOAD ou RUN un programme enregistré sur cassette.
- \* Tapez NEW [ENTER] qui remet à zéro toutes les variables et la mémoire à programme sans modifier le mode et les couleurs.

Vous allez maintenant établir une touche de fonction qui va taper automatiquement [ENTER] CLS : LIST [ENTER] ce qui a l'avantage d'accélérer l'écriture et le développement des programmes. Il suffit de taper:

```
KEY 138, CHR$(13)+"CLS:LIST"+CHR$(13)
```

Appuyez alors sur la touche avec un petit point sur le pavé numérique. Il est possible de pré-programmer 32 touches du clavier et vous pouvez choisir les touches que vous voulez redéfinir si jamais vous voulez garder l'usage du pavé numérique. La description de la commande KEY se trouve au chapitre 8.

Si vous avez des programmes longs, vous pouvez définir cette touche de la manière suivant:

**KEY 138, CHR\$(13)+"CLS:LIST"**

Cela vous permettra de préciser l'ensemble des lignes que vous voulez afficher, ou de LISTer tout le programme en appuyant deux fois de suite sur la touche.

Quand on fait des essais avec les couleurs, on peut se perdre dans une combinaison de couleurs où on ne peut pas voir ce qui est affiché car le fond et les caractères prennent la même couleur; ainsi si vous faites:

**KEY 139, CHR\$(13)+"mode2:ink1,0:ink0,9"+CHR\$(13)**

Vous aurez seulement à presser la petite touche [ENTER] du pavé numérique pour retrouver des couleurs visibles et sans effacer le programme de la mémoire.

Les codes des touches pré-programmables sont remis à zéro en même temps que la machine car elles doivent être disponibles pour différents programmes; lorsque vous aurez trouvé celles qui vous sont le plus utiles, écrivez un programme et sauvegardez le sur cassette pour le charger quand vous en avez besoin.

## 1.2.7 Comment EDITer

Vous ferez obligatoirement des fautes en tapant vos programmes. Bienvenue à ceux qui arrivent dans ce paragraphe et viennent du paragraphe 1.2.2!

Le CPC464 a essayé de rendre la correction des erreurs aussi simple que possible, et en même temps d'éviter les problèmes posés quand on efface des caractères que l'on ne voulait pas changer.

Le pavé de touches des curseurs permet d'attirer l'attention de l'ordinateur sur l'endroit précis que vous voulez modifier.

Quand on a fait une erreur dans une ligne numérotée:

**10 FON=332 TO 255**

on a plusieurs options:

- 1) On peut faire [ENTER] et retaper la ligne entière. La ligne incorrecte sera remplacée dans la mémoire par la ligne qui commence par le même numéro.
- 2) On peut presser la touche [**←**] et placer le curseur sur la faute elle-même (le caractère en trop)

**10 FOR N=332 to 255**

Notez que le caractère sous le curseur est montré en vidéo inverse, autrement dit le caractère qui a normalement la même couleur que le curseur prend la même couleur que le fond (PAPER), ce qui fait qu'on le voit au travers du curseur.

Appuyez alors la touche [CLR] (abrégé de Clear = effacer) et le caractère sous le curseur disparaît, et l'espace où était le chiffre 3 disparaît:

**10 FOR N=32 to 255**

Appuyez sur [ENTER] à nouveau, et la ligne ainsi corrigée sera celle que l'ordinateur range dans sa mémoire. Le curseur n'a pas besoin d'être à la fin de la ligne, l'ordinateur range la ligne entière, quelle que soit la position du curseur.

3) On peut aussi amener le curseur sur le caractère juste après l'erreur:

**10 FOR N=332 TO 255**

Appuyez maintenant sur [DEL] (abrégé pour delete = enlever), le caractère à gauche du curseur est enlevé, la ligne se ramasse sans toucher au 2 sous le curseur.

Faites [ENTER] et la ligne rentre en mémoire comme dans le cas précédent.

**10 FOR N=32 TO 255**

### 1.2.8 L'erreur passa inaperçue

Les méthodes précédentes sont valables si vous voyez l'erreur avant d'avoir fini la ligne et tapé [ENTER]. La plupart des erreurs se produisent sans qu'on s'en rende compte et refont surface à l'exécution du programme quand un message d'erreur apparaît (il y a 30 messages d'erreurs dans l'Appendice VIII).

Un certain nombre d'erreurs conduisent l'ordinateur à afficher la ligne fautive, avec le curseur placé à gauche au tout début de la ligne. Dans ce cas, vous pouvez utiliser une des procédures mentionnée au paragraphe précédent.

Si l'erreur n'amène pas l'affichage de la ligne fautive, il faut LISTer le programme, trouver la cause du problème et s'attaquer à la solution.

### 1.2.9 EDITer avec le curseur de copie

Commençons par taper le programme suivant en faisant exprès une erreur, (!) à la ligne 20, nous tapons un S au lieu du signe \$ (dollar). (Si vous n'avez pas encore fait la faute, dites vous qu'elle risque très fort d'arriver).

Le symbole \$ indique une chaîne de caractères prise dans son ensemble; nous y reviendrons avec plus de précision au chapitre 4.

Tapez donc:

```
5 CLS  
10 FOR N=n 32 to 255  
20 PRINT CHR$(N)  
30 NEXT N
```

Appuyez et gardez appuyée la touche [SHIFT] et appuyez sur le curseur [ $\uparrow$ ]. On peut taper sur le curseur à petits coups, et tenir appuyé, mais attention, il se balade vite; (vous redescendez avec [SHIFT] et [ $\downarrow$ ]).

Ce faisant, le **COPY CURSOR** (curseur de copie) se sépare du curseur principal (ils sont identiques) et le curseur de copie va sur la ligne que vous voulez corriger. Au départ, le curseur de copie est sur le premier caractère de la ligne.

```
5 CLS  
10 FOR N = 32 TO 255  
20 PRINT CHR$(N)  
30 NEXT N  
Ready  
■
```

Si vous aviez pris le curseur principal pour corriger de cette manière, sans tenir la touche **[SHIFT]**, l'ordinateur ne verrait dans cette action rien de valable, car seuls les caractères "entrés" après le curseur principal sont reconnus comme valables.

Si vous êtes embarqués dans cette direction, vous pouvez facilement sortir de cette impasse en tapant sur la touche **[ESC] AVANT de faire [ENTER]**.

Le curseur de copie étant en place, tapez la touche verte **[COPY]** jusqu'à ce que le curseur de copie arrive sur ce qu'il faut changer. (Quand vous serez habitué à la vitesse du curseur de copie, vous pourrez garder le doigt sur la touche **[COPY]** et déplacer le curseur plus rapidement)

```
5 CLS  
10 FOR N = 32 TO 255  
20 PRINT CHR$(N);  
30 NEXT N  
Ready  
20 PRINT CHR$
```

Relâchez la touche **[COPY]** et tapez, **\$**, il apparaîtra sous le curseur **PRINCIPAL**, qui bougera alors d'une place vers la droite

```
20 PRINT CHR$
```

Il faut maintenant faire passer le curseur de copie au delà du **S** erroné, et pour ce faire, appuyez sur **[SHIFT]** et pressez le curseur **[→]** une fois seulement. Le curseur de copie est sur la parenthèse **( )**. Lâchez la touche **[SHIFT]** et appuyez sur la touche **[COPY]** jusqu'à dépasser la fin de la ligne. Faites **[ENTER]** et la ligne corrigée vient remplacer en mémoire la ligne incorrecte.

Vous pouvez combiner toutes ces techniques en **[COPY]**ant entièrement la ligne fautive et, avant de taper sur **[ENTER]**, modifier cette ligne en utilisant les possibilités d'édition du curseur principal, en se servant des touches curseurs et des touches **[CLR]** et **[DEL]** sans appuyer sur **[SHIFT]**. En appuyant sur **[CTRL]** et le curseur droit **[→]** ou bien le curseur gauche **[←]** déplace le curseur tout à droite ou tout à gauche de la ligne d'un seul coup.

Essayez et vous verrez qu'avec un peu de pratique, c'est très facile.

Enfin on peut éditer en tapant:

```
EDIT 20
```

L'ordinateur répond par:

```
20 PRINT CHR$(N)
```

Il faut maintenant faire passer le curseur de copie au delà du S erroné et pour ce faire, appuyez sur [SHIFT] et pressez le curseur [→] une fois seulement. Le curseur de copie est sur la parenthèse ). Lâchez la touche [SHIFT] et appuyez sur la touche [COPY] jusqu'à dépasser la fin de la ligne. Faites [ENTER] et la ligne corrigée vient remplacer en mémoire la ligne incorrecte.

Vous pouvez combiner toutes ces techniques en [COPY]ant entièrement la ligne fautive et, avant de taper sur [ENTER], modifier cette ligne en utilisant les possibilités d'édition du curseur principal, en se servant des touches curseurs et des touches [CLR] et [DEL] sans appuyer sur [SHIFT]. En appuyant sur [CTRL] et le curseur droit [right arrow] ou bien le curseur gauche [left arrow] déplace le curseur tout à droite ou tout à gauche de la ligne d'un seul coup.

Essayez et vous verrez qu'avec un peu de pratique, c'est très facile.

Enfin on peut éditer en tapant:

**EDIT 20**

L'ordinateur répond par:

**20 PRINT CHR\$(N)**

En se servant du curseur principal avec les touches [CLR] et [DEL] comme on l'a dit plus haut, on arrive au résultat désiré et on tape sur [ENTER] comme décrit auparavant. S'il y a des pépins, soyez brefs et tapez sur [ESC] et écrivez LIST à nouveau, la ligne où vous étiez avant de faire [ESC] est toujours là, intacte.

Faites donc LIST à nouveau et vous verrez la version corrigée du programme affichée. Si c'est encore faux, essayez à nouveau!

Jusqu'à présent, nous avons seulement esquissé le canevas du CPC464. Pour avoir une idée des deux autres modes, il suffit de faire:

**MODE 0**

**RUN**

L'écran s'efface et le programme affiche les caractères sur 20 colonnes.



Pour retourner aux caractères de départ, faisons:

## MODE 1

RUN

C'est le retour à la case départ. Pour voir ce que cela donne sur 80 colonnes, on fait:

MODE 2

RUN

Voilà. Nous avons parcouru la première étape, et vous devriez avoir satisfait une partie de votre curiosité initiale. Les chevronnés doivent déjà essayer d'adapter leurs programmes favoris pour qu'ils marchent avec le BASIC de l'AMSTRAD, les autres doivent procéder plus calmement et suivre la progression logique, qui permettra d'avoir une vue d'ensemble des caractéristiques du BASIC de la machine.

# 2 Lecteur de Cassettes Datacorder

Changement et fonctionnement du système Datacorder

## Sujets abordés dans ce chapitre

- \* Ressemblances et différences entre les cassettes musicales et les cassettes de programmes ou de données
- \* Chargement et exécution des programmes sur cassette - la cassette de Bienvenue
- \* les différentes vitesses de chargement
- \* la sauvegarde des programmes sur cassette
- \* les erreurs de lecture

La mémoire du CPC464 est volatile, c'est à dire qu'elle ne fonctionne que lorsque l'ordinateur est branché. Si vous voulez conserver des programmes ou des fichiers quand on a coupé l'alimentation en courant de l'ordinateur, il faut avoir sauvegardé auparavant sur cassette ou sur une autre mémoire non volatile comme les disquettes en option.

### 2.1 Les commandes de contrôle

A droite du clavier se trouve le lecteur de cassette appelé DATAORDER (figure 2.1). Les caractéristiques mécaniques de ces appareils sont identiques à ceux des magnétocassettes de musique, mais l'électronique qui contrôle les échanges de signaux est spécialement optimisée pour la sauvegarde et la lecture informatisée.

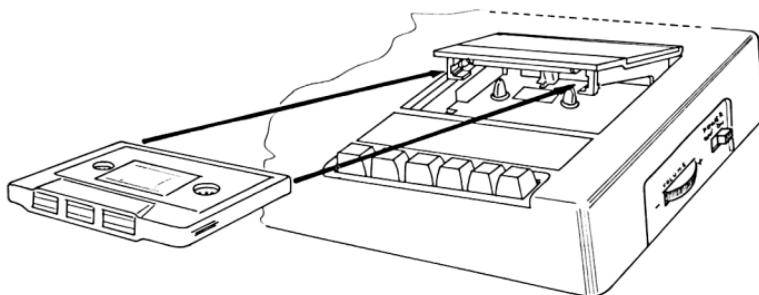


Figure 2.1 Méthode d'introduction de la cassette dans le DATAORDER.

Le fonctionnement des touches de commandes sur la Datacorder est identique à celui de la plupart des magnéto-cassettes:

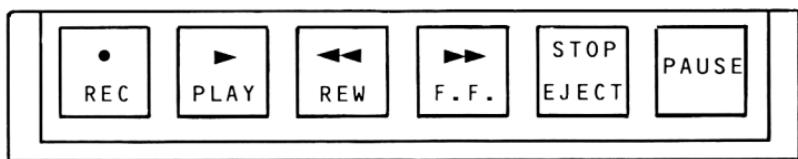


Figure 2.2: Les touches de contrôle du DATACORDER

Il est à noter que ces touches nécessitent une pression plus importante que les touches du clavier des lettres et chiffres.

**[REC]** = touche d'enregistrement, qui fonctionne en même temps que la touche **[PLAY]** = lecture, pour sauvegarder les données et programmes. Ces touches ne peuvent pas fonctionner si les taquets d'écriture sont absents (souvent appelés taquets ou pattes de protection à tort car la cassette est protégée contre l'effacement lorsque les taquets sont absents).

Les touches ne fonctionnent que si la porte du lecteur est fermée.

Pour l'utiliser vous devez d'abord presser sur **[REC]** et tout en maintenant cette touche enfoncée, appuyez sur la touche **PLAY**. L'ordinateur va alors écrire les données sur la cassette si un programme le lui commande ou par une commande directe **SAVE** tapée au clavier.

**[PLAY]** = fait la lecture de la cassette pour charger (= **LOAD**), ou exécuter (= **RUN**) un programme à partir de la cassette. L'ordinateur lit alors les données de la cassette quand le programme le demande, ou lorsqu'il y a une commande directe.

Les touches **[REC]** et **[PLAY]** sont annulées par un moyen mécanique lorsque la bande est terminée.

**[REW]** = réenrouleur de la bande

**[F.F.]** = avance rapide de la bande

Ces deux touches ne s'arrêtent pas d'elles mêmes et il faut donc appuyer sur **[STOP/EJECT]** lorsque c'est fini ou le moteur pourrait surchauffer.

**[STOP/EJECT]** Arrête toute opération de la cassette et remet toutes les touches de fonctionnement à zéro. Si on appuie une deuxième fois, le boîtier s'ouvre, permettant de mettre ou d'enlever une cassette. On ne peut enlever une cassette tant que la cassette tourne.

**[PAUSE]** Comme les arrêts momentanés de la cassette sont contrôlés par l'ordinateur, cette touche de pause, qu'il ne faut pas appuyer quand on écrit ou on lit une cassette (une erreur s'afficherait) sera très peu utilisée.

## 2.2 Protection contre l'écriture accidentelle

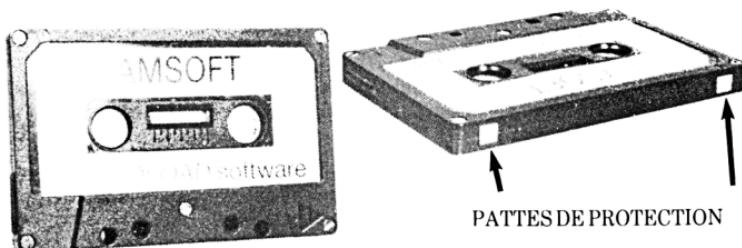


Figure 2.3 Pattes de protection

Si on veut empêcher l'effacement accidentel des données sur une cassette, il faut enlever les pattes de protection (avec des pinces à épiler) et il ne sera plus possible d'enregistrer sur cette cassette. (**ATTENTION:** Si vous appuyez sur [REC], la touche refuse de marcher: ne cherchez pas à forcer).

Ce système protège un côté de la cassette. Si vous voulez protéger les deux côtés de la cassette il faut enlever les deux pattes de protection.

Lorsqu'on veut pouvoir récrire sur la cassette, il faut alors couvrir soigneusement avec du ruban adhésif les petits ouvertures derrière la cassette.

## 2.3 Chargement de la cassette

Sur la figure 2.1, on voit le moyen correct d'introduction de la cassette dans le boîtier, en se servant de la cassette fournie avec le CPC464.

La cassette doit être remise au départ en appuyant sur [REW]. Si la bande est sortie de la cassette, il faut la réenrouler à la main avant de mettre la cassette dans le boîtier, ou vous pourriez perdre une partie de vos programmes.

Il est à noter que les cassettes de données supportent beaucoup moins les mauvais traitements que les cassettes audio qui peuvent fonctionner avec des défauts dans la bande alors que celles utilisées pour un ordinateur ne peuvent pas.

Par exemple, si vous endommagez une cassette (bande sortie et pliée par exemple) et que vous apercevez que vous pouvez encore charger et exécuter le programme, sauvegardez le rapidement sur une cassette neuve (alors que le programme est en sécurité dans la mémoire du CPC464) et jetez la cassette endommagée avant d'être tenté de la ré-utiliser.

## 2.4 La cassette de Bienvenue

La cassette fournie avec le CPC464 contient un certain nombre de démonstrations du son et des capacités graphiques de l'ordinateur et de son logiciel intégré, l'AMSTRAD BASIC et le système d'exploitation (MOS = Machine Operating System)

Une partie de ce système d'exploitation est le système logiciel qui dirige le fonctionnement de la cassette, avec une série de commandes pour lire et écrire les données sur la bande magnétique. Les plus fréquemment utilisées sont **LOAD** (charge) et **SAVE** (sauvegarde).

Pour faciliter l'utilisation de l'ordinateur, le CPC464 possède une série de fonctions spéciales qui simplifient l'usage du clavier pendant que vous apprenez à vous en servir.

Vous avez mis l'ordinateur en marche et le message d'accueil est là avec le mot:

**Ready**

Indroduisez la cassette (voir Figure 2.1), remettez le compteur à 000 en appuyant sur le petit bouton à côté du compteur, puis appuyez sur la touche **[CTRL]** et en la gardant appuyée, pressez la petite touche **[ENTER]**, en bas à droite du pavé numérique. Le message suivant s'affiche:

**RUN"**

**Press PLAY then any key:**

Ce qui veut dire appuyez sur la touche **PLAY** du lecteur de cassette, puis n'importe quelle autre touche.

(Pour référence, c'est un exemple de ce qu'on appelle jeton d'expansion, qu'on a vu avec la commande **KEY** introduite brièvement dans le chapitre 1, où l'ordinateur doit faire une série de commandes quand on appuie sur une succession de touches et cela permet d'aller plus vite).

Vous auriez pu taper **RUN"** sur le clavier et puis **[ENTER]** mais cela va plus vite comme cela.

La plupart du temps, les deux touches **[ENTER]** font la même chose mais il y a des cas où la petite touche peut être modifiée (on dit redéfinie) pour servir à autre chose. Nous en reparlerons plus tard.

La touche **[PLAY]** est la touche sur le Datacorder, qui doit faire un déclic lorsqu'elle est enclanchée.

Quand ou dit "**any key**" (n'importe quelle touche) c'est une touche du clavier dont il s'agit. Pour ma part je pense qu'il vaut mieux utiliser la grande touche **[ENTER]**: c'est ce qui va le plus vite et il vaut mieux prendre cette habitude.

Quand vous pressez donc une touche, rien ne s'affiche, seule la cassette se met en marche. Si elle ne se met pas en marche vérifiez que vous n'avez pas appuyé par erreur sur la touche **[PAUSE]**.

Notez que nous n'avons pas donné de nom de programme. S'il n'y a rien de tapé après

**RUN"**

L'ordinateur va chercher la premier programme sur la cassette et commencer à le charger. Quand il a trouvé le début du programme, le message suivant apparaît:

**Loading BIENVENUE 1 block 1**

Ceci vous informe que l'ordinateur a trouvé le premier d'une série de "blocs" du programme appelée **BIENVENUE 1**. Chaque programme est sauvegardé sous la forme de blocs de données (jusqu'à 2K Octets chacun), chacun étant séparé par un petit signe sur la bande et le message vous informe du bloc en train d'être lu par l'ordinateur. Après chaque bloc, la bande s'arrête un moment et repart, et le numéro du bloc change sur l'écran.

Si à un moment, l'ordinateur découvre des données corrompues, il affiche un message d'erreur (voir la liste en Appendice VIII) qui annonce la nature du problème. La seule chose à faire est d'exécuter le programme à nouveau jusqu'à ce que le chargement se fasse.

Supposons donc que la cassette est en train d'être chargée, lisez les instructions sur l'écran, la cassette Bienvenue fera le reste.

## 2.5 Super Prudent et Charge Rapide

Le CPC464 possède deux vitesses: SuperPrudent à 1000 baud (bits de données par seconde) et Charge Rapide à 2000 baud. En Charge Rapide, les données sont écrites et lues deux fois plus vite qu'en mode SuperPrudent, avec un sacrifice au niveau de la fiabilité dans le cas de bandes magnétiques de qualité douteuse et les erreurs dues à des alignements différents des têtes de lecture.

Pour les programmes sauvegardés et chargés sur la même machine, la Charge Rapide sera fiable avec des cassettes de bonne qualité, elle pourra charger la plupart des progiciels vendus dans le commerce sans erreurs, mais AMSTRAD conseille d'enregistrer ces programmes en utilisant les deux vitesses.

L'ordinateur lit automatiquement la cassette à la vitesse à laquelle elle a été enregistrée. En sauvegardant un programme, il faut préciser si ou veut la Charge Rapide, ou alors le mode SuperPrudent est utilisé si on ne précise rien.

Pour avoir la vitesse rapide, vérifiez que le message Ready est là et tapez:

**SPEED WRITE 1**

pour revenir à la vitesse normale SUPERPRUDENT, vous devez taper:

**SPEED WRITE 0**

## 2.6 Sauvegarde des programmes et données sur une cassette.

Le BASIC a plusieurs commandes qui concernent l'écriture de données sur cassette. Les voici brièvement avec quelques exemples.

### 2.6.1 SAVE ".Nom du dossier."

La méthode la plus simple de sauvegarde sur cassette est l'utilisation de la commande **SAVE** quand le CPC464 a affiché le message Ready après avoir exécuté ou fait la liste d'un programme. Si on prend l'exemple du petit programme d'affichage des caractères du chapitre 1 comme cobaye pour notre commande **SAVE**:

Le **.nom du dossier.** peut être une combinaison quelconque de 16 caractères (y compris les espaces). Si vous essayez un nom plus long, le 17 ème caractère et les suivants ne seront pas retenus. Quand ce programme est dans la mémoire de l'ordinateur, tapez

**SAVE "CARACTERES"**

L'ordinateur répond avec le message

**press REC and PLAY then any key:**

Appuyez sur **REC** et **PLAY**, puis sur la grande Touche **[ENTER]** et la cassette va tourner et l'ordinateur va sauvegarder le programme sous le nom **CARACTERES**.

### IMPORTANT

L'ordinateur ne peut pas vérifier si vous avez vraiment appuyé sur les touches de cassette correctes, donc si vous avez appuyé seulement sur **[PLAY]**, la cassette va démarrer et le programme semblera sauvegardé alors qu'il ne l'est pas.

**ATTENTION:** Si vous appuyez accidentellement sur **[REC]** et **[PLAY]** quand vous voulez seulement lire ou charger un programme sur cassette, vous allez effacer ce qu'il y a sur la cassette : si vous n'appuyez pas alors vite sur la touche **[ESC]**, la cassette va se dérouler jusqu'au bout et tout sera effacé; pour éviter cela avec les programmes que vous ne voulez pas perdre, enlevez les pattes de protection avant d'avoir appris à vos dépendre.

Il y a quatre méthodes de sauvegarde (**SAVE**) sur le CPC464. Nous venons de voir la méthode générale, mais il y a trois autres moyens pour des usages particuliers.

## **2.6.2 SAVE "<nom du dossier>" , A**

Cette procédure est la même que la précédente, excepté que le suffixe, A, demande à l'ordinateur de sauvegarder le programme ou les données sous la forme d'un dossier ASCII au lieu de la méthode normale précédente.

Cette méthode de sauvegarde est utilisée dans les dossiers créés par des traitements de textes et autres progiciels. Nous en reparlerons, quand nous verrons ce genre de programmes.

## **2.6.3 SAVE "<nom du dossier>" , P**

Le suffixe ,P indique à l'ordinateur de protéger (en mettant des messages invisibles) les données pour qu'on puisse exécuter un programme, mais pas en faire la liste. Des programmes protégés comme cela peuvent être appelés seulement en utilisant les commandes **CHAIN** ou **RUN**. Si vous devez modifier le programme ultérieurement, il vaut mieux avoir aussi une version non protégée.

## **2.6.4 SAVE "<nom du dossier>" , B, <adresse départ>, <longueur>[, <point d'entrée>]**

Cette option vous permet de faire une sauvegarde en Binaire, quand un bloc complet de données emmagasiné dans la mémoire RAM est stocké sur cassette exactement comme dans la mémoire. Il est nécessaire de préciser où est le point de départ, la longueur du dossier et l'adresse mémoire du point de départ, si on veut exécuter le programme.

Cette capacité de sauvegarde en binaire permet de sauvegarder un écran graphique directement sur la cassette sous la forme d'une image-mémoire. Un de ses avantages est de pouvoir faire des séquences de titres qui se chargent rapidement.

## **2.6.5 Dossiers sans nom et C ATalogue**

Si on sauvegarde sans donner un nom de dossier:

**SAVE"**

Le BASIC va sauvegarder comme un dossier sans nom. On peut sauvegarder plusieurs dossiers du même nom (ou sans nom) sur une cassette, les uns après les autres, contrairement aux disquettes qui demande un nom unique.

Mais vous allez vite vous y perdre si vous ne donnez pas à vos programmes des noms faciles à mémoriser, et nous vous conseillons d'ajouter un code pour la date au nom du programme pour savoir plus tard quels sont les plus récents.

On peut faire le **C ATalogue** d'un cassette en faisant la commande **CAT** et en suivant les instructions de l'écran:

**Press PLAY then any key:**

(appuyer sur **PLAY** sur le datacorder puis la grande touche **[ENTER]**)

BASIC va vous donner tous les ·noms de dossier· en majuscules suivis par le nombre de blocs présents, puis un caractère unique qui vous indique le genre de dossier.

- \$ est un programme standard BASIC
- % est un programme BASIC protégé
- \* est un dossier ASCII
- & est un dossier en binaire

Un OK à la fin de la ligne veut dire que le chargement aurait été OK si on avait fait LOAD. Lorsqu'on donne la commande CAT, le programme qui est dans la mémoire ne sera pas affecté.

## 2.7 Erreurs de lecture

Si vous avez sur l'écran un message "Read error" pendant que le CPC464 essaie de charger un programme depuis la cassette, la cassette va continuer à marcher et l'ordinateur va poursuivre la lecture après l'erreur, mais ne va pas essayer de charger, sauf si c'est le bloc 1 du programme qu'il avait déjà essayé de charger sans succès.

Ce qui veut dire qu'après une erreur de lecture, il faut arrêter avec le bouton [STOP/EJECT], puis réenrouler avec [REW], puis [PLAY] de nouveau. L'ordinateur peut alors essayer de nouveau et si vous avez de la chance, ça marchera.

Les erreurs ont plusieurs causes différentes, la plus commune venant de la bande magnétique (pliée, étirée, abimée ou autre problème similaire). Elles arrivent aussi si on arrête l'ordinateur pendant que les boutons [REC] ou [PLAY] sont enfoncés.

C'est parce que quand ces touches sont enfoncées, la bande est en position sur la tête de lecture et une petite décharge électrique peut passer et affecter la bande bien qu'immobile.

Des erreurs de lecture peuvent aussi arriver, si on a utilisé la touche de [PAUSE] pendant l'enregistrement ou la lecture ou si la bande a été enregistrée sur un autre CPC464 où les têtes de lecture sont mal alignées.

Des erreurs arriveront aussi pour des raisons imprévisibles ou inconnues. Le magnéto-cassette n'a pas été conçu à l'origine pour l'informatique et a quelques inconvénients qui le distinguent des systèmes de bande magnétique conçus pour les gros ordinateurs.

Malgré tout, les cassettes sont un moyen standard excellent de sauvegarde pour les ordinateurs économiques. Les caractéristiques limitées de la bande magnétique et les contraintes de vitesse par rapport à la tête de lecture imposent un seuil de vitesse de lecture - écriture qu'il ne vaut mieux pas dépasser pour des raisons de fiabilité, surtout quand on considère les méthodes de reproduction de cassettes dans les progiciels de bas de gamme.

## **2.8 Quelles cassettes, choisir?**

Le Datacorder peut utiliser des cassettes jusqu'à C90, mais nous vous conseillons fortement de n'utiliser que des cassettes C12 (6 minutes par côté) ou au plus des C30.

Si les cassettes sont trop longues, il devient difficile de retrouver un dossier rapidement. Si on veut écrire par dessus un autre dossier, il faut faire très attention au point de départ.

Autrement dit, des cassettes C12 pour avoir peu de programmes sur chacune et les retrouver rapidement et si on en abîme une, on risque d'y perdre beaucoup moins.

Enfin, rappelez-vous que les logiciels du commerce sont fournis avec des conditions strictes de droits d'auteur (Copyright).

Vous ne devez pas essayer de copier ou de reproduire des logiciels autrement qu'en accord avec les termes de vente (certains programmes vous conseillent de faire des copies de sauvegarde) même si c'est seulement 'pour un copain'. La position juridique est en voie de modification et la situation va changer substantiellement dans les années qui viennent; ces lois pourront même avoir effet rétrospectif.



# 3 Une brève introduction aux programmes écrits en BASIC AMSTRAD.

Sujets abordés dans ce chapitre

- \* les règles de syntaxe et description de la grammaire.
- \* la commande PRINT, les canaux et l'agencement de l'écran.
- \* la ZONE !?

## 3.1 La base du BASIC

Les éléments essentiels qui rattachent le BASIC intégré du CPC464 au fonctionnement interne de l'ordinateur sont esquissés dans l'Appendice II. Si vous n'avez pas encore utilisé un ordinateur, nous allons essayer de vous guider lentement, mais il est possible que nous fassions des suppositions qui jettent la confusion chez les débutants et ceux qui n'ont jamais fait d'Anglais. Le monde de l'informatique étant ce qu'il est, une connaissance de certains mots anglais est indispensable. De toute façon, il existe de bons livres d'initiation au BASIC, si vous trouvez ces passages un peu trop ardu斯 pour les débutants.

Vous devriez cependant être capable de vous y retrouver dans ce chapitre en faisant les exercices simples demandés sans être obligé de comprendre tout ce qui se passe, bien que l'apprentissage soit plus facile au fur et à mesure que vous avancez dans ce guide.

BASIC est le langage fourni intégré au CPC464. Il est déjà là quand on met en marche et vous accueille avec ce mot

**Ready**

ce qui veut presque dire "toujours prêt."

BASIC est le plus simple des langages. Il est organisé avec des mots clairement définis, une grammaire propre et fonctionne d'une manière parfaitement logique, pourvu qu'on comprenne et respecte les règles.

Le BASIC d'AMSTRAD peut exécuter toutes les commandes détaillées dans ce chapitre. Chaque commande est composée d'un ou plusieurs mot-clés (ou mots réservés = on ne peut pas les utiliser avec un sens différent dans un programme), et peut avoir un certain nombre de paramètres, dont certains sont optionnels. En général chaque paramètre peut être une expression avec des constantes, des variables et des fonctions.

Les combinaisons de lettres et de chiffres sont appelées des chaînes (**STRING** en anglais) et on peut utiliser des données numériques sous la forme décimale, hexadécimale et binaire.

Les dossiers sur cassette sont rangés d'une manière séquentielle (à la suite les uns des autres) par opposition à aléatoire, où un dossier peut être choisi au milieu de beaucoup d'autres sans avoir à passer par les dossiers qui ne sont pas nécessaires.

### 3.2 La structure d'un programme en BASIC

Un programme en BASIC se présente sous la forme de lignes. Une ligne peut comprendre plusieurs commandes, séparées par des deux-points (:) et limitée en longueur à 255 caractères. Un caractère est un nombre, une lettre, un espace ou encore un symbole.

En *Mode direct*, on tape les lignes sur le clavier et il ne faut pas mettre de numéro de ligne. (Par exemple quand on fait **RUN**" puis **[ENTER]**, on est en *Mode Direct*).

En *Mode Programme*, les lignes sont lues à partir du programme qui est dans la mémoire et sont exécutées dans l'ordre strict des nombres qui apparaissent au début de la ligne.

Dans le BASIC d'**AMSTRAD**, on peut ajouter et enlever des lignes à un programme en mode direct et modifier des lignes qui existent déjà. Avant d'exécuter (**RUN**) ou de donner la **LISTe** du programme, le BASIC réorganise l'ordre des lignes dans l'ordre montant, sans se soucier de l'ordre dans lequel on les a tapées.

### 3.3 La Frappe d'une ligne...

En BASIC, une ligne peut avoir 255 caractères, et on doit la terminer en frappant la touche **[ENTER]**. Pendant qu'on tape une ligne il est possible de l'édition (la modifier) et de se servir du curseur de **COPY** pour insérer des caractères venant d'un autre endroit de l'écran (voir chapitre 1, paragraphe 1.2.7).

Chaque mot réservé doit être suivi d'un séparateur, c'est-à-dire un espace, ou un signe d'opération (+, - etc). Ceci parce qu'on peut utiliser ces mots s'ils sont 'noyés' dans d'autres mots (par exemple on pourra avoir une variable appelée 'Listedesclients', car l'ordinateur connaît le mot réservé **LIST**, mais si on met un - à la fin, liste, c'est pour lui un mot différent.)

L'AMSTRAD BASIC accepte les mots réservés en majuscules ou en minuscules.

La commande **PRINT** peut s'abréger en ? (point d'interrogation) et sous cette forme, on n'a pas besoin d'un séparateur. Les symboles d'opération (+, -, \*, et **MODulo**) peuvent aussi faire office de séparateurs.

De même, l'apostrophe ' (obtenue en pressant **[SHIFT]** et 7) peut remplacer la commande **REM** (= REMarque).

Les espaces en trop sont ignorés et on peut s'en servir pour mieux présenter les programmes, quand il y a des boucles, etc.

### 3.4 La terminologie

Pour pouvoir décrire les commandes et mots réservés du BASIC, des termes formels mais simples doivent être utilisés. Chaque commande est décrite comme elle doit apparaître quand on la tape au clavier, avec tout ce qui est variable, options, paramètres indiqués par des indicateurs de type parenthèses qui se rapportent à des éléments précisés dans la description qui suit.

Ces éléments sont représentés par des noms entre les parenthèses en coin, . . Par exemple, si on veut représenter un nombre, on écrira:

·expression numérique·

Quand il n'y a pas de parenthèses en coin, il faut l'écrire tel quel. Par exemple la commande STOP s'écrit.

STOP

Quand quelque chose est facultif dans une définition, ce qui est facultatif est mis entre des parenthèses droites []. Si par exemple une expression numérique est facultative, on écrira:

[·expression numérique·]

Si une option (donc facultative) est répétée, une astérisque est mise après les parenthèses droites; ainsi, si on veut avoir une chaîne de chiffres, on écrira:

[·chiffre·]\*

ce qui peut être

- rien du tout
- ou 3
- ou 3456

Souvent une liste d'éléments séparés par des virgules est utilisée, souvent en version réduite du type suivant

·liste d': ·expression· veut dire: ·expression·[, ·expression·]\*

la première ·expression· est obligatoire, les suivantes facultatives. ou encore

·liste de: [#]·nombre· veut dire [#]·nombre·[, [#]·nombre·]\*

ce qui donne par exemple ou on encore

3,4  
3,8,2  
3,2,5,7,8 etc.

la liste peut avoir un seul élément, mais s'il y en a plusieurs, il faut placer une virgule avant chaque élément supplémentaire.

Les nombres sont représentés de différentes manières

- a. ·nombres réels· ..... sont les nombres normaux, sans exposants.
- b. ·nombres E· ..... nombres à la puissance 10 suivant la forme 2E4 (2 fois 10 à la puissance 4, ou  $2 \cdot 10^4$ )  
... l'exposant pouvant-être positif ou négatif.

c. ·nombre en base X)

X pouvant être 10, 16 ou 2

base 10 (décimal)	100
base 16 (hexadécimal)	&64 ou bien &H64 .....
base 2 (binaire)	8X1100100

### 3.5 La pratique au secours de la théorie:

De l'utilisation de la commande PRINT

Pour mieux montrer l'usage de cette terminologie, voici quelques exemples du BASIC en plein travail.

Un des mots du BASIC qui utilise la plupart des éléments de cette terminologie est la commande PRINT. (Qui signifie aussi bien Afficher qu'Imprimer, cela dépend des cas comme nous allons le voir).

Une commande est un mot réservé ou une affirmation qui marche avec l'ordinateur aussi bien en mode direct qu'en mode programme. Une fonction a besoin de la présence d'une commande pour 'invoquer la fonction'.

**PRINT FRE ("")**

où nous demandons à l'ordinateur combien de mémoire libre il nous reste.

Pour que l'ordinateur vous donne une réponse à une question, il faut lui donner trois éléments:

1. Où voulez-vous la réponse - l'écran ou l'imprimante ou 'ailleurs'?
2. Vous devez donner à l'ordinateur des données de travail
3. Vous devez dire à l'ordinateur ce qu'il faut faire avec les données.

PRINT indique à l'ordinateur de mettre les résultats sur un 'canal' de sortie, où le canal est identifié par un nombre de 0 à 9, que l'on appelle ·numéro du canal·:

0...7 sont des canaux de sortie texte sur des fenêtres définies auparavant par la commande WINDOW.

8 est la sortie imprimante qui marche seulement si on a branché une imprimante de type parallèle Centronics.

9 est le canal de sortie sur cassette, que l'on doit d'abord avoir 'ouvert' auparavant dans le programme.

Une forme concise de la commande PRINT (qui n'emploie pas le format PRINT USING) se présente donc ainsi:

**PRINT [#·numéro du canal·]·.liste d'affichage·]**

Les parenthèses droites signifient qu'il est facultatif de préciser le ·numéro du canal· ou qu'on peut ne pas donner à la commande PRINT une liste des choses à afficher (dans ce cas, l'ordinateur affiche une ligne vierge, essayez donc). Si vous ne précisez pas le numéro du canal pour la sortie, le CPC464 suppose que c'est le canal 0, c'est à dire l'écran où est le curseur. Essayez cette ligne (sans oublier [ENTER] à la fin de la ligne pour dire à l'ordinateur que la ligne est finie):

**PRINT "Salut les copains"**

L'ordinateur répond

**Salut les copains**

Les guillemets ont disparu: ils servent au BASIC à savoir où la liste à afficher doit débuter et finir.

Maintenant tapez:

**PRINT #0, "Salut les copains"**

...et le résultat est le même.

Mais si vous faites

**PRINT #4, "salut les copains"**

la ligne va s'afficher en haut et à gauche de l'écran car c'est la première ligne à utiliser le canal 4, qui couvre tout l'écran, à moins qu'on ne lui ait donné une autre valeur avec la commande **WINDOW**. Le message d'accueil a utilisé le canal 0 (celui qui est utilisé par défaut, si on ne dit rien) et le texte a été envoyé sur le canal qui se trouvait après les caractères déjà affichés.

Cette caractéristique du BASIC d'AMSTRAD est particulièrement puissante, car elle permet des images et des affichages complexes avec des commandes simples et contribue à l'agencement d'un écran bien présenté.

BASIC va afficher (**PRINT**) tout ce que vous mettez entre les guillements sans se soucier de savoir si cela a du sens ou non. Des mots réservés peuvent être mis dans la liste à afficher:

**PRINT "4\*4"**

et l'ordinateur répond par

**4\*4**

Pour dire au BASIC de faire la multiplication (4 fois 4, et le symbole de multiplication est \*), il ne faut pas mettre les guillements:

**PRINT 4\*4**

...et le BASIC vous donne la réponse

**16**

Le résultat est une ligne plus loin que d'habitude car le BASIC réserve la place pour le signe négatif dans le cas d'un nombre négatif.

La commande **PRINT** a plusieurs autres formes, surtout en utilisant la capacité à donner une forme suivant une série de cadres standards.

La liste à afficher dans la commande **PRINT** se rapporte à la liste des détails à imprimer. Ce peut être un nombre, une expression variable ou expression en chaîne, ce qui signifie qu'une chaîne variable a été définie auparavant (exemple **HELLO\$**), ou toute chose entre guillemets " ".

**PRINT USING** donne un cadre pour afficher les nombres, et pour que les colonnes soient alignées et pour se débarrasser des restes dans une division ou une fraction.

### 3.6 le cadre PRINT USING et les ZONES

Au départ, le BASIC définit la largeur de l'écran en ZONES de 13 colonnes de large. Quand il y a une virgule dans la commande PRINT, l'élément suivant est imprimé dans la ZONE suivante. S'il y a moins de colonnes disponibles sur une ligne que la ZONE a défini, alors le BASIC va débuter l'élément suivant sur une ligne suivante. Un élément n'est pas coupé par un bord de ligne.

S'il n'y a pas de format précisé par USING, le BASIC affiche les nombres positifs précédés d'un espace, les négatifs précédés du signe - (moins). Tous les nombres sont suivis d'un espace.

**Mais ATTENTION:** en anglais et en BASIC, ils utilisent le point à la place de la virgule:

3,5 (3 virgule 5) s'écrit 3.5 (3 point 5)

Ce point décimal comme ils l'appellent est oublié quand la partie fractionnaire est peu significative.

Le BASIC d'AMSTRAD n'utilise pas la touche [TAB] comme touche de colonnes, car il y a de grandes différences sur le sens et la fonction de cette touche dans le dialecte BASIC. Quand on presse la touche [TAB], une flèche [right arrow] s'affiche, (même chose que [CTRL] et I pressés ensemble), mais n'a pas de signification dans le BASIC d'AMSTRAD.

Mais il y aura des progiciels qui utilisent cette touche comme TABulateur notamment les traitement de textes et la gestion de tableaux.

### 3.7 PRINT TAB(<nombre entier>) (<liste à afficher>)

L'effet de ce cadre de commande est mis en évidence par l'exemple suivant:

```
5 MODE 2: INK 1,0: INK 0,9  
10 FOR N=1 TO 5  
20 ZONE 4030 PRINT TAB(N*4)"HI",N  
40 NEXT
```

Ce programme illustre à la fois ZONE avec une virgule, et la fonction TAB( ) au travail. On peut recommencer en changeant la ligne 10 en

```
10 FOR N= -5 TO 5
```

L'instruction TAB déplace le début de PRINT (affichage) par le nombre d'espaces précisé dans le <nombre entier>. (Zone peut avoir une valeur de 1 à 255, voir au chapitre 8.)

La forme PRINT USING est utilisée pour arranger le résultat de calculs où des nombres réels (ou fractionnaires) donneraient des ensembles de chiffres peu présentables. C'est assez compliqué et il vaut mieux le présenter par des exemples car la forme technique serait.

PRINT [<numéro du canal>,<list à afficher>,<clause USING>][<séparateur>]

et la clause **USING** se subdivise en  
**USING** ·expression en chaîne·;{·**USING** liste·}  
qui se divise alors en  
·expression·;·séparateur··expression··]\*

Vous me suivez ?

C'est aussi facile à comprendre que Polytechnique et les MARX BROTHERS réunis.

