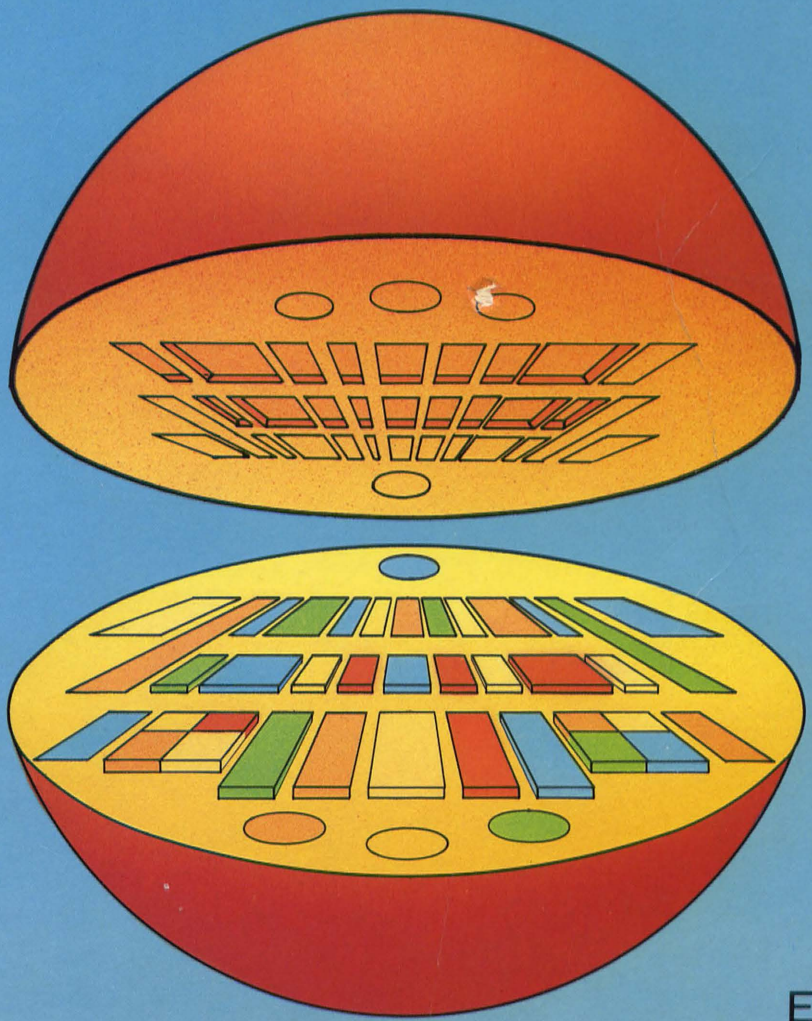
TOUT SAVOIR SUR ATMOS



EYROLLES

ROGER POLITIS . BRUNO VANRYB

COLLECTION MICROPLUS

TOUT SAVOIR SUR ATMOS

par

Roger POLITIS
Bruno VANRYB

EYROLLES 61, boulevard Saint-Germain – 75005 PARIS
1984

Introduction

En 1982, la société anglaise Tangerine créait une mini-révolution dans le monde de la Micro-informatique "familiale" avec le lancement de son Oric-1. La machine se démarquait nettement par rapports à ses concurrents directs, en raison notamment de son Basic extrêmement complet pour cette gamme de prix.

Les mêmes récidivent en 1984, avec l'Atmos qui est en quelque sorte l'Oric de la deuxième génération: L'expérience acquise sur l'Oric-1 a été mise à profit, les défauts corrigés, les quelques lacunes comblées. Cela va du clavier, qui est maintenant entièrement mécanique du type "machine à écrire", jusqu'aux quelques instructions qui manquaient encore au Basic pour exploiter pleinement l'Ordinateur. On trouve ainsi maintenant des instructions spéciales pour la sauvegarde de variables, le positionnement curseur, et même la fusion de programmes!

Les différences entre les deux machines sont en fait souvent minimes, mais relativement nombreuses. Pour cette raison, cet ouvrage a été conçu selon le même principe que "Tout savoir sur Oric" en exploitant les nouvelles

possibilités créées par l'Atmos. Cette fois encore, il s'agit d'un ouvrage d'approfondissement et non d'initiation.

Il est orienté selon trois axes:

— *L'exploitation la plus complète des possibilités du Basic, aussi bien dans le domaine des fonctions "standard" que dans ceux, plus spécialisés, du graphique et du son.*

— *La compréhension de la structure et de l'organisation de la mémoire, que ce soit au niveau du stockage des programmes, des écrans texte et graphique ou des jeux de caractères. Cela débouche sur l'utilisaion extensive des pointeurs et des variables-système, permettant d'obtenir nombre de fonctions impossibles en Basic.*

— *Enfin, sur la programmation concrète. En effet, une très large part est faite aux programmes, qui ne sont pas des "exemples" inutilisables mais bien des véritables programmes d'applications, remplissant des fonctions précises d'une manière intéractive et avec une présentation attrayante.*

Vous trouverez ainsi une gestion de carnet d'adresse, un générateur de caractères très performant, un mini-traitement de textes et un programme complet de D.A.O., entre autres... Et les amateurs de jeux n'ont pas non plus été oubliés, qui découvriront plusieurs exemples d'applications, trucs et astuces, ainsi qu'un grand jeu en temps réel du type "jeu d'arcades".

Table des matières

1

Le Basic de l'Oric Atmos

1.1. Introduction

Ce chapitre n'a pas pour but de présenter in extenso toutes les commandes Basic dont on peut trouver l'explication dans le Manuel Atmos, par ailleurs fort bien fait. Nous nous étendrons plutôt sur toutes celles d'entre elles qui, soit parce qu'elles sont peu fréquentes, soit parce que leur utilisation n'est pas évidente au premier abord, nécessitent une définition plus détaillée.