Voyons plutôt un exemple

**PRINT** 123.456 , **USING** "###.##"; 4567.896

et le résultat est

123.456 %4567.90

Ceci illustre un certain nombre de points importants: d'abord, ce qui est avant le **USING** n'est pas modifié. Ensuite la clause **USING** réserve un certain nombre de places et si le nombre à imprimer dépasse le nombre à gauche cela imprime quand même, mais si cela dépasse à droite on voit un % apparaître et le nombre est arrondi au plus près.

En outre la virgule après le nombre 123.456 a conduit le nombre suivant à être imprimé dans la ZONE suivante. S'il y avait eu un point virgule, le nombre aurait été affiché un espace après 123.456. Il y a toujours un espace au moins pour séparer deux nombres (évidemment).

Essayez ceci:

**PRINT** 123.456 , **USING** "###.##+";4567.899

et le signe apparaît à la fin du nombre (voir chapitre 8 ).

**PRINT USING** est une commande très utile pour la présentation de tableaux. Il vous prévient toujours si le format pose des problèmes (avec le symbole %).



# 4 Variables, opérations et données

*Manipulation de l'information dans un programme en BASIC.*

Sujets abordés dans ce chapitre

- \* Faisons connaissance
- \* types de variables: réelles, entières et chaînes
- \* opérations, expressions logiques
- \* rangées, arrangements, tables
- \* DATA (= données)

## 4.1 Où est donc ce mot réservé

Notons d'abord que les commandes et les autres mots réservés du BASIC d'AMSTRAD sont délimités par des espaces, des signes de ponctuation ou d'opération. Les programmes sont plus faciles à vérifier car, même si vous tapez ces mots réservés en minuscules, quand le programme est LISTé, tous les mots réservés seront devenus des mots en majuscule. (Si vous tapez origin 50,100, cela devient après un LIST: ORIGIN 50,100. Si vous tapez origine 50,100, cela reste origine 50,100 après un LISTing et vous avez une Syntax error si vous essayez RUN.)

AMSTRAD BASIC vous permet de 'noyer' un mot réservé dans une variable: dans le BASIC d'AMSTRAD, une variable est un nom que l'on donne à un élément particulier. Ce peut être seulement une lettre (une variable doit toujours commencer par une lettre, pas un chiffre), mais il est plus facile de comprendre un long programme si vous utilisez des noms qui se rapportent à la variable:

**REPONSE-4\*4: print REPONSE**

Les variables du BASIC d'AMSTRAD peuvent avoir jusqu'à 40 caractères (la première doit être une lettre) et toutes les 40 sont importantes. Il ne doit pas y avoir d'espace ou alors le BASIC va lire seulement les lettres jusqu'à l'espace et répondre alors

**Syntax error**

Ce qui veut dire qu'on a mis une succession de lettres non acceptable. (Voir Appendice VIII.) Si vous voulez avoir des phrases avec des mots séparés mettez un point entre chaque mot où vous voudriez un espace. Toutes les formes de variables sont permises. (Pour ceux qui savent déjà, rappelons qu'il est nécessaire de DIMENSIONNER les arrangements, voir ce chapitre au paragraphe 4.11.)

## 4.2 Les Raccourcis

Pour ne pas avoir à taper **PRINT** toutes les cinq minutes, vous pouvez mettre **? à la place et BASIC comprendra **PRINT** (à moins que ce ne soit entre des guillemets et il le prendra alors pour un point d'interrogation). Avec l'option **?** on n'a pas besoin de l'espace après le point d'interrogation. Quand on tape la ligne comme un programme:**

```
10?4*4  
run  
16
```

Le résultat ne change pas, mais si on fait **L I S T**, on voit apparaître:

```
10 PRINT 4*4
```

BASIC a remplacé **?** par **PRINT** et mis un espace après. On peut aussi écrire:

```
10 ? "HELLO, comment vas-tu?"
```

Le point d'interrogation entre les guillemets ne change pas. On peut aussi ne pas mettre le dernier guillement mais ce n'est pas une bonne habitude à prendre.

## 4.3 Lignes à déclarations multiples et calculs composés.

On peut faire plusieurs opérations en BASIC, en fait autant que vous voulez dans la limite des 255 caractères. Comme d'habitude, les déclarations doivent être séparées par deux points:

```
?2*8/5+5-4*777E3/9
```

donne pour résultat:

```
-1.036E+12
```

Cependant il est essentiel de comprendre l'ordre dans lequel les opérations mathématiques sont effectuées par le BASIC ou vous allez faire des erreurs fondamentales:

↑	Exponentiation = élévation d'un nombre à une puissance
-	signe moins d'un nombre négatif
*	Multiplication
/	Division
\	Division par un entier, le résultat étant un entier et le reste étant laissé de côté par le BASIC
+	Addition
-	Soustraction
<b>MOD</b> n	Division modulo n, le reste étant le résultat (uniquement dans l'ensemble des entiers).

Tout ce qui est entre parenthèses ( ) est effectué en premier et si le contenu des parenthèses est un calcul composé, l'ordre est celui annoncé ci-dessus. Il faut toujours avoir dans ce cas autant de parenthèses à droite que de parenthèses à gauche ou alors le système répond par une Syntax Error.

## 4.4 Mise en route

Nous avons avancé à grands pas et vous devriez avoir assez appris sur les règles du BASIC pour commencer à programmer sérieusement . Les mots du BASIC seront introduits quand on en aura besoin, pour plus de détails reportez vous au Chapitre 8 pour le guide alphabétique des mots réservés, ou si vous voulez mieux comprendre).

Le mode direct (ou immédiat) vous permet de rentrer un programme en séparant les commandes par: mais une fois que vous avez fait [ENTER] le programme est exécuté puis disparaît de la mémoire. On utilise donc ce mode pour des commandes uniques (RUN, LIST etc) et des petits programmes courts dont on a besoin une seule fois.

## 4.5 Déclarations conditionnelles et logiques.

Le BASIC utilise beaucoup la caractéristique de l'ordinateur de pouvoir faire des opérations répétées vite et sans se fatiguer (quelle chance!). Un certain nombre de commandes sont utilisées pour programmer ces boucles (loops); des commandes qui débutent, continuent et déterminent la fin d'une boucle, si un ensemble prévu de conditions est rempli.

Les derniers de ces éléments de contrôle d'une boucle sont les relations, qui servent à comparer des données entre elles ou à des données de référence. Ces relations sont les suivantes:

- < moins grand que
- = moins grand que ou égal à
- = égal à
- > plus grand que
- = plus grand ou égal à
- <> non égal à (différent de)

Voilà un bref programme pour exposer ces relations, sur un sujet qui nous intéresse.

Si vous n'avez pas le message Ready sur votre écran, faites [CTRL] [SHIFT] et [ESC] en les pressant en même temps.

Vous pouvez maintenant taper le programme suivant en vous reportant au chapitre 1.2.7 si vous avez des problèmes.

```
10 INPUT "QUEL EST VOTRE SALAIRE MENSUEL";SALAIRE
20 IF SALAIRE <10000 THEN GOTO 30 ELSE 40
30 PRINT "DEMANDEZ UNE AUGMENTATION":END
40 PRINT "DEMANDEZ UNE VOITURE DE SOCIETE PLUS
    PUISSANTE"
```

Exécutez ce petit programme en tapant:

RUN

puis [ENTER], obligatoirement. Le ordinateur vous demande:

QUEL EST VOTRE SALAIRE MENSUEL?

(remarquez que l'ordinateur ajoute toujours le point d'interrogation quand vous utilisez le mot INPUT)

Répondez à la question par un nombre - pas de lettres ni de signes ni virgules, puis faites [ENTER].

Ajoutez la ligne 5 après le message Ready:

5 CLS

Faites RUN de nouveau pour avoir un écran net. Ajoutez à votre programme la ligne 50 comme suit:

```
5 CLS
10 INPUT "QUEL EST VOTRE SALAIRE MENSUEL";SALAIRE
20 IF SALAIRE <10000 THEN GOTO 30 ELSE 40
30 PRINT "DEMANDEZ UNE AUGMENTATION":END
40 PRINT "DEMANDEZ UNE VOITURE DE SOCIETE PLUS
    PUISSANTE"
50 IF SALAIRE >30000 THEN PRINT "ET PRENEZ UN BON
CONSEILLER FISCAL"
run
```

Le END (= la fin) de la ligne 30 arrête le programme et donne le message Ready. Comme il n'y pas de fin (END) à la ligne 40, le programme lit la ligne 50 et vous donne un autre bon conseil!

Pour continuer, ajoutons une ligne 60.

```
5 CLS
10 INPUT "QUEL EST VOTRE SALAIRE MENSUEL";SALAIRE
20 IF SALAIRE <10000 THEN GOTO 30 ELSE 40
30 PRINT "DEMANDEZ UNE AUGMENTATION":END
40 PRINT "DEMANDEZ UNE VOITURE DE SOCIETE PLUS
    PUISSANTE"
50 IF SALAIRE >30000 THEN PRINT "ET PRENEZ UN BON
CONSEILLER FISCAL"
60 IF SALAIRE >25000 THEN PRINT "DIS DONC, T'AS
    PAS CENT BALLES ?"
run
```

Notez que les relations · et · servent de limites et qu'on n'est pas obligé de mettre un espace avant ou après ces symboles. Si vous répondez 26000 en exécutant ce programme, le BASIC ne s'arrête pas à la ligne 50 et passe directement à la ligne 60.

Vous pouvez ainsi construire un programme en utilisant les sujets abordés jusqu'à présent. Remarquez comment les programmes s'étendent, la plupart des programmes sont comme cela, ce qui introduit peut-être le concept le plus important de la programmation en BASIC...

## 4.6 Evolution: L'origine des programmes.

La manière dont le BASIC vous permet de construire des programmes en ajoutant des lignes au fur et à mesure est une des choses les plus agréables du BASIC. Les puristes diront que ce n'est pas %<structuré%>; les现实ists considéreront l'intérêt de celui qui veut apprendre et doit avancer par petites étapes.

Prenons notre programme de nouveau, avec une boucle à la ligne 70 qui fait revenir à la ligne 5 (GOTO 5) après avoir fait une pause assez longue pour que vous puissiez lire l'écran:

```
5 CLS
10 INPUT "QUEL EST VOTRE SALAIRE MENSUEL";SALAIRE
20 IF SALAIRE <10000 THEN GOTO 30 ELSE 40
30 PRINT "DEMANDEZ UNE AUGMENTATION":END
40 PRINT "DEMANDEZ UNE VOITURE DE SOCIETE PLUS
    PUISSANTE"
50 IF SALAIRE >30000 THEN PRINT "ET PRENEZ UN BON
    CONSEILLER FISCAL"
60 IF SALAIRE >25000 THEN PRINT "DIS DONC, T'AS
    PAS CENTBALLES?"
70 for n=1 to 900:next n:goto 5
run
```

Remarquez que cette ligne supplémentaire est tapée en minuscules pour vous rappeler que le BASIC d'AMSTRAD fait la différence entre un nom de variable et un mot réservé: faites [ESC] deux fois et LIST et voyez comme l'ordinateur change les mots réservés pour les mettre en majuscules et laisse la variable n en minuscule.

La ligne 70 fait attendre pendant que l'ordinateur compte de 1 à 900 avant de passer à la commande suivante - GOTO 5. De cette manière le programme repart au départ et ne peut s'arrêter. Le seul moyen d'en sortir est de presser [ESC] deux fois consécutives: si on presse une fois, on arrête momentanément le programme, et en pressant deux fois on repasse en mode direct sans perdre le contenu de la mémoire.

En fait, à moins que vous pressiez [ESC] pendant que l'ordinateur compte (ligne 70), le programme s'arrête immédiatement car l'ordinateur ne fait rien lorsqu'il attend qu'on réponde (= il attend un INPUT). La ligne où il attendait est indiquée:

**Break in 10** (arrêt à la ligne 10)

Si vous arrivez à appuyer sur [ESC] deux fois pendant qu'il compte, vous aurez le message:

**Break in 70**

Si vous arrêtez pendant la boucle de la ligne 70, l'opération est suspendue, et on peut recommencer en pressant une touche quelconque. Si vous avez arrêté un programme vous pouvez repartir après le Break (fi arrêt) avec la commande:

**CONT**

et le programme va **CONTinuer** en recommençant là où il s'était arrêté.

Quoique vous fassiez avec la touche [ESC], vous ne perdez pas ce que vous avez dans la mémoire, à moins que vous ne disiez à l'ordinateur d'effacer tout ce qu'il a en mémoire avec la commande:

**NEW**

(= nouveau = neuf) ou bien vous faites une remise à zéro en appuyant sur les trois touches **[CTRL][SHIFT]** et **[ESC]**.

Vous ne devriez donc pas remettre à zéro par accident et si vous devez faire une remise à zéro, demandez vous d'abord si vous ne devez pas sauvegarder sur la cassette avant de commencer.

## 4.7 Encore des variables, et puis des chaînes.

L'essence de l'informatique est la variable. Si une expression mathématique contient une variable, alors le résultat aussi est variable.

Les variables ont trois caractéristiques: un nom, un type et une 'organisation'. Nous avons parlé des noms auparavant (4.1) - le type est une option, on pourrait donc définir une variable selon les règles du paragraphe (3.4) comme:

**.nom· | ·symbole du type· ]**

Ces symboles sont:

% pour des nombres entiers. Toute partie décimale ou fractionnaire est ignorée. Les nombres entiers prennent moins de place dans la mémoire et les programmes où il n'y a pas de manipulations de parties décimales peuvent marcher plus vite si les variables sont **DEFinies** comme des entiers.

La commande **DEFINT** dit que les variables sont des nombres entiers et leur valeur peut varier de **-32768 à +32767**

! Précise qu'une variable est réelle c'est-à-dire qu'il peut y avoir une partie décimale. L'ordinateur suppose que tout variable est réelle, donc on a seulement besoin de **DEFinir** des variables réelles si on a utilisé auparavant la commande **DEFINT**. Les variables réelles peuvent prendre toute valeur de **2.9E-39 à 1.7E+38**. (Autrement dit des nombres avec 39 et 38 chiffres respectivement: vous voyez pourquoi on prend cette notation!)

**\$** indique une variable en chaîne, où le contenu peut être des lettres, des chiffres, un mélange des deux, etc... En résumé un ensemble de caractères que l'on met entre deux guillemets. Par exemple:

NOM \$ = "JEAN DUPONT"

En continuant avec notre programme évolutif, ajoutons la ligne 6 et modifions la ligne 60.

```
5 CLS
6 INPUT "QUEL EST VOTRE NOM";NOM$
10 INPUT "QUEL EST VOTRE SALAIRE MENSUEL";SALAIRE
20 IF SALAIRE <10000 THEN GOTO 30 ELSE 40
30 PRINT "DEMANDEZ UNE AUGMENTATION":END
40 PRINT "DEMANDEZ UNE VOITURE DE SOCIETE PLUS
PUISANTE"
50 IF SALAIRE >30000 THEN PRINT "ET PRENEZ UN BON
CONSEILLER FISCAL"
60 IF SALAIRE >25000 THEN PRINT "DIS DONC, T'AS
PAS CENT BALLES "
70 for n=1 to 900:next n:goto 5
run
```

On a placé un espace après balles pour que le nom ne soit pas collé. Essayez si vous ne me croyez pas! Le point virgule; après une commande **PRINT** ou **INPUT** empêche l'ordinateur d'aller à la ligne. On peut aussi travailler sur les nombres entiers en ajoutant la ligne 61. Vous devez taper la ligne 61 et modifier la ligne 70 car vous avez besoin de plus de temps pour lire ce qu'il y a sur l'écran.

```
5 CLS
6 INPUT "QUEL EST VOTRE NOM";NOM$
10 INPUT "QUEL EST VOTRE SALAIRE MENSUEL";SALAIRE
20 IF SALAIRE <10000 THEN GOTO 30 ELSE 40
30 PRINT "DEMANDEZ UNE AUGMENTATION":END
40 PRINT "DEMANDEZ UNE VOITURE DE SOCIETE PLUS
PUISANTE"
50 IF SALAIRE >30000 THEN PRINT "ET PRENEZ UN BON
CONSEILLER FISCAL"
60 IF SALAIRE >25000 THEN PRINT "DIS DONC, T'AS
PAS CENT BALLES "
61 SALAIREPARJOUR=SALAIRE/30: PRINT"CELA FAIT
";SALAIREPARJOUR;" FRANCS PAR JOUR"
70 for n=1 to 500:next n: goto 5
run
```

Ajoutons la ligne 62 car le résultat en chiffres ne fait pas net. Il faut arrondir:

```
5 CLS
6 INPUT "QUEL EST VOTRE NOM";NOM$
10 INPUT "QUEL EST VOTRE SALAIRE MENSUEL";SALAIRE
20 IF SALAIRE <10000 THEN GOTO 30 ELSE 40
30 PRINT "DEMANDEZ UNE AUGMENTATION":END
40 PRINT "DEMANDEZ UNE VOITURE DE SOCIETE PLUS
    PUISSANTE"
50 IF SALAIRE >30000 THEN PRINT "ET PRENEZ UN BON
    CONSEILLER FISCAL"
60 IF SALAIRE >25000 THEN PRINT "DIS DONC, T'AS
    PAS CENT BALLES "
61 SALAIREPARJOUR=SALAIRE/30 PRINT"CELA FAIT
    ";SALAIREPARJOUR;" FRANCS PAR JOUR"
62 SALAIRE.ENTIER%=SALAIREPARJOUR:PRINT
    "OUBIEN ";SALAIRE.ENTIER%;" FRANCS, SI LES
    CENTIMES NE VOUS PREOCCUPENT PAS TROP"
70 FOR n=1 TO 5000:NEXT n: GOTO 5
```

et RUN de nouveau.

Faites attention à garder le symbole %, car on peut avoir deux variables, une réelle et une entière avec le même nombre entier, le nombre entier ayant le symbole % pour le distinguer. Voyez aussi comme l'ordinateur coupe les lignes trop longues. Il vaut mieux utiliser le MODE 2 pour écrire de longs programmes, car il est beaucoup plus facile de lire des programmes qui n'ont pas trop de lignes.

Pour avoir le MODE 2, avec des couleurs lisibles, faisons en commande directe.

```
MODE 2
INK 1,0
INK 0,13
BORDER 0,13
```

puis faites LIST encore une fois.

## 4.8 Présentation de l'écran

Une partie de l'évolution de votre programme consiste à mettre de l'ordre de temps à autre. La première chose que nous pouvons faire, c'est utiliser la commande RENUM (refaire les numéros) pour numérotter les lignes de 10 en 10.

Après la ligne Ready, tapez

```
RENUM
```

et puis LIST:

```
10 CLS
20 INPUT "QUEL EST VOTRE NOM";NOM$
30 INPUT "QUEL EST VOTRE SALAIRE MENSUEL";SALAIRE
40 IF SALAIRE <10000 THEN GOTO 50 ELSE 60
50 PRINT "DEMANDEZ UNE AUGMENTATION":END
60 PRINT "DEMANDEZ UNE VOITURE DE SOCIETE PLUS
    PUISSANTE"
70 IF SALAIRE >30000 THEN PRINT "ET PRENEZ UN BON
    CONSEILLER FISCAL"
80 IF SALAIRE >25000 THEN PRINT
    "DIS DONC, T'AS PAS CENT BALLES"
90 SALAIREPARJOUR=SALAIRE/30
    PRINT"CELA FAIT ";SALAIREPARJOUR;" FRANCS
    PAR JOUR"
100 SALAIRE.ENTIER%=SALAIREPARJOUR:PRINT
    "OUBIEN ";SALAIRE.ENTIER%;" FRANCS, SI LES
    CENTIMES
    NE VOUS PREOCCUPENT PAS TROP"
110 FOR n=1 TO 5000: NEXT n : GOTO 10
```

Toutes les lignes ont été renumérotées, y compris celles à l'intérieur du programme (GOTO 5 est devenu GOTO 10), car cela ne serait pas très intéressant autrement.

Nous allons améliorer la présentation et pour cela suspendons provisoirement la ligne 110 en mettant une REMarque:

```
10 CLS
20 INPUT "QUEL EST VOTRE NOM";NOM$
30 INPUT "QUEL EST VOTRE SALAIRE MENSUEL";SALAIRE
40 IF SALAIRE <10000 THEN GOTO 50 ELSE 60
50 PRINT "DEMANDEZ UNE AUGMENTATION":END
60 PRINT "DEMANDEZ UNE VOITURE DE SOCIETE PLUS
    PUISSANTE"
70 IF SALAIRE >30000 THEN PRINT "ET PRENEZ UN BON
    CONSEILLER FISCAL"
80 IF SALAIRE >25000 THEN PRINT
    "DIS DONC, T'AS PAS CENT BALLES"
90 SALAIREPARJOUR=SALAIRE/30
    PRINT"CELA FAIT ";SALAIREPARJOUR;" FRANCS
    PAR JOUR"
100 SALAIRE.ENTIER%=SALAIREPARJOUR:PRINT
    "OUBIEN ";SALAIRE.ENTIER%;" FRANCS, SI LES
    CENTIMES
    NE VOUS PREOCCUPENT PAS TROP"
110 REM:FOR N=1 TO 5000: NEXT N : GOTO 10
```

Mettre REM: au début d'une ligne fait que le reste de la ligne est ignoré par le BASIC qui termine le programme et retourne au message Ready, en laissant l'image sur l'écran. Faisons maintenant:

### 15 Mode 1

La ligne 15 va fixer le mode d'écran pour le programme ce qui fait que même si vous étiez auparavant dans un autre mode, la ligne 15 va faire exécuter le programme en mode 2. La commande Mode produit un CLS (efface l'écran), la ligne 10 est inutile mais on va la laisser là pour le moment.

Maintenant, exécutez (RUN) le programme et répondez

```
QUEL EST VOTRE NOM?JEAN
QUEL EST VOTRE SALAIRE MENSUEL? 40000
DEMANDEZ UNE VOITURE DE SOCIETE PLUS PUISSANTE
ET PRENEZ UN BON CONSEILLER FISCAL
DIS DONC, T'AS PAS CENT BALLES JEAN
CELA FAIT 1333.333333 FRANCS PAR JOUR"
OU BIEN 1333 FRANCS, SI LES CENTIMES
NE VOUS PREOCCUPENT PAS TROP
```

La présentation n'est pas fameuse, surtout la coupure... Améliorons...

```
25 PRINT:PRINT
85 PRINT
```

et éditons la ligne 100

```
100 SALAIRE.ENTIER%=SALAIRE\REPARJOUR:PRINT
    "OU BIEN ";SALAIRE.ENTIER%;" FRANCS, SI LES
    CENTIMES":PRINT
    "NE VOUS PREOCCUPENT PAS TROP"
```

Faites RUN de nouveau

La présentation a changé.

Ajoutons la ligne 120 pour déplacer le Ready en tapant

```
120 ?:?:?:?:
```

ou supprimez le Ready en faisant

```
120 GOTO 120
```

Une fois que vous avez fait cela, le seul moyen d'arrêter le programme est de faire [ESC] deux fois. Rappelez vous que ? est un moyen rapide d'écrire PRINT. Faisons LIST pour vérifier.

```

10 CLS
15 MODE 1
20 INPUT "QUEL EST VOTRE NOM";NOM$
25 PRINT
30 INPUT "QUEL EST VOTRE SALAIRE MENSUEL";SALAIRE
40 IF SALAIRE <10000 THEN GOTO 50 ELSE 60
50 PRINT "DEMANDEZ UNE AUGMENTATION":END
60 PRINT "DEMANDEZ UNE VOITURE DE SOCIETE PLUS
    PUISSANTE"
70 IF SALAIRE >30000 THEN PRINT "ET PRENEZ UN BON
    CONSEILLER FISCAL"
80 IF SALAIRE>25000 THEN PRINT
    "DIS DONC, T'AS PAS CENT BALLES"
85 PRINT
90 SALAIREPARJOUR=SALAIRE/30
    PRINT"CELA FAIT";SALAIREPARJOUR;""
    FRANCS PAR JOUR"
100 SALAIRE.ENTIER%=SALAIREPARJOUR:PRINT
    "OUBIEN";SALAIRE.ENTIER%;" FRANCS,
    SI LES CENTIMES : PRINT "
    NE VOUS PREOCCUPENT PAS TROP"
110 REM:FOR n=1 TO 5000: NEXT n : GOTO 10
120 GOTO 120

```

## 4.9 LOCATE (=Place)

Jusqu'à maintenant, la plupart des commandes en BASIC ont été utilisées avec la syntaxe BASIC standard comprise par la plupart des micros. LOCATE est une commande spécifique du BASIC d'AMSTRAD (qu'on trouve sur d'autres BASIC) qui vous permet de placer le curseur à une position quelconque de l'écran.

**LOCATE 10,4**

Place le curseur de texte à la dixième colonne, quatrième ligne à partir du haut de l'écran. Si on utilise le mode direct, le curseur va se placer correctement mais aussitôt après le message Ready apparaît et le curseur se déplace à gauche. Tapez.

**16 LOCATE 10,4**  
**RUN**

La première question change de position. La ligne suivante commence à gauche vu que la ligne 25 met des lignes vides. Faites [ESC] deux fois, puis ajoutez la ligne 26.

**26 LOCATE 10,4**  
**RUN**

La deuxième question est écrite par dessus la première. Vous pouvez placer un texte où vous voulez sur l'écran, et utiliser les grilles de coordonnées de l'Appendice VI.

Si vous voulez avoir toutes les questions et commentaires sur la même ligne, il faut précéder chaque LOCATE par un CLS (efface l'écran).

Les coordonnées de LOCATE peuvent être des variables elles-mêmes, produites dans le programme. Le programme "SALAIRE" est bien usé maintenant, nous allons donc aborder quelque chose de nouveau. Si vous voulez sauvegarder ce programme, faites-le en utilisant les commandes de cassette du chapitre 2. Vous avez peut-être commencé à l'apprécier et vous pourrez ensuite l'agrandir.

## 4.10 IF...THEN (si...alors)

Son usage est fréquent. IF ·expression logique· THEN GOTO ·numéro de ligne· est une des formes de la commande. (S'il y a un GOTO, le THEN n'est pas obligatoire. Voir chapitre 8). IF (= si) vérifie si le résultat de ·l'expression logique· est vrai, auquel cas (THEN) il effectue ce qu'on lui dit. La commande IF peut intervenir dans des boucles récursives (à répétition). Remettez l'ordinateur à zéro en faisant NEW pour changer (notez qu'il reste quelque chose sur l'écran) et tapez:

```
1 MODE 1
10 AMSTRAD=0
20 PRINT "Ordinateur Personnel Couleurs AMSTRAD
CPC464"
30 AMSTRAD=AMSTRAD+1
40 IF AMSTRAD <10 GOTO 20
RUN
```

La commande d'affichage de la ligne 20 est répétée jusqu'à ce que la ligne 40 soit complète. Ainsi la ligne 40 fait une boucle de retour sur la ligne 20. Voyez la signification du terme variable quand la valeur donnée à AMSTRAD change avec chaque boucle.

Si vous voulez voir ce qui arrive à la valeur d'AMSTRAD pendant le programme, ajoutez la ligne 35:

```
35 LOCATE 1,20: PRINT AMSTRAD: LOCATE 1, AMSTRAD
run
```

Si cela va trop vite, faites une boucle de ralenti

```
36 FOR T=1 TO 500: NEXT
```

Ajoutez y une touche de couleur (si vous avez la couleur) en faisant.

```
34 BORDER AMSTRAD
```

Cette ligne change la couleur du cadre suivant le nombre de la ligne 30. Faites de nouveau LIST.

```
1 MODE 1
10 AMSTRAD=0
20 PRINT "Ordinateur Personnel Couleurs AMSTRAD
CPC464"
30 AMSTRAD=AMSTRAD+1
34 BORDER AMSTRAD
35 LOCATE 1,20:PRINT AMSTRAD:LOCATE 1,AMSTRAD
36 FOR n=1 TO 500:NEXT
40 IF AMSTRAD <10 GOTO 20
RUN
```

Et comme vous voulez voir toutes les couleurs de l'AMSTRAD, modifiez la ligne 40

```
40 IF AMSTRAD <26 GOTO 20
```

Exécutez le programme et cela varie du plus foncé au plus clair. On peut faire un message plus clair à la ligne 35:

```
35 LOCATE 1,20: PRINT "cadre"; AMSTRAD:  
LOCATE 1, AMSTRAD  
run
```

A propos de BORDER (le cadre pour nous, ou la bordure si vous préférez mais le Cadre Noir a de l'allure) quand le programme est terminé et que vous êtes Ready, faites donc

```
BORDER 14,6
```

joli, n'est-ce pas?

(vous en apprendrez plus au chapitre suivant sur les graphiques) Et puis, pour finir en beauté

```
ink 1,18,16
```

```
puisspeed ink 1,5
```

après cela 2 [ESC] consécutifs, un cachet d'aspirine et vous pouvez sauvegarder sur une cassette (SAVE "Colorama", par exemple). Puis remise à zéro.

## 4.11 Arrangements

Il y a des programmes qui nécessitent beaucoup de variables pour manipuler les données. Cela pose un problème de savoir quelle variable est utilisée pour une donnée particulière. Le BASIC a la solution avec les arrangements de données, un arrangement étant un ensemble de données, à une ou plusieurs dimensions.

Les mots tableaux, tables ou matrices sont aussi utilisés.

Pour mieux comprendre prenons l'exemple du jeu de 52 cartes. Supposons un programme de simulation d'un jeu de cartes, il faut se rappeler des cartes distribuées et des cartes encore dans le paquet: on pourrait avoir 52 variables et indiquer par un 1 et un 0 si la carte a été distribuée ou pas.

Il faudrait évidemment avoir 52 variables et se rappeler quelle variable pour quelle carte, c'est là qu'un arrangement est utile.

On donne un nom à cet arrangement d'abord, par exemple PAQUET; si on décide que les coeurs sont les premières cartes du paquet, alors le six de coeur est représenté par PAQUET (6) le 10 par PAQUET (10) et le Roi par PAQUET (13).

Mais il faut prévenir l'ordinateur de la dimension de l'arrangement, pour qu'il puisse réserver de la place dans la mémoire pour ses calculs; on lui dit.

```
DIM PAQUET (52)
```

Cela indique à l'ordinateur de se préparer à avoir 52 variables (53 en réalité car le 0 peut aussi être utilisé).

On peut maintenant faire l'esquisse d'un sous-programme de distribution de cartes:

```
10 DIM PAQUET(52)
20 FOR X=1 TO 52
30 LET PAQUET(X)=0
40 NEXT X
*****
1000 CARTE=RND(52)+1
1010 IF PAQUET(CARTE)=1 THEN GOTO 1000
1020 PAQUET(CARTE)=1
1030 RETURN
```

La commande **DIM**ension est la première du programme. Chaque arrangement doit être dimensionné une fois seulement, au début du programme.

Les lignes de 20 à 40 donnent la valeur zéro à chaque carte (0 = non distribuée). Le sous-programme qui commence à la ligne 1000 choisit une carte au hasard et vérifie si la carte a déjà été distribuée: si elle l'est déjà, il en cherche une autre jusqu'à ce qu'il en trouve une non distribuée. Le programme change alors l'indicateur de la carte en un 1 pour qu'on se rappelle après qu'elle a été distribuée.

(La fonction **RND(52)+1** est la fonction du hasard! Voir chapitre 8 pour plus de détail).

Les arrangements peuvent avoir plusieurs dimensions: pour le jeu de cartes, on aurait pu faire.

**DIM PAQUET (4,13)**

Ce qui aurait arrangé en quatre couleurs de treize cartes, bien utile pour jouer au bridge - si on veut retrouver le 7 de trèfle, il sera l'élément (4,7) si on prend l'ordre PIQUE, COEUR, CARreau, TREFLE, le trèfle étant la quatrième couleur (et le sans-atout, alors?)

Les arrangements ne doivent pas obligatoirement avoir des nombres, il peut y avoir des chaînes de lettres; par exemple un arrangement sera utilisé pour enregistrer les noms des gens qui ont loué une place de théâtre ou d'avion.

## 4.12 DATA (=donnée)

Cette commande fait la paire avec la commande **READ** (=lire). Les données sont rassemblées sur une ligne, chaque donnée étant séparée par une virgule et la liste précédée par la commande **DATA**. Les données peuvent être lues dans l'ordre par la commande **READ**.

Programme pour l'exemple:

```
10 READ X,Y,Z
20 PRINT X;"+";Y;"+";Z;" = ";X+Y+Z
30 DATA 1,3,5
```

Les données peuvent être numériques, algébriques, alphabétiques; Si les données ne tiennent pas dans une ligne, il suffit d'ajouter une nouvelle ligne commençant par **DATA**. Quand l'ordinateur rencontre un **READ**, il cherche dans le programme où se trouve les données suivantes. Attention à avoir assez de données pour chaque **READ**, ou une erreur se produira (erreur no. 4 Data exhausted, Appendice III).

La seule manière d'interrompre une entrée de données dans un certain ordre est la commande RESTORE qui remet le 'pointeur' de données à la première donnée pour qu'on puisse lire (= READ) plusieurs fois la même donnée. Le programme suivant le montre:

```
10 FOR C=1 TO X
20 READ X$
30 PRINT X$;"";
40 NEXT C
50 RESTORE
60 GOTO 10
70 DATA Hello,comment,allez,vous, aujourd'hui
```

Deux fois [ESC] pour sortir de ce programme.

Bien que les lignes DATA soient à la fin, elles peuvent être placées n'importe où dans le programme.

La commande DATA n'est pas seulement pour des informations qui doivent être affichées. On peut aussi l'utiliser avec des sons:

```
10 FOR n=1 TO 30
20 READ s
30 SOUND 1,s,40,5
40 NEXT n
50 DATA 100,90,100,110,120,110,100,0
60 DATA 130,120,110,0,120,110,100,0
70 DATA 100,0,100,110,120,110,100,0
80 DATA 130,0,100,0,120,150
```

Si vous n'entendez rien, ajustez le volume sur le côté droit de l'ordinateur.

Pour conclure cette brève introduction au BASIC, voilà un programme pour jouer au Vingt et un avec (ou contre) le CPC464. Il utilise beaucoup de caractéristiques du BASIC et peut se comprendre grâce aux noms connus des variables. On peut y ajouter des graphiques, ajouter des bruits et effets sonores, en un mot développer ce programme à la manière des meilleurs programmes en BASIC qui viennent d'une simple idée.

L'objet du jeu est d'arriver à un total le plus proche de 21 en ajoutant les valeurs des cartes dans votre main, et la banque (le CPC464) essaiera de faire aussi bien ou mieux sans évidemment dépasser 21 et faire bang. Après la ligne 1, faites la commande AUTO en mode direct, les numéros de ligne seront automatiquement affichés.

```

1 REM VINGT ET UN
10 REM INITIALISE
20 YC=2:CC=2
30 ACES=0
40 CACES=0
50 S=0
60 T=0
70 DIM SUITS$(4)
80 SUITS$(1)="TREFLES"
90 SUITS$(2)="COEURS"
100 SUITS$(3)="PIQUES"
110 SUITS$(4)="CARREAUX"
120 CLS
130 DIM PAQUET (52)
140 FOR X=1 TO 52
150 PAQUET (X)=0
160 NEXT X
170 REM DONNE DEUX CARTES A CHAQUE JOUEUR
180 LOCATE 10,3
190 PRINT "VOUS";SPC(15)"LA BANQUE"
200 LOCATE 3,5
210 GOSUB 740
220 S=S+F
230 IF F=11 THEN ACES=ACES+1
240 LOCATE 3,6
250 GOSUB 740
260 S=S+F
270 IF F=11 THEN ACES=ACES+1
280 LOCATE 24,5
290 GOSUB 740
300 T=T+F
310 IF F=11 THEN CACES=CACES+1
320 LOCATE 24,6
330 GOSUB 740
340 T=T+F
350 IF F=11 THEN CACES=CACES+1
360 REM INPUT OPTION-TWIST (T) OR STICK(S)
370 X$=INKEY$:IF X$<>"S" AND X$<>"T" THEN 370
380 IF X$="S" THEN 560
390 LOCATE 3,YC+5
400 YC=YC+1
410 GOSUB 740
420 S=S+F
430 IF F=11 THEN ACES=ACES+1
440 REM VERIFIÉ LE SCORE AND LES AS
450 IF S<22 THEN 370
460 IF ACES=0 THEN 500
470 ACES=ACES-1
480 S=S-10
490 GOTO 450

```

```

500 LOCATE 12,19
510 PRINT "VOUS AVEZ PERDU"
520 PRINT: PRINT:PRINT"UN AUTRE JEU (Y/N)"
530 X$=INKEY$:IF X$<>"Y" AND X$<>"N" THEN 530
540 IF X$="Y" THEN RUN
550 END
560 IF T>16 THEN GOTO 700
570 CC=CC+1
580 LOCATE 24,CC+4
590 GOSUB 740
600 T=T+F
610 IF F=11 THEN CACES=CACES-1
620 IF T<21 THEN 560
630 IF CACES=0 THEN 670
640 CACES=CACES-1
650 T=T-10
660 GOTO 620
670 LOCATE 12,19
680 PRINT "VOUS GAGNEZ"
690 GOTO 520
700 LOCATE 12,19
710 IF T<S THEN 680
720 PRINT "LA BANQUE GAGNE"
730 GOTO 520
740 REM DONNE UNE CARTE
750 LET CARTE=INT (RND(1)*52+1)
760 IF PAQUET(CARTE)=1 THEN GOTO 750
770 PAQUET(CARTE)=1
780 F=CARTE-13*INT(CARTE/13)
790 IF F=0 THEN F=13
800 IF F=1 OR F>10 THEN GOTO 850
810 PRINT F;" de ";
820 IF F>10 THEN F=10
830 PRINT SUIT$(INT((CARTE-1)/13)+1)
840 RETURN
850 IF F=11 THEN PRINT "VALET DE ";
860 IF F=12 THEN PRINT "DAME DE ";
870 IF F=13 THEN PRINT "ROI DE ";
880 IF F<>1 THEN GOTO 820
890 F=11
900 PRINT "AS DE ";
910 GOTO 830

```

Mettez T pour votre carte en plus de celles déjà distribuées ou S pour passer et jouer contre la Banque. Ce n'est pas le jeu le plus compliqué pour votre CPC464, mais c'est un début auquel vous pouvez ajouter des graphiques et des effets sonores.

## 4.13 Expressions logiques

Une des plus grandes différences entre un calculateur et un ordinateur est la capacité de l'ordinateur à manipuler les opérations logiques dans la séquence IF...THEN (si...alors) par exemple.

Pour ce faire, les opérateurs logiques traitent les valeurs auxquelles ils sont appliqués comme des ensembles de bits et opèrent sur les bits individuels... la description et son utilisation est entièrement... logique, mais il est très difficile de décrire la logique en termes simples.

Disons qu'une expression logique comprend deux arguments et un opérateur logique.

Les opérateurs logiques sont, avec leur effet sur chaque argument:

**AND** (= et) résultat 0 sauf si les deux arguments sont vrais (bits indicateurs tous les deux 1)

**OR** (= ou inclusif) résultat 1 sauf si les deux arguments sont faux (bits indicateurs tous les deux 0)

**XOR** (= ou exclusif) résultat 1 sauf si les deux arguments sont identiques (bits indicateurs identiques).

AND est le plus courant et ne veut pas dire plus ("")

PRINT 10 and 10 résultat 10!

mais

PRINT 10 AND 12 résultat 8!

et

PRINT 10 AND 1000 résultat 8 aussi!

Drôles de mathématiques, mais rappelez vous que les opérateurs logiques ne connaissent que le système binaire 0 et 1, vrai ou faux

Explication:

10 en binaire	1010
1000 en binaire	1111101000

L'opérateur AND vérifie les 1 qui sont dans les deux nombres à la même position et trouve

0000001000

qui en système décimal est égal à 8

L'opérateur logique AND (= et) vérifie si deux conditions sont vraies en même temps. Petit programme explicatif:

```
10 CLS:INPUT "numero du jour";jour
20 INPUT "numero du mois";mois
30 IF jour=25 AND mois=12 GOTO 50
40 GOTO
50 PRINT "Joyeux Noel!"
```

Passons à l'exemple avec OR ou bien inclusif:

```
10 CLS:  
20 INPUT "NUMERO DU MOIS";MOIS  
30 IF MOIS=12 OR MOIS=10 OR MOIS=2 GOTO 50  
40 CLS:GOTO 10  
50 PRINT "CE DOIT ETRE L'HIVER"
```

Ou encore

```
10 CLS  
20 INPUT "numero du mois";mois  
30 IF NOT (mois=6 OR mois=7 OR mois=8) goto 50  
40 CLS:GOTO 10  
50 PRINT "cen'est pas l'ETE!"
```

On peut aussi mélanger les opérateurs logiques (il vaut mieux alors ne pas se mélangler les pinceaux) et cela donne:

```
10 CLS:INPUT"numero du jour";jour  
20 INPUT "numero du mois";mois  
30 IF NOT (mois=12 OR mois=1) AND jour=29 GOTO 50  
40 CLS:GOTO 10  
50 Print" Cen'est ni Decembre ni janvier  
mais c'est peut être une année bissextile."
```

Ajoutons deux lignes pour compliquer

```
60 PRINT NOT (mois=12 OR mois=1)  
70 PRINT (mois=12 OR mois=1)
```

Finalement, XOR (ou bien eXclusif) donne un résultat vrai si les deux arguments sont différents.

Résumons par un tableau qui donne certainement une idée plus claire de la situation.

ARGUMENT A ARGUMENT B	1(vrai) 0(faux)	0(faux) 1(vrai)	1(vrai) 1(vrai)	0(faux) 0(faux)
Résultat AND(et)	0(faux)	0(faux)	1(vrai)	0(faux)
Résultat OR(ou)	1(vrai)	1(vrai)	1(vrai)	0(faux)
Résultat XOR (ou exclusif)	1(vrai)	1(vrai)	0(faux)	0(faux)



# 5 Les Eléments Graphiques

Comment s'y retrouver dans les couleurs et les graphiques du CPC464

Sujets abordés dans ce chapitre

- \* **les modes d'affichage et les pixels**
- \* **les couleurs**
- \* **INK (encre), PAPER (papier) et PEN (stylo)**
- \* **Dessins de lignes**
- \* **Les FENETRES**

## 5.1 Des graphiques différents

Jusqu'à maintenant, la description et l'utilisation du BASIC de l'AMSTRAD sont proches de celles rencontrées dans d'autres BASIC. Les programmes faisant appel à l'arithmétique marcheront avec de légères modifications. Mais dans le domaine des graphiques (et des positions du curseur de textes) l'AMSTRAD innove et les commandes du BASIC se rapportent plus à l'organisation particulière du CPC464 dans son contrôle de l'affichage et doivent être bien comprises pour tirer profit des avantages du CPC464.

Comme d'habitude les mots-clés réservés sont montrés avec les caractères d'imprimerie différents, et les précisions sur ces mots sont regroupées dans le chapitre 8.

### 5.1.1 Sélection des Couleurs

Le noir, qui est pour l'ordinateur l'absence de couleur est considéré comme une couleur dans les descriptions suivantes. BORDER (=e cadre) peut avoir n'importe quelle couleur, indépendamment du mode, et varier entre deux couleurs (pas trop longtemps, cela fatigue les yeux.)

Le nombre d'encre (INK) qu'on peut avoir en même temps sur l'écran dépend du MODE d'affichage choisi.

Chaque encre (INK) peut avoir deux couleurs en même temps, c'est-à-dire de type flash, ou une couleur fixe. Le nombre d'encre utilisables à chaque moment dépend du mode défini auparavant. Le papier (PAPER pour le texte), les stylos (PEN pour le texte, PEN pour les graphiques) peuvent prendre la couleur d'une encre disponible.

### 5.1.2 L'option de transparence et la relation entre PEN, INK et PAPER.

A l'exception des couleurs alternées pour donner un effet de flash, deux encres sont utilisées pour afficher sur l'écran: une encre (INK) détermine la couleur du PEN (stylo), une autre encre détermine la couleur du papier PAPER.

NB. Attention, ne mélangez pas vos pinceaux: le numéro associé à PAPIER ou à PEN n'est pas le numéro de la couleur indiqué à l'Appendice IV, mais la couleur de l'encre définie par ce numéro. Je m'explique:

Mode 1

INK 0,9 (encre 0, couleur verte)  
INK 1,6 (encre 1, couleur rouge vif)  
INK 2,24 (encre 2, couleur jaune vif)  
INK 3,10 (encre 3, couleur turquoise)

maintenant si vous écrivez

PAPER 1  
PEN 3

Le papier prend la couleur rouge vif, (couleur No. 6) le stylo prend la couleur turquoise (couleur No. 10), car on déclare en écrivant PAPER 1 que la couleur du papier est celle définie avec la commande INK 1,6.

Si on ne précise pas la couleur du papier et du stylo (en réalité, on devrait dire la couleur de l'encre du papier, la couleur de l'encre du stylo), le papier prend automatiquement la couleur de l'encre 0 et le stylo la couleur de l'encre 1.

Ainsi, si vous écrivez

ink 0,9  
ink 1,0

le PAPER sera vert (= 9) et le PEN sera noir (= 0) sans avoir à le préciser.

Si vous donnez la même valeur à PAPER et à PEN, vous ne voyez plus rien: essayez donc d'écrire avec un stylo noir sur du papier noir!

L'écriture peut être définie comme transparente, ou opaque en se servant de caractères de contrôle. Avec les caractères transparents, on peut ignorer la couleur du papier et écrire sur des graphiques ou écrire par dessus le fond.

Brève rencontre avec un programme qui explique mieux qu'un long discours, (n'oubliez pas [CTRL][SHIFT] et [ESC] ou bien je ne réponds pas des résultats)

```
[CTRL][SHIFT][ESC]  
10 MODE 1  
20 INK 2,19  
30 DRAW 200,200,2  
40 LOCATE 1,21  
50 PRINT "1 NORMAL"  
60 PRINT CHR$(22)+CHR$(1)  
70 ORIGIN 0,0  
80 DRAW 500,200,2  
90 LOCATE 12,18  
100 PRINT"2 TRANSPARENT"  
110 PRINT CHR$(22)+CHR$(0)  
120 LOCATE 22,15  
130 PRINT"3 NORMAL de NOUVEAU"
```

La commande **DRAW** de la ligne 30 est effectuée avant qu'on ait défini le mode transparent, mais la deuxième commande **DRAW** de la ligne 80 est après la ligne 60 qui établit le mode transparent, **CHR\$(22)+CHR\$(1)**. Observez et notez les différences.

Echangez les lignes 60 et 110 qui établissent et suppriment respectivement la transparence et voyez ce qui se passe...

## 5.2 MODES ÉCRAN

Il y a trois modes d'affichage sur l'écran (textes et graphiques).

a) Normal

**MODE 1:** 40 colonnes x 25 lignes, 4 INK (encre)  
320 x 200 pixels, chacun pouvant avoir une des 4 couleurs.

b) Multicolore

**MODE 0:** 20 colonnes x 25 lignes, 16 INK différentes  
160 x 200 pixels, chacune pouvant avoir une des 16 couleurs.

c) Haute Résolution

**MODE 2:** 80 colonnes x 25 lignes, 2 INK,  
640 x 200 pixels, deux couleurs au choix.

La différence vient des pixels, non pas des petits animaux rares, mais des éléments de l'affichage: disons pour simplifier que ce sont des points ayant des coordonnées qui forment ce qu'on appelle l'adresse du point.

Le changement de **MODE** efface l'écran complètement (même effet que les commandes **CLS** et **CLG**) mais ne touche pas au contenu de la mémoire.

### 5.2.1 MODE 0, haut en couleurs.

16 couleurs choisies dans la palette de 27 peuvent apparaître simultanément à l'écran. L'affichage comprend 160 pixels par ligne horizontale, 200 par ligne verticale. Il y a 20 caractères sur chacune des 25 lignes.

### 5.2.2 MODE 1 est le mode standard

Le Mode 1 est le mode d'origine quand on branche l'ordinateur ou quand on remet à zéro. 4 couleurs peuvent être choisies parmi les 27, mais on peut passer par les 27 couleurs avec des changements astucieux, (il n'y aura que 4 couleurs visibles au même instant, mais vous pouvez avoir les 27 couleurs dans l'espace de 10 secondes). 320 pixels de largeur, 200 pixels en hauteur, 40 caractères de texte sur 25 lignes.

Appendice VI contient des grilles pour mieux présenter vos affichages, que ce soit en Mode 1,0 ou 2.

### 5.2.3 Mode 2 est le mode haute résolution

2 couleurs en mode 2, utilisable surtout pour les traitements de textes avec ses 80 colonnes par ligne, qui permet d'écrire plus facilement les programmes puisqu'on en voit plus sur l'écran. (Absolument nécessaire dès qu'on veut l'utiliser au bureau, gestion de tableaux, lettres, factures, etc). En Mode 2 on a 640 pixels par rang horizontal et 20 verticalement.

### 5.2.4 Transformez l'essai suivant:

après [CTRL] [SHIFT] et [ESC]

```
5 REM EXEMPLE de DEMONSTRATION GRAPHIQUE
10 MODE 1
15 INK 2,0
16 INK 3,6:REM DEFINIT LA COULEUR POUR
    LA LIGNE 90
17 BORDER 1:REM BLEU FONCE
20 CLG:REM BLEU FONCE
30 b%=RND*5+1:REM DONNE DES ENTIERS AU HASARD
40 c%=RND*5+1
50 ORIGIN 320,200:REM FIXE L'ORIGINE GRAPHIQUE
60 FOR a=0 TO 1000 STEP PI/30
70 x%=1000*COS(a)
80 MOVE x%,x%:REM DEPLACE LE CURSEUR GRAPHIQUE
90 DRAW 200*COS(a/b%),200*SIN(a/c%),3
    :REM DESSINE LA LIGNE
91 IF INKEY$<>"" THEN 20
100 NEXT :REM REVIENT A LA LIGNE 60 SAUF EN
    CAS D'ARRET LIGNE 91
110 GOTO 20
```

Faites RUN, puis appuyez une touche pour avoir un autre motif de dessin. Il démontre plusieurs caractéristiques importantes de la machine et de son logiciel intégré. Le CPC464 'écrit' sur l'écran d'une manière douce, sans à-coups et nous disposons de commandes qui permettent des effets sophistiqués avec le minimum d'effort.

Les REMarques sont là à titre de remarques! (bizarre, bizarre! Vous avez dit bizarre?), on n'en a pas besoin pour l'exécution du programme mais c'est utile pour ceux qui n'ont pas écrit le programme ou veulent se rappeler six mois après.

Vous pouvez constater que les numéros de lignes indiquent des éléments rajoutés après coup. Bien qu'il soit possible de mettre de l'ordre avec la commande RENUM, cela vous aide à suivre l'évolution d'un programme si nous laissons la numérotation telle quelle. Sauvegardons sur cassette:

```
SAVE "GRAPHIQUES 5.SEPTE.84"
```

Le programme suivant donne des motifs différents.

```
new
5 REM DESSINS ET FORMES
10 a$=INKEY$:REM PRESSEZ UNE TOUCHE
    QUELCONQUE POUR AVOIR UN AUTRE DESSIN
20 IF a$="" THEN 10
30 CLS
40 m=INT(RND*3):REM CHOIX D'UN NOMBRE
    ENTRE 0 ET 3
50 IF m>2 THEN 40:REM ESSAI SI LA VALEUR
    EST PLUS QUE 2
60 MODE m
70 i1=RND*26:REM CHOIX D'ENCRÈS
80 i2=RND*26
90 IF ABS(i1-i2)<5 THEN 70
100 INK 0, i1:INK 1,i2
110 s=RND*5+3
120 ORIGIN 320,-100
130 FOR x=-1000 TO 0 STEP s
140 MOVE 0,0
150 DRAW x,300:DRAW 0,600
160 MOVE 0,0
170 DRAW -x,300: DRAW 0,600
180 a$=INKEY$
190 IF a$<>"" THEN 30:REM ARRETE LA BOUCLE SI
    UNE TOUCHE EST TAPEE
200 NEXT x
210 GOTO 10
```

Les deux programmes précédents illustrent de simples concepts mathématiques d'une manière colorée et visuelle. Les deux font des additions de nombres choisis au hasard, (random = RND = au hasard) mais avec un peu de direction (comme les têtes de série au tennis).

Voyons maintenant un serpent se dessiner, appelé une sinusoïde en mathématiques.

```
10 REM DESSIN DE SINUSOÏDE
20 MODE 2
30 INK 1,21
40 INK 0,0
50 CLS
60 DEG
70 ORIGIN 0,200
80 FOR n=0 TO 720
90 y=SIN(n)
100 PLOT n*640/720,198*y,1
110 NEXT
```

L'affirmation **PLOT** de la ligne 100 est ce qui dessine la courbe. Elle produit un pixel (point) pour chaque calcul de la boucle **FOR...NEXT** des lignes 80 et 110 - et le résultat est sur l'écran.

Le CPC64 a beaucoup de commandes simples et performantes, on peut y ajouter de l'effet avec

**15 BORDER 6,9**

qui donne une bordure de couleur alternée (6 = rouge, 9 = vert) pour que le programme soit continu, faisons une boucle avec:

**120 GOTO 150**

Les couleurs 'flash' de la bordure ont continué même quand le programme est arrêté, car la commande **BORDER** est indépendante du reste. Pour revenir à une image plus stable, faisons **[ESC]** deux fois puis:

**15 BORDER 2**

exécutez le programme (**RUN**) et plus de flash couleur. Pour changer les couleurs du fond et de la courbe, il faut modifier la valeur de **INK** dans les lignes 30 et 40. Après avoir demandé la **LIS Te**, cela doit donner:

```
10 REM sinusoide
15 BORDER 2
20 MODE 2
30 INK 1,2
40 INK 0,20
50 CLS
60 DEG
70 ORIGIN 0,200
80 FOR n=0 TO 720
90 y=SIN(.)
100 PLOT N*640/720,195*y,1
110 NEXT
120 GOTO 50
```

Le numéro 1, à la fin de la commande **PLOT** à la ligne 100, dit à l'ordinateur de dessiner la courbe avec la couleur précisée par la commande **INK 1,2**. Reportez vous à la définition de **PLOT** dans la liste des mots clés du chapitre 8. En regardant la courbe attentivement, vous verrez que ce n'est pas une ligne continue, mais qu'elle est divisée en petits segments. Les plus petits de ces segments sont des exemples de pixel décrits plus haut.

### 5.2.5 Le curseur graphique et le dessin des lignes.

Vous avez maintenant essayé plusieurs moyens de transformer un programme pour des effets graphiques, et nous avons introduit plusieurs commandes et concepts graphiques. Quand on dessine des lignes sur l'écran, il y a des règles à respecter pour éviter la confusion.

D'abord, quand on utilise la commande **NEW**, les éléments graphiques restent en mémoire. Le seul moyen de revenir au départ est de faire **[CTRL] [SHIT]** et **[ESC]** mais souvenez vous qu'il faut sauvegarder (**SAVE**) vos programmes avant, si vous voulez le conserver.

Vous pouvez le vérifier en tapant:

**NEW: CLS**

puis dessinons:

**DRAW 200,100**

L'instruction **DRAW** dessine une ligne droite en partant de la dernière place du curseur graphique pour aller au point de coordonnées (x,y) qui est (200,100) dans notre cas.

Le curseur graphique est un concept invisible qui indique de quel endroit l'opération graphique suivante va partir. Pour trouver où il est, vous utilisez les fonctions **XPOS** et **YPOS** qui vous disent où se trouve le curseur.

**PRINT XPOS**

réponse

**200**

**PRINT YPOS**

réponse

**100**

Notons que si les graphiques montent sur l'écran (quand on tape du texte ou presse **[ENTER]**), le curseur graphique reste à la même position. Essayez en appuyant sur la flèche [down arrow] plusieurs fois et faites **PRINT XPOS** et **PRINT YPOS**.

Pour indiquer une couleur de ligne à dessiner, on met un numéro après la commande, et il faut donc avoir précisé **INK** auparavant. EXEMPLE:

```
10 MODE 1
20 INK 0,10
30 ORIGIN 0,0
40 INK 1,26
50 INK 2,0
60 DRAW 320,400,1
70 DRAW 640,0,2
```

Voyons ensuite un exemple récapitulatif qui utilise tous les éléments mentionnés jusqu'à présent et introduit quelques concepts nouveaux: la ligne 10 établit les couleurs pour être sûr que ce qu'il y avait avant a été effacé et que les couleurs d'encre (**INK**) sont bien celles que nous voulons:

```

10 INK 0,0:INK 1,26:INK 2,6:INK 3,18: BORDER 0
20 REM ce programme dessine des formes
30 mode 1:DEG
40 PRINT "3,4 ou 6 COTES ?";
50 LINE INPUT p$
60 IF p$="3" THEN sa=120:GOTO 100
70 IF p$="4" THEN sa=135:GOTO 100
80 IF p$="6" THEN sa=150:GOTO 100
90 GOTO 50
100 PRINT:PRINT "Je Calcule ";
105 IF p$="3" THEN ORIGIN 0,-50,0,640,0,400 ELSE
    ORIGIN 0,0,0,640,0,400
110 DIM cx(5),cy(5),r(5),lc(5)
120 DIM np(5)
130 DIM px%(5,81),py%(5,81)
140 st=1
150 cx(1)=320:cy(1)=200:r(1)=80
160 FOR st=1 TO 4
170 r(st+1)=r(st)/2
180 NEXT st
190 FOR st=1 TO 5
200 lc(st)=0:np(st)=0
210 np(st)=np(st)+1
220 px%(st,np(st))=r(st)*SIN(lc(st))
230 py%(st,np(st))=r(st)*COS(lc(st))
240 lc(st)=lc(st)+360/r(st)
245 IF (lc(st) MOD 60)=0 THEN PRINT "patience ";
250 IF lc(st) < 360 THEN 210
252 px%(st,np(st)+1)=px%(st,1)
254 py%(st,np(st)+1)=py%(st,1)
260 NEXT st
265 CLS:ink 1,2
270 st=1
280 GOSUB 340
290 LOCATE 1,1
300 EVERY 25,1 GOSUB 510
310 EVERY 15,2 GOSUB 550
320 EVERY 5,3 GOSUB 590
330 GOTO 330
340 REM dessine un cercle plus 3,4 or 6 autour
350 cx%=cx(st):cy%=cy(st):lc(st)=0
360 FOR x%=1 TO np(st)
370 MOVE cx%,cy%
380 DRAW cx%+px%(st,x%),cy%+py%(st,x%),1+(st
    MOD 3)
390 DRAW cx%+px%(st,x%+1),cy%+py%(st,x%+1),1+(st
    MOD 3)
400 NEXT x%
410 IF st=5 THEN RETURN
420 lc(st)=0
430 cx(st+1)=cx(st)+1.5*r(st)*SIN(sa+lc(st))

```

```

440 cy(st+1)=cy(st)+1.5*r(st)*COS(sa+lc(st))
450 st=st+1
460 GOSUB 340
470 st=st-1
480 lc(st)=lc(st)+2*sa
490 IF (lc(st) MOD 360)<>0 THEN 430
500 RETURN
510 ik(1)=1+RND*25
520 IF ik(1)=ik(2) OR ik(1)=ik(3) THEN 510
530 INK 1,ik(1)
540 RETURN
550 ik(2)=1+RND*25
560 IF ik(2)=ik(1) OR ik(2)=ik(3) THEN 550
570 INK 2,ik(2)
580 RETURN
590 ik(3)=1+RND*25
600 IF ik(3)=ik(1) OR ik(3)=ik(2) THEN 590
610 INK 3,ik(3)
620 RETURN

```

A l'exécution de ce programme, la question de la ligne 40 apparaît, répondez 3 si vous voulez quelque chose de rapide. Le programme va alors répondre "Je calcule", puis de temps en temps "patience!" pour confirmer qu'il calcule toujours.

Les sous-programmes des lignes 300,320 alternent les couleurs à la vitesse déterminée par la commande **EVERY**. Si vous voulez que le changement soit moins rapide, modifiez les lignes 300 à 320 (en utilisant **EDIT**).

```

300 EVERY 250,1 GOSUB 510
310 EVERY 150,2 GOSUB 550
320 EVERY 50,3 GOSUB 590

```

Pour voir le procédé, regardez la commande **EVERY** au chapitre 8 (et au chapitre 10 si vous voulez vraiment tout savoir) car c'est une commande les plus utiles du BASIC de l'AMSTRAD.

Un des effets intéressant est la manière dont la commande **EVERY** continue à travailler quand on presse **[ESC]** une fois (et une fois seulement): appuyez sur **[ESC]**, attendez quelques secondes puis appuyez sur la barre d'espace. L'affichage va s'activer pour rattraper les instructions qui ont été mises en attente et après cela, cela redévient normal.

### 5.3 WINDOWS (= 'fenêtres')

L'utilisateur peut sélectionner huit fenêtres de texte pour écrire des caractères et une fenêtre pour les graphiques. Les fenêtres sont remises à l'origine quand le mode est changé.

NB le déroulement du texte sur l'écran est plus lent quand on a une fenêtre plus petite que l'écran car le déroulement est fait par le logiciel et pas par la machine.

La commande **WINDOW** précise les coins de la fenêtre dans l'ordre suivant gauche, droite, haut, bas en utilisant les coordonnées de texte du mode concerné (20 par 25 en mode 0, 40 par 25 en mode 1, 80 par 25 en mode 2). Les fenêtres peuvent se superposer. Mais avant, tapez:

```
KEY 139,"mode 2:Paper 0:Ink 1,0:  
ink 0,9:list"+chr$(13)
```

Cela définit la petite touche **[ENTER]** comme touche de récupération lorsqu'on s'est emmêlé les pinceaux dans ses couleurs et qu'on ne voit plus rien sur l'écran.

Le programme suivant dessine quelques fenêtres en couleurs sur l'écran, et illustre deux éléments principaux.

```
5 MODE 0  
10 FOR n=0 TO 7  
20 WINDOW #n,n+1,n+6,n+1,n+6  
30 PAPER #n,n+4  
40 CLS #n  
50 FOR c=1 TO 200:NEXT  
60 NEXT
```

Le premier élément est que chaque nouvel écran écrit par dessus l'ancien et que les messages apparaissent sur le canal 0, à moins qu'on ne lui donne une commande spéciale. Faites:

LIST

et le programme s'affiche normalement. Puis

LIST #5

et le programme va dans la fenêtre 5.

Puis

CLS #6

Montrant que le dernier écran défini écrit par dessus le reste, et que le message du système Ready apparaît toujours sur canal 0.

Pour utiliser la commande **WINDOW SWAP**, ajoutez la ligne 55

```
55 IF n=3 THEN WINDOW SWAP 7,0
```

Vous avez peut-être votre petite idée sur ce qui va se passer. Exécutez (RUN) et voyez donc si vos suppositions étaient correctes.

# 6 Le Son: la théorie

Les effets sonores viennent d'un haut-parleur dans l'ordinateur lui-même. Si vous utilisez l'adaptateur Péritel MP1 et un poste de télé, mettez le réglage du volume au minimum.

Le niveau sonore est contrôlé par le bouton **VOLUME** sur le côté droit de l'ordinateur. Le son peut aussi aller alimenter votre système stéréo, en branchant la prise de sortie (**I/O**) sur le panneau arrière de l'ordinateur.

Vous pourrez ainsi écouter le son en stéréo, avec vos haut-parleurs ou votre casque d'écoute individuel.

*Le CPC464 est aussi performant dans le domaine sonore que dans les autres domaines. Pour obtenir les meilleurs résultats, il faut comprendre comment fonctionne le système des structures qui concernent le temps (appelé horloge, ou chronomètre interne)*



Sujets abordés dans ce chapitre

- \* période de ton (la note, pour simplifier)
- \* commandes sonores
- \* enveloppes
- \* les queues et la synchronisation

Le bruit le plus élémentaire, le 'bip' est produit en faisant:

```
PRINT CHR$(7)
```

Vous pourriez arrêter là... mais ce serait manquer un des aspects les plus intéressants du CPC464.

Ce chapitre est certainement un peu difficile d'accès, mais il est indispensable si vous voulez utiliser les capacités de votre micro dans le domaine sonore. Vous pourrez alors jouer des notes sur des instruments différents et utiliser votre CPC464 comme un chef d'orchestre.

## 6.1 La structure fondamentale du son (SOUND)

Quand vous entendez le son produit par la note d'un air de musique, il y a plusieurs éléments à considérer.

- 1) Hauteur et variations de hauteur pendant la durée de la note.
- 2) Le volume et ses variations
- 3) La longueur de la note.

## 6.2 Hauteur

Dans une note musicale, la hauteur est le facteur essentiel. Une note musicale peut être décrite comme une oscillation et toutes les oscillations ont une fréquence, une période et une amplitude. (Voir Fig. 1)

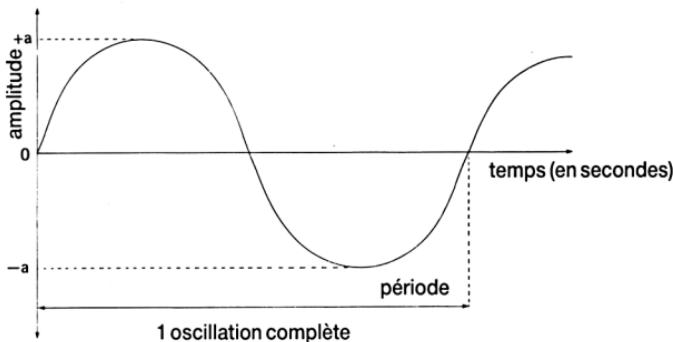


Figure 1: les caractéristiques d'une note musicale.

La fréquence est le nombre d'oscillations par seconde, et la période est la longueur en temps de chaque oscillation. L'amplitude dépend du volume et n'a pas d'importance dans la définition de la hauteur d'une note.

La fréquence et la période sont liées par la formule suivante:

Fréquence (exprimée en Hertz) = 1/période (exprimée en secondes)

La relation entre la fréquence et la 'période de ton' de la commande SOUND est

$$\text{Fréquence (en Hertz} = \text{Hz}) = 125000 : \text{période de ton.}$$

Une ·période de ton· de 1000 donne une note de 125Hz et une ·période de ton· de 125 donne une note de 1000Hz (1kHz).

Le BASIC Amstrad a choisi la ·période de ton· pour définir la hauteur. Quand la période augmente, la hauteur diminue. La période peut varier de 0 à 4095 et doit être donnée en valeur entière, ce qui donne parfois une petite erreur relative, qui n'est pratiquement pas audible. (voir appendice VII).

Quand une note est jouée sur un instrument véritable, la hauteur peut varier, quelquefois après (vibrato). En utilisant la commande ENT (ENveloppe de Ton), on peut concevoir une variation de la période pendant la note, que l'on utilise dans la commande SOUND.

## 6.3 Volume

C'est la mesure de la force de la note, le volume initial étant défini par un nombre entier compris entre 0 et 15.

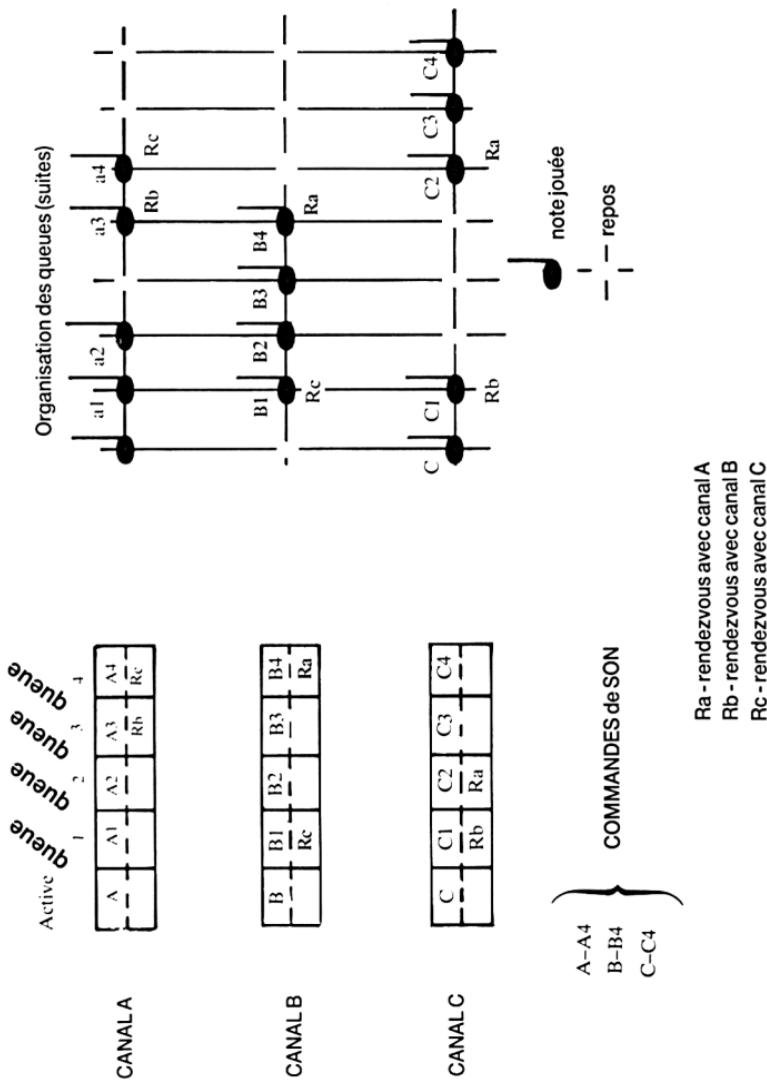
La valeur est établie dans la commande SOUND.

Une note simple n'a pas d'intérêt tant que la variation de VOLUME n'est pas définie, ce qu'on appelle en termes musicaux l'attaque et la chute de la note et chaque instrument de musique tire une partie de ses caractéristiques par la nature de ces deux points: attaque et chute de la note, et c'est ce qu'on définit sur l'ordinateur avec la commande ENV (ENveloppe de Volume).

## 6.4 Longueur de la note

Définie dans la commande SOUND comme la durée exprimée en unités de un centième de seconde, et peut être aussi déterminée avec la durée de l'enveloppe de volume.

# Rendezvous pour les canaux sonores



## **6.5 Autres sons (SOUNDs)**

Des bruits qui n'ont rien de musical peuvent être produits par le CPC464, comme fond sonore des airs musicaux ou comme des effets spéciaux, explosions ou bruits extra-terrestres(!!?).

## **6.6 Sons multiples et canaux**

La plupart des airs de musique sont écrits avec la clef de sol et la clé de fa. Pour tenir compte de cette approche, il y a trois canaux sonores, appelés A, B et C. Ils peuvent jouer indépendamment, ou coïncider. La sélection des canaux est un des paramètres de la commande son, le premier dénommé ·canal·.

## **6.7 Suites de sons (queues)**

Chaque canal a une suite de SOUNDs (sons) à jouer, avec de la place pour cinq commandes SOUND séparées dans cette suite. L'ordinateur continue à travailler pendant qu'il joue des notes, ne retournant que pour aller chercher d'autres commandes SOUND.

## **6.8 Etat d'un canal**

Le mot-clé SQ est utilisé pour déterminer l'état d'un canal que vous voulez interroger et donne des informations sur les places libres dans les queues, les 'rendezvous' et les attentes. ON SQ GOSUB est une commande d'interruption pour rappeler à l'ordinateur de revenir à la partie sonore d'un programme.

## **6.9 Rendezvous et attentes.**

Pour forcer les canaux à se synchroniser, on utilise les 'rendezvous'. C'est un indicateur qui force l'exécution simultanée de plusieurs commandes SOUND.

ATTENTE (HOLD en anglais) est important pour démarrer un air et les suites de sons doivent être déjà prêtes. On utilise une séquence de HOLD (attentes) et RELEASE pour cela.

## **6.10 Organisation de la commande SOUND.**

Elle a la forme: SOUND G,H,I,J,K,L,M

ou

- G: canal (statut du canal)
- H: période de ton
- I: durée
- J: volume
- K: enveloppe de volume
- L: enveloppe de ton
- M: période de bruit

**SOUND** est une commande où tous les paramètres sont des entiers et seulement les deux premiers sont obligatoires, les autres étant facultatifs, et prenant une valeur par défaut si on ne précise pas leur valeur.

#### 6.10.1 Description des paramètres

**G:** canal (statut du)

Valeur: de 1 à 255  
obligatoire

Avec le CPC464 on peut jouer quatre sons en même temps, en ayant trois canaux de **SOUND** appelés A, B et C. Le nombre entier est donné en valeur décimale, puis changé en valeur de bits significatifs, de la manière suivante

DECIMAL	BIT	COMMANDE
1	0 LSB	son dirigé sur canal A
2	1	son dirigé sur canal B
4	2	son dirigé sur canal C
8	3	rendez vous avec canal A
16	4	rendez vous avec canal B
32	5	rendez vous avec canal C
64	6	hold
128	7 MSB	'flush'

par exemple

2 = envoie le son (**SOUND**) sur canal B envoie le son sur A et C

5 =  $64+32+2$

98 = envoie le **SOUND** sur canal de sortie B, rendez vous avec C

= et attente.

L'attente sur une combinaison de canaux arrête la musique et les suites (queues) sonores jusqu'à ce qu'elle soient lâchées par une commande **RELEASE** (ou une commande **SOUND** avec **FLUSH** dedans).

**FLUSH**

Flush dans une commande **SOUND** vide les sons dans ce canal.

**H:** période de ton

Valeur 0 à 4095  
obligatoire

Quand on met 0, il n'y a pas de fréquence et c'est utile pour les bruits.

### I: Durée

Valeur: de -32768 à +32767  
par défaut: 20

Pour les valeurs supérieures à zéro, la durée est exprimée en centièmes de seconde.  
Pour zéro, c'est la durée donnée par la commande ENV. Pour les valeurs négatives,  
la valeur absolue donne le nombre de répétitions de l'enveloppe de volume.

### J: volume

Valeur:

0 à 15

0 à 7 (s'il n'y a pas de commande ENV)

par défaut: 12 (4 s'il n'y pas de commande ENV)

C'est le volume de départ, modifiable par ENV. Zéro signifie pas de volume.

### K: enveloppe de volume

Valeur: de 0 à 15

par défaut: 0

La valeur indique une enveloppe déjà définie par une commande ENV.

### L: enveloppe de ton

Valeur: de 0 à 15

par défaut: 0

La valeur indique une enveloppe déjà définie par une commande ENT.

## 6.11 La commande ENV pour les enveloppes de volume

L'attaque et la chute d'une note est ce qui donne du corps et de la vie à une note.  
Cette commande ENV (ENveloppe de Volume) donne de la forme à la note.

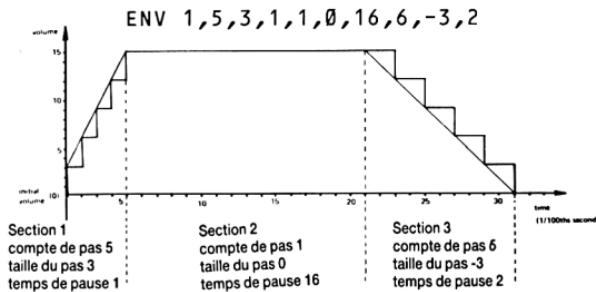
Forme de la commande en divisant l'enveloppe en sections

ENV N,P1,Q1,R1,P2,Q2,R2,P3,Q3,R3,P4,Q4,R4,P5,Q5,R5

N	numéro d'enveloppe	valeur 1-15
P1...5 :	décompte des pas (section 1 à 5)	valeur 0-127
Q1...5 :	taille du pas (id)	valeur -128 "127
R1...5 :	temps de pause (id)	valeur 0 255

Le numéro d'enveloppe est obligatoire, ainsi qu'une section les autres sections sont facultatives.

Exemple dans le diagramme suivant



## 6.12 La commande ENT et les enveloppes de ton.

Même type de structure qu'une enveloppe de volume, à ceci près qu'elle donne des variations de fréquence de la note, un espèce de vibrato, comme dans le diagramme suivant:

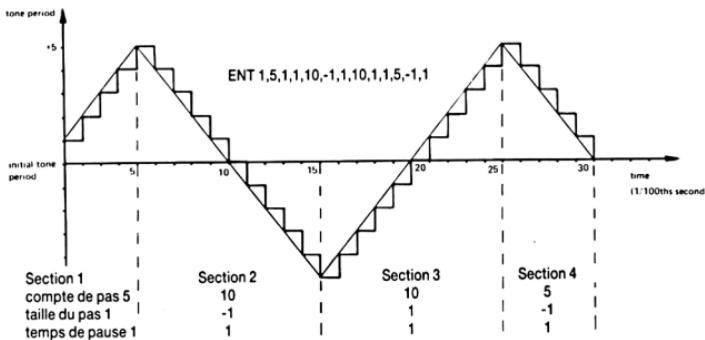


Figure 4: caractéristiques d'une enveloppe de ton

La forme de la commande est la même que pour ENV, avec 5 sections possibles.

**ENT S, T1,V1,T1,W1,T2,U2,W2,...,T5,V5,W5**

S est obligatoire et chaque section doit être complète si on l'utilise.

## 6.13 Autres Fonctions et Commandes

### SQ(x)

X est le numéro du canal, avec pour valeur 1, 2 ou 4 qui représentent les canaux A, B et C. C'est une demande d'état du canal. Cette fonction donne un entier de 0 à 255 sous forme de bit significatif.

Bit 0..2	nombre de place libres dans la queue
Bit 3	rendezvous avec canal A
Bit 4	rendezvous avec canal B
Bit 5	rendezvous avec canal C
Bit 6	attente au début de la queue
Bit 7	canal en train de jouer.

### ON SQ(y) GOSUB ·numéro de ligne·

y étant le numéro du canal, 1, 2 ou 4.

C'est une interruption qui permet de contrôler les sons, et ce sous-programme finit avec un RETURN comme un sous-programme normal.

### RELEASE

### RELEASE z

z indique les numéros de canal, avec un entier de 0 à 7.

Comme décrit dans la commande SOUND on peut signaler une attente (HOLD) sur un canal avec une commande SOUND particulière. La commande RELEASE libère ces attentes. Fonctionne avec des bits significatifs.

Bit 0:	canal A
Bit 1:	canal B
Bit 2:	canal C



# 7 Imprimantes et manettes de jeux

Le CPC464 n'a pas besoin d'interface supplémentaire pour une ou deux manettes de jeux et pour une imprimante de type Centronics parallèle.

Sujets abordés dans ce chapitre

- \* Joysticks (= manettes de jeux)
- \* Imprimante type parallèle
- \* Interfaces

## 7.1 Joysticks

Le joystick AMSOFT modèle JY1 est un appareil en option que vous pouvez acheter si vous utilisez le micro-ordinateur CPC464 pour des jeux conçus pour fonctionner avec des joysticks et faire feu (= FIRE). Le JY1 se branche derrière l'ordinateur en utilisant la fiche à 9 plots marquée USER PORTS (I/O). Le CPC464 peut avoir 2 manettes de jeux, le deuxième se branchant sur le côté du premier. Les branchements de cet appareil sont détaillés dans l'appendice V avec les autres possibilités d'entrées-sorties (I/O) = Input-Output en Anglais).



Le logiciel du CPC464 accepte 1 ou 2 joysticks et ils sont considérés comme faisant partie du clavier et peuvent être interrogés par les commandes **INKEY\$** et **INKEY** comme si c'étaient des touches du clavier. S'il y a seulement un bouton FEU sur votre joystick, il correspond probablement au **FIRE 2** dans la terminologie de l'AMSTRAD CPC464.

Une fonction particulière est utilisée pour analyser les joysticks directement, **JOY(0)** pour le premier joystick et **JOY(1)** pour le second. Cette fonction indique un bit significatif qui représente l'état des interrupteurs lors de la dernière analyse du clavier. Comme il y a cinquante de ces analyses par seconde, le résultat et quasiment l'état instantané de ces interrupteurs.

Les joysticks renvoient les valeurs suivantes où **KEY** (= touche) est la valeur à utiliser avec la fonction **INKEY** et **MIRROR** (= MIROIR) est la touche correspondante du clavier.

Premier Joystick	<b>JOY(0)</b>	<b>KEY</b>	Deuxième Joystick	<b>JOY(1)</b>	<b>KEY</b>	<b>MIRROR</b>
Haut (monte)	Bit 0	72	Haut	Bit 0	48	6
Bas (descent)	Bit 1	73	Bas	Bit 1	49	5
à gauche	Bit 2	74	à gauche	Bit 2	50	R
à droite	Bit 3	75	à droite	Bit 3	51	T
FIRE 2 (Feu)	Bit 4	76	FIRE 2	Bit 4	52	G
FIRE 1	Bit 5	77	FIRE 1	Bit 5	53	F

Il est à remarquer que lorsque le second joystick est interrogé le CPC464 ne peut faire la différence entre le joystick et les touches du clavier correspondante. En pratique, il y a peu de chance de confusion, le clavier pouvant être utilisé comme deuxième joystick.

Le deuxième joystick est identique au premier, et se branche simplement dans le premier. La fiche à 9 plots **USER PORTS (I/O)** est compatible avec les joysticks standard des micros, quoique vous ne puissiez ajouter un deuxième joystick sans un adaptateur spécial. Mais il ne faut pas chercher à utiliser ce type de joystick comme deuxième joystick et le brancher sur un joystick AMSOFT.

Les auteurs de logiciels peuvent définir une option en début de programme pour permettre de choisir entre le joystick ou le clavier, (la touche **[COPY]** ou une autre touche étant définie comme bouton feu).

## 7.2 Interface d'imprimante

Le CPC464 d'AMSTRAD peut s'utiliser avec une imprimante de type Centronics Parallèle.

Le cable d'imprimante est simplement fabriqué comme une connection simple entre la sortie **PRINTER** (= imprimante) et la fiche parallèle de l'imprimante; il y a deux 'doigts' en moins sur le circuit imprimé de l'ordinateur comparé au connecteur de l'imprimante, ceci pour permettre l'utilisation d'une interface standardisée. (Détails en Appendice V).

Les branchements doivent être ainsi faits que plot 1 de l'ordinateur est connecté au plot 1 de l'imprimante, plot 19 au plot 19, avec les plots 18 et 36 de l'imprimante non connectés au CPC464.

L'ordinateur utilise le signal **BUSY** (No. 11, busy signifie occupé), pour la synchronisation avec l'imprimante et attendra si l'imprimante est **OFF LINE** (non branchée).

Il n'y a pas de commandes spéciales de mise en route, et la sortie est dirigée sur l'imprimante en précisant canal 8:

**LIST #8**

imprime le programme BASIC de la mémoire, pourvu qu'il puisse être imprimé (par exemple non protégé).

A l'intérieur d'un programme, ou peut faire marcher l'imprimante en faisant (supposons que c'est à la ligne 200).

**200 PRINT #8, "Ceci doit être imprimé"**

Beaucoup d'imprimantes changent automatiquement de ligne si on atteint la fin de ligne en sortie, vérifiez dans le manuel d'instruction de l'imprimante. Le BASIC AMSTRAD fait de même avec la commande **WIDTH** (= largeur). Si on ne précise pas, c'est établi à 132 et peut être redéfini avec une nouvelle valeur, exemple **WIDTH 80**.

Si la valeur spéciale 255 est précisée, l'AMSTRAD BASIC laisse à l'imprimante le soin de faire les sauts à la ligne suivante. Le BASIC a en mémoire la position de l'imprimante qui est interrogée avec l'aide de la fonction **POS**:

**IF POS (#8)>50 THEN GOTO 100**

Le CPC464 envoie un ordre de changement de ligne, **CHR\$(10)**, et un ordre de retour de chariot, **CHR\$(13)**, à la fin d'une ligne.

L'imprimante a généralement un interrupteur de définition et vous vous en apercevrez rapidement en essayant d'imprimer.

## **7.3 Impression des GRAPHIQUES**

Le manuel fourni avec votre imprimante va préciser les codes de contrôle, généralement sous la forme.

**PRINT CHR\$(n)**

Des imprimantes peuvent avoir des caractères semblables aux caractères graphiques représentés dans l'Appendice III, mais il est peu probable qu'ils correspondent, et il faudra alors faire une table de conversion.

L'interface d'imprimante a été conçue pour l'utilisation avec des imprimantes matricielles économiques, on peut utiliser des imprimantes à marguerite, des traceuses graphiques, imprimantes à jet d'encre, pourvu qu'elles disposent d'une interface parallèle standard.

## **7.4 Imprimante Amstrad DMP1**

Cette imprimante matricielle de type Centronics a été spécialement conçue pour fonctionner avec l'AMSTRAD CPC464 et vous pourrez utiliser toutes ses possibilités avec l'aide du manuel spécial, notamment la possibilité d'imprimer les accents simples.

# 8 Guide de référence aux mots-clés du BASIC AMSTRAD

Une liste de mots-clés du BASIC, illustrée par des exemples, donnant le type d'utilisation et les mots associés.

Sujets abordés dans ce chapitre

- \* Le genre de notation
- \* Les caractères spéciaux et leur sens
- \* Tous les mots-clés réservés du BASIC AMSTRAD par ordre alphabétique.

Ce chapitre contient un dictionnaire des fonctions et commandes du BASIC contenu dans le ROM (mémoire morte) du CPC464. L'ensemble représente le double avantage d'un BASIC standard et d'extensions très utiles qui tiennent compte des capacités particulières du CPC464 en tant que machine.

## 8.1 Notation

### Caractères Spéciaux

& ou bien &H	Préfixe pour un nombre en hexadécimal
&X	Préfixe pour un nombre binaire
:	Sépare deux affirmations sur une même ligne
#	Préfixe pour un envoi sur un canal (d'affichage ou de sortie)

### Types de données

Les chaînes peuvent avoir de 0 à 255 caractères et sont désignées par « chaîne alphanumérique ». On peut mettre une chaîne après une autre avec le signe '+', pourvu que la chaîne qui en résulte ne dépasse pas 255 caractères.

Les données numériques peuvent être entières ou réelles.

Les entiers varient de -32768 à + 32767 et les nombres réels de -1.7E+38 à 1.7E±38, le plus petit nombre au dessus de zéro étant 2.9E-39 et chaque nombre ayant de 9 à 10 chiffres significatifs.

Les différents symboles de données sont

% entiers  
! réels  
\$ chaîne alphanumérique

Une ·expression numérique· est une expression qui prend une valeur numérique. Cela peut être des nombres, ou une variable numérique, ou des nombres qui opèrent sur des variables numériques, à peu près tout ce qui n'est pas chaîne alphanumérique.

Un ·numéro de canal· se rapporte à un nombre qui désigne l'écran (0 à 7), l'imprimante (8) ou la cassette (9) quand on veut envoyer le résultat sur un canal.

Une ·valeur impropre· signifie qu'une ·expression numérique· a pris une valeur en dehors des limites permises ou qu'un paramètre de commande n'est pas accepté et que le BASIC ne reconnaît pas comme valable.

## MOTS-CLÉS

Nous allons donner les mots-clés du BASIC AMSTRAD sous la forme suivante:

MOT-CLÉ  
Présentation  
Exemple  
Description  
MOTS-CLÉS associés

## IMPORTANT

### Les Mots-Clés sont

ou bien des **COMMANDES**: opérations qu'on exécute directement  
ou bien des **FONCTIONS**: opérations qui interviennent dans une expression.