D'autre part, et malgré le fait que le Basic de l'Atmos se soit encore étendu par rapport à celui de son "frère aîné" l'Oric-1, certaines commandes ou fonctions lui font encore défaut. Les plus importantes d'entre elles sont remplacées par des "astuces" ou des courts programmes, effectuant la même tâche.

1.2. Quelques rappels

Avant toutes choses, nous pensons qu'il est utile de préciser quelques notions de base, sur le fonctionnement des micro-ordinateurs, et celui de l'Atmos en particulier.

Celui-ci utilise un langage évolué, le Basic, pour communiquer avec nous. Les ordres donnés en Basic sont ensuite interprétés et exécutés un à un. Le programme interne chargé de cette traduction s'appelle l'**interpréteur Basic**, et il a été écrit une fois pour toutes : On ne peut pas le modifier, et la mémoire e contenant s'appelle mémoire morte ou MEM en français, et ROM (Read Only Memory) en anglais. La syntaxe du langage Basic se compose d'un ensemble d'**Instructions**, de **Fonctions**, d'**Opérateurs** et d'**Opérandes** ou **Arguments** ; voici l'explication de ces termes :

1.2.1. Instructions et fonctions

— Une **Instruction** est un mot-clé compris par le Basic, qui implique l'exécution d'une tâche précise par l'ordinateur. Certaines instructions doivent être suivies d'un argument, certaines non.

Exemples :

Dans PRINT"BONJOUR", PRINT est l'instruction et "BONJOUR" est l'argument. Mais PRINT peut aussi être utilisé seul.

Par contre, CLS (Effacement de l'écran) est une instruction utilisée toujours seule, alors que INPUT implique nécessairement l'emploi d'un ou plusieurs arguments.

— Une **Fonction** est également un mot-clé, mais il s'agit plutôt d'une "question" que l'on pose à l'ordinateur : celui-ci doit nous retourner une valeur en réponse. Une fonction s'utilise toujours à la suite d'une instruction, en général PRINT ou LET.

Exemples :

Dans PRINT SIN(PI/2), SIN est la fonction et PI/2 son argument.

A=HEX$(1ØØ) donnera à la variable A la valeur hexadécimale de l'argument, ici le nombre 1ØØ. (N'oublions pas que A=X est équivalent à LET A=X).

1.2.2. Variables et constantes

— Les **Opérandes** sont les valeurs, nombres ou séquences de caractères, servant d'argument aux instructions et fonctions. Les opérandes numériques sot de deux sortes, les constantes et les variables:

Les constantes, qui peuvent être soit décimales en notation ordinaire (1ØØ, 3.146, etc.) ou scientifique (2.2E3, 4.7E—12, ces deux nombres étant équivalents à 2.2 $1Ø^3$ et 4.7 $1Ø^{-12}$), soit hexadécimales si elles sont précédées du symbole # (#2Ø, #B1ØØ, etc.).

Les variables, qui contrairement aux constantes peuvent changer de valeur en cours de programme. Celle-ci peut être définie par l'utilisateur, ou représenter le résultat d'un calcul. Une variable est caractérisée par un nom comportant au maximum deux caractères alphanumériques, dont le premier doit être une lettre. Deux types de variables sont disponibles sur Atmos:

1) Les variables simples en virgule flottante, dont la valeur peut être comprise entre 2.9 $1Ø^{-39}$ et 1.7 $1Ø^{+38}$ et comporte 9 chiffres significatifs. A1, N, KM sont des noms de variables simples.

2) Les variables entières, qui suivent les mêmes règles d'appellation que les variables simples avec la différence que leur nom doit être suivi du symbole %. Elles ne peuvent contenir que des nombres entiers, dont la valeur doit être comprise entre —32768 et +32767.
A%, R2% ou AG% sont des noms de variables entières.

Nous avons enfin les opérandes littérales, qui ne sont plus considérées comme des nombres, mais comme des suites de caractères. Il s'agit bien sûr des **chaînes de caractères**, et elles sont aussi subdivisées en constantes et variables:

Une *constante de chaîne* est une suite quelconque de caractères pouvant aller jusqu'à 255, toujours incluse entre guillemets (pour la distinguer d'un nom de variable).

"PROGRAMME 1", "TOTO", "1234QWERTY" sont des constantes de chaîne.

Les noms des *variables de chaîne* suivent les mêmes règles que ceux des variables numériques, mais ils doivent être suivis du symbole $ (dollar) : A$, F1$, etc.

Remarquons que tous les types de variables peuvent être constitués en tableaux. Par exemple, A(3,3) représente un tableau de nombres à deux dimensions que l'on peut assimiler à un damier de 3×3=9 cases, chacune d'entre elles contenant un nombre. Le premier élément ("en haut à gauche") sera A(1,1), et le dernier ("en bas à droite") sera A(3,3). Les nombres entre parenthèses, permettant d'appeler individuellement chaque élément d'un tableau s'appellent des indices: un élément de tableau s'appelle une variable indicée, et il est à noter que les indices peuvent être des constantes (comme dans notre exemple) ou des variables. Atmos accepte les tableaux comportant un nombre quelconque de dimensions. Bien entendu, tous les éléments d'un tableau doivent être de même nature: variables simples, variables entières ou chaînes de caractères. Pour un tableau de chaînes à deux dimensions, il suffit de spécifier le nombre de chaînes qu'il contiendra; la longueur de chacune d'elles est laissée libre.