### Parenthèses:

( ) sont nécessaires pour certaines fonctions ou commandes. Les autres types de parenthèses (.,[ ]) ne doivent pas être tapées dans une ligne de programme.

[.] contient des options

.. contient des expressions définies dans la description

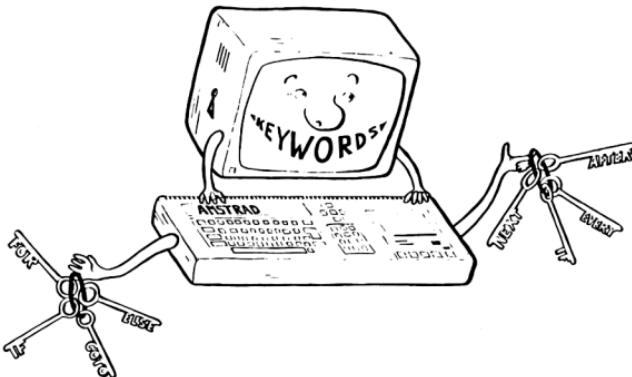
### Les guillemets

Seuls les doubles guillemets " " font partie de la structure d'un programme en BASIC. Les petits guillemets simples " sont là pour mettre en évidence des points particuliers et ne doivent pas apparaître normalement dans une fonction ou un exemple excepté à l'intérieur d'une ·chaîne alphanumérique·.

## La frappe

Le BASIC transforme tous les mots-clés tapés en minuscules en majuscules quand un programme est LISTé. Les exemples sont en majuscules puisque c'est ainsi qu'ils apparaissent après un LISTing, et si vous les tapez en minuscules, c'est mieux, car vous vous apercevrez plus facilement de vos erreurs de frappe car les mots-clés contenant une erreur resteront en minuscules.

Les mots-clés doivent être séparés des autres mots (la plupart du temps avec un espace) car le BASIC AMSTRAD permet de 'noyer' des mots-clés dans un nom de variable : Liste des clients sera accepté comme variable par l'AMSTRAD BASIC mais refusé par beaucoup d'autres.



## **ABS**

**ABS** (*·expression numérique·*)

```
PRINT ABS (-67.98)  
67.98
```

FONCTION: donne la valeur absolue de l'expression entre parenthèses -ce qui signifie que les nombres négatifs perdent le signe moins.

Mot-clé associé : **SGN**

## **AFTER**

**AFTER** (*·entier·*,*·nombre entier·*) **GOSUB** (*·numéro de ligne·*)

```
AFTER 200,2 GOSUB 320
```

COMMANDÉ: Appelle un sous-programme après (= **AFTER** en anglais) qu'un certain temps se soit écoulé. Le premier *·nombre entier·* indique la durée de l'attente en multiples de 0.02 seconde et le second *·nombre entier·*, (qui peut être 0,1,2 ou 3) précise lequel des quatre chronomètres d'attente il faut utiliser.

Mots-clés associés: **EVERY**, **REMAIN**

## **ASC**

**ASC** (*·chaîne alphanumérique·*)

```
PRINT ASC ("X")  
88
```

FONCTION: donne la valeur numérique du premier caractère d'une chaîne de caractères ASCII.

Mot-clé associé: **CHR\$**

## **ATN**

**ATN** (*·expression numérique·*)

```
PRINT ATN(1)  
0.785398163
```

FONCTION: Calcule l'Arc Tangente (réduisant l'*·expression numérique·* à un nombre réel en radians compris entre  $-\pi/2$  et  $+\pi/2$ ) de la valeur donnée.

Mots-clés associés: **SIN**, **COS**, **TAN**, **DEG**, **RAD**

## AUTO

**AUTO** [*·numéro de ligne·*], [*·différence·*]

**AUTO 100,50**

**COMMANDÉ:** donne automatiquement les numéros de ligne. Le *·numéro de ligne·* donne le premier numéro que vous voulez dans votre programme. La valeur *·différence·* est la différence entre deux numéros de ligne. Si vous ne précisez pas, les deux prennent la valeur 10 par défaut.

Quand un programme est déjà en mémoire, le BASIC met une étoile \* après le numéro de ligne pour prévenir qu'il y a un risque d'effacement.

## BIN\$

**BIN\$** (*·nombre entier sans signe·*, *·nombre entier·*)

**PRINT BIN\$ (64,8)**  
**01000000**

**FONCTION:** convertit le nombre entier en base 10 en nombre binaire, remplissant avec autant de zéros au début du nombre pour qu'il ait autant de chiffres qu'indiqué par le deuxième nombre entier.

## BORDER

**BORDER** [*·numéro de couleur·*], [*·numéro de couleur·*]

**BORDER 3,2**

**COMMANDÉ:** Pour changer la couleur de la bordure sur l'écran. Si deux couleurs sont indiquées, elles alternent à la vitesse déterminée par la commande SPEED INK, si elle est précisée. Les valeurs sont de 0 à 26.

Mot-clé associé: SPEED INK

## CALL

**CALL** [*·adresse·*], [*·liste de paramètres·*]

**CALL &BD19**

**COMMAND:** Permet à un sous-programme externe d'être appelé à partir du BASIC. A utiliser avec précaution, ce n'est pas pour les novices. Le CALL ci-dessus est sans danger, il est utilisé pour arranger le mouvement des caractères sur l'écran quand on fait des effets d'animation.

Mot-clé associé: UNT

## CAT

CAT

CAT

COMMANDÉ: demande au BASIC de commencer à lire une cassette et à donner les titres de tous les fichiers trouvés. Ceci n'affecte pas le programme en mémoire et peut être utilisé pour vérifier qu'un programme a été enregistré avant d'effacer la mémoire. Un message vous dit d'appuyer sur PLAY et quand un programme a été trouvé:

NOMDUDOSSIER - numéro du bloc - indicateur - OK; l'indicateur indique le type de sauvegarde fait à l'enregistrement:

- \$ un programme en BASIC
- % un programme protégé en BASIC
- \* un dossier ASCII
- & un dossier en binaire

D'autres caractères pourront être utilisés comme indicateurs si le dossier ne provient pas du BASIC.

Mots-clés associés: LOAD, RUN, SAVE

## CHAIN

### CHAIN MERGE

CHAIN ·nom du dossier·[,·numéro de ligne·]  
CHAIN MERGE ·nom du dossier·[,·numéro de ligne·]  
[,·DELETE ·ensemble de lignes·]

CHAIN "TEST",350

COMMANDÉ: CHAIN charge un programme à partir de la cassette dans la mémoire, remplaçant le programme existant. CHAIN MERGE amalgame un programme sur cassette avec un programme dans la mémoire. Le ·numéro de ligne· indique la ligne où doit commencer le programme quand le nouveau programme est enchainé à l'autre. Si aucun numéro de ligne n'est précisé, BASIC prend le plus petit disponible.

S'il n'y a pas de nom de dossier, BASIC va enchaîner le programme qu'il trouve sur la cassette. Si le premier caractère d'un dossier est !, il est enlevé du dossier et cela supprime les messages habituels de la lecture d'une cassette.

CHAIN MERGE conserve les variables déjà en mémoire mais les fonctions définies et les dossiers ouverts sont oubliés. ON ERROR GOTO ne fonctionne plus, RESTORE intervient, les fonctions DEFINT, DEFREAL & DEF STR sont remises à zéro et toutes les FOR, WHILE et GOSUB commandes du programme précédent sont oubliées. Les dossiers protégés ne peuvent s'amalgamer (= MERGE).

Mots-clés associés: LOAD, MERGE

## **CHR\$**

**CHR\$** (.nombre entier.)

```
PRINT CHR$(100)
d
```

FONCTION: Change une valeur numérique en un caractère équivalent (en utilisant le jeu de caractères de l'AMSTRAD CPC465 défini dans l'Appendice III)

Mots-clés associés: **ASC, LEFT\$, RIGHTS\$, MID\$, STR\$**

## **CINT**

**CINT** (.expression numérique.)

```
10 n=578.76543
20 PRINT CINT(n)
RUN
579
```

FONCTION: convertit une valeur numérique en un entier compris entre -32768 et 32767

Mots clés associés: **CREAL, INT, FIX, ROUND, UNT**

## **CLEAR**

**CLEAR**

**CLEAR**

COMMANDÉ: efface toutes les variables et fichiers.

## **CLG**

**CLG** [.masked ink.]

**CLG**

COMMANDÉ: efface l'écran graphique

Mots-clés associés: **CLS, ORIGIN**

## **CLOSEIN**

**CLOSEIN**

**CLOSEIN**

**COMMANDÉ:** Ferme le dossier d'entrée ouvert sur la cassette. Les commandes comme **NEW** et **CHAIN MERGE** abandonneront tous les fichiers ouverts.

Mots-clés associés: **OPENIN**, **CLOSEOUT**

## **CLOSEOUT**

**CLOSEOUT**

**CLOSEOUT**

**COMMANDÉ:** Ferme le dossier sur cassette en sortie.

Mots clés associés: **OPENOUT**, **CLOSEIN**

## **CLS**

**CLS** [..numéro de canal..]

**COMMANDÉ:** Pour effacer l'écran ou la fenêtre d'écran et lui donner sa couleur de papier: (**PAPER INK**)

## **CONT**

**CONT**

**CONT**

**COMMANDÉ:** Continue l'exécution du programme après un **\*Break\***, **STOP** ou **END** si le programme n'a pas été modifié. Des commandes directes peuvent être tapées.

## COS

**COS** (<expression numérique>)

```
?COS(34)
-0.848570274
```

et

```
deg:=?COS(34)
0.829037573
```

FONCTION: calcule le cosinus d'un angle et suppose par défaut la valeur en radian, à moins qu'on ait précisé degré (DEG) auparavant remarquez qu'on a utilisé? à la place de PRINT et mis des mots-clés en minuscules et /a marche avec le BASIC AMSTRAD.

Mots-clés associés: **SIN, TAN, ATN, DEG, RAD**

## CREAL

**CREAL** (<expression numérique>)

```
5 DEFINT n
10 n=75.765
20 d=n/34.6
30 PRINT d
40 PRINT CREAL(n)
50 PRINT n/55.4
run
2.19653179
76
1.37184116
Ready
```

FONCTION: Convertit une valeur en nombre réel (par opposition à nombre entier).

Mots-clés associés: **CINT**

## DATA

```
DATA ·liste de· · constantes·  
10 REM Proofreader list  
20 DIM Proofer$(5)  
30 DIM ProoferSurname$(5)  
40 FOR n=1 to 5  
50 READ Proofer$(n)  
60 READ ProoferSurname$(n)  
65 PRINT Proofer$(n); "ProoferSurname$(n)  
70 DATA Bob,Smith,Dicky,Jones,Malcolm,Green,  
    Alan,Brown,Ivor,Curry  
90 NEXT
```

COMMANDÉ: Déclare qu'une donnée est constante pour la durée du programme. Une des caractéristiques fondamentales du BASIC. La donnée (= DATA) doit correspondre aux variables qui l'appellent. Une ligne de DATA peut apparaître n'importe où dans le programme.

Mots-clés associés: READ, RESTORE

## DEF FN

DEF FN ·nom[ ·paramètres formels · ]= ·expression générale ·

```
10 DEF FN interêt(capital)=1.19*capital  
20 INPUT "Quel est le montant du capital";capital  
30 PRINT "Le total après un an avec les  
    intérêts est";FN interêt(capital)
```

COMMANDÉ: Le BASIC permet au programme de définir et de se servir de fonctions simples de calcul de valeurs. DEF Fonction est la partie définitive du mécanisme de création d'une fonction spécifique qui fonctionne d'une manière similaire aux fonctions déjà dans la BASIC (COS, SIN, ATN etc). Ceci peut être appelé à tout moment du programme et doit être placé en dehors des boucles d'exécution.

## DEFINT DEFSTR DEFREAL

DEF type ·lettres concernées ·

```
DEFINT I-N  
DEFSTR A,W-Z  
DEFREAL
```

COMMANDÉ: définit le type de variable par défaut, le 'type' pouvant être entier, réel ou chaîne. La variable sera déterminée suivant la première lettre du nom de la variable, qu'on peut mettre en majuscules ou minuscules.

Mots-clés associés: LOAD, RUN, CHAIN, NEW, CLEAR

## **DEG**

```
DEG
DEG
```

**COMMANDÉ:** Etablit le mode de calcul en degrés. Par défaut, les fonctions **SIN**, **COS** utilisent les radians pour les données numériques. La commande reste valable jusqu'à ce qu'on utilise les commandes **CLEAR** or **RAD** ou si on charge un autre programme.

Mot-clé associé:

## **DELETE**

```
DELETE (ensemble de lignes>
DELETE 100-200
```

**COMMANDÉ:** Une commande qui efface une partie du programme en mémoire. On ne peut pas récupérer si on se trompe, aussi il vaut mieux l'utiliser avec soin et faire attention aux erreurs de frappe avant de faire **[ENTER]**

Mot-clé associé: **NEW**

## **DI**

```
DI
10 CLS
20 TAG
30 EVERY 10 GOSUB 100
40 X1=RND*320:X2=RND*320
50 Y=200+RND*200
60 FOR X=320-X1 TO 320+X2 STEP 2
70 DI:PLOT 320,0,1:MOVE X-2, Y:PRINT " ";:MOVE
X,Y:PRINT "#";:EI
80 NEXT
90 GOTO 40
100 MOVE 320,0
110 DRAW X+8,Y-16,0
120 RETURN
```

**COMMANDÉ:** Annule une interruption (autre que **\*Break\***) jusqu'à ce qu'elle soit remise en route directement par une commande **EI** ou indirectement par un **RETURN** à la fin d'une routine **GOSUB** d'interruption.

On l'utilise quand le programme doit s'exécuter littéralement sans interruption, quand deux sous-programmes sont en compétition pour utiliser les ressources de l'ordinateur, par exemple les ressources graphiques dans le programme ci-dessus.

Mot-clé associé: **EI**

## DIM

DIM ·liste de: variables définies·

```
10 CLS:PRINT "Enter 5 names....":PRINT
20 DIM B$(5)
30 FOR N=1 TO 5
40 PRINT "Name"N"please";
50 INPUT B$(N)
60 NEXT
70 FOR N=1 TO 5
80 PRINT "How wise of you";B$(N);
      "to buy a CPC464"
90 NEXT
```

COMMANDÉ: elle réserve de l'espace dans la mémoire et définit le nombre maximum. On doit prévenir de la dimension d'un arrangement ou la valeur est 10 par défaut. Une fois fixée, la dimension d'un arrangement (ou matrice) ne peut être changée, ou une erreur se produit.

La liste des variables définies représente le nombre de dimensions dans l'arrangement et la quantité d'éléments de chaque dimension. DIM A\$(10,20) peut représenter la dimension d'un immeuble de 10 étages avec 20 appartements à chaque étage.

Comprendre les arrangements (appelés tableaux ou matrices dans certains livres) est un des éléments essentiels de la programmation avancée en BASIC.

Mot-clé associé: ERASE

## DRAW

DRAW ·coordonnée x·, ·coordonnée y·[, ·encre masquée·]

DRAW 200,200,13

COMMANDÉ: Dessine une ligne sur l'écran à partir de la position du curseur graphique vers une position absolue. Les coordonnée· sont les mêmes pour les trois modes. (Exemples au chapitre 5).

Mots-clés associés: DRAWR, PLOT, PLOTR, MOVE, MOVER, TEST, TESTR, XPOS, YPOS, ORIGIN

## DRAWR

DRAWR 200,200,13

COMMANDÉ: dessine une ligne sur l'écran à partir du curseur graphique vers une position relative par rapport à ce curseur graphique.

Mots-clés associés: DRAW, PLOT, PLOTR, MOVE, MOVER, TEST, TESTR, XPOS, YPOS, ORIGIN

## **EDIT**

**EDIT** ·line number·

**EDIT 110**

COMMANDÉ: édite (modifie) un programme en appelant une ligne précise. Voir chapitre 1.4.

Mot-clé associé: **L I S T**

## **EI**

**E I**

**E I**

COMMANDÉ: pour rétablir un interrupteur supprimé par la commande **D I** mot-clé associé: **D I**

## **END**

**E N D**

**E N D**

COMMANDÉ: Fin du programme. Il y a **END** (=fin) implicite dans l'AMSTRAD BASIC quand on arrive à la fin du programme. **END** ferme les dossiers sur cassette et revient en mode direct. Les sons continuent jusqu'à la fin des queues.

Mot-clé associé: **S T O P**

## ENT

ENT ·numéro d'enveloppe·[,·sections d'enveloppe·]

10 ENT 1,100,2,20

20 SOUND 1,100,1000,4,0,1

COMMANDÉ: En faisant un son, on peut faire varier le ton. Une enveloppe de ton définit comment faire varier le ton. Voir chapitre 6 pour une description plus détaillée des effets sonores.

On peut donner jusqu'à cinq sections d'enveloppe et chacune peut prendre une des deux formes suivantes

·décompte les pas·,·taille du pas·,·temps de pause·,

ou bien

= ·periode de ton·,·temps de pause·

·décompte de pas· donne le nombre de pas dans la section, un entier de 0 à 239.

·taille du pas· donne la variation de la période pour chaque pas de l'enveloppe, de -128 à +127

·temps de pause· donne le temps d'attente entre les pas, un entier de 0 à 255 (ou 0 à la valeur 256!)

·période de ton· est la nouvelle valeur de la période, un entier de 0 à 4095.

La commande SOUND donne la période de ton initial et précise une des quinze enveloppes. Si aucune enveloppe n'a été définie, le ton reste constant.

Mots-clés associés: ENV , SOUND

## ENV

ENV ·numéro d'enveloppe·[,·sections d'enveloppe·]

10 ENV 1,100,2,20

20 SOUND 1,100,1000,4,1

COMMANDÉ: Lorsqu'un son est produit, on peut faire varier le volume en se servant de cette commande.

Les paramètres et valeurs sont les mêmes que pour l'enveloppe de ton (voir ci-dessus), le ·numéro d'enveloppe· étant un entier de 1 à 15 précisant le volume.

Mots-clés associés: ENT, SOUND

## **EOF**

```
EOF  
PRINT EOF  
-1
```

FONCTION: Pour tester si l'entrée cassette est à la fin du fichier. Donnez -1 si c'est vrai (à la fin) ou 0 si c'est vrai (à la fin) ou 0 si c'est faux (pas encore à la fin).

Mot-clé associé: **OPENIN**

## **ERASE**

```
ERASE .list of;.nom de la variable.  
ERASE A,B$
```

COMMANDÉ: Quand un arrangement n'est plus nécessaire, il peut être effacé (ERASEd) et la mémoire utilisée pour d'autres choses.

Mot-clé associé: **DIM**

## **ERR**

## **ERL**

```
ERR  
ERL  
  
10 CLS  
20 ON ERROR GOTO 1000  
30 DATA SALLY,EMMA,JOANNE,HELEN,GEMMA  
40 READ A$  
50 PRINT A$  
60 GOTO 40  
70 REM error detection starts here  
1000 IF ERR=4 THEN PRINT "I have spotted a DATA  
EXHAUSTED error!"  
1010 IF ERL<70 AND ERL>20 THEN PRINT ".....in  
lines 30-60"  
1020 END
```

VARIABLES: Ces variables sont utilisées pour les sous-programmes de détection d'erreurs, leur numéro d'ordre et la ligne d'où elle vient. Voir les messages d'erreurs en Appendice 8.

Mots-clés associés: **ON ERROR, ERROR**

## **ERROR**

**ERROR** ·nombre entier·

**ERROR 17**

**COMMANDÉ:** Décide une action consécutive à une erreur numérotée l'erreur peut être une erreur déjà reconnue par le BASIC (Appendice VIII) auquel cas l'action est la même que celle prévue par le BASIC. Au delà, on peut programmer ses propres messages d'erreur.

Mots-clés associés: **ON ERROR ,ERR ,ERL**

## **EVERY**

**EVERY** ·nombre entier·[, ·nombre entier·] **GOSUB** ·line number·

**EVERY 500,2 GOSUB 50**

**COMMANDÉ:** Le CPC464 possède une horloge en temps réel. La commande **EVERY** permet à un programme en BASIC de s'arranger pour que des sous-programmes soient appelés à des intervalles réguliers. Quatre chronomètres indépendants sont disponibles, précisés par le deuxième ·nombre entier· de valeur 0,1,2 ou 3, chacun pouvant avoir un sous programme associé.

Mots-clés associés: **AFTER , REMAIN**

## **EXP**

**EXP** ·expression numériques·

**PRINT EXP (6.876)**  
**968.743625**

**FONCTION:** calcule 'e' à la puissance donnée par l'expression numérique, où 'e' est égal à 2.7182818, la base des logarithmes népériens, le nombre dont le logarithme naturel est 1.

Mot-clé associé: **LOG**

## **FIX**

```
FIX (.expression numérique.)
PRINT FIX (9.99999)
9
```

FONCTION: à la différence de CINT, FIX n'arrondit pas à l'entier le plus proche, il enlève simplement la partie décimale. (Arrondissant toujours par défaut)

Mots-clés associés: CINT, INT, ROUND

## **FOR**

```
FOR .variable simple.=.début. TO .la fin.[STEP.valeur.]
FOR Semaine=1 TO 52 STEP 2
```

COMMANDÉ: exécute le corps d'un programme un certain nombre de fois, le saut d'une valeur à l'autre étant commandé par la valeur du STEP. Si on ne précise pas, STEP prend la valeur 1 par défaut.

Mots-clés associés: NEXT, WHILE

## **FRE**

```
FRE (.expression numérique.)
FRE (.chaîne alphanumérique.)
PRINT FRE(0)
PRINT FRE("")
```

FONCTION: informe combien de mémoire reste disponible. La forme FRE ("") force l'ordinateur à mettre de l'ordre avant de donner la valeur disponible.

## **GOSUB**

```
GOSUB .numéro de ligne.
GOSUB 210
```

COMMANDÉ: Appelle un sous-programme BASIC en se branchant sur la ligne indiquée. Voir RETURN

Mot-clé associé: RETURN

## GOTO

**GOTO** ·numéro de ligne·

**GOTO 90**

COMMANDÉ: se branche sur la ligne indiquée

## HEX\$

**HEX\$** (·nombre entier sans signe·[,·nombre entier·])

**PRINT HEX\$(65534) FFFE**

FONCTION: change un nombre entier en un nombre hexadécimal

Mots-clés associés: **BIN\$**, **STR\$**

## HIMEM

?HIMEM

43903

VARIABLE: donne l'adresse de l'octet le plus élevé dans la mémoire utilisée par le BASIC et peut être utilisé dans des expressions numériques.

Avant de redéfinir l'octet le plus haut avec la commande **MEMORY**, il est souhaitable d'écrire **mm=HIMEM**. Vous pourrez après retourner à la capacité de mémoire précédente avec la commande **MEMORY mm**.

Mots-clés associés: **FRE**, **MEMORY**

## IF

**IF** ·expression logique· **THEN** ·option· [**ELSE** ·option·]

**IF** ·expression logique· **GOTO** ·numéro de ligne· [**ELSE** ·option·]

**IF A>B THEN A=C ELSE A=D**

**IF A>B GOTO 1000 ELSE 300**

COMMANDÉ: Une commande très fréquemment utilisée qui détermine les points de branchement conditionnel dans un programme. La partie logique est évaluée, et si elle est vraie, la partie **THEN** ou **GOTO** est exécutée, si elle est fausse, le programme exécute la partie **ELSE**, ou saute à la ligne suivante s'il n'y a pas de **ELSE**. Les conditions peuvent s'emboîter les unes dans les autres, la limite étant le nombre de 255 caractères. Le **IF** est terminé par la fin de la ligne. On ne peut avoir de déclarations indépendantes de **IF** sur la même ligne.

Mots-clés associés: **THEN**, **ELSE**, **GOTO**, **OR**, **AND**, **WHILE**

## INK

**INK** [,couleur[,couleur]]

**INK** 0,1,9

COMMANDÉ: Suivant le **MODE** choisi pour l'écran (chapitre 5), un certain nombre d'encre sont possibles. La couleur ou les couleurs (si elles sont intermittentes) utilisées pour une **INK** (encre) peuvent être changées par une commande **INK**, selon la table des valeurs de couleurs à l'Appendice VI.

Mots-clés associés: **PEN**, **PAPER**

## INKEY

**INKEY** (,nombre entier,)

```
10 CLS: IF INKEY(55)=32 THEN 30 ELSE 20
20 CLS: GOTO 10
30 PRINT "You're pressing [SHIFT] and V"
40 FOR t=1 TO 1000:NEXT:GOTO 10
```

FONCTION: Cette fonction interroge le CLAVIER pour indiquer quelles touches sont pressées. Le clavier est analysé tous les cinquantièmes de seconde. La fonction est surtout utile pour reconnaître les réponses oui-non, puisqu'il y a un des cas où on n'a pas besoin de savoir si **[SHIFT]** est pressé (donc peu importe si c'est une majuscule ou minuscule) l'exemple ci-dessus détermine si **[SHIFT]** et **V** sont pressés en même temps.

**[SHIFT]** et **[CONTROL]** sont identifiés par les valeurs suivantes.

Valeur	[SHIFT]	[CTRL]	KEY
-1	?	?	UP
0	UP	UP	DOWN
32	DOWN	UP	DOWN
128	UP	DOWN	DOWN
160	DOWN	DOWN	DOWN

Mots-clés associés: **INPUT**, **INKEY\$**

## INKEY\$

INKEY\$

```
10 CLS
20 PRINT "Etes vous intelligent (O ou N)?"
30 a$=INKEY$:IF a$=""GOTO 30
40 IF a$="N" OR a$="n" THEN PRINT "Vous l'etes
certainement puisque vous m'avez acheté!":END
50 IF a$=Y" OR a$="y" THEN PRINT "Vous etes trop
modeste!!!":END
60 GOTO 20
```

FONCTION: lit une touche du clavier pour avoir interaction avec l'opérateur sans avoir à utiliser la touche [ENTER] après chaque réponse. Si une touche est frappée, alors la fonction intervient, autrement il continue à tourner dans une boucle vide jusqu'à ce qu'une touche soit frappée.

Mots-clés associés: INPUT, INKEY

## INP

```
INP (·numéro d'entrée·)
PRINT INP (&FF77)
255
```

FONCTION: une fonction qui donne la valeur d'entrée dans le canal entrée-sortie (I/O) précisé par l'adresse.

Mots-clés associés: OUT, WAIT

## INPUT

INPUT [#·numéro du canal·,||;||·chaîne·,];liste de:[variable].  
ou bien INPUT [#·numéro du canal·,||;||·chaîne·,];liste de:[variables].

```
10 CLS
20 INPUT "Donne-moi deux nombres séparés par une
virgule";A,B
30 IF A=B THEN PRINT "Les deux nombres
sont égaux"
40 IF A>B THEN PRINT A "est plus grand que" B
50 IF A<B THEN PRINT A "est plus petit que" B
60 CLEAR:GOTO 20
```

**COMMANDÉ:** lit les données venant du canal précisé. Un point virgule après INPUT supprime le passage à la ligne à la fin de ce qu'on tape. Un point virgule après la chaîne fait qu'un point d'interrogation apparaît, une virgule supprime le point d'interrogation si on frappe quelque chose de faux (un 0 à la place d'un ?), alors un message BASIC apparaît:

?Redo from start

et on voit le texte original

Toute réponse doit se terminer avec [ENTER].

Quand un canal de cassette est précisé, il n'y a pas d'invite. Un élément sera lu dans le dossier pour chaque variable donnée dans la liste. Il faut que ce soit compatible avec ce qui est indiqué dans la commande INPUT, c'est-à-dire: variable numérique, avec une virgule, ou à la ligne, ou un espace ou fin du fichier. Les chaînes entre guillemets seront lues mot à mot jusqu'à la fin des guillemets. Des chaînes sans guillemets se terminent comme les valeurs numériques.

Mots-clés associés: LINE INPUT , READ , INKEY\$

## INSTR

```
INSTR ([·nombre entier·],·chaine·,·chaine·)  
PRINT INSTR (2, "BANANE","AN")
```

**FONCTION:** cherche dans la première chaîne si la deuxième chaîne est un morceau de la première en commençant au caractère désigné par le nombre entier facultatif, autrement commence au premier caractère de la première chaîne.

Mots-clés associés: MID\$, LEFT\$, RIGHT\$

## INT

```
INT (·expression numérique·)  
PRINT INT (-1.995)  
-2
```

**FONCTION:** Arrondit au plus petit nombre entier, enlevant la partie fractionnaire. Identique à FIX pour les nombres positifs, il donne 1 de moins que FIX pour les nombres négatifs qui ne sont pas des entiers.

Mots clés associés: CINT , FIX , ROUND

## **JOY**

**JOY (·nombre entier·)**

**10 IF JOY(0)=8 THEN GOTO 100**

FONCTION: La fonction **JOY** lit le bit significatif du **JOYSTICK** précisé par le ·nombre entier· (0 ou 1)

Bit	Décimal
0: Haut	1
1: Bas	2
2: Gauche	4
3: Droite	8
4: Fire 2	16
5: Fire 1	32

Mots-clés associés: **INKEY**

## **KEY**

**KEY ·nombre entier·,[CHR\$(n)+]·chaîne+[CHR\$(n)]**

**KEY 140, "RUN"+ CHR\$(13)**

COMMANDÉ: définit une nouvelle touche de fonction. Il y a 32 caractères d'expansion sur le clavier entre les valeurs 128-159, les touches du pavé numérique ayant les valeurs 128 à 140. Chaque chaîne peut avoir un maximum de 32 caractères, le total des caractères d'expansion ne pouvant dépasser 100. La commande **KEY** associe une chaîne à un caractère d'expansion.

Mot-clé associé: **KEY DEF**

## **KEY DEF**

**KEY DEF ·numéro de touche·,·répétition·[,·normal·[,·avec shift·[,·avec control·]]]**

**KEY DEF 46,1,63**

COMMANDÉ: change la valeur produite par une touche quelconque comme défini dans l'appendice III. L'exemple ci-dessus change la touche **N** et lui fait afficher le point d'interrogation? (No. de caractère 63).

Pour revenir à ce qui est normal:

**KEY DEF 46,1,110**

où le caractère 110 est la minuscule 17.

Mot clé associé: **KEY**

## **LEFT\$**

**LEFT\$ (·chaine·,·nombre entier·)**

```
10 CLS
20 A$="AMSTRAD"
30 B$=LEFT$(A$,3)
40 PRINT B$
RUN
```

[L'écran s'efface]

AMS  
Ready

FONCTION: extrait les caractères à la gauche de, et y compris la position précisée par le ·nombre entier·, dans la ·chaine·. Si la chaîne est plus courte que la longueur requise, la ·chaine· complète est affichée ou utilisée.

Mots-clés associés: **MID\$, RIGHT\$**

## **LEN**

**LEN (·chaine·)**

```
A$="AMSTRAD": PRINT LEN(A$)
7
```

FONCTION: donne le nombre de caractères de la ·chaine alphanumérique· y compris les espaces.

## **LET**

**LET ·expression logique·**

```
LET X=100
```

COMMANDÉ: un reste des BASIC historiques, où on devait annoncer ses variables. Seulement utile pour être compatible avec des programmes antérieurs. En AMSTRAD BASIC, il suffit d'écrire:

X=100

## LINE INPUT

```
LINE INPUT [#·numéro de canal,] ; [·chaîne·;]·variable en chaîne·  
LINE INPUT A$  
LINE INPUT "NAME";NS
```

COMMAND: lit une ligne entière du canal indiqué. Un point virgule après LINE INPUT supprime le retour à la ligne. Par défaut, le canal 0 est l'ensemble clavier -moniteur (= la console de travail).

Mots-clés associés: READ, INPUT, INKEY\$, INPUT\$

## LIST

```
LIST [·ensemble de lignes·] , #·lnuméro de canal·]  
LIST 100-100, #1
```

COMMAND: fait la LISTe du programme sur le canal désiré. 0 est l'écran, 8 est l'imprimante. On peut arrêter le LISTing une fois en pressant [ESC] une fois et reprendre en appuyant sur une autre touche; appuyer deux fois de suite sur [ESC] ramène au mode direct. On peut envoyer la liste d'un programme qu'on vérifie sur une fenêtre définie précédemment pour ne pas effacer l'écran sur lequel on travaille. On peut faire un LISTing à partir d'une ligne, qui finit à une ligne donnée, exemples:

LIST 30- ou LIST-200

## LOAD

```
LOAD ·nom du dossier· [, ·adresse·]  
LOAD "INVENT"
```

COMMAND: Pour lire un programme BASIC de la cassette pour le mettre en mémoire, en remplaçant tout programme qui y était déjà, ou, si on utilise l'option de l'adresse, pour charger un dossier binaire. Voir chapitre 2.

## LOCATE

```
LOCATE [#·numéro du canal·,]·coord.x·,·coord.y·  
10 MODE 1  
20 LOCATE 20,12  
30 PRINT CHR$(249)
```

COMMAND: déplace le curseur de texte vers une nouvelle position relative à l'origine de la fenêtre (WINDOW). Le coin en haut à gauche de la fenêtre est 1,1 et c'est par défaut tout l'écran qui est la fenêtre.

Mot-clé associé: WINDOW

## **LOG**

**LOG (·expression numérique·)**

? LOG(9999)

9.21024037

FONCTION: calcule le logarithme naturel de l'expression numérique et donne un résultat réel même si l'expression numérique est entière.

Mots-clés associés: **EXP, LOG10**

## **LOG10**

**LOG10 (·expression numériques·)**

? LOG10(9999)

3.99995657

FONCTION: calcule le logarithme à base 10 de l'expression numérique.

Mots-clés associés: **EXP, LOG**

## **LOWER\$**

**LOWER\$ (·chaîne alphanumérique·)**

A\$="AMSTRAD":PRINT LOWER\$(A\$)

amstrad

FONCTION: Change dans la chaîne numérique toutes les Majuscules en minuscules. Utile quand on s'attend à des réponses pouvant être des majuscules ou des minuscules ou un mélange de deux.

Mot-clé associé: **UPPER\$**

## **MAX**

**MAX (·list d'·expressions numériques·)**

10 n=66

20 PRINT MAX (1,n,3,6,4,3)

FONCTION: donne dans la liste la valeur la plus grande.

Mot-clé associé: **MIN**

## **MEMORY**

**MEMORY** ·adresse·

**MEMORY &20AA**

**COMMANDÉ:** Rétablit les paramètres mémoire du BASIC pour changer la quantité de mémoire disponible en fixant l'adresse de l'octet le plus élevé. Voir la description du mot-clé **HIMEM**. Pour analyser la mémoire, utilisez la commande **FRE**.

Mots-clés associés: **HIMEM, FRE**

## **MERGE**

**MERGE** [·nom du dossier·]

**MERGE"PLAN"**

**COMMANDÉ:** 'mélange' un programme sur cassette avec un programme qui est déjà dans la mémoire. Cela ajoute le contenu d'un dossier au programme déjà en mémoire. S'il n'y a pas de nom, BASIC va essayer d'amalgamer le premier dossier valable trouvé sur la cassette. Si le premier caractère sur le dossier est !, c'est enlevé du nom du dossier et cela supprime les messages ordinaires de la lecture de cassette.

Quand on fait cette commande **MERGE**, il faut être sûr que les numéros de lignes sont différents et il faut utiliser la commande **RENUM** (**RENUM** éroter les lignes).

Toutes les variables, fonctions définies et dossiers ouverts sont abandonnés. **ON ERROR GOTO** disparaît, **RESTORE** intervient et les fonctions **DEFINT**, **DEFREAL** et **DEFSTR** sont remises à zéro. Les dossiers protégés ne peuvent pas être l'objet de **MERGE**.

Mots-clés associés: **LOAD, CHAIN, CHAIN MERGE**

## **MID\$**

**MID\$** (·chaîne·,·nombre entier·[,·nombre entier·])

**A\$="ORDINATEUR":PRINT MID\$(A\$,3,4)**

**DINA**

**COMMANDÉ** et **FONCTION:** **MID\$** indique une partie de chaîne qui peut être l'objet d'une opération (**MID\$** utilisé comme commande) ou comme élément d'une expression en chaîne (**MID\$** comme fonction). Le premier ·nombre entier· précise la position du premier caractère dans la chaîne. Le deuxième nombre entier indique la longueur de la sous-chaîne que l'on demande. Si ce n'est pas précisé, cela va jusqu'à la fin de la chaîne.

Mots-clés associés: **LEFT\$, RIGHTS\$**

## **MIN**

```
MIN (..liste d'expressions numériques..)
PRINT MIN (3,6,2.95,8,9)
2.95
```

FONCTION: L'opposé de MAX

Mot-clé associé: MAX

## **MODE**

```
MODE <nombre entier>
```

```
MODE 2
```

COMMANDÉ: Pour changer le mode d'écran (0,1 ou 2) et efface l'écran pour avoir INKØ qui n'est peut être pas le PAPER INK utilisé à ce moment. Toutes les fenêtres de textes et de graphiques sont retournées à l'écran entier et les curseurs de texte et graphique retournent à leur point d'origine.

Mots-clés associés: WINDOW, ORIGIN

## **MOVE**

```
MOVE <coord.x>,<coord.y>
MOVE 32,57
```

COMMANDÉ: Positionne le curseur graphique à une position absolue. XPOS et YPOS sont les fonctions à utiliser pour savoir la position actuelle du curseur.

Mots-clés associés: MOVER, PLOT, PLOTR, DRAW, DRAWR, TEST, TESTR, XPOS, YPOS

## **MOVER**

```
MOVER <x relatif>,<y relatif>
MOVER 20,30
```

COMMANDÉ: Pour positionner le curseur graphique relativement à la position du curseur à ce moment là. XPOS et YPOS sont les fonctions qui donnent la position du curseur graphique à tout moment

Mots-clés associés: MOVE, PLOT, PLOTR, DRAW, DRAWR, TEST, TESTR, XPOS, YPOS

## **NEW**

**NEW**

**NEW**

**COMMANDÉ:** efface le programme et les variables en mémoire. Les définitions de touches (KEY) ne sont pas effacées et l'affichage reste ce qu'il était. C'est un point irréversible, à utiliser avec précautions, comme le RESET complet ([CTRL], [SHIFT] et [ESC] qui lui, efface tout.

## **NEXT**

**NEXT** [.liste de variables.]  
**FOR** T=1 TO 1000:**NEXT**

**COMMANDÉ:** Donne la fin d'une boucle qui commence avec **FOR**. La commande **NEXT** peut être anonyme, ou peut se rapporter au **FOR** qui est concerné, ce qui dans notre exemple donnerait.

**NEXT** T

Mot-clé associé: **FOR**

## **ON GOSUB**

## **ON GOTO**

**ON** <expression donnant un nombre entier> **GOSUB** <liste de:>numéros de ligne>  
**ON** <expression donnant un nombre entier> **GOTO** <liste de:>numéros de ligne>

**10 ON jour GOSUB 100,200,300,400,500**  
**20 ON taux GOTO 100,200,300,400,500**

**COMMANDÉ:** GOSUB vers un sous-programme, GOTO vers une ligne indiquée par le résultat de l'expression. Si le résultat est 1, (jour=1) la première ligne après **GOSUB** est choisie (dans notre exemple) si jour=2, le branchement se fait vers la deuxième ligne (200) et ainsi de suite.

Mots-clés associés: **GOTO**, **GOSUB**

## ON BREAK GOSUB

```
ON BREAK GOSUB ·numéro de ligne·  
10 ON BREAK GOSUB 40  
20 PRINT "le programme marche"  
30 GOTO 20  
40 CLS  
50 PRINT "presser [ESC] deux fois de suite  
appelle une routine GOSUB "  
60 FOR t=1 TO 2000:NEXT  
70 RETURN
```

COMMANDÉ: Appelle un sous-programme lorsqu'il y a un BREAK dans le programme ([**ESC**] deux fois de suite).

Mots-clés associés: **ON BREAK STOP, RETURN**

## ON BREAK STOP

```
ON BREAK STOP  
10 ON BREAK GOSUB 40  
20 PRINT "le programme marche"  
30 GOTO 20  
40 CLS  
50 PRINT "presser [ESC] deux fois de suite  
appelle une routine GOSUB "  
60 FOR t=1 TO 2000:NEXT  
65 ON BREAK STOP  
70 RETURN
```

COMMANDÉ: Quand on l'emploie au cours d'un sous-programme **ON BREAK**, **ON BREAK STOP** annule le retour, mais n'a pas d'effet immédiat. Dans le programme ci-dessus, qui comprend **ON BREAK STOP**, le "piège à BREAK" **ON BREAK GOSUB** ne marchera qu'une fois.

Mot-clé associé: **ON BREAK GOSUB**

## **ON ERROR GOTO**

```
ON ERROR GOTO·line number·  
10 ON ERRO GOTO 80  
20 CLS  
30 PRINT "S'il ya une erreur,j'aimerais bien"  
40 PRINT "avoir la liste du programme, pour"  
50 PRINT "voir la position de l'erreur"  
60 FOR t=1 TO 4000:NEXT  
70 GOTO 200  
80 CLS: PRINT"IL Y A UNE ERREUR A LA LIGNE";ERL:  
    PRINT  
90 LIST
```

COMMANDÉ: se dirige vers la ligne indiquée du programme quand une erreur se produit; dans l'exemple la ligne 70 contient une erreur.

Mots-clés associés: **ERR, ERL, RESUME**

## **ON SQ GOSUB**

```
ON SQ ·canal sonore· GOSUB ·numéro de ligne·  
ON SQ (2) GOSUB 2000
```

COMMANDÉ: permet une interruption quand il y a de la place dans une queue sonore. Le ·canal sonore· est un nombre entier indiquant une des valeurs:

- 1: pour canal A
- 2: pour canal B
- 3: pour canal C

Mots-clés associés: **SOUND, SQ**

## **OPENIN**

```
OPENIN ·nom du dossier·  
100 OPENIN "! INFORMATION"
```

COMMANDÉ: Ouvre un dossier existant sur la cassette pour avoir des données qui vont être utilisées par le programme déjà en mémoire. Si le premier caractère dans le ·nom du dossier· est un point d'exclamation, alors les messages affichés d'habitude sont supprimés et le programme lit le premier bloc de la cassette, prêt à utiliser les données.

Mots-clés associés: **CLOSEIN, OPENOUT**

## **OPENOUT**

**OPENOUT** ·nom du dossier·

**OPENOUT** " !FACTS "

**COMMANDÉ:** Ouvre un dossier de sortie sur la cassette utilisable avec le programme en mémoire. Si le premier caractère est ! les messages de manipulation de cassette sont supprimés, et le programme crée le premier bloc de données, avec le dossier et le nom qu'on lui a donné. Chaque bloc contient 2K de données et rien n'est écrit tant que le 'buffet' de 2K n'est pas rempli ou qu'une commande **CLOSEOUT** a fermé le dossier.

**ATTENTION:** la commande **NEW** abandonne tout dossier ouvert et les données seront perdues.

Mots-clés associés: **CLOSEOUT**, **OPENIN**

## **ORIGIN**

**ORIGIN** ·x·, ·y·, ·l·gauche·, ·droite·, ·haut·, ·bas· |

**COMMANDÉ:** détermine le point de départ du curseur graphique. La partie [facultative] de la commande contient la définition d'une nouvelle fenêtre graphique, qui sera utilisable dans tous les modes graphiques à cause du système d'adresses en pixels utilisé.

**ORIGIN** est le point de coordonnées 0,0 en BAS à GAUCHE de l'écran, les coordonnées allant en montant et vers la droite. (On peut utiliser les grilles de fenêtres pour les graphiques, appendice VI) Si un des coins de la fenêtre est hors de l'écran,, on suppose qu'ils représentent la position visible la plus éloignée de l'écran.

Mot-clé associé: **WINDOW**

## **OUT**

**OUT** ·numéro de sortie·, ·nombre entier·

**OUT** & F8F4,10

**COMMANDÉ:** envoie la valeur du ·nombre entier· (qui doit avoir la valeur 0...255) vers la sortie précisée par l'adresse du numéro de sortie.

Mots-clés associés: **INP**, **WAIT**

## PAPER

PAPER |#·.numéro du canal· , l'encre masquée·

```
10 MODE 0
20 FOR p=0 TO 15
30 PAPER p:CLS
40 PEN 15-p
50 LOCATE 6,12:PRINT "PAPER"p
60 FOR t=1 TO 500:NEXT t
70 NEXT p
```

COMMANDÉ: établit la couleur du fond pour les caractères. Quand les caractères s'affichent sur l'écran, la cellule du caractère (la grille) est remplie par l'encre correspondant au papier (PAPER INK) avant que le caractère ne soit écrit, à moins qu'on ait choisi le mode transparent.

Le nombre de couleurs disponibles dépend du mode choisi. Si l'encre masquée· n'est pas disponible, le papier prend une autre couleur disponible.

Mots-clés associés: INK, WINDOW, PEN

## PEEK

PEEK (·expression avec adresse·)

```
10 MODE 2
20 INK 1,0: INK 0,12 : BORDER 12
30 INPUT "Adresse de depart de l'examen";first
40 INPUT "Adresse de la fin de l'examen";last
50 FOR n=first TO last
60 VALUE$=HEX$(PEEK(n),2)
70 PRINT VALUE$;
80 PRINT" pour ";HEX$(n,4),
90 NEXT
```

FONCTION: Examine le contenu d'un endroit de la mémoire. Ceci vous permet de voir dans la mémoire VIVE RAM du CPC464. Vous pouvez alors lire le RAM dans les zones basses (&0000-&FFFH) et hautes (&C000-FFFF) mais pas la mémoire morte ROM.

Mot-clé associé: POKE

## PEN

PEN [#·numéro du canal·,]·encre masquée·

PEN 1,2

COMMANDÉ: PEN établit l'encre à utiliser pour écrire sur le canal d'écran précisé, 0 par défaut.

Mots-clés associés: INK, PAPER

## PI

PI

```
PRINT PI
3.14159265
10 REM DESSIN EN PERSPECTIVE
20 MODE 2
30 RAD
40 INK 1,0
50 INK 0,12
60 BORDER 970 FOR N=1 TO 200
80 ORIGIN 420,0
90 CRAW 0,200
100 REM Dessine des angles vers l'infini
110 DRAW 30*N*SIN(N*PI4),(SIN(PI/2))*N*SIN(N)
120 NEXT
130 MOVE 0,200
140 DRAWR 0,50
150 DRAWR 40,0
160 WINDOW 1,40,1,10
170 PRINT "Maintenant vous pouvez finir le
programme du pendu!"
```

FONCTION: donne la valeur approchée de Pi (3,141592653468251) et on l'utilise beaucoup dans les graphiques.

Mots-clés associés: DEG, RAD

## PLOT

```
PLOT coordonnée x, coordonnée y[, encre masquée]  
10 MODE 2: PRINT "donnez quatre nombres  
    séparés par des virgules":PRINT  
20 PRINT "donnez l'origine X (0-639), l'origine Y  
(0-399), le rayon et l'angle ":INPUT x,y,r,s  
30 ORIGIN x,y  
40 FOR angle=1 to 360 STEP s  
50 XPOINT=r*COS(angle)  
60 YPOINT=r*SIN(angle)  
70 PLOT XPOINT,YPOINT  
74 REM MOVE 0,0  
75 REM DRAW XPOINT, YPOINT  
80 NEXT
```

COMMANDÉ: Essayez 320,200,20,1 comme première réponse. **PLOT** dessine comme **DRAW**, mais dessine un seul point. Si vous enlevez la **REM** dans la ligne 75 et mettez un **REM** à la ligne 70 pour la rendre inutile, vous voyez la différence. (Enlevez le **REM** de la ligne 74 pour remplir le cercle.)

Le programme est en mode RADian et chaque **STEP** (étape) est plus grande qu'un degré. Faites 25 **DEG** et **RUN** de nouveau pour voir la différence.

Mots-clés associés: **DRAW**, **DRAWR**, **PLOT**, **PLOTR**, **MOVE**, **MOVER**, **ORIGIN**, **TEST**, **TESTR**, **XPOS**, **YPOS**

## PLOTR

```
PLOTR (coordonnée x, coordonnée y[, encre masquée])  
10 MODE 2: PRINT "donnez quatre nombres  
    séparés par des virgules":PRINT  
20 PRINT "donnez 'origine X (0-639), l'origine Y  
(0-399), le rayon et l'angle ":INPUT x,y,r,s  
30 ORIGIN x,y  
40 FOR angle=1 to 360 STEP s  
50 XPOINT=r*COS(angle)  
60 YPOINT=r*SIN(angle)  
70 PLOTR XPOINT, YPOINT  
80 NEXT:GOTO 40
```

COMMANDÉ: essayez 320,0,20,1 comme première réponse. **PLOTR** est le même que **DRAWR**, seul le pixel d'arrivée est affiché.

Mots-clés associés: **DRAW**, **DRAWR**, **PLOT**, **PLOTR**, **MOVE**, **MOVER**, **ORIGIN**, **TEST**, **TESTR**, **XPOS**, **YPOS**

## **POKE**

**POKE** ·expression avec adresse·, ·nombre entier·

**POKE &00FF,10**

COMMANDÉ: donne accès direct à la mémoire de la machine et charge un ·nombre entier· entre 0 et 255 directement à l'adresse indiquée. On peut facilement faire ·éclater· le système si on ne fait pas attention. Seulement pour les mordus du code Machine.

Mot-clé associé: **PEEK**

## **POS**

**POS** (#·numéro de canal·)

**PRINT POS (#0)  
1**

FONCTION: donne la position pour un canal donné.

Ecran: donne la coordonnée X du curseur de texte, par rapport à la position de l'origine de la fenêtre, qui est le coin haut à gauche de coordonnées 1,1.

Imprimante: donne la position du chariot, où 1 est la marge à gauche. Tous les caractères ASCII plus grand que &F(=31) sont inclus.

Cassette: comme pour l'imprimante (mais c'est un chariot virtuel.)

Mot-clé associé: **VPOS**

## **PRINT**

**PRINT** [#·numéro du cana·,][·iste à imprimer·]  
[·USING option·][· séparateur·]]

**PRINT #0, "abc"** ·print list· est une ·expression· [·séparateur··élément à imprimer··]\* ·élément à imprimer· est un expression, ou bien

**SPC (·nombre entier·)**  
**TAB (·nombre entier·)**

Pour écrire des données sur un dossier sur cassette

**10 OPENOUT "DONNEES"**  
**20 PRINT #9, "Bonjour"**  
**30 CLOSEOUT**

Pour écrire des données sur une imprimante

Pour écrire des données sur une imprimante

```
10 SALAIRE DE JULES=23*PI
20 PRINT #8, USING "####.##";SALAIREDEJULES
30 PRINT #0, SALAIREDEJULES
RUN
72.25663
```

Pendant ce temps, sur l'imprimante

72.26

COMMANDÉ: imprime et affiche au ·numéro de canal· précisé en utilisant le format demandé. Voyez la table à la page suivante pour les indicateurs.

Quand aucun format n'est précisé, BASIC affiche (imprime) en 'format libre' où une virgule après l'élément à imprimer envoie cet élément dans la ZONE suivante (13 colonnes dans une ZONE par défaut). Une point virgule sépare les expressions utilisant un seul espace.

SPC (·nombre entier·) imprime le nombre donné d'espaces, donnant 0 par défaut si le nombre est négatif. Si l'expression est plus large que la largeur de l'appareil (écran, imprimante) elle est ramenée aux limites utilisables. SPC n'a pas besoin d'être terminé par une virgule ou un point-virgule, le point-virgule est sous-entendu dans tous les cas.

TAB (·nombre entier·) indique aux espaces de se déplacer jusqu'à la position indiquée, si le nombre est négatif, 1 est supposé par défaut. TAB est réduit automatiquement suivant la largeur de l'appareil indiqué par le canal.

Si la position atteinte le demande, il y a un passage à la ligne et des blancs sont imprimés jusqu'à la position demandée. Un point-virgule est sous-entendu dans une commande TAB.

**PRINT USING** ".Indicateurs de format."

## NUMERIQUE

INDICATEUR	CHIFFRES POSSIBLES	CARACTERES	DEFINITION	EXEMPLE
#	1	1	Champ numérique	####
.	0	1	Point décimal	# . #
-	0	1	Imprime un signe devant ou derrière le nombre + pour les nombres positifs (un nombre négatif ne peut avoir le signe - devant.)	+###+
-	0	1	Imprime le signe - si c'est négatif laisse un blanc si c'est autre	## - ## -
**	2	2	Astérisque de début	**####.##
\$\$	1	2	Dollar \$ flottant placé devant le premier chiffre.	\$\$##.##
*\$	2	3	Astérisques pour remplir et dollar flottant.	*\$#.#
,	1	1	Met la virgule tous les trois chiffres (à gauche du point décimal uniquement)	,###,##,
↑↑↑↑↑	0	4	Format exponentiel. Le nombre est placé de telle manière que le premier chiffre n'est pas zéro	#,##↑↑↑

**STRING [= chaîne alphanumérique]**

!	Premier caractère seulement	!
·espace·	Nombre d'espaces indiqués plus un espace avant et après un ensemble de caractères	
&	Domaine à longueur variable On dit aussi 'zone' et 'champ' à la place de domaine.	&

## **RAD**

**RAD**

**RAD**

**COMMANDÉ:** Etablit le mode en RADians

Mots clés associés: **DEG, SIN, COS, TAN, ATN**

## **RANDOMIZE**

**RANDOMIZE** [**·expression numérique·**]

```
10 RANDOMIZE 23  
20 PRINT RND(6)
```

**COMMANDÉ:** Le générateur de nombres au hasard du BASIC produit une séquence de pseudo hasard, chaque nombre dépendant du précédent et à partir d'un nombre donné, la séquence est toujours la même. **RANDOMIZE** donne une nouvelle valeur de départ, soit automatique, soit donnée par l'utilisateur. **RANDOMIZE TIME** produit une séquence difficile à répéter.

Mot-clé associé: **RND**

## **READ**

```
READ Liste des: <variables>  
10 FOR X=1 TO 4  
20 READ N$  
30 PRINT N$  
40 DATA PAUL, ANNIE, ZEBULON, RICHARD  
50 NEXT
```

**COMMANDÉ:** **READ** cherche les données dans une liste de constantes fournies dans les lignes de **DATA** correspondantes et les attribue à des variables, lisant automatiquement l'élément suivant dans la suite des **DATA**. **RESTORE** remplace l'indicateur au début de la ligne des **DATA**. Voir le mot-clé **DATA**.

Mots-clés associés: **DATA, RESTORE**

## **RELEASE**

**RELEASE** ·canaux sonores·

**RELEASE 4**

**COMMANDE:** Quand un son est placé dans une queue, il peut y avoir un état d'attente. Si un des canaux indiqués dans ce canal sonore est en attente, alors il est libéré (**RELEASEd**). Voir chapitre 6 si les sons vous intéressent.

Mot-clé associé: **SOUND**

## **REM**

**REM** ·reste de la ligne·

```
10 REM Les Monstres Métagalactiques
20 REM Un jeu hyper super extra intersidéral
30 REM Copyright AMSOFT 1985
```

**COMMANDE:** utilisée pour placer des REMarques ou des rappels dans un programme sans affecter le résultat et le fonctionnement du programme. Tout ce qui est après le **REM** dans une ligne est ignoré. Un simple apostrophe ' dans une ligne (mais pas dans un chaîne ou un DATA) est équivalent à: **REM**.

## **REMAIN**

**REMAIN** (.nombre entier.)

```
REMAIN(3)
PRINT #6,REMAIN(0);
```

**FONCTION:** Supprime le chronomètre indiqué (0,1,2 ou 3). Lit le temps qui restait à courir. Zéro est indiqué si le chronomètre n'avait pas été mis en route.

Mot-clé associé: **AFTER**, **EVERY**

## RENUM

**RENUM** [*nouveau numéro de ligne*] [*ancien numéro de ligne*] [*différence entre deux lignes*]

```
RENUM  
RENUM 1000,200,20
```

COMMANDÉ: **RENUM** renumérote les lignes suivant les indications données dans la commande. Par défaut, le *nouveau numéro de ligne* est 10, l'*ancien numéro de ligne* est la première ligne du programme, et la *différence* est 10. Dans l'exemple plus haut, **RENUM** va renuméroter l'ancien programme à partir de la ligne 200, donner le numéro 1000 à cette ligne et numérotter les suivantes de 20 en 20 (1020, 1040, 1060 etc) **RENUM** s'occupe de tous les **GOSUB**, **GOTO** et autres appels de ligne. Si tous les indicateurs manquent, le programme est renumérotré comme si on avait dit **RENUM 10,,10**

Les numéros de lignes valables vont de 1 à 65535.

## RESTORE

**RESTORE** [*numéro de ligne*]

```
RESTORE 300  
10 FOR N=1 TO 6  
20 READ A$  
30 PRINT A$ "";  
40 DATA des donnees,RESTOREes,peuvent,etre,lues,  
de nouveau  
50 NEXT  
60 PRINT  
70 RESTORE  
80 GOTO 10
```

COMMANDÉ: Remet la position de l'indicateur de lecture au début de la ligne indiquée par le *numéro de ligne*. Si on ne mentionne pas le *numéro de ligne*, l'indicateur pointe la première ligne commençant par **DATA**.

Mots-clés associés: **READ**, **DATA**

## RESUME

**RESUME** [*numéro de ligne*]

ou bien **RESUME NEXT**  
**RESUME 300**

COMMANDÉ: Lorsqu'une erreur a été décelée par une commande **ON ERROR GOTO** et a été corrigée, **RESUME** (= recommencer) permet au programme de recommencer, la ligne de redépart pouvant être indiquée.

## **RETURN**

```
RETURN  
RETURN
```

COMMANDÉ: signale la fin d'un sous-programme. Le BASIC retourne à la position juste après le GOSUB qui appelait le sous-programme.

Mots-clés associés: GOSUB, ON X GOSUB, ON SQ GOSUB, AFTER n GOSUB, EVERY n GOSUB, ON BREAK GOSUB.

## **RIGHT\$**

RIGHT\$ (·chaîne alphanumérique·, nombre entier·)

```
10 CLS  
20 A$="AMSTRAD"  
30 B$=RIGHT$(A$,3)  
40 PRINT B$  
RUN
```

[L'écran s'efface]

```
RAD  
Ready
```

FONCTION: extrait les caractères à droite de la chaîne, la position indiquée par un ·nombre entier· étant le nombre de caractères obtenus, sauf si la chaîne est plus petite auquel cas toute la chaîne est affichée.

Mots-clés associés: MID\$, LEFT\$

## **RND**

RND [(·expression numérique·)]

```
10 RANDOMIZE 23  
20 PRINT RND(6)
```

FONCTION: donne un nombre au hasard, qui peut être le suivant d'une séquence, le même ou le premier d'une nouvelle séquence. La commande RANDOMIZE dans le programme ci-dessus fait que RND(6) donne le même nombre à chaque fois (0.146940658).

RND(0) donne le dernier chiffre produit au hasard. Quand l'expression numérique est négative, la séquence est facile à prévoir.

Mot-clé associé: RANDOMIZE

## ROUND

ROUND (*expression numérique*,*nombre entier*)

```
10 x=0.123456789
20 FOR r=9 TO 0 STEP-1:PRINT r,ROUND(x,r):NEXT
25 x=123456789
30 FOR r=0 TO-9 STEP-1
40 PRINT r,ROUND (x,r)
50 NEXT
```

FONCTION: Arrondit une *expression numérique* après la virgule, à une place indiquée par le *nombre entier*. Si le *nombre entier* est négatif, la valeur est arrondie en un entier suivi d'autant de zéros que l'entier

Mots-clés associés: INT, FIX, CIN, ABS

## RUN

RUN *chaîne alphanumérique*:

```
RUN "BIENVENUE"
```

COMMANDÉ: Charge un programme qui est sur cassette et commence à l'exécuter. Si on a "", le BASIC cherche à charger et exécute le premier dossier qu'il rencontre sur la cassette. Si le premier caractère dans la chaîne est !, les messages de manipulation de cassette sont supprimés.

ATTENTION: BASIC effectue une commande NEW au moment où il lit un *nomdedossier* sur la cassette.

Mot-clé associé: LOAD

## RUN

RUN [*line number*] :

```
RUN 100
```

COMMANDÉ: Commence à exécuter le programme à la ligne indiquée ou à partir du début si aucune ligne n'est indiquée. Les dossiers sur cassette sont abandonnés et tout ce qui est emmagasiné dans les 'buffets' (=mémoire tampon) est perdu.

Mot-clé associé: LOAD

## RUN

RUN [·line number·]

RUN 100

COMMANDÉ: Commence à exécuter le programme à la ligne indiquée ou à partir du début si aucune ligne n'est indiquée. Les dossiers sur cassette sont abandonnés et tout ce qui est emmagasiné dans les " buffets " (mémoire tampon) est perdu.

Mot-clé associé: LOAD

## SAVE

SAVE ·nom du dossier·[,·type de dossier·][,·paramètres binaires·]

SAVE "PROG",P

COMMANDÉ: Sauvegarde le programme en mémoire avec le nom ·nom du dossier·

,A sauvegarde un programme ASCII

,P protège le programme

,B sauvegarde le programme sous forme binaire

Mots-clés associés: LOAD, RUN ·nom du dossier·, MERGE, CHAIN, CHAIN  
MERGE

## SGN

SGN (·expression numérique·)

```
10 INPUT "Combien avez vous sur votre compte
bancaire";CASH
20 IF SGN(CASH) <1 GOTO 30 ELSE 40
30 PRINT "Mondieu, mondieu !":END
40 PRINT "Vous avez plus d'argent que moi"
```

FONCTION: détermine le signe d'une ·expression numérique·. Donne -1 si l'expression numérique est négative, 0 si elle est nulle et 1 si elle est positive.

Mot-clé associé: ABS

## SIN

**SIN** (·expression numérique·)

**PRINT SIN (PI/2)**

**1**

FONCTION: calcule la valeur réelle du SINus de l'expression numérique utilisant les radians sauf si on a précisé que l'on donnait des degrés avec la commande **DEG**.

Mots-clés associés: **COS, TAN, ATN, DEG, RAD**

## SOUND

**SOUND** ·canal·, ·période du ton·[, ·durée·[, ·volume·[, ·enveloppe de volume·[, ·enveloppe de ton·[, ·période du bruit·]]]]]

**SOUND 1,200,1000,7,0,0,1**

COMMANDÉ: C'est une des commandes les plus complexes du CPC464 et elle est étudiée en détail au chapitre 8.

Mots-clés associés: **ENV, ENT**

## SPACES\$

**SPACE\$** (·nombre entier·)

**SPACE\$(190)**

FONCTION: crée une chaîne d'espaces blancs d'une longueur donnée.

Mots-clés associés: **PRINT, SPC, TAB**

## SPEED INK

**SPEED INK** ·nombre entier·, ·nombre entier·

**5 INK 0,9,12:INK 1,0,26**

**10 BORDER 12,9**

**20 SPEED INK 50,20**

COMMANDÉ: les commandes **INK** et **BORDER** peuvent avoir deux couleurs associées intermittentes. Les temps sont mesurés en 50 èmes de seconde, le premier pour la première couleur, le deuxième pour la deuxième couleur.

Faites attention pour éviter des effets psychédéliques en choisissant les couleurs et les répétitions.

Mots-clés associés: **INK, BORDER**

## SPEED KEY

**SPEED KEY** ·attente· , ·période de répétition·

**SPEED KEY 20,3**

COMMANDÉ: Si on presse sans les lâcher les touches du CPC464 elles se répètent automatiquement après un délai d'attente. Les réglages se font au cinquantième de seconde de 1 à 255 avec une valeur par défaut de 10,10.

La commande **KEY DEF** permet de modifier ou d'établir une période de répétition.

## SPEED WRITE

**SPEED WRITE** ·nombre entier·

**SPEED WRITE 1**

COMMANDÉ: On peut écrire les données sur la cassette à 2000 Baud (nombre entier = 1) ou par défaut à 1000 Baud (nombre entier = 0). Quand on charge une cassette, le CPC464 établit automatiquement la vitesse correcte de lecture, et il n'est pas nécessaire que l'utilisateur le précise.

Quand on veut enregistrer sur des cassettes de qualité douteuse, il vaut mieux utiliser la vitesse 1000 Baud (**SPEED WRITE 0**). Voir chapitre 2.

Mot-clé associé: **SAVE**

## SQ

**SQ** (.canal sonore.)

```
10 MODE 1
20 FOR n=20 TO 0 STEP -1
30 PRINT n;
40 SOUND 1,10+n,100,7
50 WHILE SQ(1)>127:WEND
60 NEXT
```

FONCTION: On utilise cette fonction **SQ** pour vérifier le nombre de places libres dans la queue pour un canal donné, canal A étant numéro 1, B numéro 2 et C numéro 3. La fonction précise si le canal est actif, et s'il ne l'est pas, pourquoi on attend. On a le résultat par bit significatif.

- 0,1,2      nombre d'entrées libre dans la queue
- 3,4,5      état de Rendezvous au début de la queue
- 6            si la tête de la queue attend
- 7            si le canal est actif.

Mots-clés associés: **SOUND, ON SQ GOSUB**

## SQR

SQR (·expression numérique·)

```
PRINT SQR(9)  
3
```

FONCTION: calcule et donne la racine carrée d'un nombre.

Mot-clé associé: PRINT

## STOP

```
STOP  
300 IF n<0 THEN STOP
```

COMMANDÉ: pour stopper l'exécution d'un programme, mais laisse le BASIC dans un état où le programme peut repartir après la commande STOP. Utile si l'on veut voir les erreurs dans un long programme.

Mots-clés associés: CONT, END

## STR\$

STR\$ (·expression numériques·)

```
PRINT STR$(8766)  
1894  
PRINT STR$(84)  
84
```

FONCTION: change l'expression numérique en une représentation décimale de la même forme que celle utilisée dans la commande PRINT.

Mots-clés associés: VAL, PRINT, HEX\$, BIN\$

## STRING\$

STRING\$ (·nombre entier·, ·caractère·)

```
PRINT STRING$(13,"*")  
*****
```

FONCTION: affiche ou imprime une chaîne de caractères.

Mot-clé associé: SPACE\$

## **SYMBOL**

**SYMBOL** ·numéro du caractère· , ·liste de: rangées·

```
5 MODE 2
10 SYMBOL AFTER 90
20 SYMBOL 93,&80,&40,&20,&10,&88,&84,&82,&81
30 FOR n=1 TO 2000
40 PRINT CHR$(93);
50 NEXT
60 GOTO 60
```

**COMMANDÉ:** la commande **SYMBOL** redéfinit un caractère indiqué auparavant par la commande **SYMBOL AFTER**. Le caractère est choisi parmi les caractères ASCII ou ceux du CPC464 et les chiffres suivants définissent le nouveau caractère sur une matrice. Voir appendices II et III. Dans l'exemple, on a défini une barre diagonale qui s'affiche lorsqu'on appuie sur la touche **J**.

Mot-clé associé: **SYMBOL AFTER**

## **SYMBOL AFTER**

**SYMBOL AFTER** ·nombre entier·

**SYMBOL AFTER 90**

**COMMANDÉ:** le nombre de caractères que l'on peut redéfinir est fixé par la commande **SYMBOL AFTER**. Par défaut, c'est 240, ce qui donne 16 caractères redéfinissables. Si le nombre entier est 32, tous les caractères de 32 à 255 sont redéfinissables.

Après une commande **SYMBOL AFTER**, tous les caractères définis auparavant reprennent leur type d'origine.

Mot-clé associé: **SYMBOL**.

## TAG

```
TAG [#·numéro du canal·]  
10 MODE 2  
11 BORDER 9  
14 INK 0,12  
15 INK 1,0  
20 FOR n=1 TO 100  
30 MOVE 200+n,320+n  
40 TAG  
50 IF n<70 GOTO 60 ELSE 70  
60 PRINT "Hello";:GOTO 80  
70 PRINT "Adieu";  
80 NEXT  
90 GOTO 20
```

COMMANDÉ: du texte envoyé sur un canal peut être redirigé pour être écrit à la position graphique du curseur. Cela permet de mélanger textes et symboles avec des graphiques. Si on ne met pas le numéro du canal, cela va sur le canal 0. Le haut du caractère est attaché au curseur graphique, et les caractères de contrôle qui ne s'affichent pas d'habitude seront affichés si la commande **PRINT** n'est pas suivie d'un point-virgule.

Mot-clé associé: **TAGOFF**

## TAGOFF

```
TAGOFF [#·numéro du canal·]  
TAGOFF #0
```

COMMANDÉ: annule la commande **TAG** d'un canal particulier et renvoie le texte où était le curseur de texte avant la commande **TAG**.

Mot-clé associé: **TAG**

## TAN

```
PRINT TAN(45)
```

FONCTION: calcule les tangentes pour des angles exprimés en RADians, à moins que la fonction DEGré soit établie.

Mot-clés associés: **COS**, **SIN**, **ATN**, **DEG**, **RAD**

## TEST

```
TEST(· coordonnée x ·, · coordonnée y ·)  
PRINT TEST(300,300)
```

FONCTION: Donne la valeur de l'encre à un endroit spécifique de l'écran graphique en coordonnées absolues.

Mots-clés associés: TESTR, MOVE, MOVER, PLOT, PLOTR, DRAW, DRAWR

## TESTR

```
TESTR(·x relatif, y relatif ·)  
TESTR(5,5)
```

FONCTION: donne la valeur de l'encre en coordonnées relatives par rapport à la position présente du curseur.

Mot-clés associés: TEST, MOVE, MOVER, PLOT, PLOTR, DRAW, DRAWR.

## TIME

```
TIME  
10 DONNEE=INT(TIME/300)  
20 Battement=((TIME/300)-DONNEE)  
30 PRINT Battement  
40 GOTO 20
```

FONCTION: donne le temps écoulé depuis la mise en route, sans compter les temps de lecture et d'écriture sur cassette. L'unité de temps est 1/300 ème de seconde.

## TRON

## TROFF

```
TRON  
TROFF
```

COMMANDÉ: BASIC a la possibilité de vérifier l'exécution d'un programme en donnant le numéro de chaque ligne entre des parenthèses droites, avant l'exécution de la ligne. TRON met en marche, TROFF l'enlève.

Mot-clé associé: RUN

## **UNT**

```
UNT (adresse.)  
PRINT UNT(&FF66)
```

FONCTION: convertit un entier sans signe en un entier entre -32768 et +32767.

Mots-clés associés: INT, FIX, CINT, ROUND

## **UPPER\$**

```
UPPER$ (.chaîne alphanumérique.)  
PRINT UPPER$("amstrad")  
AMSTRAD
```

FONCTION: transforme les minuscules d'une chaîne alphanumérique en majuscules.

Mot-clé associé: LOWER\$

## **VAL**

```
VAL (.chaîne.)  
10 A$="7 est/mon chiffre porte-bonheur"  
20 PRINT VAL(A$)
```

FONCTION: extrait une expression numérique du début d'une chaîne alphanumérique. L'opposé de STR\$

Mot-clé associé: STR\$

## **VPOS**

```
VPOS (#.numéro du canal.)  
PRINT VPOS (#0)
```

FONCTION: donne la position verticale du curseur pour le canal demandé qui est obligatoire.

Mot-clé associé: POS

## **WAIT**

**WAIT** ·numéro de fiche· , ·masque· [, ·inversion·]  
**WAIT &FF34,20,25**

COMMANDÉ: suspend une opération jusqu'à ce qu'une fiche d'ENTREE-SORTIE ait une valeur entre 0 et 255. Le BASIC fait une boucle pendant qu'il lit cette fiche. Strictement pour les connaisseurs.

Mots-clés associés: **INP**, **OUT**

## **WEND**

**WEND**

```
10 MODE 1: REM Horloge BASIC
20 INPUT "Tapez l'heure, minute, et les secondes
(h,m,s)";heure,minute,seconde
30 CLS:datum=INT(TIME/300)
40 WHILE heure<13
50 WHILE minute<60
60 WHILE tic<60
70 tic=(INT(TIME/300)-datum)+seconde
80 LOCATE 70,4
90 PRINT #0,USING "##";heure,minute,tic
100 WEND
110 tic=0
115 seconde=0
120 minute=minute+1
130 GOTO 30
140 WEND
150 minute=0
160 heure=heure+1
170 WEND
180 heure=1
190 GOTO 40
```

COMMANDÉ: la boucle WHILE/WEND exécute une partie du programme sans interruption jusqu'à ce qu'une condition soit vérifiée. Dans l'exemple on voit l'élégance des programmes construits avec des boucles WHILE/WEND. Plusieurs détails peuvent être ajoutés à ce chronomètre. La commande WEND termine la boucle commencée par WHILE.

Mot-clé associé: **WHILE**

## **WHILE**

**WHILE** ·expression logique·

**WHILE JOUR<0**

Voir exemple à la commande précédente **WEND**

**COMMANDÉ:** Une boucle **WHILE** exécute une partie du programme jusqu'à ce qu'une condition soit vraie. La commande **WHILE** définit le début de la boucle et donne la condition. **WEND** termine la boucle.

Mot-clé associé: **WEND**

## **WIDTH**

**WIDTH** ·nombre entier·

**WIDTH 86**

**COMMANDÉ:** indique au BASIC la largeur de l'imprimante en nombre de caractères, pour que le BASIC puisse effectuer le retour de chariot quand c'est nécessaire.

Mots-clés associés: **PRINT, POS**

## **WINDOW**

**WINDOW** [# ·numéro du canal· , ·gauche· , ·droite· , ·haut· , ·bas·

```
10 MODE 1
20 BORDER 6
30 WINDOW 10,30,7,18
40 PAPER 2:PEN 3
50 CLS
60 PRINT CHR$(143);CHR$(242);c"est la position"
70 PRINT "1,1 dans la fenetre de texte"
80 GOTO 80
```

**COMMANDÉ:** établit une fenêtre (= window) de texte pour un canal donné de l'écran.

Mot-clé associé: **ORIGIN**

## **WINDOW SWAP**

**WINDOW SWAP** ·numéro du canal· , ·numéro du canal·  
**WINDOW SWAP** 0,2

**COMMANDÉ:** Echange le contenu de deux fenêtres. Par exemple les messages envoyés sur le canal 0 peuvent être envoyés sur un autre canal pour développer un programme.

Mots-clés associés: **WINDOW, PEN, PAPER, TAG**

## **WRITE**

**WRITE** [# ·numéro de canal· , ] ·liste à écrire·  
**WRITE** #2, "BONJOUR",4,5  
"BONJOUR",4,5

**COMMANDÉ:** affiche ou écrit des expressions ou valeurs sur un canal donné, en les séparant par une virgule et en mettant les chaînes entre deux guillemets.

Mot-clé associé: **PRINT**

## **XPOS**

**XPOS**  
**PRINT XPOS**

**FONCTION:** donne la position horizontale du curseur graphique.

Mots-clés associés: **YPOS, MOVE, MOVER, ORIGIN**

## **YPOS**

**YPOS**  
**PRINT YPOS**

**FONCTION:** donne la position verticale du curseur graphique.

Mots-clés associés: **XPOS, MOVE, MOVER, ORIGIN**

## **ZONE**

**ZONE** ·nombre entier·

```
10 Print 1,2,3  
20 ZONE 19  
30 PRINT 4,5,6
```

**COMMANDÉ:** change la largeur de la zone d'affichage ou d'impression utilisée avec **PRINT**, partant de 13 par défaut pour lui donner une valeur de 1 à 255. C'est remis à zéro par les commandes **NEW**, **LOAD**, **CHAIN** et **RUN** ·nom de dossier·

Mots-clés associés: **PRINT**, **WIDTH**

# 9 Et pour quelques programmes de plus...

Sujets abordés dans ce chapitre

- \* Retour à la position légale du curseur
- \* Caractères de contrôle
- \* Le système d'exploitation de la Machine (MOS)
- \* Les interrupteurs.

## 9.1 La place du curseur et les extensions du code de contrôle.

Dans un certain nombre de progiciels (ou programmes d'application), le curseur de texte peut être en dehors de la fenêtre utilisée. Plusieurs opérations peuvent ramener le curseur à sa position légale (= normale = là où il devrait être), avant d'être effectuées, et ce sont:

- écrire un caractère
- dessiner le bloc du curseur
- obéir au codes de contrôle marqué d'un astérisque dans les pages suivantes.

La procédure qui ramène le curseur à une position légale est la suivante:

- a) si le curseur est complètement à droite, il est déplacé complètement à gauche de la ligne suivante
- b) s'il est à gauche du bord gauche, il repart à droite sur la ligne au dessus.
- c) si le curseur est au-dessus du bord supérieur, alors la fenêtre fait descendre tout d'une ligne et le curseur se place sur la première ligne de la fenêtre.
- d) si le curseur est en-dessous du bord inférieur, la fenêtre fait tout monter d'une ligne et le curseur se place sur la dernière ligne.

Les tests et opérations se font dans l'ordre indiqué.

Les valeurs de caractère de 0 à 31 (voir appendice III) ne produisent rien sur l'écran (et peuvent coincer l'ordinateur si on les utilise sans ménagement). ent) mais sont interprétées comme codes de contrôle (CONTROL CODES). Certains codes changent le sens de un ou plusieurs caractères suivants, qui sont les paramètres du code.

Les caractères de contrôle envoyés sur l'écran graphique vont s'afficher (voir TAG au chapitre 8) et des caractères comme (&07"BEL") vont afficher un symbole précis en rapport avec leur fonction si on les appelle à partir du clavier (ex: [CTRL] G) mais n'exécuteront la commande correspondante que si on les appelle avec la forme PRINT CHR\$(&07).

Les codes avec un astérisque \* forcent le curseur à aller vers une position légale avant d'être exécutés, mais peuvent laisser le curseur dans une position non légale. Les codes sont référencés d'abord avec leur valeur HEX (&XX) puis l'équivalent décimal.

### **Commandes de contrôle des caractères supplémentaires: normalement inaccessibles par la touche CTRL du clavier.**

VALEUR	Nom	Paramètre	Signification
&00	0	NUL	Pas d'effet. Ignoré
&01	1	SOH	0..255 Affiche le symbole donné par la valeur du paramètre. Ceci permet aux symboles de 0 à 31 d'être affichés
&02	2	STX	Supprime le curseur de texte
&03	3	ETX	Met le curseur de texte. Le BASIC a son propre interrupteur de curseur qui dépend de la frappe au clavier.
&04	4	EOT	0..2 Etablit le Mode écran: équivalent à une commande MODE
&05	5	ENQ	0..255 Envoie le caractère paramètre sur le curseur graphique
&06	6	ACK	Etablit l'écran de textes (voir &15, NAK)
&07	7	BEL	Fait sonner la clochette. (Vide les queues sonores)
&08	8 *	BS	Curseur vers la gauche
&09	9 *	TAB	Curseur vers la droite
&0A	10 *	LF	Curseur vers le bas
&0B	11 *	VT	Curseur vers le haut

&0C	12	FF		Efface l'écran de texte et met le curseur en haut à gauche; équivalent à une commande CLS.
&0d	13 *	CR		Passage à la ligne suivante
&0E	14	SO	0..15	Etablit l'encre du Papier = Commande PAPER
&0F	15	SI	0..15	Etablit l'encre du stylo = Commande PEN
&10	16 *	DLE		Efface le caractère. Remplit la place du caractère avec l'encre du Papier.
&11	17 *	DC1		Efface depuis le bord gauche de la fenêtre jusqu'au caractère présent. Rempli avec l'encre du Papier.
&12	18 *	DC2		Efface à partir de,et y compris le dernier caractère jusqu'au bord droit de la fenêtre.
&13	19 *	DC3		Efface depuis le début de la fenêtre jusqu'au caractère présent. Rempli avec l'encre du Papier.
&14	20 *	DC4		Efface à partir du caractère jusqu'à la fin de la fenêtre. Remplissage avec l'encre du papier.
&15	21	NAK		Arrête l'écran de texte. Rien ne s'affichera jusqu'à ce que un signal ACK (&06) ait été envoyé.
&16	22	SYN	0..1	0 supprime le mode transparent 1 établit le mode transparent
& 17	23	ETB	0..3	0 Met le Mode d'Encre Graphique NORMAL 1 Met le Mode d'Encre Graphique XOR 2 Met le Mode d'Encre Graphique AND 3 Met le Mode d'Encre Graphique OR

&18	24	CAN		Echange des couleurs PAPER et PEN
&19	25	EM	0..255 0..255 0..255 0..255 0..255 0..255 0..255 0..255 0..255	Equivalent à la commande SYMBOL.Neuf paramètres: le premier pour le numéro du caractère, les suivants correspondent chacun à une ligne de la matrice des caractères.
&1A	26	SUB	1..80 1..80 1..25 1..25	Etablit la fenêtre = commande WINDOW. Les deux premiers paramètres définissent les bords gauche et droit, les deux derniers le haut et le bas.
&1B	27	ESC		Pas d'effet. Ignoré
&1C	28	FS	0..15 0..31 0..31	Attribue deux couleurs à une encre. Equivalent à une commande INK. Le premier paramètre définit le numéro de l'encre, les suivants les couleurs.
&1D	29	GS	0..31 0..32	Attribue deux couleurs à la Bordure(BORDER)
&1E	30	RS		Déplace le curseur vers le coin en haut à gauche de la fenêtre.
&1F	31	US	1..80 1..25	Déplace le curseur à une position donnée dans la fenêtre. Equivalent à une commande LOCATE.

## 9.2 Système d'Exploitation de la Machine

(M.O.S = machine operating system)

L'organisation interne du CPC464 est assurée par un système d'exploitation sophistiqué travaillant en temps réel. Le système d'exploitation dirige le trafic dans l'ordinateur depuis les entrées jusqu'aux sorties.

Il sert principalement de lien entre la machine et l'interpréteur du BASIC -par exemple les couleurs qui flashent ou le BASIC passe les paramètres et le système d'exploitation organise le travail, d'un côté ce qu'il faut faire, de l'autre quand il faut le faire.

Le système d'exploitation fait partie du logiciel intégré à la machine (que les anglais appellent firmware, c'est à dire à la fois du hardware et du software) et comprend les routines en code machine qui sont appelées par le BASIC.

Bien qu'il soit (presque) impossible de coincer la machine avec du BASIC autrement que par une commande **ON BREAK GOSUB** mal placée, il est assez facile de causer des problèmes qui ne peuvent être résolus que par une remise à zéro complète (et perte de vos programmes et données) quand on cherche à modifier le système d'exploitation.

Si vous voulez faire **POKE** dans la machine ou **CALL** pour des sous-programmes, sauvegardez vos programmes et **LIS**tings avant de le faire. Ce logiciel intégré est décrit dans le manuel pour utilisateurs chevronnés et va au delà de ce guide.

## 9.3 L'assembleur AMSOFT

Pour les mordus du code-machine, il faut un assembleur. L'assembleur d'AMSOFT comprend un assembler Z80 adaptable, avec un éditeur, un disassembleur et un moniteur.



# 10 Interruptions

Sujets abordés dans ce chapitre

- \* interruptions en temps réel
- \* AFTER
- \* EVERY
- \* REMAIN
- \* le chronomètre principal

Si vous ne l'avez pas déjà remarqué, une des innovations majeures dans le logiciel du CPC464 est sa capacité à faire des interruptions directement à partir du BASIC, ce qui veut dire que le BASIC d'AMSTRAD peut faire un certain nombre d'opérations simultanées mais séparées dans un programme.

On appelle parfois cette capacité 'multitâche' et c'est introduit par l'utilisation de deux nouvelles commandes, AFTER et EVERY.

Cette capacité est clairement démontrée dans la manière dont les sons sont traités avec le système des queues et des Rendezvous.

Tous les aspects concernant le temps sont contrôlés par le chronomètre principal (la grande horloge si vous préférez) qui est une sorte de montre à quartz qui s'occupe des chronométrages et de la synchronisation. Quand une fonction dans la machine dépend du temps, on est sûr qu'elle passe par le chronomètre principal.

Le résultat dans le système logiciel donne les commandes EVERY et AFTER:

AFTER = après qu'un temps ce soit écoulé, il faut faire une action particulière  
EVERY = tous les x secondes, il faut faire une action particulière.

## 10.1 AFTER

Le CPC464 a donc son horloge interne. La commande AFTER permet d'avoir dans un programme BASIC des sous-programmes appelés dans le futur. Quatre chronomètres sont disponibles, chacun pouvant avoir son sous-programme associé.

AFTER ·nombre entier·[, ·nombre entier·] GOSUB ·numéro de ligne·

Le premier ·nombre entier· précise dans combien de temps il faut appeler le sous-programme. Le temps est mesuré en cinquantièmes de seconde. Le deuxième nombre entier (de 0 à 3) indique quel chronomètre doit être utilisé. S'il n'y a pas de nombre précisé, 0 est supposé.

Quand le temps écoulé est passé, le sous-programme est appelé automatiquement, comme si un GOSUB avait été mis dans le programme à cet endroit là. Quand le sous programme est terminé avec la commande habituelle RETURN, le programme principal continue là où il avait été interrompu.

Les chronomètres ont des priorités différentes. Chronomètre 3 a la priorité la plus haute, chronomètre 0 la priorité la plus basse.

Les commandes AFTER peuvent être données à tout moment, rétablissant le sous-programme et le temps associé à l'un des chronomètres. Ces chronomètres sont les mêmes que ceux utilisés par la commande EVERY, ainsi un AFTER remplace un EVERY précédent s'ils ont le même chronomètre et vice versa.

Voici un programme simple pour illustrer cette commande:

```
10 MODE 1:X=0
20 AFTER 45 GOSUB 100
30 AFTER 100,1 GOSUB 200
40 PRINT "AMSOFT"
50 WHILE X<100
60 LOCATE #1,30,1:PRINT #1,X:X=X+1
70 WEND
80 END
100 PRINT "peripheriques"
110 RETURN
220 PRINT "et logiciels"
210 RETURN
```

Notez l'utilisation de deux fenêtres (WINDOW) pour avoir un affichage différent dans le programme principal (50-80) et les sous-programmes appellés par AFTER

## 10.2 EVERY

La commande EVERY est très semblable à la commande AFTER. La seule différence est qu'avec EVERY, on va effectuer le sous-programme régulièrement, toutes les 3 secondes par exemple, ou tous les 10 èmes de seconde suivant le chiffre indiqué après la commande EVERY.

L'exemple suivant utilise les commandes DI et EI (arrête l'interrupteur et remet l'interrupteur):

```
10 MODE 1:X=0
20 P100=0:EVERY 10 GOSUB 10030 P200=0:EVERY 12,1
   GOSUB 200
40 PRINT "AMSOFT"
50 WHILE X<200
60 LOCATE #1,30,1:PRINT #1,X:X=X+1
70 WEND
80 LOCATE 1,20:END
100 DI:PEN P100:LOCATE 1,2:PRINT "peripheriques":
   EI
105 IF P100=0 THEN P100=1 ELSE P100=0
110 RETURN
200 PEN P200:LOCATE 1,3:PRINT "et logiciels"
205 IF P200=2 THEN P200=3 ELSE P200=2
210 RETURN
```

Comme vous pouvez le constater, les commandes DI et EI permettent aux sous-programmes d'être exécutés en entiers sans être interrompus par un autre EVERY au milieu du sous-programme.