Enfin, il est intéressant de savoir que des variables de types différents peuvent avoir le même nom: L'Atmos ne confondra pas A, A%, A$, A(2Ø) ou A$(1ØØ).

1.2.3. Les opérateurs

Les Opérateurs sont de trois types:

1) *Les opérateurs arithmétiques*, que tout le monde connaît bien: +, —, * (multiplié par), / (divisé par) et ↑ (élévation à une puissance). Ces opérateurs s'utilisent de la manière normale sur les constantes et variables numériques, mais remarquons que "+" peut aussi s'utiliser sur les chaînes. Les chaînes ne sont pas additionnées, mais "concaténées", c'est-à-dire mise l'une à la suite de l'autre.

Exemple:

```
Si A=123 et B=456, alors A+B=579. Par contre, si
   A$="123" et B$="456", A$+B$="123456".
```

2) *Les opérateurs relationnels*, >, >=, <, <=, =, <> (supérieur, supérieur ou égal, inférieur, inférieur ou égal, égal, différent). Ces opérateurs

peuvent s'utiliser indifféremment sur des nombres ou des chaînes. Dans le cas de deux chaînes, celles-ci sont comparées caractère par caractère, et sont classées par ordre alphabétique comme dans un dictionnaire, une lettre précédant une autre étant considérée comme son inférieure. Pour Atmos, une condition VRAIE vaut −1, une condition FAUSSE vaut Ø.

Exemples:

— 5>2 est vrai, donc l'expression vaut −1.

— "TOTO"<"JOJO" est faux, donc l'expression:
(5>2)="TOTO">"JOJO" est VRAIE, ainsi que l'expression:
(5>2)=NOT ("TOTO"<"JOJO").

3) *Les opérateurs logiques*, AND, OR et NOT (ET, OU et NON) s'utilisent pour combiner des expressions utilisant des opérateurs relationnels.

Exemple:

```
IF A>2 AND A<5 THEN PRINT A      signifie:
SI A EST COMPRIS ENTRE 2 ET 5 ALORS PRINT A.
```

1.2.4. Mode direct et mode programme

Dans tout ce qui va suivre, nous nous référerons constamment au mode "direct" et au mode "programme". Rappelons que toute instruction ou séquence d'instructions entrée au clavier et non précédée d'un nombre, est exécutée dès que la touche RETURN est tapée: C'est le mode direct.

Par contre, si une séquence d'instructions est précédée d'un nombre, celui-ci sera considéré comme un numéro de ligne. Le fait de frapper RETURN n'exécutera pas les instructions,mais les **mémorisera** pour une exécution ultérieure: C'est le mode "différé", ou mode programme; il faudra taper l'instruction RUN pour exécuter cette séquence.

Précisons aussi qu'Atmos accepte des lignes de programme de 78 caractères maximum. Une même ligne peut contenir plusieurs instructions si celles-ci sont séparées par des ":", et les commandes et noms de variables doivent être tapés en majuscules pour être acceptés par le Basic (les

minuscules ne sont utilisables que dans des chaînes de caractères, leur emploi dans une commande provoque une erreur de syntaxe).

Une forme abrégée est acceptée pour deux Instructions du Basic:

PRINT peut être remplacé par "?" (point d'interrogation).
REM (REMarque) peut s'écrire "'" (apostrophe).

Enfin, les espaces entre les mots ne sont pas significatifs, contrairement à la pratique courante dans beaucoup d'autres Basic comme le Microsoft. On pourra donc les omettre pour tirer le meilleur parti de la mémoire ou de la longueur de ligne disponible, mais n'oubliez pas que la lisibilité de vos programmes s'en ressentira. (Dans tous les programmes présentés ci-après, des espaces ont été introduits entre les diverses commandes pour des raisons de clarté).

1.2.5. Comment augmenter l'efficacité de vos programmes

Voici quelques renseignements qui intéresseront certainement les fanas de l'efficacité:

— Les variables normales en virgule flottante occupent 7 octets en mémoire, contre 4 pour les variables entières. Par contre et paradoxalement, les variables normales sont traitées plus rapidement, plus vite même que les constantes.

— On ne peut utiliser une variable entière pour contrôler une boucle (FOR-NEXT), etc.). Leur usage est donc surtout conseillé lorsque vous voulez forcer une variable à prendre une valeur entière, en évitant d'utiliser INT. Notons que le résultat est le même que celui fourni par INT, c'est-à-dire arrondi à l'entier inférieur et non pas un arrondi exact.

— Par contre, un tableau de variables entières n'occupera que les 2/5^{e} de l'espace mémoire exigé pour un tableau identique en virgule flottante: 2 octets par élément contre 5; à utiliser de préférence quand l'occupation de la mémoire est critique!

— Le mode de stockage des tableaux de chaînes est identique à celui du Basic Microsoft: La longueur de chaque élément du tableau n'est pas fixée d'avance, mais peut évoluer en cours de programme. Cette solution s'avère la meilleure en ce qui concerne l'espace occupé (il faut tout de même 3 octets par élément), mais pas pour la rapidité de l'accès et du traitement.

— En règle générale, et si la vitesse est primordiale, utilisez des variables normales attribuées le plus tôt possible dans le programme et dont le nom ne comporte qu'un seul caractère. Les tableaux sont traités plus rapidement si des variables simples sont utilisées comme indices.

1.3. *Conventions d'écriture*

Pour faciliter la compréhension, nous présenterons par la suite les divers mots-clés du Basic de la manière suivante:

Syntaxe: Donne la syntaxe complète de la commande.

Application(s): Donne les détails de son utilisation.