### 10.3 REMAIN

Cette fonction interroge pour savoir combien de temps il reste sur un chronomètre avant qu'il ne démarre un sous-programme. Elle arrête le chrono et indique zéro s'il est déjà arrêté. La commande prend la forme:

**REMAIN (·nombre entier·)**

Ce nombre pouvant être 0,1,2 ou 3 c'est à dire les numéros des chronomètres.



# Appendice I

## *L'Art du possible*

Les débutants en informatique commencent souvent par établir des points de référence qui aident à comprendre les possibilités et les limites de l'ordinateur. Cette partie du livre essaie d'être un guide général sur ce qui se passe dans un ordinateur et pourquoi.

### **Pan sur les envahisseurs de l'espace!**

Même si la seule raison de votre achat du CPC464 est de pouvoir jouer à des jeux vidéos sophistiqués sur cette machine, vous demandez ce qu'il y a dans ce 'matériel', pris souvent par opposition à progiciel.

Certaines actions de l'ordinateur dépendent du matériel, et c'est au logiciel d'utiliser les capacités du matériel pour produire des textes et des graphiques sur l'écran.

Le matériel dirige le canon à électrons à l'intérieur du tube du Moniteur, le logiciel y met de l'ordre et de l'intelligence pour distribuer et contrôler les séquences qu'on voit sur l'écran: comment le vaisseau spatial décolle, ou comment une lettre est recomposée devant vous avec une marge différente.

### **Question: Pourquoi un ordinateur est-il meilleur qu'un autre?**

Le matériel sans logiciel ne vaut rien. La réciproque est aussi vraie, la valeur vient de l'ensemble intégré qui permet de réaliser différentes tâches. Il y a plusieurs éléments qui permettent de juger les performances aux niveaux logiciels et matériels.

Les références généralement acceptées pour les ordinateurs personnels sont les suivantes:

1 La résolution de l'écran: le plus petit élément visible sur l'écran

C'est une combinaison d'éléments différents, le nombre de couleurs disponibles pour le programmeur, le nombre de pixels (petits points), le nombre de caractères de texte définissable sur l'écran, et vous verrez que le CPC a des caractéristiques très favorables dans ce domaine.

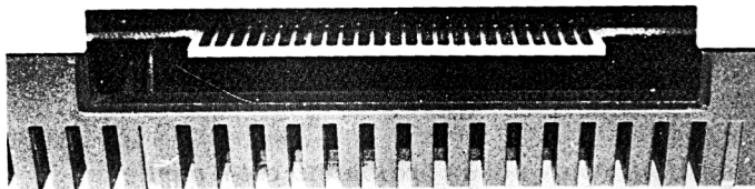
## 2 L'interpréteur BASIC

Presque tous les ordinateurs familiaux possèdent un interpréteur en BASIC qui permettent d'écrire rapidement des programmes. Le BASIC intégré à la machine est lui-même un programme, mais un programme très élaboré et compliqué qui a nécessité des millions d'heures de travail pour sa mise au point, depuis son 'invention' aux USA. Ce 'Beginners All Purpose Symbolic Instruction Code' (= langage symbolique tous usages pour débutants) est le langage le plus utilisé de l'informatique personnelle et comme tous les langages, il a beaucoup de dialectes.

La version du CPC464 est une des plus développées des dialectes du BASIC et peut utiliser la plupart des progiciels écrits pour le système d'exploitation pour disquette appelé CPM. C'est une variante très rapide du BASIC, et même si vous pensez qu'il n'y a pas grande différence entre 0,05 secondes et 0,125 secondes, quand il y a des milliers d'instructions à effectuer, l'un sera presque trois fois plus rapide que l'autre.

Vous entendrez souvent parler de code machine: c'est le seul langage que la machine comprend directement; il va très vite mais il est très long à programmer, c'est pourquoi on utilise le BASIC avec un interprète qui traduit en code-machine.

Le BASIC dans votre Amstrad est un des plus rapides et les plus complets qu'on puisse trouver sur un micro familial et a des capacités performantes qui aident le programmeur averti à accélérer certaines lenteurs inhérentes à tout langage de haut niveau comme le BASIc, et permet des effets graphiques et sonores surprenants.



## 3 Expansion

La plupart des ordinateurs font attention à prévoir des extensions de matériels, imprimantes, joysticks, lecteurs de disquettes. C'est donc un paradoxe de voir tant de micro-ordinateurs familiaux où vous devez acheter des interfaces d'expansion avant de pouvoir branchez des choses aussi simples qu'une imprimante ou un joystick.

L'acheteur ne pense pas toujours au développement de ses besoins, car une machine qui possède déjà une interface pour une imprimante (type Centronics) et une interface pour manette de jeux est plus économique à long terme.

LE CPC464 possède d'origine une interface Centronics, une interface pour deux joysticks, une sortie pour son stéréo, et un bus d'expansion pour ajouter des lecteurs de disquettes, des cartouches ROM, une interface série (RS232) etc.

Un ROM (mémoire morte = Read Only Memory) est un circuit intégré qui contient des programmes. Le BASIC de votre machine est dans un ROM et on peut avoir des programmes additifs ou de remplacement dans un autre ROM.

La cartouche est un circuit ROM dans un boîtier plastique avec une fiche de branchement pour qu'on le mette ou l'enlève facilement. Il a le même rôle qu'un programme sur cassette, mais il est disponible immédiatement sans avoir à attendre comme pour une cassette, et c'est vraiment plus pratique.

Mais ce qui est dans le ROM est fixé et on ne peut pas le modifier comme un programme sur cassette (d'où le nom mémoire morte, quoique le mot mémoire permanente serait plus indiqué!)

Expansion est un mot qui indique que votre ordinateur peut se développer et bénéficier des progrès des logiciels et des périphériques. Le CPC464 possède des possibilités d'expansion considérables prévues à la conception de la machine.

#### 4 Les effets sonores

Les caractéristiques sonores d'un micro varient du système à faire des bruits plus ou moins cacophoniques à une reproduction électronique de la musique. Le CPC464 utilise 3 canaux et 8 octaves pour produire les sons, qui produit une musique de bonne qualité avec contrôle total de l'amplitude et des enveloppes de ton. De plus, le son est vraiment stéréo avec un canal vers la gauche, un vers la droite et un au milieu.

Cela donne des performances qui peuvent se traduire par des effets sonores impressionnantes, surtout quand ils vont de pair avec des graphiques synchronisés.

Finalement, c'est à vous de voir ce qui est le plus important pour vous. Nous sommes sûrs que vous y trouverez des qualités que nous n'imaginions même pas.

### **Pourquoi un micro ne peut-il pas faire cela?**

Souvent les gens se demandent pourquoi un ordinateur ne peut faire des images comme on en voit sur la télévision.

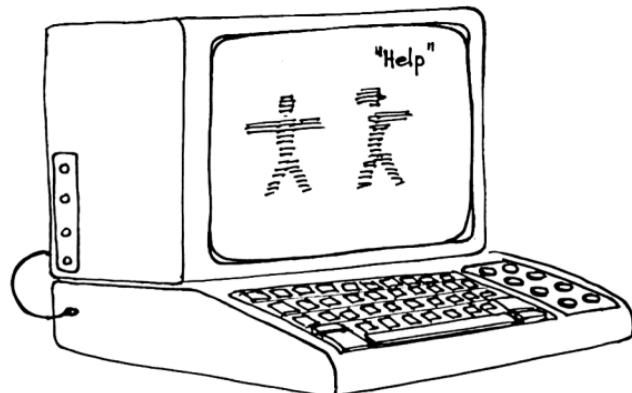
**Question: Pourquoi, par exemple, un micro ne peut-il animer un dessin comme dans un dessin animé ou un film, pourquoi les personnages sont-ils toujours simplifiés.**

La réponse est simple et complexe. La réponse simple est que l'écran d'un ordinateur est beaucoup plus simple qu'un écran télé, et que même s'il utilise un poste télé, il n'utilise qu'une partie des circuits.

Ce qu'on voit sur un image retransmise à la télé est parfois 20 fois plus important que ce que montre un ordinateur, en terme de mémoire nécessaire.

De plus, pour être animé, tous les points de cette image doivent changer 50 fois par seconde. On peut le faire avec un ordinateur mais sa mémoire et son prix sont exorbitants.

Les micros doivent se débrouiller avec la mémoire relativement petite comparée aux mega-ordinateurs et il en résulte des mouvements plus saccadés et une résolution d'écran moins élevée. On est encore assez loin des dessins animés comme on en voit à la télévision, mais on y vient.



### **Question. Pourquoi ne peut-on simplement s'asseoir devant un micro et taper une page de texte sans la machine?**

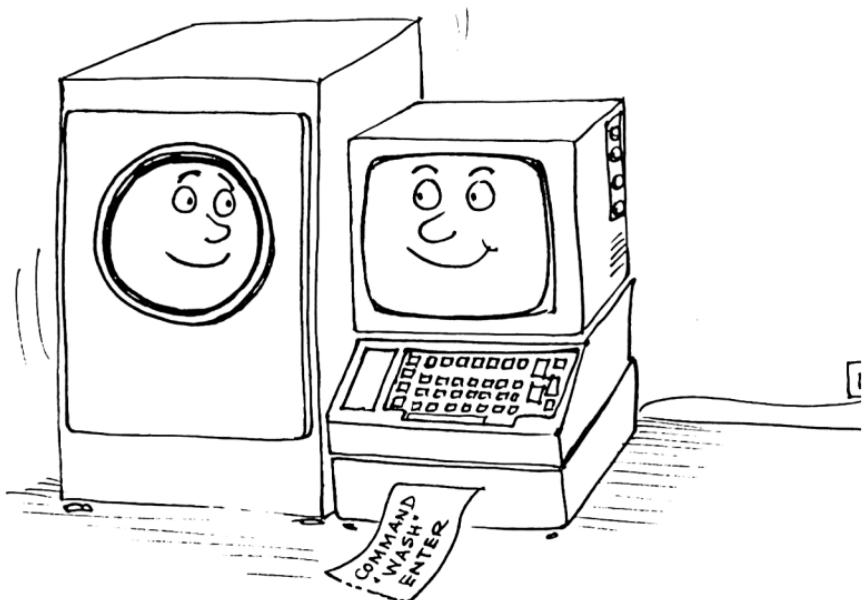
Ne pas se laisser tromper par le fait qu'un ordinateur ressemble à une machine à écrire avec un écran; c'est un appareil qui communique avec vous par l'intermédiaire d'un langage de programmation. Et à moins que vous n'ayez un progiciel de traitement de textes, il vous répondra par le message

#### **Syntax Error**

Si vous tapez n'importe quelle phrase et faites [**ENTER**].

Mais si vous avez mis en mémoire un programme de traitement de textes, alors vous pouvez taper une phrase et [**ENTER**] et l'ordinateur va se comporter presque comme une machine à écrire. Mais auparavant vous aviez chargé un programme qui disait à votre ordinateur: 'nous allons taper une lettre'.

L'ordinateur "semble" combiner plusieurs appareils familiers, un écran comme la TV, un clavier comme une machine à écrire, un datacorder comme une magnétocassette, mais vous devez vous rappeler que les ressemblances sont superficielles et que l'ordinateur a des capacités et un usage complètement différents qui lui donnent sa personnalité!





# Appendice II

## *Introduction à l'informatique*

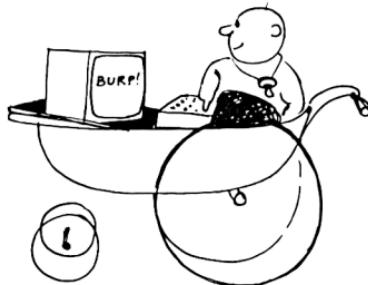
### **Mais qui a peur du Jargon Informatique?**

Comme beaucoup d'industries spécialisées, l'informatique a développé son jargon, un moyen rapide de parler de choses compliquées qui demanderaient beaucoup d'explications en langage courant. Ce n'est pas particulier à la haute technologie, tous les corps de métiers utilisent plus ou moins une forme de jargon qui permet de parler des choses du métier de la façon la plus concise possible.

Une des différences par exemple avec le jargon légal est que la confusion vient non pas de la façon dont les mots sont utilisés mais des mots eux-mêmes. Et en plus, comme beaucoup d'entre eux ont été adaptés de l'anglais ou l'américain, cela n'arrange rien.

N'oubliez pas que dans l'informatique, il faut essayer de décrire des concepts nouveaux avec des mots anciens. Le mot éditer prend un sens particulier en informatique, il signifie plus modifier un texte ou ce qu'on voit sur l'écran que faire le travail de Monsieur Hachette.

L'informatique est comprise rapidement par les jeunes qui apprécient la précision et la simplicité des idées, et on voit plus de programmeurs de 10 ans que d'avocats du même âge.



## **BASIC de Base**

Pratiquement tous les micro-ordinateurs familiaux utilisent le langage appelé BASIC une abréviation d'une phrase anglaise signifiant langage symbolique pour les débutants, (Beginners All-Purpose Symbolic Instruction Code pour les puristes Anglo-saxons) mais depuis quelque temps il est devenu un langage performant pas si simple que son nom l'indique.

Mais son nom reste synonyme de facilité et attire les débutants de l'informatique et il est le langage le plus utilisé pour l'informatique personnelle, du simple programme à des programmes de statistiques élaborés.

### **Basics:**

Le BASIC est un langage interactif qui interprète un certain nombre de commandes bien définies puis fait des opérations sur les données qu'on lui a fournies pendant le programme. Contrairement aux langages humains avec un langage de 5000-8000 mots, BASIC doit se débrouiller avec environ 200 mots. Les programmes écrits en BASIC doivent suivre des règles simples et rigides, avec une syntaxe qu'on doit respecter sous peine d'avoir le message:

#### **Syntax error**

Ce n'est pas aussi draconien que cela paraît, c'est un moyen d'indiquer que l'ordinateur ne comprend pas ce que vous voulez dire, et c'est souvent à cause d'une virgule ou de guillemets mal placés, ou encore du O que vous avez mis à la place du 0 (zéro). Et c'est là qu'on voit à la fois la limite des ordinateurs et pourquoi il faut être précis avec eux. Vous voyez la différence entre 0 et zéro 0 à cause du contexte, mais c'est exactement le même symbole quand on l'écrit et l'ordinateur a besoin qu'on lui indique si on veut la lettre ou le chiffre.

### **Des chiffres mais pas de lettres**

Si vous avez les œuvres complètes de Victor Hugo sur un ordinateur, vous ne trouverez pas une seule lettre ou un seul mot dans le système: toute donnée, tout renseignement est d'abord transformé en chiffres et en nombres avant que l'ordinateur soit capable de manipuler ces données et de les placer.

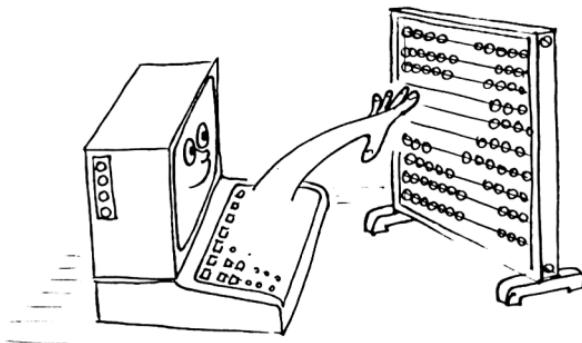
Le BASIC interprète les mots comme des nombres que l'ordinateur peut manipuler avec des additions et soustractions et l'algèbre de BOOLE qui permet de comparer des données (tables de vérité, oui-non etc.), de savoir si deux conditions sont remplies en même temps (et, ou inclusif, ou exclusif).

Avec un programme l'ordinateur divise toute tâche en une infinité d'opérations oui-non (0 ou 1, 0 il n'y a pas de courant électrique, 1 il y en a, c'est comme si l'ordinateur manipulait des milliers de petits interrupteurs pour arriver à son résultat.)

Un ordinateur ne sait même pas multiplier directement: pour multiplier 35 par 10 (35\*10, car c'est le signe de la multiplication pour l'ordinateur, il ne reconnaît x que comme une lettre ) il ajoute 35 dix fois de suite.

Une partie du microprocesseur qu'on appelle CPU (= central processing unit, les informaticiens adorent les initiales) contient l'information 35 sous la forme binaire 100011. Chaque fois qu'il ajoute 35, un autre endroit du CPU qui a le nombre 10 (1010 en binaire) diminue d'une unité jusqu'à ce qu'il arrive à zéro et envoie le nombre 350 (trop long à écrire en binaire) à une autre partie du CPU pour qu'il l'envoie sur la sortie.

Un peu compliqué, n'est-ce pas?



Mais c'est là la base du fonctionnement d'un ordinateur, qui est simplement une machine qui fait des tâches répétitives très, très vite et avec une grande précision.

Le procédé qui consiste à passer d'un état 0 à l'état 1 est souvent appelé digital par opposition à analogue où les choses varient d'une manière continue.

Mais dans un monde où il n'y a que deux chiffres 0 et 1, comment allons nous compter au delà?

## Bits et Octets

Nous sommes habitués au système décimal, ou système à base 10, qui utilise 10 chiffres de 0 à 9. Le système qui utilise les 2 chiffres 0 et 1, s'appelle le système binaire, qu l'ordinateur utilise, et les unités où le système binaire opère s'appellent des bits.

Pour convertir un nombre décimal en nombre binaire (nombre en base 2) il faut partir des puissances de 2

$$2^0 = 1$$

$$2^1 = 2 = 2(2^0)$$

$$2^2 = 4 = 2 \times 2 = 2(2^1)$$

$$2^3 = 8 = 2 \times 2 \times 2 = 2(2^2)$$

$$2^4 = 16 = 2 \times 2 \times 2 \times 2 = 2(2^3)$$

$$2^5 = 32 = 2 \times 2 \times 2 \times 2 \times 2 = 2(2^4)$$

et ainsi de suite

Si on veut représenter un nombre en base 2 (avec des zéros et des 1 seulement) les chiffres peuvent être considérés comme des indicateurs pour savoir si une puissance de 2 est présente ou non:

$2^5$	$2^4$	$2^3$	$2^2$	$2^1$	$2^0$	
0	1	0	0	1	1	
(0	16	0	0	2	1)	= 19
0	0	0	1	1	1	
(0	0	0	4	2	1)	= 7
1	1	1	1	1	1	
(32	16	8	4	2	1)	= 63

Pour pouvoir manipuler les bits, on les groupe par ensemble de 8, ce qui s'appelle un octet. Le plus grand nombre qu'on peut écrire avec un octet est

11111111 (binaire) = 255 (décimal)

ce qui fait avec le nombre 00000000, 256 nombres qui correspondent bien à  $2^8 = 256$

Les ordinateurs manipulent les nombres avec les octets. 256 n'est pas un nombre trop grand et pour faciliter les manipulations, 2 octets sont utilisés dans une matrice avec une adresse verticale et une adresse horizontale qui permet de positionner chaque élément de la matrice.

0	1	2	3	4	5	6	7	8
1								
2								
3								
4								
5								
6								
7								
8								

Cette matrice peut placer (8x8) = 64 éléments, les éléments placés avec un 1 étant reconnus comme 5,3 et 2,4.

Ainsi une matrice binaire 256x256 peut manipuler 65536 cases différentes en utilisant un octet pour l'adresse verticale et un octet pour l'adresse horizontale.

Pour raccourcir, on utilise le Kilo octet (K ou KO) qui est égal à 1024 (ou, le kilo ne vaut pas 1000 en informatique), et cela explique pourquoi un ordinateur comme le CPC464 est appelé 64K car il a 65536 octets (64 x 1024) dans sa mémoire.

Heureusement, le BASIC fait ces conversions pour vous, et on peut devenir un très bon programmeur sans trop connaître le binaire. Mais cela ne peut que vous aider à comprendre pourquoi certains nombres ont plus d'importance que d'autres en informatique.

## Cependant

Bien que simple et élégant, le système binaire est difficile à lire car les nombres deviennent un peu longs et ne sautent pas aux yeux.

On a donc choisi de recourir à un système de notation en base 16, car c'est une puissance de 2 et permet de traduire les nombres binaires d'une manière plus élégante. C'est le système hexadécimal, HEX pour faire plus court.

### Décimal

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	10

Rappelez-vous, la base d'un système s'écrit toujours 10 dans ce système.

Le système hexadécimal peut décomposer les 8 bits d'un octet en deux blocs de quatre bits, puisque 15 est un nombre à quatre bit: 1111 en binaire.

Considérons la table avec binaire, décimal et hexadécimal

Décimal	Binaire	Hexadécimal
0	0	0
1	1	1
2	10	2
3	11	3
4	100	4
5	101	5
6	110	6
7	111	7
8	1000	8
9	1001	9
10	1010	A
11	1111	B
12	1100	C
13	1101	D
14	1110	E
15	1111	F
16	10000	10

Un nombre à 8 bits peut être décomposé en deux nombres de 4 bits (appelés nibbles en anglais, ce qu'on peut traduire par petit bouchée...)

exemple 11010110 s'écrit Hex D6

Dans ce livre, les nombres hexadécimaux sont désignés avec le symbole &, &D6 par exemple et sont utilisés par les programmeurs qui utilisent le langage assembleur, à mi-chemin entre le code machine et le BASIC.

Quand vous utilisez l'hexadécimal, il faut faire attention à multiplier le premier chiffre par 16 et ajouter le deuxième. Ainsi &D6 donne  $(13 \times 16) + 6 = 214$  et non pas 13 et 6 ou 136.

C'est le même procédé que vous faites avec le système decimal, mais il est plus facile de multiplier par 10 que par 16.

Si vous avez tout compris jusque là, bravo, vous saurez rapidement utiliser le CPC464 pour quantité de programmes. Si cela vous paraît compliqué, choisissez un livre d'initiation qui vous expliquera plus en détail ce que nous ne pouvons qu'aborder dans ce livre.



# Appendice III

## Les caractères ASCII, et les caractères graphiques du CPC464

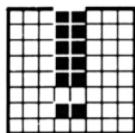
### III.1 ASCII

Comme référence utile, nous reproduisons ici le jeu de caractères standard ASCII avec la notation décimale, octale et hexadécimale avec le code ASCII quand il s'applique.

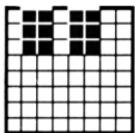
DEC	OCTAL	HEX	ASCII characters	DEC	OCTAL	HEX	ASCII	DEC	OCTAL	HEX	ASCII
0	000	00	NUL ([CTRL]@)	50	062	32	2	100	144	64	d
1	001	01	SOH ([CTRL]A)	51	063	33	3	101	145	65	e
2	002	02	STX ([CTRL]B)	52	064	34	4	102	146	66	f
3	003	03	ETX ([CTRL]C)	53	065	35	5	103	147	67	g
4	004	04	EOT ([CTRL]D)	54	066	36	6	104	150	68	h
5	005	05	ENQ ([CTRL]E)	55	067	37	7	105	151	69	i
6	006	06	ACK ([CTRL]F)	56	070	38	8	106	152	6A	j
7	007	07	BEL ([CTRL]G)	57	071	39	9	107	153	6B	k
8	010	08	BS ([CTRL]H)	58	072	3A	:	108	154	6C	l
9	011	09	HT ([CTRL]I)	59	073	3B	:	109	155	6D	m
10	012	0A	LF ([CTRL]J)	60	074	3C	<	110	156	6E	n
11	013	0B	VT ([CTRL]K)	61	075	3D	=	111	157	6F	o
12	014	0C	FF ([CTRL]L)	62	076	3E	>	112	160	70	p
13	015	0D	CR ([CTRL]M)	63	077	3F	?	113	161	71	q
14	016	0E	OE ([CTRL]N)	64	100	40	@	114	162	72	r
15	017	0F	SI ([CTRL]O)	65	101	41	A	115	163	73	s
16	020	10	DLE ([CTRL]P)	66	102	42	B	116	164	74	t
17	021	11	DC1 ([CTRL]Q)	67	103	43	C	117	165	75	u
18	022	12	DC2 ([CTRL]R)	68	104	44	D	118	166	76	v
19	023	13	DC3 ([CTRL]S)	69	105	45	E	119	167	77	w
20	024	14	DC4 ([CTRL]T)	70	106	46	F	120	170	78	x
21	025	15	NAK ([CTRL]U)	71	107	47	G	121	171	79	y
22	026	16	SYN ([CTRL]V)	72	110	48	H	122	172	7A	z
23	027	17	ETB ([CTRL]W)	73	111	49	I	123	173	7B	{
24	030	18	CAN ([CTRL]X)	74	112	4A	J	124	174	7C	-
25	031	19	EM ([CTRL]Y)	75	113	4B	K	125	175	7D	}
26	032	1A	SUB ([CTRL]Z)	76	114	4C	L	126	176	7E	-
27	033	1B	ESC	77	115	4D	M				
28	034	1C	FS	78	116	4E	N				
29	035	1D	GS	79	117	4F	O				
30	036	1E	RS	80	120	50	P				
31	037	1F	US	81	121	51	Q				
32	040	20	SP	82	122	52	R				
33	041	21	!	83	123	53	S				
34	042	22	"	84	124	54	T				
35	043	23	#	85	125	55	U				
36	044	24	\$	86	126	56	V				
37	045	25	%	87	127	57	W				
38	046	26	&	88	130	58	X				
39	047	27	'	89	131	59	Y				
40	050	28	(	90	132	5A	Z				
41	051	29	)	91	133	5B	C				
42	052	2A	*	92	134	5C	\				
43	053	2B	+	93	135	5D	J				
44	054	2C	,	94	136	5E	†				
45	055	2D	-	95	137	5F	-				
46	056	2E	.	96	140	60	.				
47	057	2F	/	97	141	61	a				
48	060	30	Ø	98	142	62	b				
49	061	31	1	99	143	63	c				

## III.2 Le jeu de caractères particulier du CPC464

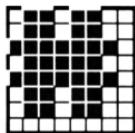
Les caractères représentés ici sont montrés avec la matrice standard 8x8 utilisée pour l'affichage sur l'écran du CPC464. Des caractères définis par l'utilisateur peuvent être groupés pour des effets spéciaux et mis bout à bout-voir la description de la commande SYMBOL au chapitre 8.



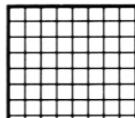
33  
&H21  
&X00100001



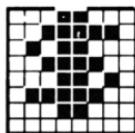
34  
&H22  
&X00100010



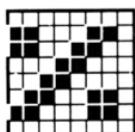
35  
&H23  
&X00100011



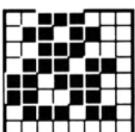
32  
&H20  
&X00100000



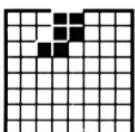
36  
&H24  
&X00100100



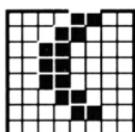
37  
&H25  
&X00100101



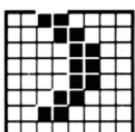
38  
&H26  
&X00100110



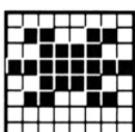
39  
&H27  
&X00100111



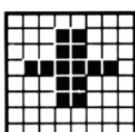
40  
&H28  
&X00101000



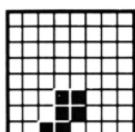
41  
&H29  
&X00101001



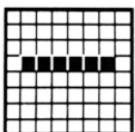
42  
&H2A  
&X00101010



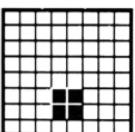
43  
&H2B  
&X00101011



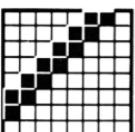
44  
&H2C  
&X00101100



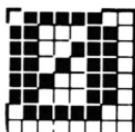
45  
&H2D  
&X00101101



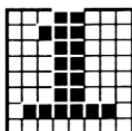
46  
&H2E  
&X00101110



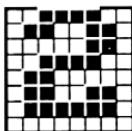
47  
&H2F  
&X00101111



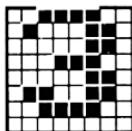
48  
&H30  
&X00110000



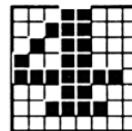
49  
&H31  
&X00110001



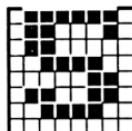
50  
&H32  
&X00110010



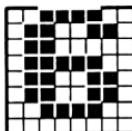
51  
&H33  
&X00110011



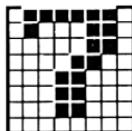
52  
&H34  
&X00110100



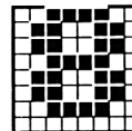
53  
&H35  
&X00110101



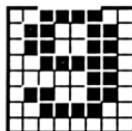
54  
&H36  
&X00110110



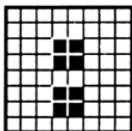
55  
&H37  
&X00110111



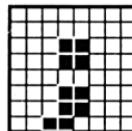
56  
&H38  
&X00111000



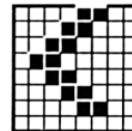
57  
&H39  
&X00111001



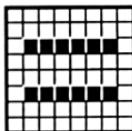
58  
&H3A  
&X00111010



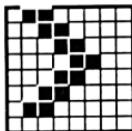
59  
&H3B  
&X00111011



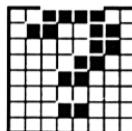
60  
&H3C  
&X00111100



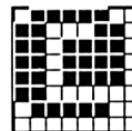
61  
&H3D  
&X00111101



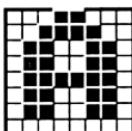
62  
&H3E  
&X00111110



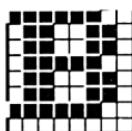
63  
&H3F  
&X00111111



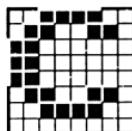
64  
&H40  
&X01000000



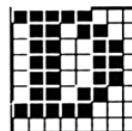
65  
&H41  
&X01000001



66  
&H42  
&X01000010

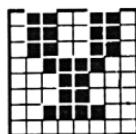


67  
&H43  
&X01000011

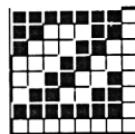


68  
&H44  
&X01000100

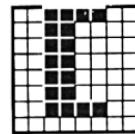




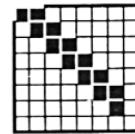
89  
&H59  
&X01011001



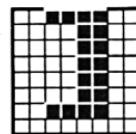
90  
&H5A  
&X01011010



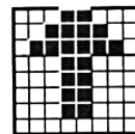
91  
&H5B  
&X01011011



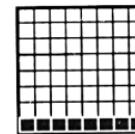
92  
&H5C  
&X01011100



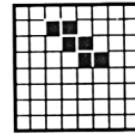
93  
&H5D  
&X01011101



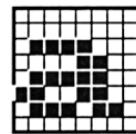
94  
&H5E  
&X01011110



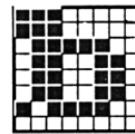
95  
&H5F  
&X01011111



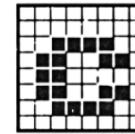
96  
&H60  
&X01100000



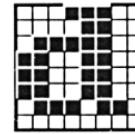
97  
&H61  
&X01100001



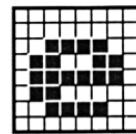
98  
&H62  
&X01100010



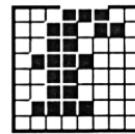
99  
&H63  
&X01100011



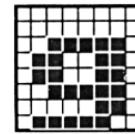
100  
&H64  
&X01100100



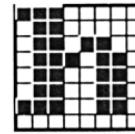
101  
&H65  
&X01100101



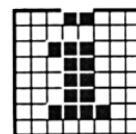
102  
&H66  
&X01100110



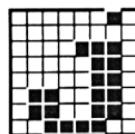
103  
&H67  
&X01100111



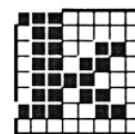
104  
&H68  
&X01101000



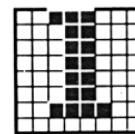
105  
&H69  
&X01101001



106  
&H6A  
&X01101010

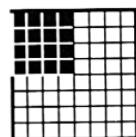


107  
&H6B  
&X01101011

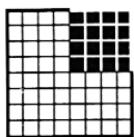


108  
&H6C  
&X01101100

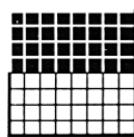




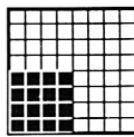
129  
&H81  
&X10000001



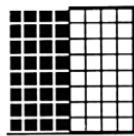
130  
&H82  
&X10000010



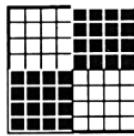
131  
&H83  
&X10000011



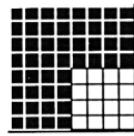
132  
&H84  
&X10000100



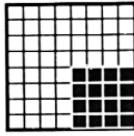
133  
&H85  
&X10000101



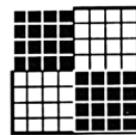
134  
&H86  
&X10000110



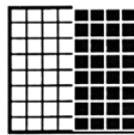
135  
&H87  
&X10000111



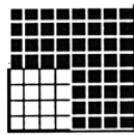
136  
&H88  
&X10001000



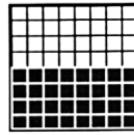
137  
&H89  
&X10001001



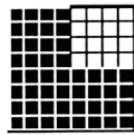
138  
&H8A  
&X10001010



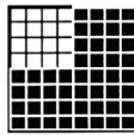
139  
&H8B  
&X10001011



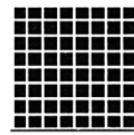
140  
&H8C  
&X10001100



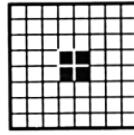
141  
&H8D  
&X10001101



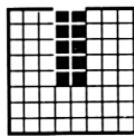
142  
&H8E  
&X10001110



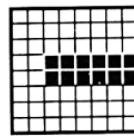
143  
&H8F  
&X10001111



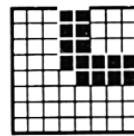
144  
&H90  
&X10010000



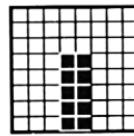
145  
&H91  
&X10010001



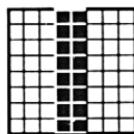
146  
&H92  
&X10010010



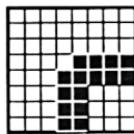
147  
&H93  
&X10010011



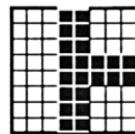
148  
&H94  
&X10010100



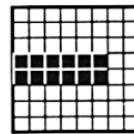
149  
&H95  
&X10010101



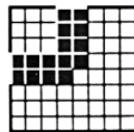
150  
&H96  
&X10010110



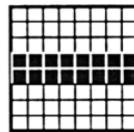
151  
&H97  
&X10010111



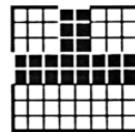
152  
&H98  
&X10011000



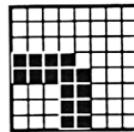
153  
&H99  
&X10011001



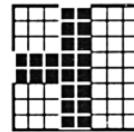
154  
&H9A  
&X10011010



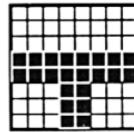
155  
&H9B  
&X10011011



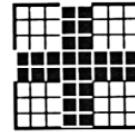
156  
&H9C  
&X10011100



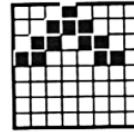
157  
&H9D  
&X10011101



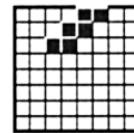
158  
&H9E  
&X10011110



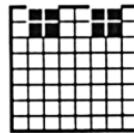
159  
&H9F  
&X10011111



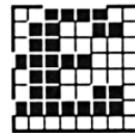
160  
&HA0  
&X10100000



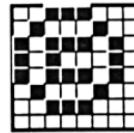
161  
&HA1  
&X10100001



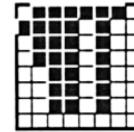
162  
&HA2  
&X10100010



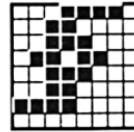
163  
&HA3  
&X10100011



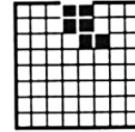
164  
&HA4  
&X10100100



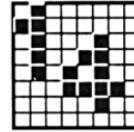
165  
&HA5  
&X10100101



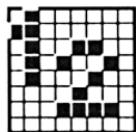
166  
&HA6  
&X10100110



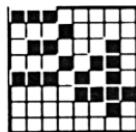
167  
&HA7  
&X10100111



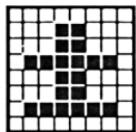
168  
&HA8  
&X10101000



169  
&HA9  
&X10101001



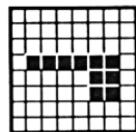
170  
&HAA  
&X10101010



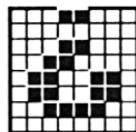
171  
&HAB  
&X10101011



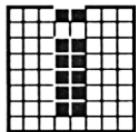
172  
&HAC  
&X10101100



173  
&HAD  
&X10101101



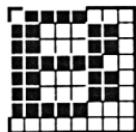
174  
&HAE  
&X10101110



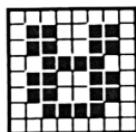
175  
&HAF  
&X10101111



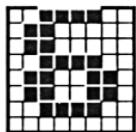
176  
&HB0  
&X10110000



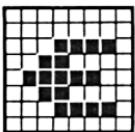
177  
&HB1  
&X10110001



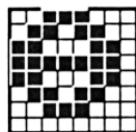
178  
&HB2  
&X10110010



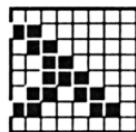
179  
&HB3  
&X10110011



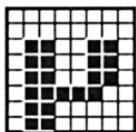
180  
&HB4  
&X10110100



181  
&HB5  
&X10110101



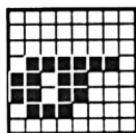
182  
&HB6  
&X10110110



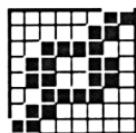
183  
&HB7  
&X10110111



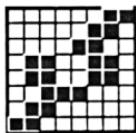
184  
&HB8  
&X10111000



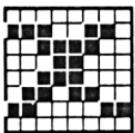
185  
&HB9  
&X10111001



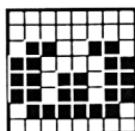
186  
&HBA  
&X10111010



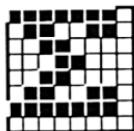
187  
&HBB  
&X10111011



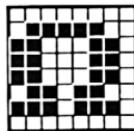
188  
&HBC  
&X10111100



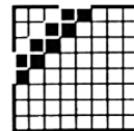
189  
&HBD  
&X10111101



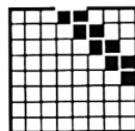
190  
&HBE  
&X10111110



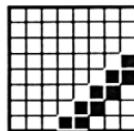
191  
&HBF  
&X10111111



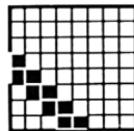
192  
&HCO  
&X11000000



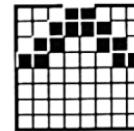
193  
&HC1  
&X11000001



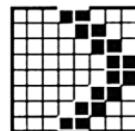
194  
&HC2  
&X11000010



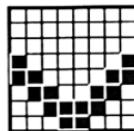
195  
&HC3  
&X11000011



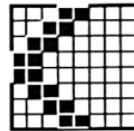
196  
&HC4  
&X11000100



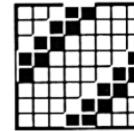
197  
&HC5  
&X11000101



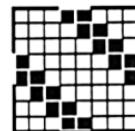
198  
&HC6  
&X11000110



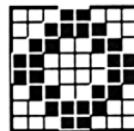
199  
&HC7  
&X11000111



200  
&HC8  
&X11001000



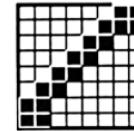
201  
&HC9  
&X11001001



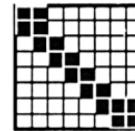
202  
&HCA  
&X11001010



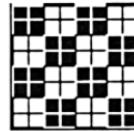
203  
&HCB  
&X11001011



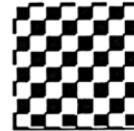
204  
&HCC  
&X11001100



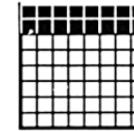
205  
&HCD  
&X11001101



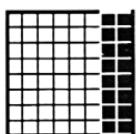
206  
&HCE  
&X11001110



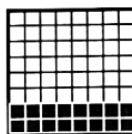
207  
&HCF  
&X11001111



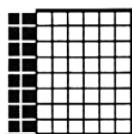
208  
&HD0  
&X11010000



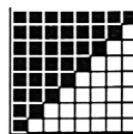
209  
&HD1  
&X11010001



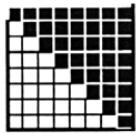
210  
&HD2  
&X11010010



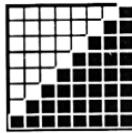
211  
&HD3  
&X11010011



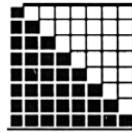
212  
&HD4  
&X11010100



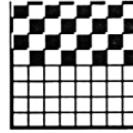
213  
&HD5  
&X11010101



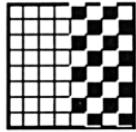
214  
&HD6  
&X11010110



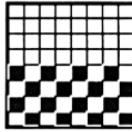
215  
&HD7  
&X11010111



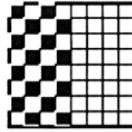
216  
&HD8  
&X11011000



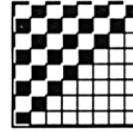
217  
&HD9  
&X11011001



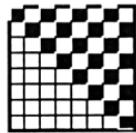
218  
&HDA  
&X11011010



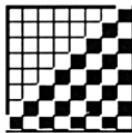
219  
&HDB  
&X11011011



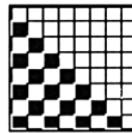
220  
&HDC  
&X11011100



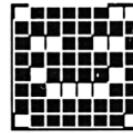
221  
&HDD  
&X11011101



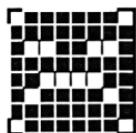
222  
&HDE  
&X11011110



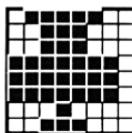
223  
&HDF  
&X11011111



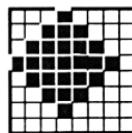
224  
&HE0  
&X11100000



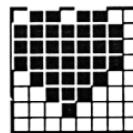
225  
&HE1  
&X11100001



226  
&HE2  
&X11100010

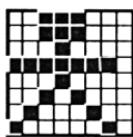


227  
&HE3  
&X11100011

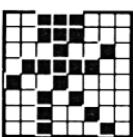


228  
&HE4  
&X11100100

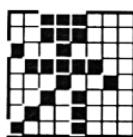




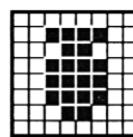
249  
&HF9  
&X11111001



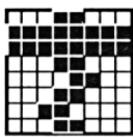
250  
&HFA  
&X11111010



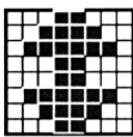
251  
&HFB  
&X11111011



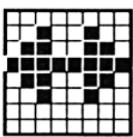
252  
&HFC  
&X11111100



253  
&HFD  
&X11111101

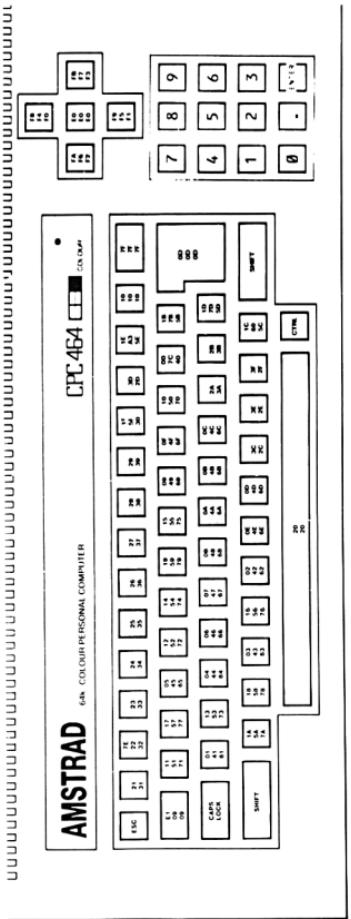


254  
&HFE  
&X11111110



255  
&HFF  
&X11111111

### Valeurs ASCII par défaut



JOYSTICK 0

OB  
OB

26 36

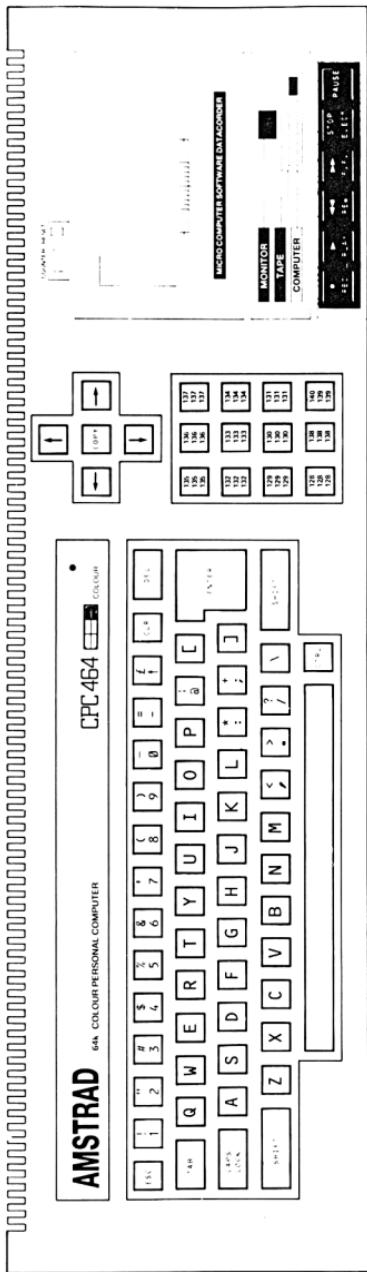
JOYSTICK 1

25

FIRE 2	FIRE 1					
08 ← 58	5A → 09					
08 ← 58	5A → 09					
		12 ← 52	07 ← 47	06 ← 46	14 → 54	
		72	67	66	74	

↓

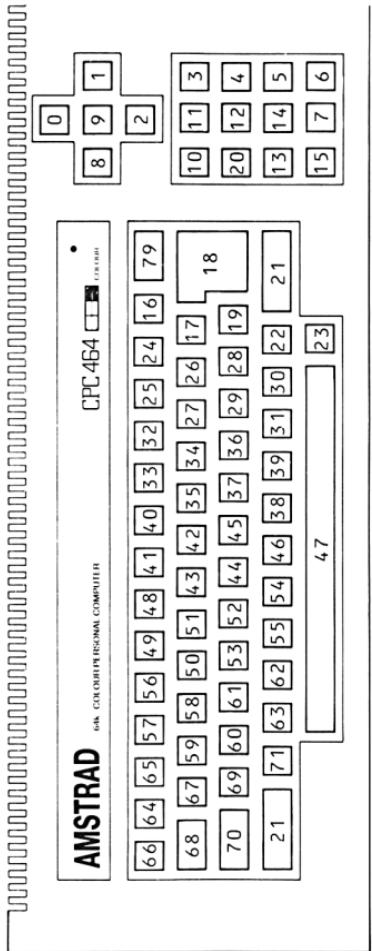
OA



Caractères d'expansion,valeurs et emplacements par défaut

CARAEXP	valeur	ascii
128	0	30
129	1	31
130	2	32
131	3	33
132	4	34
133	5	35
134	6	36
135	7	37
136	8	38
137	9	39
138	.	2E
139	[enter] run"	0D,55,4E,22,0D
140		

## Numéros des touches et des positions des joysticks



# Appendice IV

## Le CPC464 :une approche pour les mordus des micros

Le CPC464 est un ordinateur économique utilisant des technologies qui ont fait leurs preuves de manière innovative et performante, présenté sous un format offrant un rapport qualité-prix très attractif et des capacités considérables d'expansion qui le rendent intéressant à la fois pour les novices et ceux qui ont l'expérience de la micro.

Le matériel et le logiciel en ROM ont été conçus pour donner aux programmeurs débutants aussi bien qu'aux chevronnés un environnement agréable où les logiciels sont facilement adaptables et où de nouveaux logiciels peuvent bénéficier des capacités étendues offertes par l'AMSTRAD CPC464.

**Les principaux éléments du système sont:**

### **Z80 CPU**

Le microprocesseur le plus utilisé dans l'informatique familiale, avec la base de logiciels la plus importante, à quoi s'ajoute le potentiel du système d'exploitation CPM. La structure unique du système d'interruptions a permis au CPC464 d'innover avec les commandes BASIC, AFTER et EVERY et autres capacités en 'temps réel' contrôlant le son et les chronomètres internes.

## **64K RAM**

Une ration généreuse de RAM, dont plus de 42K réellement disponibles pour l'utilisateur, grâce à la technique astucieuse de la superposition du ROM dans l'utilisation du BASIC.

## **Ecran**

Le CPC464 possède trois modes d'affichage dont un mode 80 colonnes, plus une palette de 27 couleurs et une résolution de 640 x 200 pixels.

## **Un vrai clavier**

Un clavier de type machine à écrire avec un pavé curseur ergonomique et un pavé numérique qui peut servir pour les touches de fonction.

## **Magnéto-cassette incorporé.**

La sauvegarde et la charge des données utilisent ce lecteur enregistreur de cassettes incorporé qui évite l'amoncellement de fils et connections. Pas de problème de niveau et de réglage, et l'avantage de pouvoir enregistrer à deux vitesses différentes (choisies d'une manière logicielle) 1K Baud et 2K Baud, la lecture se faisant automatiquement à la vitesse de l'enregistrement.

## **BASIC**

Un BASIC écrit en Angleterre, plus rapide et polyvalent qu'on ne pourrait s'y attendre d'un BASIC, avec beaucoup d'extensions pour les graphiques et le son, judicieusement supporté par le logiciel intégré.

## **Jeu de caractères étendus**

Un jeu de caractères complet avec symboles graphiques disponibles à partir du clavier ou en utilisant la fonction CHR\$(n).

## **Chronométrage**

Les interruptions sont produites par des analyseurs instantanés qui mesurent les différences de temps.

## **Touches redéfinissables**

L'utilisateur peut redéfinir 32 touches pour son usage particulier une fonction pouvant avoir jusqu'à 32 caractères. On peut redéfinir les paramètres de répétition. Un jeu de 255 caractères (tous les ASCII plus une centaine d'autres) sont redéfinissables.

## **Sous-programmes**

Un grand nombre de sous-programmes en assembleur peuvent être appelés à partir du BASIC.

## **MODES D'ECRAN**

trois mode d'opération

a) Normal

Mode 1: 40 colonnes x 25 lignes, 4 couleurs de texte (INK) 320 x 200 pixels, chacun adressable individuellement en quatre couleurs.

b) Mode multicouleurs

Mode 0: 20 colonnes x 25 lignes, 16 couleurs de texte 160 x 200 pixels, adressables individuellement en 16 couleurs.

c) Haute résolution

Mode 2: 80 colonnes x 25 lignes, 2 couleurs 640 x 200 pixels, adressables individuellement en 2 couleurs.

## **Sélection des Couleurs**

(tout au long de ce guide, noir, c'est à dire luminance zéro, est considéré comme une couleur pour faciliter les descriptions).

La bordure peut avoir 2 couleurs quelconques, indépendamment du mode, deux couleurs intermittentes ou une couleur fixe.

Le nombre d'encre (INK) disponibles dépend du mode; chaque encre peut avoir deux couleurs intermittentes ou une couleur fixe. Les papiers et stylos, texte ou graphique peuvent alors être associés à une encre.

L'écriture peut-être transparente ou opaque: elle peut ignorer la couleur du papier et écrire sur les graphiques ou écrire sur le fond.

## Fenêtres

L'utilisateur peut choisir jusqu'à 8 fenêtres de textes où il peut écrire du texte et une fenêtre graphique pour dessiner.

Les fenêtres prennent une valeur par défaut quand on redéfinit le mode d'écran.

**N.B.** Si la fenêtre de texte est par défaut l'écran entier, le déroulement rapide des lignes est fait par le matériel. Si la fenêtre est plus petite que l'écran, le déroulement est contrôlé par le logiciel donc beaucoup plus lent.

## Curseur

Le curseur ne marche plus quand le processeur n'a pas besoin d'entrées au clavier, faisant du curseur un avis d'appel automatique. Le curseur est représenté par un carré de couleur inverse.

## Son polyphonique

Les capacités sonores du CPC464 proviennent d'un processeur générateur de son de la famille AY8910 de General Instruments. Il fonctionne avec 3 canaux (voix), qui peuvent être programmés séparément en ton et en amplitude. Des bruits peuvent être ajoutés à volonté.

Les trois canaux apparaissent à gauche, à droite et au centre (quand une utilise la fiche stéréo). Le haut-parleur interne produit un son mixé.

Le contrôle des capacités d'amplitude et de ton provient du logiciel, celui venant du processeur n'est généralement pas utilisé.

## Interface d'imprimante

Une interface parallèle de type Centronics est d'origine. Elle utilise le signal 'Busy' pour les protocoles de reconnaissance.

## Support d'Expansion

Plusieurs interfaces utilisables grâce à l'adressage indirect et aux blocs de saut, fournissent une grande capacité d'expansion pour les logiciels. Amstrad dispose de lecteurs de disquettes et d'interfaces série avec contrôle du logiciel en ROM.

## Coordonnées

L'origine texte est en haut à gauche et les positions changent suivant le Mode d'écran.

L'origine graphique est en bas à droite et considère l'écran comme haute résolution même si les calculs pour l'encre (INK) sont effectués correctement dans tous les modes d'écran.

L'axe vertical a des coordonnées variant de 0 à 399 divisées par deux pour donner une position physique de 0 à 199. Cela permet de garder les proportions sur l'écran.

En écran normal, chaque pixel a deux adresses horizontales, en Mode multicolore 0 chaque pixel a quatre adresses et chacune peut-être utilisée pour définir la position.

## ROM d'expansion

Tous les ROM occupent les 16K supérieures de la mémoire (où se trouve le BASIC) et le logiciel intégré peut appeler 240 ROM additionnels de 16K chacun. (Une certaine quantité de décodage d'adresse du matériel doit être effectué d'une manière externe à la machine).

# Vue d'ensemble

Un bref sommaire des caractéristiques principales du matériel et du logiciel intégré du CPC464.

## 1) Le Matériel

### 1.1 à l'intérieur du CPC464

L'ordinateur, le clavier, le magnétocassette et le haut-parleur. Sortie RGB et luminance.

#### 1.1.1 Les microprocesseurs LSI

Z80A tournant à 4MHz

64K octets (64K x 1) de mémoire vive RAM réactivée par accès à la mémoire écran.  
32K octets de ROM contenant le BASIC et le système d'exploitation MOS.

Le contrôleur 6845 CRT produit les signaux d'analyse pour la RAM vidéo.

**NB.** L'organisation est complexe et change avec le Mode d'écran. Le CRTC peut servir au déroulement horizontal (de coté par zones de 140 ème de largeur) et vertical (vers le haut et le bas par bauds de 8 lignes) si logiciel le demande. Des paramètres du CRTC choisissent le numéro de lignes, la vitesse d'affichage, la position et la largeur des bordures.

Générateur de son GI AY 3-8912: 3 voix. Le son est pris sur les trois canaux et mixé pour produire une sortie mono sur le haut parleur interne, contrôlé par un bouton de volume. Il y a aussi une sortie de son stéréo où:

Gauche = Canal A + 1/2 Canal C

Droit = Canal B + 1/2 Canal C

Le microprocesseur reçoit aussi des informations du clavier et de la sortie manette de jeux.

Une Entrée-Sortie sert d'interface entre le Bus et le processeur de son GI. Elle analyse le clavier, la sortie manette de jeux, les connections en option et contrôle la cassette.

### 1.1.2 Les fiches externes

Des connecteurs (de côté) PCB pour l'expansion d'ordre général et pour l'imprimante (parallèle, Centronics). Des fiches pour les manettes de jeux (9 plots DIN), son stéréo, sortie vidéo (RVB et sync ou composite vidéo ou luminance et sync.)

## 1.2 A l'extérieur de la machine

Le CPC464 est fourni avec un choix de deux types d'Entrée directes sur le Moniteur Vidéo, chacun ayant une alimentation en 5V pour l'ordinateur. Il existe en plus un adaptateur PERITEL, le MP1.

Des câbles sont nécessaires pour le branchement de l'imprimante et de l'ampli Hi-Fi.

## 1.3 Caractéristiques de l'affichage.

L'affichage dispose de 16K de mémoire. La machine a un choix de 27 couleurs pour le choix de la palette. Le nombre d'encre sur la palette dépend du mode d'écran. Un certain nombre d'encre peuvent avoir la même couleur si nécessaire. Les pixels sur l'écran sont définis comme des points d'une certaine couleur. (Notez que le fond ou papier (PAPER), a besoin d'une encre de la palette, la bordure ou cadre extérieur (BORDER) peut avoir une ou deux (*intermittentes*) couleurs indépendantes choisies parmi les 27.

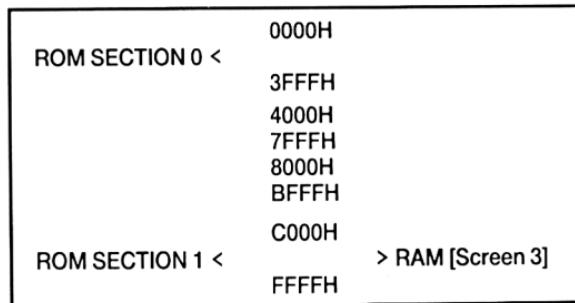
Mode	Nombre d'encre	Points vertic.	Points Horiz.	Caract. Horiz.
Normal	4	200	320	40
Haute résol	2	200	640	80
Multicoloré	16	200	160	20

NB. L'ordinateur produira des effets de gris différents si on les observe sur un moniteur monochrome (tons de vert pour le GT64). L'ordre des couleurs en intensité croissante est

Niveau de Gris	Couleur	Niveau de Gris	Couleur
0	Noir		
1	Bleu	14	Bleu Pastel
2	Bleu Vif	15	Orange
3	Rouge	16	Rose
4	Magenta	17	Magenta Pastel
5	Mauve	18	Vert Vif
6	Rouge Vif	19	Vert Marin
7	Pourpre	20	Turquoise Vif
8	Magenta Vif	21	Vert Citron
9	Vert	22	Vert Pastel
10	Turquoise	23	Turquoise Pastel
11	Bleu Ciel	24	Jaune Vif
12	Jaune	25	Jaune Pastel
13	Blanc	26	Blanc Brillant

## 1.4 Implantation de la mémoire

Les 64K de mémoire vive sont organisés comme suit, tenant compte qu'une partie de la mémoire ROM se superpose à l'écran RAM, libérant le maximum d'espace pour l'utilisateur pour le fonctionnement en BASIC.



Quand il y a ROM et RAM à la même adresse, la LECTURE accède au ROM et l'ECRITURE au RAM. Chaque section du ROM peut être interrompue donnant accès à la lecture RAM à la même adresse.

## 1.5 Extension

### 1.5.1 ROM latéraux ('sideways')

On a prévu de pouvoir sélectionner des ROM supplémentaires à la place du ROM de la machine. Le choix des adresses et la logique de selection des unités ROM sera contenue dans un BUS d'expansion, mais tous les signaux passent par le BUS d'expansion.

### 1.5.2 Extensions RAM

Les extensions RAM suivront un principe analogue aux extensions ROM, passant aussi par le BUS d'expansion. Cette mémoire sera pour la lecture à la base et un procédé concernant l'implantation Entrées-Sorties sera nécessaire pour écrire sur cette extension RAM.

### 1.5.3 Entrées-Sorties supplémentaires

La plupart des adresses d'Entrée-Sortie sont réservées à l'ordinateur et en particulier les adresses inférieures à **7FXX** ne doivent pas être utilisées. Les suivantes doivent être utilisées par le matériel extérieur:

**F8XX, F9XX, FAXX, FBXX**

Les périphériques du bus d'expansion doivent décoder les adresses **A0** à **A7** pendant que l'adresse **A10** est peu élevée. Les canaux d'expansion d'entrée-sortie des adresses **F800** à **FBFF** sont réservées comme suit:

#### Adresse A0-A7

00 - 7B	* ne pas utiliser*
7C - 7F	Reservées à l'interface disquette
80 - 8B	* ne pas utiliser*
BC - BF	Reservées à des usages ultérieurs
CO - DB	* ne pas utiliser*
DC - DF	Reservées pour les interfaces de communication.
E0 - FF	Disponibles pour l'utilisateur

Notez que les instructions Z80 qui placent le registre B sur la moitié supérieure du bus d'adresse (A15,A8) doivent être utilisées.

## 2 Clavier

Une remise à zéro complète est effectuée avec **[CTRL]** **[SHIFT]** **[ESC]**. Les touches de caractères ou de mouvement du curseur se répètent automatiquement sous contrôle du logiciel, à l'exception des touches du pavé numérique.

**[ESC]** suspend l'exécution du programme. Suivi d'un autre **[ESC]**, l'exécution est terminée, suivi d'une autre touche le programme reprend.

**[CAPS LOCK]** verrouille et déverrouille les majuscules. Le verrouillage de **[SHIFT]** se fait avec **[CTRL]** et **[CAPS LOCK]** pressés ensemble.

Le curseur de copie se sépare du curseur normal en appuyant sur [**SHIFT**] et les touches curseur. On peut modifier les caractères sous le curseur de copie en pressant la touche de [**COPY**].

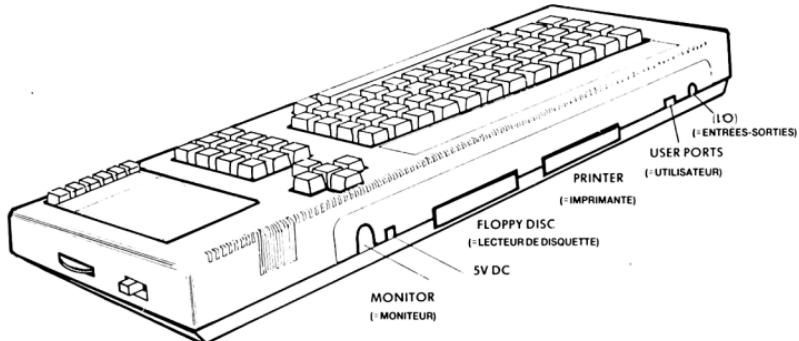
Les touches curseur permettent d'éditer ce qui est dans le 'buffer' d'Entrée qui peut comporter plusieurs lignes. Les touches curseur peuvent être utilisées pour placer le début d'une entrée sur le clavier avant qu'une entrée ait été faite à partir du clavier. Une fois qu'on a pressé sur une touche de caractère, la position sur l'écran est déterminée. Toute entrée nouvelle écrira par dessus ce qui est sur l'écran.

[**DEL**] est une touche d'effacement vers l'arrière, [**CLR**] efface devant.



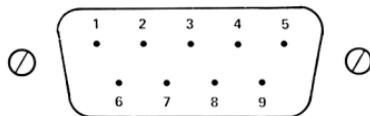
# Appendice V:

Connections du CPC464 sur le panneau arrière



SORTIE MANETTE DE JEUX PIN D)

VUE DE L'ARRIÈRE



PIN 1	UP	PIN 6	FIRE 2
PIN 2	DOWN	PIN 7	FIRE 1
PIN 3	LEFT	PIN 8	COMMON
PIN 4	RIGHT	PIN 9	COM 2
PIN 5	SPARE		

SORTIE VIDÉO

VIDEO OUTPUT CONNECTOR (6 PIN DIN)

VUE DE L'ARRIÈRE



PIN 1	RED	PIN 4	SYNC
PIN 2	GREEN	PIN 5	GND
PIN 3	BLUE	PIN 6	LUM

## SORTIE EXPANSION

EXPANSION PORT

50 WAY 0.1 EDGE CONNECTOR

### VUE DE L'ARRIÈRE



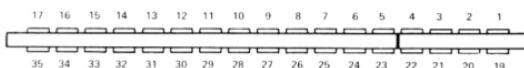
PIN 1	SOUND	PIN 18	A0	PIN 35	INT
PIN 2	GND	PIN 19	D7	PIN 36	NMI
PIN 3	A15	PIN 20	D6	PIN 37	BUSRD
PIN 4	A14	PIN 21	D5	PIN 38	BUSAK
PIN 5	A13	PIN 22	D4	PIN 39	READY
PIN 6	A12	PIN 23	D3	PIN 40	BUS_RESET
PIN 7	A11	PIN 24	D2	PIN 41	RESET
PIN 8	A10	PIN 25	D1	PIN 42	ROMEN
PIN 9	A9	PIN 26	D0	PIN 43	ROMDIS
PIN 10	A8	PIN 27	-1.5v	PIN 44	RAMRD
PIN 11	A7	PIN 28	MREQ	PIN 45	RAMDIS
PIN 12	A6	PIN 29	M1	PIN 46	CURSOR
PIN 13	A5	PIN 30	RFSH	PIN 47	L_PEN
PIN 14	A4	PIN 31	IORD	PIN 48	EXP
PIN 15	A3	PIN 32	RD	PIN 49	GND
PIN 16	A2	PIN 33	WR	PIN 50	U
PIN 17	A1	PIN 34	HALT		

## SORTIE IMPRIMANTE

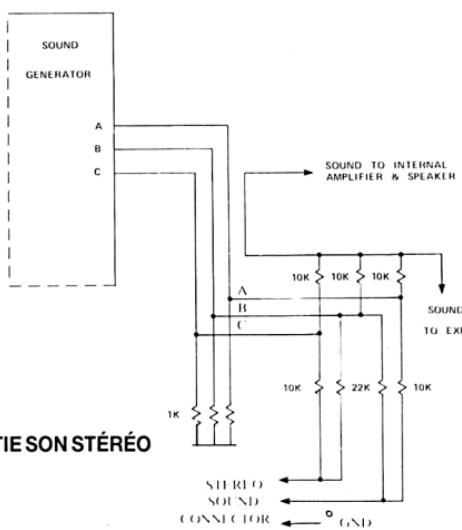
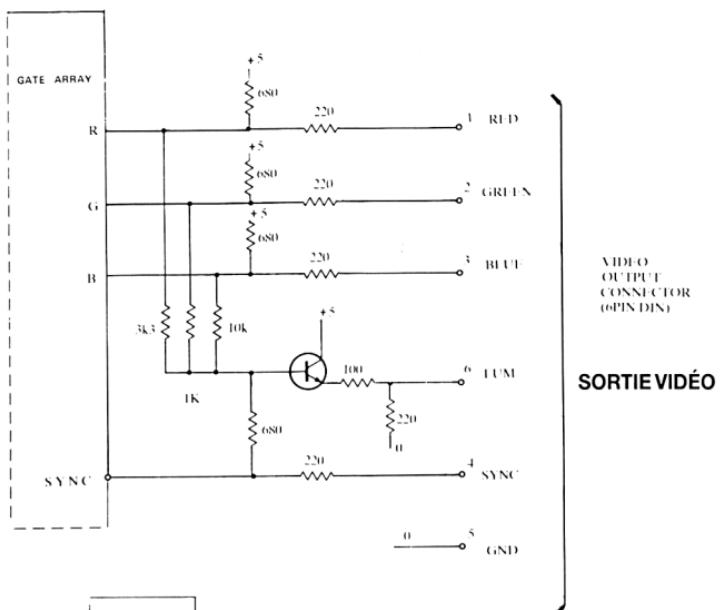
PRINTER PORT

34 WAY 0.1 EDGE CONNECTOR

### VUE DE L'ARRIÈRE



PIN 1	STROBE	PIN 19	GND
PIN 2	D0	PIN 20	GND
PIN 3	D1	PIN 21	GND
PIN 4	D2	PIN 22	GND
PIN 5	D3	PIN 23	GND
PIN 6	D4	PIN 24	GND
PIN 7	D5	PIN 25	GND
PIN 8	D6	PIN 26	GND
PIN 9	D7	PIN 27	GND
PIN 11	BUSY	PIN 28	GND
PIN 14	GND	PIN 33	GND
PIN 16	GND	All other pins	NC



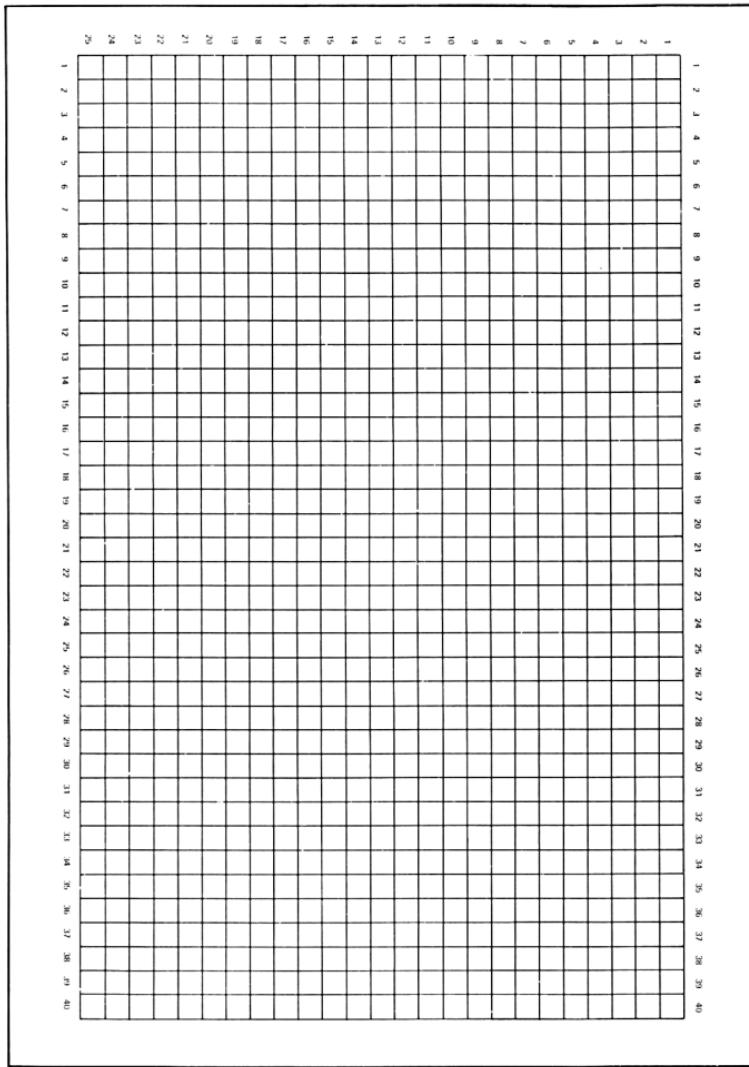


## Appendice VI

## Grille pour textes et fenêtres d'écran MODE 0 20 colonnes

# Grille pour textes et fenêtres d'écran

## MODE 1    40 colonnes



# Grille pour textes et fenêtres d'écran

## MODE 2 80 colonnes

2	4	6	8	10	12	14	16	18	20	22	24	26	28	30	32	34	36	38	40	42	44	46	48	50	52	54	56	58	60	62	64	66	68	70	72	74	76	78	80						
1																																													
2																																													
3																																													
4																																													
5																																													
6																																													
7																																													
8																																													
9																																													
10																																													
11																																													
12																																													
13																																													
14																																													
15																																													
16																																													
17																																													
18																																													
19																																													
20																																													
21																																													
22																																													
23																																													
24																																													
25																																													



# Appendice VII: Notes et Périodes de Tons.

La table suivante donne les périodes de tons pour les notes de la gamme complète des huit octaves.

La fréquence produite n'est pas totalement exacte car elle est calculée à partir d'un nombre entier et l'erreur relative est indiquée.

NOTE	FREQUENCE	PERIODE	ERREUR RELATIVE
C=do	32.703	3822	-0.007%
C#=do dièse	34.648	3608	+0.007%
D=ré	36.708	3405	-0.007%
D#=mi bémol	38.891	3214	-0.004%
E=mi	41.203	3034	+0.009%
F=fa	43.654	2863	-0.016%
F#=fa dièse	46.249	2703	+0.009%
G=sol	48.999	2551	-0.002%
G#=sol dièse	51.913	2408	+0.005%
A=la	55.000	2273	+0.012%
A#=si bémol	58.270	2145	-0.008%
B=si	61.735	2025	+0.011%

NOTE	FREQUENCE	PERIODE	ERREUR RELATIVE
C=do	65.406	1911	-0.007%
C#=do dièse	69.296	1804	+0.007%
D=ré	73.416	1703	+0.022%
D#=mi bémol	77.782	1607	-0.004%
E=mi	82.407	1517	+0.009%
F=fa	87.307	1432	+0.019%
F#=fa dièse	92.499	1351	-0.028%
G=sol	97.999	1276	+0.037%
G#=sol dièse	103.826	1204	+0.005%
A=la	110.000	1136	-0.032%
A#=si bémol	116.541	1073	+0.039%
B=si	123.471	1012	-0.038%

NOTE	FREQUENCE	PERIODE	ERREUR RELATIVE
C=do	130.813	956	+0.046%
C#=do dièse	138.591	902	+0.007%
D=ré	146.832	851	-0.037%
D#=mi bémol	155.564	804	+0.058%
E=mi	164.814	758	-0.057%
F=fa	174.614	716	+0.019%
F#=fa dièse	184.997	676	+0.046%
G=sol	195.998	638	+0.037%
G#=sol dièse	207.652	602	+0.005%
A=la	220.000	568	-0.032%
A#=si bémol	233.082	536	-0.055%
B=si	246.942	506	-0.038%
NOTE	FREQUENCE	PERIODE	ERREUR RELATIVE
C=do	261.626	478	+0.046% Middle C
C#=do dièse	277.183	451	+0.007%
D=ré	293.665	426	+0.081%
D#=mi bémol	311.127	402	+0.058%
E=mi	329.628	379	-0.057%
F=fa	349.228	358	+0.019%
F#=fa dièse	369.994	338	+0.046%
G=sol	391.995	319	+0.037%
G#=sol dièse	415.305	301	+0.005%
A=la	440.000	284	-0.032%
A#=si bémol	466.164	268	-0.055%
B=si	493.883	253	-0.038%
NOTE	FREQUENCE	PERIODE	ERREUR RELATIVE
C=do	523.251	239	+0.046%
C#=do dièse	554.365	225	-0.215%
D=ré	587.330	213	+0.081%
D#=mi bémol	622.254	201	+0.058%
E=mi	659.255	190	+0.206%
F=fa	698.457	179	+0.019%
F#=fa dièse	739.989	169	+0.046%
G=sol	783.991	159	-0.277%
G#=sol dièse	830.609	150	-0.328%
A=la	880.000	142	-0.032%
A#=si bémol	932.328	134	-0.055%
B=si	987.767	127	+0.356%

NOTE	FREQUENCE	PERIODE	ERREUR RELATIVE
C=do	1046.502	119	-0.374%
C#=do dièse	1108.731	113	+0.229%
D=ré	1174.659	106	-0.390%
D#=mi bémol	1244.508	100	-0.441%
E=mi	1318.510	95	+0.206%
F=fa	1396.913	89	-0.543%
F#=fa dièse	1479.978	84	-0.548%
G=sol	1567.982	80	+0.350%
G#=sol dièse	1661.219	75	-0.328%
A=la	1760.000	71	-0.032%
A#=si bémol	1864.655	67	-0.055%
B=si	1975.533	63	-0.435%
NOTE	FREQUENCE	PERIODE	ERREUR RELATIVE
C=do	2093.004	60	+0.462%
C#=do dièse	2217.461	56	-0.662%
D=ré	2349.318	53	-0.390%
D#=mi bémol	2489.016	50	-0.441%
E=mi	2637.021	47	-0.855%
F=fa	2793.826	45	+0.574%
F#=fa dièse	2959.955	42	-0.548%
G=sol	3135.963	40	+0.350%
G#=sol dièse	3322.438	38	+0.992%
A=la	3520.000	36	+1.357%
A#=si bémol	3729.310	34	+1.417%
B=si	3951.066	32	+1.134%
NOTE	FREQUENCE	PERIODE	ERREUR RELATIVE
C=do	4186.009	30	+0.462%
C#=do dièse	4434.922	28	-0.662%
D=ré	4698.636	27	+1.469%
D#=mi bémol	4978.032	25	-0.441%
E=mi	5274.041	24	+1.246%
F=fa	5587.652	22	-1.685%
F#=fa dièse	5919.911	21	-0.548%
G=sol	6271.927	20	+0.350%
G#=sol dièse	6644.875	19	+0.992%
A=la	7040.000	18	+1.357%
A#=si bémol	7458.621	17	+1.417%
B=si	7902.133	16	+1.134%

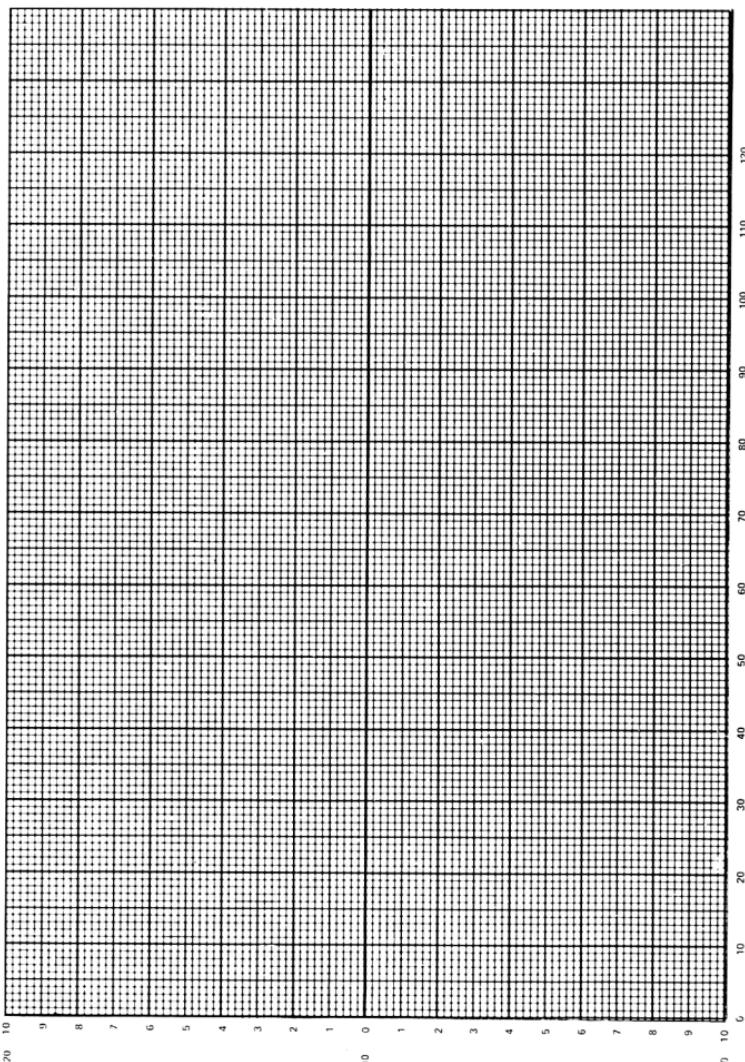
Ces valeurs sont toutes calculées à partir du la International comme suit:

$$\text{FREQUENCE} = 440 * (2^{1 \text{ OCTAVE}} + (10 - N)^{12})$$

$$\text{PERIODE} = \text{ROUND}(125000 / \text{FREQUENCE})$$

où N est égal à 1 pour do, 2 pour do dièse, 3 pour ré ...etc

# Enveloppe des sons/Feuille de musique



# Appendice VIII

## Erreurs et mots réservés

### Numéros d'erreur et messages d'erreur

Quand le BASIC rencontre dans le programme une affirmation, un mot ou une variable qu'il ne comprend pas, il va s'arrêter et afficher un message d'erreur. La forme du message (en anglais) indique ce qui est faux et parfois, si c'est une erreur typographique quand on tape le programme, BASIC se met en mode éditeur et vous présente la ligne fautive.

L'erreur la plus commune pour le débutant est la Syntax Error (ou Erreur de Syntaxe) qui a le numéro 2 et le BASIC met la ligne que vous devez corriger en évidence.

Quand on est en mode direct (pas de numéro de ligne) cela indique seulement qu'une erreur a eu lieu et suppose que la dernière ligne tapée est toujours sur l'écran, visible.

Si la commande **ON ERROR GOTO** est donnée au début d'un programme, elle peut renvoyer l'ordinateur à une ligne donnée quand une erreur est décelée. Dans l'exemple suivant, l'ordinateur est dirigé vers la ligne 1000 en cas d'erreur.

**10 ON ERROR GOTO 1000**

programme

**1000 PRINT CHR\$(7):MODE2:INK 1,0:INK 09:CLS:LIST**

auquel cas l'ordinateur fait un bip, efface l'écran, se met en mode 80 colonnes lisible et donne la liste du programme pour qu'on puisse l'examiner. Si l'erreur est une erreur de Syntaxe, la ligne apparaît en mode d'édition ( curseur sur le premier chiffre de la ligne) mais sans le message **Syntax Error**.

Souvenez-vous de mettre une ligne avec **END** avant la ligne 1000 si vous voulez garder ce qu'il y a sur l'écran.

Le BASIC ne donne pas de message d'erreurs quand les données sont valables et on peut supposer que chaque fois qu'une erreur se produit, on peut revenir à la source en s'aider des messages affichés. Comme avec beaucoup de choses, c'est avec des erreurs qu'on apprend et vous verrez que le CPC464 est très patient avec les erreurs et que vous vous épuiserez avant lui.

Voici les erreurs avec leur numéro de référence, le message que donne le BASIC, sa traduction et sa signification:

### **1 Unexpected NEXT = NEXT inattendu**

Une commande **NEXT** a été rencontrée sans que la boucle **FOR...** ait été commencée, ou la variable après **NEXT** ne correspond pas à celle de la boucle **FOR...**

### **2 Syntax Error = Erreur de Syntax**

Le BASIC ne peut pas comprendre la ligne à cause d'une construction non permise (très souvent une faute de frappe)

### **3 Unexpected RETURN = RETURN inattendu**

Une commande **RETURN** survient alors qu'il n'y a pas de sous-programme en cours.

### **4 DATA exhausted = il n'y a plus de données (= DATA)**

Une commande **READ** a essayé de lire une ligne de **DATA** qui est déjà finie.

### **5 Improper argument = valeur, argument incorrect**

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

### **6 Overflow = trop plein (mathématique)**

Se produit lorsqu'une opération arithmétique dépasse les limites. Le chiffre en virgule flottante est devenu trop grand (supérieur à 1.7E 38) ou on a essayé de l'écrire comme un nombre entier.

### **7 Memory Full = la mémoire déborde**

Le programme ou ses variables sont trop grands pour la mémoire, ou la structure des boucles est trop compliquée (trop de **GOSUB**, **WHILE** et **FOR** imbriqués).

Une commande **MEMORY** donnera cette erreur si on essayé de fixer le haut de la mémoire BASIC trop bas ou à une valeur trop haute. Attention aux fichiers sur cassette ouverts qui réservent de l'espace mémoire, tout comme les commandes **DIM**.

### **8 Line does not exist = le numéro qu vous avez demandé n'existe pas**

Le numéro de ligne tapé n'existe pas en mémoire.

### **9 Subscript out of range = indice hors limite**

Un des indices dans votre arrangement (matrice) est trop grand ou trop petit.

**10 Array already dimensioned** = arrangement déjà dimensionné.

Un des arrangements d'une affirmation **DIM** a déjà été défini.

**11 Division by zero** = Division par zéro

L'ordinateur n'aime pas diviser par zéro, que ce soit réel, entier etc...

**12 Invalid direct command** = commande directe non valable

La commande n'est pas acceptable en mode direct.

**13 Type mismatch** = les types de variables ne s'accordent pas

On a donné une valeur numérique pour une chaîne alphanumérique ou vice-versa ou un nombre non valable a été découvert pas une commande **READ** ou **INPUT**.

**14 String Space Full** = L'espace réservé aux chaînes déborde

Il y a tellement de chaînes qu'il n'y a plus de place, même après un remise en ordre.

**15 String too long** = chaîne trop longue

Une chaîne a plus de 255 caractères ce qui peut arriver quand on les accroche ensemble.

**16 String expression too complex** = chaîne trop compliquée

Des chaînes peuvent produire des valeurs intermédiaires qui, si elles sont trop nombreuses, conduisent le BASIC à donner ce message.

**17 Cannot CONTinue** = on ne peut pas CONTinuer

Le programme ne peut pas recommencer avec **CONT**, qui sert après une commande **STOP**, **[ESC] [ESC]** ou une erreur mais pas si le programme a été modifié entre temps.

**18 Unknown user function** = fonction inconnue au bataillon

On a oublié de définir la fonction FN avec la commande **DEF FN** auparavant.

**19 RESUME missing** = commande RESUME absente

On a trouvé la fin du programme au milieu d'un sous-programme venant d'une commande **ON ERROR GOTO**.

**20 Unexpected RESUME = RESUME inattendu**

On tombe sur une commande RESUME sans être dans un sous-programme de type ON ERROR GOTO

**21 Direct command found = Une commande directe tombe du ciel!**

En chargeant un programme d'une cassette, une ligne sans numéro de ligne s'est présentée.

**22 Operand missing = signe d'opération absent**

Le BASIC vient de tomber sur une expression incomplète.

**23 Line too long = ligne trop longue**

Le BASIC n'accepte pas les lignes de plus de 255 caractères.

**24 EOF met = rencontre avec la fin d'un fichier (EOF)**

EOF = End of File = fin de fichier; le programme est tombé sur la fin d'un fichier sur cassette.

**25 File type error = erreur dans le type de fichier**

Le fichier sur cassette n'est pas du type requis. OPENIN peut seulement ouvrir des fichiers TEXT ASCII. LOAD, RUN etc, ne fonctionnent qu'avec des fichiers produits par SAVE.

**26 NEXT missing = NEXT manquant**

On ne peut pas trouver un NEXT qui correspond à la commande FOR.

**27 File already open = fichier déjà ouvert**

Une commande OPENIN ou OPENOUT est exécutée avant que le fichier déjà ouvert ait été fermé.

**28 Unknown command**

Le BASIC ne trouve pas de références à cette commande externe.

**29 WEND missing = WEND manquant.**

La boucle commencée par WHILE n'est pas terminée par WEND

**30 Unexpected WEND**

Un WEND est découvert en dehors d'une boucle WHILE, ou un WEND ne correspond pas au WHILE de la boucle.

## Mots-clés du BASIC, ou mots réservés.

Voici le mots-clés du BASIC, ils sont Réservés et ne peuvent être utilisés comme variables.

ABS, AFTER, AND, ASC, ATN, AUTO

BIN\$, BORDER

CALL, CAT, CHAIN, CHR\$, CINT, CLEAR, CLG, CLOSEIN,  
CLOSEOUT,CLS,CONT,COS,CREAL

DATA, DEF, DEFINT, DEFREAL, DEFSTR, DEG, DELETE, DI,  
DIM,DRAW,DRAWR

EDIT, EI, ELSE, END, ENT, ENV, EOF, ERASE, ERL, ERR,  
ERROR,EVERY,EXP

FIX,FN,FOR,FRE

GOSUB,GOTO

HEX\$,HIMEM

IF,INK,INKEY,INKEY\$,INP,INPUT,INTR,INT

JOY

KEY

LEFT\$, LEN, LET, LINE, LIST, LOAD, LOCATE, LOG, LOG10,  
LOWER\$

MAX,MEMORY,MERGE,MID\$,MIN,MOD,MODE,MOVE,MOVER,

NEXT,NEW,NOT

ON, ON BREAK, ON ERROR GOTO, ON SQ, OPENIN, OPENOUT, OR,  
ORIGIN,OUT

PAPER,PEEK,PEN,PI,PLOT,PLOTR,POKE,POS,PRINT

RAD, RANDOMIZE, READ, RELEASE, REM, REMAIN, RENUM,  
RESTORE,RESUME,RETURN,RIGHT\$,RND,ROUND,RUN

SAVE, SGN, SIN, SOUND, SPACES\$, SPC, SPEED, SQ, SQR,  
STEP,STOP,STR\$,STRING\$,SWAP,SYMBOL

TAB, TAG, TAGOFF, TAN, TEST, TESTR, THEN, TIME, TO,  
TROFF,TRON

UNT, UPPER\$, USING  
VAL, VPOS  
WAIT, WEND, WHILE, WIDTH, WINDOW, WRITE  
XOR, XPOS  
YPOS  
ZONE

# Appendice IX

## Petit dictionnaire

Quelques mots français (et anglais) du domaine de l'informatique, expliqués aux usagers du CPC464.