REMARQUES: Dans cette rubrique seront exposées les particularités d'utilisation de la commande, ou les différences avec le Basic Microsoft qui sera pris comme référence en raison de sa très large diffusion.

Exemple(s) : Selon la complexité des cas, un ou plusieurs exemples seront donnés.

— Tous les mots écrits en majuscules représentent les mots-clés.

— Les mots écrits en minuscules et compris entre < > sont les données à fournir par le programmeur.

— Les données facultatives sont incluses entre crochets [ ].

— Toutes les ponctuations, hormis les deux précédentes, font partie intégrante de la syntaxe et doivent être tapées telles quelles.

— Les données que l'on peut répéter sont suivies de (...).

— En ce qui concerne les nombres, nous suivront la syntaxe ORIC: Les nombres décimaux seront écrits normalement, et les nombres hexadécimaux seront précédés de #.

  — X, Y, Z représentent des nombres en virgule flottante,

  — N, M des nombres entiers,

  — A représente une adresse-mémoire (entier compris entre Ø et 65535),

  — X$, A$, B$ des chaînes de caractères quelconques.

***Exemple*:**

```
INPUT [<"commentaire";>]<variable>[<,variable>,(...)]
```

— L'instruction INPUT est la commande Basic.

— Le commentaire est optionnel, sinon il doit être entre guillemets " " et suivi de ; .

— Au moins une variable doit être donnée comme argument de l'instruction.

— On peut si on le désire mentionner plusieurs variables, en les séparant par des virgules.

Ainsi, il serait correct d'écrire:

INPUT"Donner votre âge et votre mois de naissance";AG,MO$

1.4. *Les instructions du Basic*

Afin de faciliter la consultation de ce chapitre, voici la liste des instructions qui y sont présentées, avec un résumé de leur utilisation:

CALL	Appel d'une routine en langage-machine.
CSAVE	Sauvegarde de programmes ou zones-mémoire.
CLOAD	Chargement de programmes ou zones-mémoire.
DEF FN	Définition d'une fonction utilisateur.
DEF USR	Définition d'une routine utilisateur.
DIM	Déclaration des tableaux.
EDIT	Afficher une ligne de programme.
DOKE	Charger une valeur dans deux adresses-mémoire.
GET	Saisie d'un caractère au clavier.
GRAB	Récupérer la mémoire de l'écran HIRES.
HIMEM	Modifier la limite supérieure de la mémoire.
IF-THEN-ELSE	Branchements conditionnels.
INK	Changer la couleur de l'affichage.
LIST	Afficher un bloc de lignes de programme.
LORES	Sélectionner un mode d'affichage.
ON...GOTO	Branchements calculés.

ON...GOSUB	Branchements calculés.
PAPER	Modifier la couleur de l'écran.
PLOT	Afficher des attributs ou du texte à l'écran.
PRINT	Placer le curseur à l'écran.
POP	Sortir en catastrophe d'un GOSUB.
PULL	Sortir d'une boucle REPEAT
RECALL	Charger des variables sauvegardées sur cassette.
RELEASE	Rendre la mémoire à l'écran HIRES.
REPEAT-UNTIL	Boucles conditionnelles.
STORE	Sauvegarder des variables sur cassette.
TRON-TROFF	Aides à la mise au point des programmes.
WAIT	Arrêter un programme pour un temps donné.

CALL

Syntaxe: CALL A

Application: Cette instruction appelle une routine en langage-machine, préalablement écrite en mémoire et commençant à l'adresse (A) donnée comme argument; si cette routine utilise des variables, celles-ci devront lui être passées en les plaçant dans les registres appropriés du 65Ø2. Il est à noter qu'on ne retournera au Basic que si un RTS (≠6Ø, code du 65Ø2 équivalent au "RETURN" du Basic) termine la routine. Des choses amusantes se passeront si vous appelez celles de la ROM, mais l'ordinateur se bloquera souvent !

Exemple: Tapez CALL ≠FB14. Il s'agit du moyen le plus simple d'obtenir par programme la répétition sonore du clavier.

CSAVE

Syntaxe: CSAVE "<nom>"[,S][,AUTO]
ou CSAVE "<nom>", A<adresse début>, E<adresse fin>
[,S][,AUTO]

Applications: Effectue la sauvegarde sur cassette de programmes Basic ou de zones de la mémoire, définis par <nom>.

REMARQUES:

— Dans sa première syntaxe, CSAVE sert à enregistrer sur cassette le programme se trouvant en mémoire. Le nom est à votre discrétion, il doit comporter 17 caractères au maximum et peut être indifféremment une constante ou une variable. L'option AUTO provoquera le démar-

rage automatique du programme lorsque celui-ci sera rechargé. Si vous utilisez l'option S, la sauvegarde se fera à la vitesse de 3ØØ baud (environ 4Ø octets par seconde), vitesse très lente offrant une sécurité maximale. Sinon, le transfert se fait au rythme de 24ØØ baud.

— Dans sa deuxième syntaxe, cette instruction puissante permet de sauvegarder une zone-mémoire située n'importe où dans la RAM. On doit donner un nom comme précédemment, et on peut choisir sa vitesse mais l'option AUTO ne devrait être incluse que si la zone contient une routine exécutable en langage-machine. Celle-ci est définie par ses adresses de début (suivant la lettre A) et de fin (suivant la lettre E).

La première application est évidente, mais on mesurera l'importance de la deuxième, qui permet de sauvegarder aussi bien des routines en langage-machine que des jeux de caractères ou des écrans graphiques:

Exemples:

1) Sauvegarde de l'écran HIRES:

```
CSAVE"ECRAN",A#2000,E#3FFF   pour la version 16K.

CSAVE"ECRAN",A#A000,E#BF3F   pour la version 48K.
```

2) Sauvegarde des jeux de caractères:

```
CSAVE"JEU",A#3500,E#3B7F     pour la version 16K.

CSAVE"JEU",A#B500,E#BB7F     pour la version 48K.
```

Les fichiers peuvent ensuite être rechargés par CLOAD"". (Voir instruction CLOAD).

CLOAD

Syntaxe: CLOAD ["<nom>"][,J][,V][,S]

Applications: Instruction jumelle de CSAVE, CLOAD sert à recharger en mémoire des fichiers, programmes ou zones-mémoire, enregistrés sur cassette. Elle permet également de vérifier la bonne exécution d'une sauvegarde.

REMARQUES:

— Le nom suit les mêmes règles de syntaxe que CSAVE, à savoir 17 caractères au maximum, variable ou constante. Son omission est possible. Dans ce cas, l'Atmos chargera le premier fichier rencontré sur la bande. Il est évidemment indispensable de recharger celui-ci à la même vitesse de transfert que celle utilisée lors de sa sauvegarde: N'oubliez pas le "S" !

— L'option J ("join") charge le programme, mais sans effacer celui résidant en mémoire.

— L'option V ("verify") effectue une comparaison entre le programme en mémoire et celui sur cassette, et affiche les différences ou erreurs trouvées (le programme sur cassette n'est pas chargé). Option très utile pour vérifier le bon déroulement d'une sauvegarde.

— Toutes les fonctions d'entrées-sorties de l'Atmos, en particulier les divers modes d'utilisation de CLOAD et CSAVE ainsi que ceux de STORE et RECALL sont traités in extenso au § 2.4.2).

Exemple: CLOAD" ",S chargera le premier programme rencontré en vitesse lente.

DEF FN

Syntaxe: DEF FN<nom de variable>(X)=<expression numérique>

Application: Cette instruction permet au programmeur de définir ses propres fonctions, qui seraient éventuellement absentes du Basic. Le nom suit les règles d'appellation des variables; il servira par la suite à identifier la fonction. Pour l'utilisation de la fonction une fois celle-ci définie, voir aussi FN dans le chapitre suivant (Fonctions).

REMARQUE: Pour ceux qui seraient familiers avec d'autres Basic, signalons que les fonctions définies ne peuvent traiter que des nombres, et ne doivent contenir qu'une seule variable; celle-ci est une variable "fantôme", c'est-à-dire que son nom peut être utilisé indépendamment à un autre endroit du programme.

Exemples:

```
DEF FNTA(X)=SIN(X)/COS(X)       calcule la tangente de
l'angle X.

DEF FNAS(X)=ATN(X/SQR(-X*X+1))  calcule l'arcsinus de
l'angle X.
```

DEF USR

Syntaxe: DEF USR=A

Application: Cette commande permet de définir l'adresse de début (A) d'une routine en virgule flottante en langage-machine, avant qu'elle ne soit appelée. Son avantage sur CALL réside dans le fait qu'elle permet aussi bien de transmettre directement des données à la routine que d'en recevoir de celle-ci (Voir fonction USR, § 1.5).

DIM

Syntaxe: DIM<tableau 1>(<dim.1>[<,dim.2>](...)[,<tableau 2>(...)]

Application: Pour définir les dimensions des tableaux, avant leur utilisation.

REMARQUES:

— Le Basic de l'Atmos accepte un nombre quelconque de dimensions pour un tableau, avec un nombre quelconque d'éléments par dimension.

— Une fois un tableau dimensionné, toute tentative de le redimensionner se soldera par un message d'erreur. Il faut soit remettre tous les éléments du tableau à Ø, soit utiliser CLEAR ou RUN qui remettent à Ø toutes les variables. Quel que soit le nombre de ses dimensions, un tableau dont aucune dimension n'a plus de 1Ø éléments n'a pas besoin d'être déclaré par DIM.

— Rappelons qu'en ce qui concerne les tableaux de chaînes, l'élément est la chaîne, et non le caractère comme pour certains autres Basic; la conséquence pratique en est que la longueur des chaînes n'a pas besoin d'être définie à l'ayance, elle peut librement varier en cours de programme.

Exemples:

DIM A$(15) déclare un tableau à une dimension contenant 15 chaînes.

DIM A(5,5,15),A%(2Ø,2Ø) déclare un tableau numérique en virgule flottante à 3 dimensions, et un tableau d'entiers à deux dimensions.

EDIT

Syntaxe: EDIT<N° de ligne>

Application: Affiche la ligne ayant le N° spécifié.

REMARQUES: Contrairement à l'usage courant dans d'autres Basic, la commande EDIT sur Oric est pratiquement équivalente à LIST<N° de ligne>. Seules différences:

— Avec EDIT, le curseur se place au début de la ligne affichée.
— Contrairement à LIST, EDIT vous enverra un message d'erreur si vous demandez une ligne inexistante.

DOKE

Syntaxe: DOKE A,N

Application: Instruction utile pour charger deux adresses consécutives de la mémoire (A et A+1) avec un nombre N codé sur deux octets. L'adresse A reçoit l'octet de poids faible et A+1 l'octet de poids fort. A et N sont des variables ou constantes comprises entre Ø et 65535, et seront arrondis à l'entier inférieur.

REMARQUE:

```
DOKE A,X est l'équivalent exact de:
POKE A,X-(INT(X/256)*256:POKE A+1,INT(X/256)
```

Pour le détail de l'utilisation de cette instruction, ainsi que de POKE et des fonctions PEEK et DEEK, voyez aussi la deuxième partie traitant de l'organisation de la mémoire.