### Accumulateur

Un endroit de la mémoire dans le microprocesseur qui retient temporairement les données à manipuler. Très utilisé en programmation en code machine, les fans du BASIC n'ont pas besoin de savoir qu'il existe.

### Adresse

Le numéro d'une instruction qui identifie la place d'une 'cellule' dans la mémoire de l'ordinateur. Avec son adresse, un endroit de la mémoire peut être choisi et 'lu' et dans le cas d'un endroit de la mémoire vive RAM, à la fois 'écrit' et 'lu' après modification.

### Algorithme

Un nom compliqué pour une formule ou une somme,, c'est aussi une séquence d'opérations arithmétiques qui font un boulot défini en informatique.

### Alphanumérique

Des chiffres et des lettres, mais pas des caractères graphiques.

### AMSOFT

La division logicielle d'AMSTRAD qui s'occupe surtout de concevoir, organiser, publier tout ce qui est nécessaire pour votre CPC464 et ses nombreuses possibilités d'extension.

### AD Analogue

Le contraire de digital. Quand quelque chose se produit d'une manière continue on parle d'un manière analogique, et comme les ordinateurs sont des appareils digitaux il faut des convertisseurs analoguedigital (A/D) avant qu'un ordinateur puisse analyser des données analogues.

## **Animation**

Dans le genre dessins animés, l'ordinateur déplace des éléments graphiques, ce qui donne l'illusion de mouvement.

## **Architecture**

Décrit l'organisation matérielle de l'ordinateur et ses périphériques, les relations avec le bus de données, les échanges internes ou externes. Un sujet ardu.

## **Argument**

Terme mathématique désignant une variable indépendante, par exemple dans l'expression  $x+y=z$ ,  $x$  et  $y$  sont les arguments.

## **Arrangements (A = array)**

Aussi appelé table, tableau, ou matrice ou encore grille. Une organisation qui a souvent deux dimensions mais peut en avoir plus, dans laquelle les données sont référencées par leur adresse dans chaque dimension.

## **Artificielle (Intelligence)**

Une structure dans la technique des programmes qui permet à un programme d'apprendre avec ses expériences passées.

## **ASCII**

Un code presque universel (?) pour représenter les nombres, lettres et autres symboles obtenus à partir du clavier ou d'un autre type de commande.

## **Assembleur**

La méthode pratique de programmation en code machine, où les instruction en code sont appelées par des mnémoniques (lettre qui suggèrent une fonction, ADD pour addition par exemple).

## **Base**

La première considération d'un mathématicien quand il joue avec les nombres: binaire pour la base 2, hexadécimal pour la base 16 et décimale pour notre vieille base 10. Plus facile à comprendre pour les enfants que pour leurs parents.

## **BASIC**

Beginners All-Purpose Symbolic Instruction Code. C'est un langage de programmation interprété et interactif, conçu pour un apprentissage facile car on peut essayer un programme à tout moment alors que pour les programmes compilés (voir ce mot) tout le programme doit être exécuté avant de pouvoir vérifier ses éléments.

## **BAUD**

Bit par seconde: une unité de mesure du passage des données dans une transmission en série.

## **BCD (binary coded decimal)**

Un système pour coder les nombres décimaux où chaque chiffre est représenté par un groupe de 4 bits.

## **Binaire**

Système de base 2, avec deux chiffres seulement, 0 et 1.

## **Binaire (nombre)**

Nombre représenté en notation binaire. Avec le CPC464 ils sont désignés avec le préfixe &X, exemple &X0101 est égal au nombre décimal 5.

## **BIT**

Autre nom des chiffres 0 et 1 dans le système binaire, que l'ordinateur utilise dans ses circuits électroniques comme s'il avait des petits interrupteurs, 0 le courant ne passe pas, 1 le courant passe.

## **BIT significatif**

Certains bits sont plus importants que d'autres dans un Octet, et on a donc des MSB et LSB: L'informatique aime, que dis-je, adore les initiales. Quand vous aurez besoin de savoir ces initiales, vous serez déjà bien avancés.

## **BOOLE (Algèbre de)**

Utilisée dans les tables de vérité, où 0 signifie faux et 1 veut dire vrai.

## **BOUCLE**

Un procédé de programmation où l'ordinateur répète l'exécution de plusieurs lignes jusqu'à ce qu'une condition soit remplie (en BASIC, commence avec **FOR** ou **WHILE**)

## **Bug**

En anglais, une petit bête qui vous gratte. En informatique, c'est ce qui fait qu'un programme se plante. En Franais, on pourrait dire 'y'a comme un défaut'

## **BUS**

Un système de connections entre l'ordinateur, les peripheriques et le monde exterieur qui transporte l'information sur l'état du microprocesseur, le RAM et d'autres éléments materiels. Le Bus du CPC464 est la prise la plus importante à l'arrière de la machine.

## **Byte**

C'est un octet qui a 8 Bits. Donc attention, c'est pas admis chez les admirateurs de Grévisse. Il faut dire Octet.

## **CAD = CAO**

Conception assistée par Ordinateur ou Computer Aided Design, c'est à peu près la même chose au langage près, s'utilise dès que les graphiques et les plans ont de l'importance. On peut modifier un petit détail sur un dessin, tout le reste est modifié pour en tenir compte.

## **Canal (A = stream)**

Une voie de communication entre l'ordinateur et les peripheriques, l'imprimante, l'écran, la cassette etc, etc...

## **Caractère**

Tout symbole qui peut être représenté, lettres chiffres et symboles graphiques (Appendice III)

## **Caractère (grille de)**

La matrice ou grille qui permet de représenter un caractère à l'affichage en illuminant certains points de la grille.

## **Caractères (jeu de)**

Toutes les lettres, nombres et symboles disponibles sur un ordinateur ou une imprimante. Si un caractère existe sur un ordinateur, cela ne veut pas dire qu'on peut l'imprimer sur une imprimante quelconque.

## **Cartouche**

Des circuits intégrés spécialement con/u contenant un progiciel (= des programmes) qui se branchent directement et sont plus rapides que les programmes sur cassette, mais plus chers.

## **Cassette**

Le support de la majorité des programmes pour les micros familiaux. Les fichiers étant les uns après les autres (séquentiels), l'accès est assez lent comparé aux disquettes.

## **Chaîne**

Souvent sous-entendu alphanumérique. Type de données comprenant des chiffres et des lettres qui peut ne pas être traitée comme une variable numérique. Peut être numérique, mais n'est considérée comme telle que si une commande le précise.

## **Chronomètre = horloge**

L'ordinateur a besoin d'être synchronisé pour toutes ses opérations. Son horloge interne travaille en micro secondes.

## **Circuit intégré = chip**

Rien avoir avec la frite à l'anglaise, c'est une puce en fran/ais qui prend le nom de microprocesseur si elle (la puce) dirige l'ordinateur.

## **Clavier**

En fait une série d'interrupteurs électroniques qui permettent de taper comme sur une machine à écrire. Clavier QWERTY d'origine anglaise, clavier AZERTY d'origine française.

## **Code Machine**

C'est un langage de programmation qui est compris par un microprocesseur, toutes les commandes étant en nombres binaires.

## **Code à Barres**

Un code qu'on trouve sur des paquets de lessive avec des barres noires plus ou moins espacées qu'on peut lire avec un crayon spécial.

## **Commande**

Instruction que le programme doit suivre.

## **Compilateur**

Un programme très complexe qui transforme un programme en langage de haut niveau (COBOL, BASIC et) en un programme en code machine directement exécutable par la machine, donc très rapide.

## **CP/M**

Control program pour Microprocesseurs. Le système d'exploitation pour disquette le plus répandu, lancé par DIGITAL RESEARCH.

## **CPU**

Central Processing Unit ou autrement dit le microprocesseur central de l'ordinateur.

## **Crayon Optique**

Crayon qui peut parfois dessiner sur l'écran, ou saisir des données, comme sur les code à barres.

## **Curseur**

Ce petit carré (ou un autre symbole), qui indique où va apparaître le prochain caractère sur l'écran.

## **Curseur (touches de contrôle du)**

Des touches qui permettent de déplacer le curseur sur l'écran, avec des petites flèches pour indiquer la direction.

## **Décimale (notation)**

C'est notre système à base 10, avec dix chiffres de 0 à 9 facile à comprendre pour nous, mais l'ordinateur ne sait compter qu'en 0 et 1.

## **Diagnostic**

Un message produit automatiquement par l'ordinateur pour indiquer et identifier une erreur dans un programme ou dans la machine elle-même.

## **Digital**

Ce qui prend des valeurs discrètes si valeurs non continues. S'oppose à analogue.

## **Digitiseur (ou digitaliseur)**

Un moyen de transmettre des informations de type analogue à un ordinateur, souvent utilisé avec les tables graphiques.

## **Disque**

En informatique, une ou deux surfaces du disque sont recouvertes d'un oxyde magnétique et le disque est un moyen de garder les données d'une manière permanente. Il prend des formes différentes: voir disquette (floppy disk) disque dur et Winchester.

## **Disquette**

Un disque magnétique souple de diamètre 8 ou 5.25 pouces, dans une enveloppe protectrice. Plus de capacités qu'une cassette, plus rapide et plus cher.

## **Disquette (Lecteur de)**

Ce qui permet d'écrire et de retrouver les données sur une disquette.

## **Disque Dur**

Disques rigides tournant plus vite, la tête ne touchant pas le disque mais flottant un peu au dessus (quelques microns) de lui.

## **Données**

Ce sont les informations que l'ordinateur transforme avec des programmes, des mots pour un traitement de textes, des chiffres pour une comptabilité, des déplacements du joystick pour un jeu.

## **Données (saisie des)**

Le terme qui décrit le rassemblement et l'entrée des données sur un ordinateur.

## **Données (base de)**

Ensemble de données ayant des liens entre elles, pouvant aller du simple fichier à une organisation complexe de dossiers reliés les uns aux autres.(A=database)

## **DOS =disk operating system = système d'exploitation du disque**

Des systèmes qui gèrent l'utilisation des disques, les plus connus étant CP/M et MSDOS.

## **Editer**

Modifier un programme, des données ou du texte.

## **Editeur**

Un programme habituellement en ROM qui permet le processus d'édition.

## **EPROM**

Un PROM qu'on peut programmer et effacer après avec des ultra-violets.

## **Expression**

Une formule simple ou complexe qu'on utilise dans un programme pour faire un calcul sur des données, l'expression définissant souvent le type de données à manipuler.

## **Fichier**

Un ensemble d'informations habituellement sur cassette ou sur disquette.

## **Firmware**

Mot anglais qui désigne du logiciel intégré au matériel comme le BASIC et le système d'exploitation en ROM.

## **Floppy disk**

Mot anglais pour disquette

## **Formatage**

Une disquette neuve est entièrement vierge; il faut donc organiser avant un système qui permet de retrouver les dossiers et leurs références dans des secteurs particuliers de la disquette.

## **Forth**

Un langage de programmation rapide, à mi-chemin entre le haut niveau et le code machine. Pas pour débutants.

## **Générateur de son**

La partie de l'ordinateur (ce peut être un microprocesseur ou du logiciel) qui crée les sons et les bruits.

## **Générations d'ordinateurs**

Les avances technologiques ont marqué les étapes de l'évolution des ordinateurs et de l'informatique. La cinquième génération est en gestation, elle devrait permettre à un ordinateur de se programmer lui-même en utilisant l'Intelligence Artificielle.

## **Graphiques**

La partie de l'affichage qui n'a pas de lien avec les caractères. Cela comprend les lignes, cercles, diagrammes et avec une bonne imprimante, on peut avoir une copie sur papier.

## **Graphique (caractère)**

Une forme conçue spécialement pour créer des images. Le CPC464 possède un jeu complet (Appendice 3).

## **Graphique (Curseur)**

Semblable au curseur de texte, mais se rapportant à l'écran graphique. Un concept invisible sur le CPC464 mais indispensable pour dessiner.

## **Graphique (mode)**

Les premiers micros avaient du mal à mixer texte et graphiques. Le CPC464 le fait avec élégance.

## **Graphique (table)**

Un appareil qui dessine et retient les coordonnées d'un point pour que l'ordinateur puisse manipuler des dessins.

## **Hardware**

Mot anglais voulant dire beaucoup de choses, notamment la machine, le matériel, tout ce qui forme l'ordinateur physique.

## **Hexadécimal**

Système des nombres à Base 16. Dans le CPC464 on les précise avec les symboles & ou &H.

## **IEEE-488**

Une des interfaces standard pour attacher des périphériques à un ordinateur.

## **Imprimante**

Ce sont des machines à écrire automatiques branchées sur l'ordinateur, grâce à un câble, qui permettent d'avoir sur papier les listings et les textes tant désirés.

## **Imprimante à marguerite**

Imprimante de très bonne qualité de frappe, les caractères sont sur les pétales de la marguerite que l'on peut changer à volonté. Peu rapides et bruyantes, indispensables pour un courrier de qualité.

## **Imprimante matricielle**

Imprime avec une grille rectangulaire de points, la plupart des imprimantes économiques ou rapides sont de ce type. (l'AMSTRAD DMP1 en est une)

## **Informatique**

Tout ce qui touche à la manipulation des ordinateurs et de l'information dans son sens large.

## **Initialiser**

Démarrer un système ou un programme en définissant les variables ou les caractéristiques de fonctionnement d'un système.

## **Instruction**

Une réquisition ou une commande donnée à l'ordinateur pour qu'il effectue une opération particulière. Un ensemble ou un succession d'instructions forment un programme.

## **Instructions (Jeu d')**

Les procédés mathématiques et logiques accomplis par le microprocesseur composent le jeu d'instruction. Une simple commande BASIC peut faire appel à des dizaines d'instructions au niveau du microprocesseur.

## **Interactif**

Un système est interactif si l'ordinateur a besoin de réponses immédiates pour continuer son programme. Les jeux sont du mode interactif car l'action du joueur modifie la suite du programme. On dit que l'ordinateur travaille en temps réel.

## **Interface**

Un mot à la mode informatique pour désigner un lien (physique ou mental) entre l'ordinateur et le monde extérieur, y compris l'utilisateur. Les interfaces du CPC464 sont par exemple le clavier, l'écran, les connections à l'arrière de la machine, sa facilité à être utilisé par vous (si le lien utilisateur-ordinateur).

## **Interface parallèle**

Le CPC464 en a une pour l'imprimante: Il y a plusieurs fils parallèles qui transmettent des informations en même temps, d'où le nom parallèle.

## **Interface série**

Par opposition à la précédente, ne peut transmettre qu'un bit à la fois d'une manière séquentielle; la plus commune est appelée RS232.

## **Interpréteur**

Par exemple, l'interpréteur du CPC464 interprète le BASIC et le traduit en code machine, ligne par ligne. Sa lenteur vient de ce fait, qu'il traduit la même ligne plusieurs fois si elle est dans une boucle.

## **I/O**

Input/output, l'équivalent anglais des entrées-sorties.

## **Itération**

Une des bases de l'ordinateur, qui divise le travail en petits éléments simples qu'il répète maintes et maintes fois. C'est aussi le système des boucles.

## **Jeux d'Arcade et d'Aventures**

Entre les envahisseurs de l'espace qui font zap, bing bang et les aventuriers qui font crac, boum, hue, il y a des plaisirs pour tout le monde. Les jeux d'aventures qui mêlent le texte, les graphiques et les contes de fées commencent à avoir leurs fans. Les Hobbits et Le Seigneur des Anneaux y cotoient les chevaliers de la Table Ronde.

## **Joysticks**

Alias manettes de jeux. Remplace les fonctions des touches curseur pour une action plus rapide dans les jeux.

## **K**

Rien de Kafkaien, juste un peu de surréalisme. Un K, ou KO, ou Kilo Octet, contient 1024 Octets ( $2^{10}$ ) Votre CPC464 a 64K de mémoire vive, ce qui lui fait 65536 Octets. (C'est du genre 13 à la douzaine).

## **Langages**

Du langage machine qui est compris par l'ordinateur au langage de haut-niveau comme le BASIC, le COBOL ou le Pascal en passant par les langages de bas niveau comme l'assembleur qui est propre à chaque microprocesseur, l'informatique baigne dans une tour de Babel où Pascal (Blaise) n'y retrouverait pas Descartes (René). Le BASIC a ses détracteurs, surtout chez les puristes mais il domine néanmoins la scène des micros.

## **Logiciel**

Tout ce qui touche aux programmes et à ce qu'on met dans la machine pour la faire tourner. Le logiciel, c'est les programmes et c'est ce qui compte le plus.

## **Logo**

Un langage simple pour apprendre les graphiques, utilisé dans le milieu éducatif pour l'apprentissage de l'informatique.

## **Manette de jeux**

Alias joystick.

## **Matrice**

Un terme utilisé en mathématique et en informatique pour représenter des arrangements de variables dans plusieurs dimensions. Donne une grille de points si on l'utilise dans le sens physique, d'où la dénomination d'imprimante matricielle.

## **Mémoire**

Un ordinateur a sa mémoire qui flanche (RAM) et sa mémoire ROM qui elle, est permanente. On peut lire et écrire dans la mémoire vive RAM, mais ce qui est dedans disparaît quand on coupe le courant. On ne peut que lire ce qu'il y a dans la mémoire morte ROM. Il y a enfin la mémoire de masse qui est représentée par les cassettes, les disquettes, disques durs qui sont des supports de mémoire.

## **Mémoire (implantation)**

L'organisation de la mémoire, montrant les différentes adresses, et l'attribution de certaines fonctions à des parties précises de la mémoire, qui concernent l'écran, le système d'exploitation etc. etc.

## **Mémoire tampon (A=buffer)**

Ou buffet: dès qu'un ordinateur communique avec un appareil plus lent que lui, il faut des tampons ou buffets pour emmagasiner les données au fur et à mesure qu'elles ont été manipulées. C'est un poste de transit temporaire, soit pour l'écran, une imprimante ou un autre périphérique.

## **Menu**

Bien que la cuisine informatique n'ait pas l'attrait de celle de Bocuse, on utilise souvent des menus pour que l'utilisateur choisisse la substantifique moelle. Autrement dit une liste d'options qui s'affiche sur l'écran. (Un des rares mots que les Anglais nous ait piqué pour l'informatique.)

## **Micropuce**

Le cœur de l'ordinateur. Dans le CPC464 c'est un Zilog Z80A qui exécute les instructions que lui traduit l'interpréteur BASIC.

## **Modem**

Abréviation de MOdulateur DEModulateur; on l'utilise pour les communications à distance entre ordinateurs. Cet appareil peut être acoustique ou électronique.

## **Moniteur**

L'écran dédié d'un ordinateur, par opposition à un écran de Télévision qui doit généralement être modifié ou adapté pour afficher les images venant d'un ordinateur.

## **Nibble**

Mot anglais signifiant petit boucheé grignotée: il représente quatre bits d'un Octet, donc un demi-Octet!

## **Octal**

Système à base 8, où chaque chiffre (0-7) peut être écrit avec trois bits.

## **Pascal**

Un langage de haut-niveau, très prisé par Académie, et qui doit être compilé avant d'être exécuté, donc très rapide. L'étape suivante pour les étudiants du BASIC, très souvent.

## **PEEK**

Une fonction BASIC qui regarde à une adresse pour voir le contenu et la valeur à cette adresse.

## **PERIPHERIQUE**

Tous les appareils qui peuvent se brancher sur un ordinateur et développer ses capacités

## **Pixel**

La plus petite portion d'écran pouvant être contrôlée par le matériel.

## **POKE**

Le petit frère de PEEK, mais lui ce n'est pas un voyeur c'est un écrivain: il écrit une valeur à une adresse donnée.

## **Progiciels**

Ce sont des programmes d'applications spécialement développés pour une tâche particulière, tout prêt à l'emploi. Il y a des progiciels pour gérer l'alimentation équilibrée des porcs ou des progiciels un peu plus chers pour gérer une centrale atomique.

## **Programme**

Une suite d'instructions que l'utilisateur n'a même pas besoin de voir pour que le programme s'exécute. Peut avoir une ligne seulement, ou quelques milliers comme un programme de traitement de textes.

## **PROM**

Programmable ROM. Un circuit intégré sur lequel on peut écrire une seule fois; après, c'est devenu un ROM.

## **RAM**

Random Access Memory = Mémoire Vive = MEV = une mémoire sur laquelle on peut écrire et lire des données, un des éléments principaux de tout micro.

## **RANDOM**

Mot anglais qui signifie au hasard. Il est souvent utile de pouvoir produire des nombres au hasard. Avec le CPC464 on utilise les fonctions RND et RANDOMISE.

## **Registre**

Un endroit de la mémoire du microprocesseur (CPU) qui est utilisé pour stocker temporairement pendant des instructions.

## **REMarque**

Un commentaire qui ne modifie en rien l'exécution du programme, bien utile pour comprendre un programme quand on ne l'a pas écrit, ou quand on l'a écrit et oublié.

## **Réservé (mot)**

Ou mot-clé. Un mot qui doit être sans faute et avec un espace après.

## **Résolution**

La capacité à distinguer des éléments d'affichage proches les uns des autres. Le CPC464 a une haute résolution de 640 x 200 ce qui veut dire qu'on peut aligner 640 points horizontalement et 200 verticalement.

## **ROM**

Read Only Memory - mémoire morte = MEM. Mémoire figée que l'on peut lire seulement, où se trouve généralement des langages, des systèmes d'exploitation ou des programmes dont on a besoin fréquemment (cartouches ROM)

## **ROUTINE**

Programme qu'on utilise fréquemment, souvent un sous-programme auquel on fait souvent appel.

## **RS232C**

L'interface série la plus répandue, surtout utilisée pour les communications. Les appareils aux deux extrémités de cette interface doivent être configurés (déterminés dans leurs échanges) pour pouvoir se parler.

## **Séparateur**

Le plus souvent, un simple espace, mais aussi un signe d'opération peut servir de séparateur.

## **Software**

Le mot anglais pour logiciel.

## **Sortie**

Toute donnée qui provient de l'ordinateur après avoir été manipulée. Désigne aussi les interfaces qui permettent de 'sortir' les données de l'ordinateur.

## **Sous-programme**

Voir routine

## **Tableur**

Un logiciel très prisé par les gestionnaires, appelé spreadsheet par les anglais, qui permet d'avoir des colonnes et des lignes de calculs et de faire pleins de calculs scientifiques et financiers avec ces chiffres.

## **Terminal**

Autre nom d'un clavier avec ou sans moniteur qui permet d'"entrer" des données dans l'ordinateur principal.

## **Traitement de textes**

C'est un type de programme ou un logiciel. Indispensable dès qu'on veut écrire une lettre ou rédiger un document. Leur complexité et leurs capacités varient avec le prix et l'ordinateur auxquels ils sont destinés.

## **Virgule flottante**

Terme informatique qu'on utilise quand on veut représenter les grands nombres dans les calculs scientifiques.



# Appendice X

## Accents et traitements de textes

En Français, il y a beaucoup d'accents et les machines à écrire AZERTY permettent de taper les lettres accentuées. Avec la plupart des machines QWERTY (donc d'origine Anglo-Saxonne) les lettres accentuées sont difficiles à produire (il faut quelquefois appuyer sur trois touches en même temps pour avoir un accent).

Avec le CPC464, il existe une solution élégante qui utilise le pavé numérique et est compatible avec l'imprimante AMSTRAD DMP1, qui imprimera les lettres accentuées que vous voyez sur l'écran (à condition que vous ayez mis les interrupteurs - dip switches - derrière l'imprimante pour le mode d'impression Français, voir dans le manuel spécifique de l'imprimante.)

Il suffit d'utiliser une des grilles de la page suivante (vous pouvez en faire une photocopie si vous ne voulez pas découper cette feuille) en la fixant sur du carton et, en la mettant autour du pavé numérique.

Il faut ensuite taper le programme suivant, qui vous donnera les lettres accentuées; ce programme est déjà inclus dans le logiciel de traitement de textes AMLETTRES, qui vous permet l'utilisation du CPC464 pour vos lettres et documents.

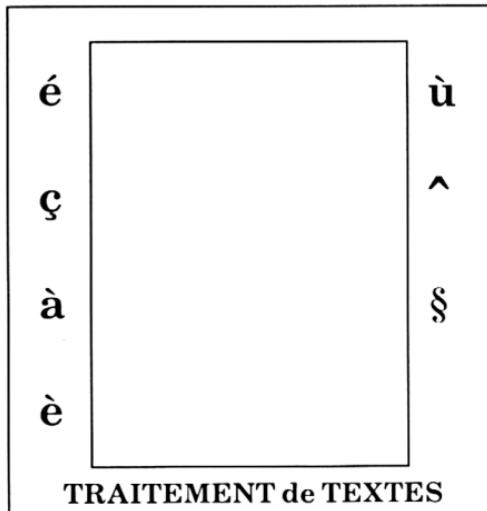
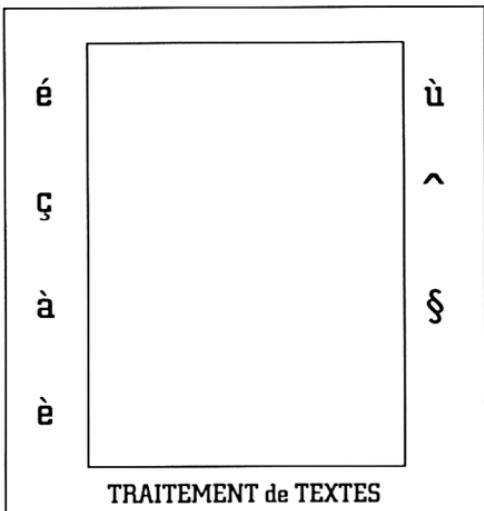
```
10 REM accents
20 SYMBOL AFTER 64
30 SYMBOL 125,&60,&10,&3C,&66,&7E,&60,&3C
40 KEY 135, CHR$(123)
50 SYMBOL 123,&6,&8,&3C,&66,&7E,&60,&3C
60 KEY 128, CHR$(125)
70 SYMBOL 64,&60,&10,&78,&C,&7C,&CC,&76
80 KEY 129, CHR$(64)
90 SYMBOL 92,&80,&80,&3C,&66,&60,&3E,&8,&18
100 KEY 132, CHR$(92)
110 SYMBOL 124,&30,&8,&66,&66,&66,&66,&3E
120 KEY 137, CHR$(124)
130 SYMBOL 94,&18,&24
140 KEY 134, CHR$(94)
150 SYMBOL 93,&1C,&32,&38,&6C,&38,&10,&90,&60
160 KEY 131, CHR$(93)
```

(Rappelez vous que **SYMBOL AFTER** redéfinit les caractères que vous auriez pu avoir défini dans le précédent programme). Ce programme définit les lettres suivantes:

é à è ù plus l'accent circonflexe (à utiliser avec un sous programme si on le veut sur la lettre) et le symbole du paragraphe.



**GRILLES**  
pour TRAITEMENT de TEXTES





# INDEX

\$ .....	page 2.8,page 8.6,page 4.7,page 8.2
% .....	page 2.8,page 8.6,page 4.6
*	page 2.8,page 2.10,page 8.6
& .....	page 2.8,page 8.1,page 8.6,page 4.6,page 8.2
?	page 3.2
&H .....	page 3.4,page 8.1
&X .....	page 3.4,page 8.1
! .....	page 4.6,page 8.2
# .....	page 8.1
: .....	page 8.1
<b>ABS</b> .....	page 8.4
<b>AFTER</b> .....	page 10.1
<b>AFTER</b> .....	page 8.4
<b>AND</b> .....	page 4.18
<b>ASC</b> .....	page 8.4
<b>ASCII</b> .....	page 1.6
<b>ASCII</b> .....	page A3.1
<b>ATN</b> .....	page 8.4
<b>AUTO</b> .....	page 8.5
<b>Accents</b> .....	Appendice X
<b>Adaptateur Péritel</b> .....	page 1.5
<b>Addition</b> .....	page E2.10
<b>Apostrophe'</b> .....	page 3.2
<b>Arrangements</b> .....	page 4.13
<b>BASIC</b> .....	Chapitre 3
<b>BASIC</b> .....	page A2.2
<b>BINS</b> .....	page 8.5
<b>BORDER</b> .....	page 8.5
<b>BORDER</b> .....	page E3.1
<b>Bienvenue (cassette de)</b> .....	page E1.6
<b>Binaire</b> .....	page A2.4
<b>Bits</b> .....	page A2.3
<b>* Break *</b> .....	page 1.4
<b>Bruit</b> .....	page E3.18

CALL .....	page 8.5
CAPS LOCK .....	page E2.2
CAT .....	page 8.6
CATalogue .....	page 2.7
CERCLES .....	page E3.11, E3.12
CHAIN .....	page 8.6
CHAIN MERGE .....	page 8.6
CHRS .....	page 8.7
CHRS(n) .....	page 1.6
CINT .....	page 8.7
CLEAR .....	page 8.7
CLOSEIN .....	page 8.8
CLOSEOUT .....	page 8.8
CLR .....	page E2.2
CLG .....	page 8.7
CLS .....	page 8.8
CLS .....	page E2.4
CONT .....	page 4.6
CONT .....	page 8.8
COPIE avec curseur .....	page E2.8
COS .....	page 8.9
CPU .....	page A4.1
CREAL .....	page 8.9
Calculs composés .....	page E2.12
Canal .....	page 3.4
Caractères .....	page A3.1
Caractères d'expansion .....	page A3.15
Caractères de contrôle .....	page 9.1, 9.2, 9.3, 9.4
Carte de la mémoire .....	page A4.7
Cassette de Bienvenue .....	page 2.4
Charge Rapide .....	page 2.5
Chargement .....	page 2.1
Chargement de cassette .....	page 2.3
Chronomètre .....	page 10.1
Chronomètre .....	page A4.2
Connections .....	Appendice V
Conventions .....	page 1.1
Couleurs .....	page E3.1
Curseur de COPY .....	page 1.13
Curseur graphique .....	page 5.7

DATA .....	page 4.14
DATA .....	page 8.10
DEF FN .....	page 8.10
DEFINT .....	page 8.10
DEFREAL .....	page 8.10
DEFSTR .....	page 8.10
DEG .....	page 8.11
DEL .....	page 8.11
DEL .....	page E2.1
DI .....	page 10.2
DI .....	page 8.11
DIM .....	page 4.1
DIM .....	page 4.13
DIM .....	page 8.12
DMP1 .....	page 7.4
DRAW .....	page 8.12
DRAW .....	page E3.10
DRAWR .....	page 8.12
Data .....	Chapitre 4
Dataorder .....	Chapitre 2
Dataorder .....	page E1.8
Dictionnaire .....	Appendice IX
Division .....	page E2.11
Donnée .....	page 4.14
Données .....	Chapitre 4

EDIT .....	page 1.13
EDIT .....	page 8.13
EDIT .....	page E2.7
EDITer .....	page E2.7
EI .....	page 10.
EI .....	page 8.13
ELSE .....	page 8.18
END .....	page 8.13
ENT .....	page 6.8
ENT .....	page 8.14
ENT .....	page E3.17
ENTER .....	page E2.1
ENTREES SORTIES .....	Appendice V
ENV .....	page 6.7
ENV .....	page 8.14
ENV .....	page E3.16
EOF .....	page 8.15
ERASE .....	page 8.15
ERL .....	page 8.15

ERR .....	page 8.15
ERROR .....	page 8.16
ESC .....	page E2.3
EVERY .....	page 10.2
EVERY .....	page 5.9
EVERY .....	page 8.16
EXP .....	page 8.16
Enveloppe de ton .....	page E3.17
Enveloppe de ton .....	page 6.8
Enveloppe de volume .....	page 6.7
Enveloppe de volume .....	page E3.16
Erreurs de lecture .....	page 2.1
Erreurs de lecture .....	page 2.8
Expansions .....	page A1.2
Extensions .....	page A1.2
FENETRES .....	Chapitre 5
FIX .....	page 8.17
FLUSH .....	page 6.6
FOR .....	page 8.17
FOR TO NEXT .....	page E2.9
FRE .....	page 8.17
Fenêtre .....	page 5.10
Flash .....	page E3.5
Fréquence .....	page 6.2
GOSUB .....	page 8.17
GOSUB RETURN .....	page E3.13
GOTO .....	page 8.18
GOTO .....	page E2.5
GRILLES et fenêtres .....	Appendice VI
Graphiques .....	Chapitre 5
Graphiques .....	page E3.7
HEXS .....	page 8.18
HIMEM .....	page 8.18
Hauteur .....	page 6.2
Hexadecimal .....	page A2.5
Horloge .....	page 10.1

IF THEN .....	page 4.12
IF THEN .....	page E2.8
IF THEN ELSE .....	page 8.18
INK .....	Chapitre 5
INK .....	page 8.19
INK .....	page E3.1
INKEY .....	page 8.19
INKEYS .....	page 8.20
INP .....	page 8.20
INPUT .....	page 8.20
INPUT .....	page E2.6
INSTR .....	page 8.21
INT .....	page 8.21
Implantation mémoire .....	page A4.7
Imprimantes .....	page 7.3
Interpréteur .....	page A1.2
Interruptions .....	page 9.6
JOY .....	page 8.22
Jeu de Caractères .....	Appendice III
Joystick .....	page E1.6
Joysticks .....	page 7.2
KEY .....	page 8.22
KEY DEF .....	page 8.22
KO .....	page A2.4
LEFT\$ .....	page 8.23
LEN .....	page 8.23
LET .....	page 8.23
LINE INPUT .....	page 8.24
LIST .....	page 1.11
LIST .....	page 8.24
LIST .....	page 8.3
LIST .....	page E2.5
LOAD .....	page 8.24
LOCATE .....	page 4.11
LOCATE .....	page 8.24
LOCATE .....	page E3.8
LOG .....	page 8.25
LOG10 .....	page 8.25

MEMORY .....	page 8.26
MERGE .....	page 8.26
MIDS .....	page 8.26
MIN .....	page 8.27
MODE .....	Chapitre 5
MODE .....	page 8.27
MODE .....	page E3.1, E3.4
MODES .....	page 5.3
MOVE .....	page 8.27
MOVER .....	page 8.27
Manette de jeux .....	page 7.1
Manette de jeux .....	page E1.6
Matrice .....	page A2.4, 4.13
Messages d'erreur .....	Appendice VIII
Mot-Clé .....	page 8.2
Mot-clé associé .....	page 8.2
Mots réservés .....	Appendice VIII
Mots réservés .....	page 4.1
Mots-clés .....	Appendice VIII
Multiplication .....	page E2.10
NEW .....	page 8.28
NEW .....	page E3.12
NEXT .....	page 8.28
NUMERIQUE .....	page 8.37
Nibble .....	page A2.5
Notes .....	Appendice VII
Numéros de touches .....	page A3.16
ON BREAK GOSUB .....	page 8.29
ON BREAK STOP .....	page 8.29
ON ERROR GOTO .....	page 8.30
ON GOSUB .....	page 8.28
ON GOTO .....	page 8.28
ON SO GOSUB .....	page 6.5
ON SO GOSUB .....	page 8.30
OPENIN .....	page 8.30
OPENOUT .....	page 8.31
OR .....	page 4.18
ORIGIN .....	page 8.31
ORIGIN .....	page E3.13
OUT .....	page 8.31
Octet .....	page A2.3

Période de ton .....	page E3.15
Périodes .....	Appendice VII
PAPER .....	Chapitre 5
PAPER .....	page 8.32
PAPER .....	page E3.1
PEEK .....	page 8.32
PEN .....	Chapitre 5
PEN .....	page 8.33
PEN .....	page E3.1
PI .....	page 8.33
PLAY .....	page 2.2
PLOT .....	page 8.34
PLOT .....	page E3.10
PLOTR .....	page 8.34
POKE .....	page 8.35
POS .....	page 8.35
PRINT .....	page 3.2
PRINT .....	page 8.35, 8.36
PRINT .....	page E2.4
PRINT TAB .....	page 3.6
PRINT USING .....	page 3.4
PRINT USING .....	page 8.37
Position legale du curseur .....	page 9.1
Puissances .....	page E2.11 & E2.13
 Queues .....	page 6.5

Résolution .....	page A1.1
RAD .....	page 8.38
RANDOMIZE .....	page 8.38
READ .....	page 8.38
REC .....	page 2.2
RELEASE .....	page 6.5
RELEASE .....	page 6.9
RELEASE .....	page 8.39
REM .....	page 8.39
REMAIN .....	page 10.3
REMAIN .....	page 8.39
RENUM .....	page 4.8
RENUM .....	page 8.40
RESTORE .....	page 8.40
RESUME .....	page 8.40
RETURN .....	page 8.41
REW .....	page 2.2

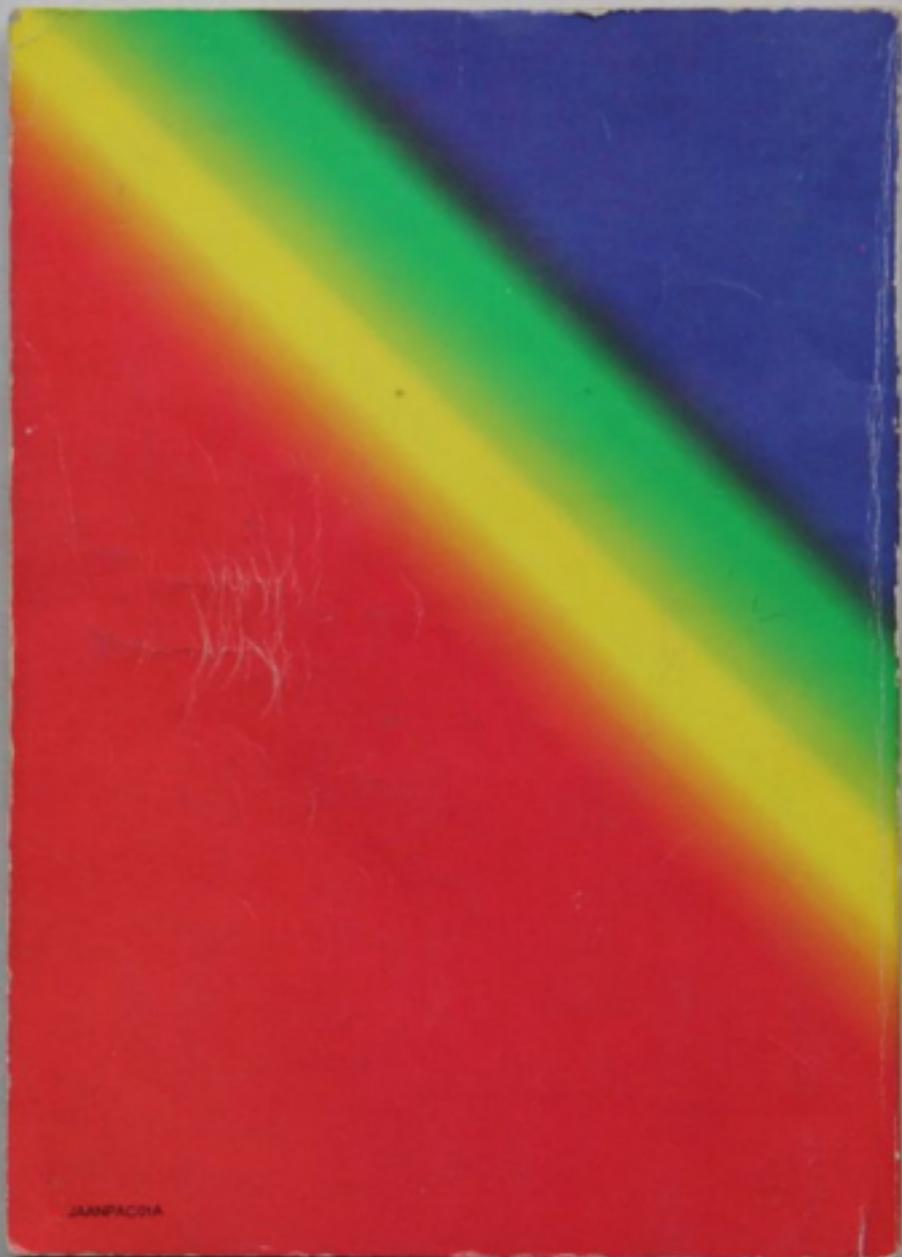
RIGHTS .....	page 8.41
RND .....	page 8.41
ROUND .....	page 8.42
RUN .....	page 8.42
RUN .....	page 8.43
RUN .....	page E2.4
Racine carrée .....	page E2.11
Racines cubiques .....	E2.11
Rendezvous .....	page 6.4
ROM .....	page A1.3

SAVE .....	page 2.6, 2.7
SAVE .....	page 8.43
SGN .....	page 8.43
SHIFT .....	page E2.1
SIN .....	page 8.44
SOUND .....	Chapitre 6
SOUND .....	page 8.44
SOUND .....	page E3.15
SPACES .....	page 8.44
SPEED INK .....	page 8.44
SPEED KEY .....	page 8.45
SPEED WRITE .....	page 8.45
SO .....	page 6.5
SO .....	page 8.45
SOR .....	page 8.46
STOP .....	page 8.46
STOP/EJECT .....	page 2.2
STRS .....	page 8.46
STRING .....	page 8.37
STRINGS .....	page 8.46
SYMBOL .....	page 8.47
SYMBOL AFTER .....	page 1.10
SYMBOL AFTER .....	page 8.47
Sauvegarde .....	page 2.1
Sauvegarde .....	page 2.6
Son et effets sonores .....	Chapitre 6
Son .....	page E3.15
Soustraction .....	page E2.10
Super Prudent .....	page 2.5
Syntax error .....	page 4.1
Système d'exploitation .....	page 9.5

TAG .....	page 8.48
TAGOFF .....	page 8.48
TAN .....	page 8.48
TEST .....	page 8.49
TESTR .....	page 8.49
TIME .....	page 8.49
TROFF .....	page 8.49
TRON .....	page 8.49
Terminologie .....	page 3.3
Touches redéfinissables .....	page A4.3
Traitemet de Textes .....	Appendice X
Transparence .....	page 5.2
UNT .....	page 8.50
UPPERS .....	page 8.50
VAL .....	page 8.50
VPOS .....	page 8.50
Valeurs ASCII par défaut .....	page A3.14
Variables .....	Chapitre 4
Verrouillage des majuscules .....	page E2.2
WAIT .....	page 8.51
WEND .....	page 8.51
WHILE .....	page 8.52
WIDTH .....	page 8.52
WINDOW .....	page 8.52
WINDOW .....	page 8.53
WINDOWS .....	Chapitre 5
WINDOWS .....	page 5.10
WRITE .....	page 8.53
XOR .....	page 4.18
XPOS .....	page 5.7
XPOS .....	page 8.53
YPOS .....	page 5.7
YPOS .....	page 8.53
ZONE .....	page 3.6
ZONE .....	page 8.54







AMSTRAD

CPC 464

CPC 464

AMSTRAD

AMSTRAD

CPC 464

GUIDE  
de  
L'UTILISATEUR





Document numérisé avec amour par

# AMSTRAD

CPC



# MÉMOIRE ÉCRITE



<https://acpc.me/>