Exemple:

```
DOKE #26D,48039  place  INT(48039/256)=187  à  l'adresse
#26E, et 48039-(187*256)=167 à l'adresse #26D.
```

GET

Syntaxe: GET X$
ou GET X

Applications: Lorsque cette instruction est rencontrée dans un programme, l'ordinateur attend qu'une touche soit frappée; dès que cela est fait, la

variable donnée comme argument contiendra le caractère correspondant à la touche actionnée; le programme continue ensuite son exécution.

REMARQUES:

— Si une variable chaîne est spécifiée, n'importe quel caractère du clavier peut être saisi. Si par contre il s'agit d'une variable numérique, seules les touches numériques pourront être utilisées (la variable contiendra alors leur valeur), toutes les autres touches provoquant une erreur.

— Cette commande est très utile lorsqu'il n'est pas nécessaire d'exécuter d'autres tâches en attendant une entrée au clavier, sinon il vaut mieux utiliser la fonction KEY$, traitée au chapitre suivant.

— L'usage de GET est interdit en mode direct.

Exemples:

1)
```
100 PRINT"POUR CONTINUER, PRESSER UNE TOUCHE"
110 GET X$
 .
(Suite du programme)
```

2)
```
100 PRINT"VOULEZ-VOUS CONTINUER ?(O/N"
110 GET X$
120 IF X$="N" THEN END
130 IF X$="O" THEN GOTO 150
140 GOTO 100
 .
(Suite du programme)
```

Dans cet exemple, on retournera à la ligne 1ØØ tant qu'une réponse correcte ("O" ou "N") ne sera pas fournie.

GRAB

Syntaxe: GRAB

Application: Récupère la zone-mémoire normalement affectée à l'écran graphique haute-résolution pour des programmes ou variables. Cette zone, inutilisée dans les modes TEXT et LORES, n'est pas accessible au Basic à moins d'exécuter un GRAB.

REMARQUE: La zone "libérée" se situe entre les adresses #18ØØ-#34ØØ sur la version 16 K, et #98ØØ-#B4ØØ sur la version 48 K. GRAB permet de récupérer 7168 octets pour le Basic, faisant passer la mémoire vive disponible à 46 Koctets environ.

HIMEM

Syntaxe: HIMEM <adresse>

Application: Définit la taille de la mémoire utile, en fixant l'adresse la plus haute de la zone accessible par le Basic. La zone située au-dessus de HIMEM sera interdite au Basic, donc disponible pour des routines en langage-machine.

REMARQUES:

— A la mise sous tension, HIMEM est fixé à 38912 (#98ØØ) pour la version 48 K, et à 6144 (#18ØØ) sur la version 16 K. L'exécution d'un GRAB fait passer ces valeurs respectivement à 46Ø8Ø (#B4ØØ) et 13312 (#34ØØ).

— HIMEM est également connu sous le nom de RAMTOP ou TOP dans d'autres Basic.

Exemples:

```
HIMEM 30000
HIMEM #17FF   (Sur version 16K)
```

IF...THEN...ELSE

Syntaxe: IF<condition> [AND / OR <condition>](...)THEN<instructions>[:(...)]

ou IF<cond.> [(...)] THEN <instr.>[:(...)] ELSE <instruction>

Application:

— Dans sa première syntaxe, exécutera la série d'instructions située après THEN si <condition> est VRAIE (=—1), sinon continue l'exécution du programme à la ligne suivante.

— Dans la seconde syntaxe, les instructions situées après THEN seront exécutées si <condition> est VRAIE (=—1), et si <condition> est FAUSSE (=Ø) le programme exécutera l'instruction placée après ELSE.

REMARQUES:

— IF<cond.> THEN GOTO <N° de ligne> peut s'écrire indifféremment:
IF<cond.> THEN <N° de ligne>, et aussi:
IF<cond.> GOTO <N° de ligne>.

— Les IF imbriqués sont autorisés si ELSE n'est pas utilisé.
On peut écrire: `IF<cond1>THEN IF<cond2> THEN....`
mais *pas*: `IF<cond1>THEN...ELSE IF<cond2>..etc..`

— ELSE ne peut pas s'utiliser suivi de plusieurs instructions, même séparées par des ":". Ce point est important, car toutes les instructions après la première suivant ELSE seront exécutée si la condition placée après IF est vraie, c'est—à-dire quand il ne le faut pas!

Exemples:

```
IF A>B AND A<C THEN PRINT"A est compris entre B et C"

IF A/2=INT(A/2) THEN PRINT"A est pair" ELSE PRINT"A est impair"

10 IF X$="OUI" THEN 100 ELSE 200
```

INK

Syntaxe: INK N Avec Ø<=N<=7

Application: Permet de choisir la couleur des caractères pour la totalité de l'écran, sans perturber l'affichage. Ne modifie pas les couleurs choisies localement par des codes appropriés.

REMARQUE: Pour l'équivalence entre les nombres de Ø à 7 et les couleurs apparaîssant à l'écran, voir § 3.4.

Exemple: INK 1 Affiche tous les caractères en rouge.

LIST

Syntaxe: LIST <N° de ligne>
ou LIST <N° de ligne début>—
ou LIST—<N° de ligne fin>
ou LIST <N° de ligne début>—<N° de ligne fin>

Application: Affiche à l'écran les lignes de programme situées entre les limites spécifiées. Si le N° de début est omis le programme est listé depuis le début, sinon lorsque le N° de fin est omis le programme est listé jusquà la dernière ligne existante.

REMARQUE: Le fait de fournir des numéros de ligne inexistants, plus petits ou plus grands que ceux du programme, ne provoque pas d'erreur.

Exemple:

```
LIST 100-200
```

LORES

Syntaxe: LORES N Avec N=Ø ou N=1

Application: Sélectionne un des trois modes d'affichage disponibles. Très proche du mode TEXT, avec les différences suivantes:

— L'affichage se fait en vidéo inverse, c'est-à-dire blanc sur fond noir.

— Si N=Ø, ce seront les caractères standards du jeu ASCII qui seront utilisés.

— Si N=1, l'affichage se fera avec le 2e jeu de caractères (pavés semi-graphiques).

REMARQUE: Si CLS ou CTRL L sont utilisés pour effacer l'écran, on repassera en mode TEXT. Il faut refaire LORESØ ou LORES1 pour effacer l'écran en restant dans le même mode. Voir aussi le § 3.6.1.

ON...GOTO et ON...GOSUB

Syntaxe: ON X GOTO <N° de ligne>[,<N° de ligne>,(...)]
et ON X GOSUB<N° de ligne>[,<N° de ligne>,(...)]

Application: Si X=1, le GOTO ou GOSUB s'effectuera vers la première ligne de la liste, si X=2 vers la deuxième et ainsi de suite.

REMARQUES:

— Si X=Ø, ou est supérieur au nombre de lignes spécifiées, le programme continuera à la ligne suivante.

— Une erreur sera provoquée si X est négatif.

— Si X provient d'un calcul et comporte une partie décimale, il sera arrondi à l'entier inférieur. (Voir § 1.6.2 pour la manière de calculer un arrondi exact).

— Il existe un autre moyen pour effectuer des branchements calculés: En effet, l'Atmos accepte parfaitement une syntaxe du type GOTO X*1ØØ+1Ø, ou GOSUB 1ØØØØ–X*1Ø.

PAPER

Syntaxe: PAPER N Avec Ø<=N<=7

Application: Instruction complémentaire de INK, permettant de changer la couleur du fond sur tout l'écran sans perturber l'affichage. Comme pour INK, les couleurs choisies localement ne sont pas affectées.

Exemple:

```
10 FOR N=0 TO 7
20 PAPER N:WAIT 50
30 NEXT N
```

Donnera à l'écran successivement chacune des couleurs.

PLOT

Syntaxe: PLOT <colonne>,<ligne>, N Avec Ø>=N>=255
ou PLOT <colonne>,<ligne>,X$

Application: Permet d'afficher des données ou des attributs (codes) à l'écran, à la position spécifiée par <colonne> et <ligne>.

— Dans la première syntaxe, si N>31, c'est le caractère de code ASCII N qui sera affiché à l'endroit désiré. Si N>159 (31+128), on obtiendra l'affichage en vidéo inverse du caractère de code ASCII N–128. Enfin, si N<32, il sera considéré comme un attribut définissant la couleur, le jeu de caractères, etc. (cf. § 3.4).

— Dans sa deuxième syntaxe, PLOT affichera à l'endroit désiré une chaîne de caractères ou variable-chaîne quelconque.

REMARQUES:

— Si l'une des deux coordonnées fournies à PLOT est située en dehors des limites de l'écran, une erreur en résultera.

— PLOT n'est pas un équivalent de l'instruction PRINT @ ou PRINT AT (voir cette instruction, ci-après), dans la mesure où le curseur n'est pas déplacé après affichage des données. De plus, PLOT contrairement à PRINT n'accepte qu'une seule donnée à afficher à la fois.

Exemples:

```
PLOT 10,10,"BONJOUR"  est équivalent à
X$="BONJOUR":PLOT 10,10,X$
```

PLOT 1Ø,1Ø,12:PLOT 11,1Ø,"BONJOUR" affiche le message au centre de l'écran, en caractères clignotants. (12 est l'attribut "caractères clignotants").

POP

Syntaxe: POP

Application: Supprime la dernière adresse de retour dans la pile des GOSUB. Conséquence pratique: Si un RETURN est rencontré après un POP, on ne retournera pas à la ligne suivant le dernier GOSUB, mais à la ligne suivant *l'avant-dernier* GOSUB rencontré si toutefois celui-ci existe; si ce n'est pas le cas, une "RETURN WITHOUT GOSUB ERROR" sera provoquée. A n'utiliser qu'avec précautions!

Exemples:

```
 10 GOSUB 100
 20 PRINT 1
  .
  .
100 REM 1ER SOUS-PROGRAMME
120 GOSUB 200
130 PRINT 2
140 RETURN
  .
  .
200 REM 2EME SOUS-PROGRAMME
210 POP:PRINT 3
220 RETURN
```

Normalement, la ligne 22Ø devrait provoquer le retour à la ligne 13Ø. Mais avec le POP de la ligne 21Ø, c'est à la ligne 2Ø que le programme reprendra: L'adresse de retour à la ligne 13Ø a été éliminée.

PRINT @

Syntaxe: PRINT @ <colonne>,<ligne>;<liste de données>(...)

Application: Cette instruction, qui n'existait pas sur la précédente version du Basic Oric, autorise le positionnement du curseur, et donc l'affichage de données, à n'importe quel endroit de l'écran.

REMARQUES:

— <colonne> doit être compris entre Ø et 39, et <ligne> entre Ø et 26. Toute valeur hors de ces limites provoque une erreur. Ces paramètres sont obligatoires: Il faut utiliser PRINT si on veut les omettre.

— PRINT @ , également connu sous le nom de PRINT AT dans d'autres Basic, peut également s'écrire ? @ en forme abrégée.

— Il est parfaitement possible de chaîner l'instruction, sans écrire PRINT à chaque fois. Ainsi:

```
PRINT@6,10;"JE SUIS"@15,10;"ATMOS"
```

est parfaitement correct. Par contre, on ne peut pas omettre le point-virgule après la ligne-colonne, alors que celui-ci peut très bien être omis entre les données (sauf si son absence prête à confusion):

```
PRINT@10,20;"A est egal a:";A;"B$=";B$;"N=";N%
```

Peut très bien s'écrire:

```
PRINT@10,20;"A est egal a:"A"B$="B$"N="N%
```

— PRINT @ permet d'accéder aux deux colonnes protégées à gauche de l'écran, contenant respectivement les couleurs du fond et de l'affichage. Celles-ci correspondent aux valeurs Ø et 1 de <colonne>. Donc, tout affichage normal se fera à partir de la colonne 2: PRINT@2,X..,etc... Pour l'utilisation des colonnes réservées, voyez aussi le § 3.3.

Exemple:

```
10 CLS:PRINT CHR$(4)CHR$(17)
20 CO=RND(1)*37+2:CH$=CHR$(RND(1)*93+33)
30 LI=INT(RND(1)*24)):IF LI/2=INT(LI/2)THEN LI=LI+1
40 PRINT@ CO,LI;CHR$(27)"J"CH$
50 WAIT 30:GOTO 20
```

Ce petit programme remplit l'écran de caractères aléatoires en double taille, affichés à des positions également aléatoires.

PULL

Syntaxe: PULL

Application: A exactement la même fonction que POP, mais pour les boucles REPEAT-UNTIL. Si deux boucles de ce type sont imbriquées, un PULL exécuté une seule fois dans la boucle intérieure provoquera un saut au REPEAT externe. Encore plus traître que POP!

REMARQUE: Un PULL au mauvais endroit vous vaudra un BAD UNTIL ERROR, ou bien engagera le programme dans une boucle infinie. Donc, testez bien vos boucles AVANT d'inclure un PULL. Il s'agit de toute manière (de même que POP, d'ailleurs) d'une solution de "bricolage", et un programme bien conçu doit pouvoir se passer complètement de ces instructions.

RECALL

Syntaxe: RECALL<Tableau>,<Nom du fichier>[,S]

Application: Instruction utilisée pour recharger en mémoire des fichiers enregistrés sur cassette à l'aide de l'instruction STORE (cf. ci-après). Le fichier dont le nom est spécifié dans l'instruction est chargé dans un tableau préalablement dimensionné.

REMARQUES:

— Le nom du fichier doit être identique à celui utilisé lors de la sauvegarde, espaces compris. Celui-ci peut être fourni par une constante, ou une variable. Si aucun nom n'est fourni, l'Atmos tentera de charger le premier fichier du même type rencontré sur la bande.

— Le tableau doit être dimensionné correctement par DIM avant le chargement, avec le même nombre de dimensions et une taille identique ou supérieure à celle utilisée lors de la sauvegarde.

— Le tableau doit également être du même type (virgule flottante, entiers ou chaîne), spécifié sous la forme A, A% ou A$, mais ne doit pas nécessairement avoir le même nom (il ne sera pas confondu avec d'éventuelles variables simples nommées A, A% ou A$).

Syntaxe: PAPER N Avec Ø⩽=N⩽=7

si N⩽32,

— L'option "S" permet, comme pour CLOAD et CSAVE, de choisir la vitesse lente (3ØØ bauds). Évidemment, la vitesse de lecture doit être la même que celle employée pour la sauvegarde.

L'utilisation de STORE et RECALL est décrite en détail au § 2.4.2.

RELEASE

Syntaxe: RELEASE

Application: Commande complémentaire de GRAB (cf. cette instruction), permet de restituer à l'écran graphique haute-résolution la mémoire que vous lui avez indûment volée pour vos programmes Basic.

REPEAT-UNTIL

Syntaxe: REPEAT

.
(lignes de programme)
.
.
UNTIL<condition> [AND <condition>](...)
[OR

Application: Toutes les lignes de programme situées entre REPEAT et UNTIL sont exécutées en boucle, jusqu'à ce que les conditions situées après UNTIL soient VRAIES (=—1).

REMARQUE: Ces commandes sont l'équivalent ORIC du WHILE-WEND du Basic Microsoft. Avec une différence cependant: Alors que WHILE teste la condition au début, UNTIL teste en fin de boucle. Ce qui implique que même si la condition est fausse avant d'entrer dans la boucle, celle-ci sera exécutée au moins une fois.

Exemple:

```
10 REPEAT
20 X=RND(1)*36+2:Y=RND(1)*26
30 PLOTX,Y,"*"
40 UNTIL X$<3
```

Ce programme dessinera des étoiles sur l'écran à des endroits aléatoires, jusqu'à ce qu'une position particulière soit atteinte.

STORE

Syntaxe: STORE <Tableau>,<Nom du fichier>[,S]

Application: Cette instruction très importante autorise le stockage de variables sur cassette, qui seront ensuite lues et rechargées par RECALL.

REMARQUES:

— <Tableau> est le nom du tableau de variables à sauvegarder. Il peut avoir n'importe quelle taille et un nombre quelconque de dimensions. Les trois types (numérique en virgule flottante, entiers et chaînes) sont acceptés. Le tableau est spécifié sous la forme: A, A% ou A$, et ne sera pas confondu avec une variable simple portant le même nom.

— **Attention**: Le nom du fichier, s'il peut avoir une longueur allant jusqu'à 16 caractères, ne peut en aucun cas être une variable. La syntaxe est acceptée, mais les données sauvegardées sont ensuite irrécupérables. Il faut donc impérativement utiliser une constante entre guillemets.

— L'option "S" permet de spécifier une vitesse lente de 3ØØ baud pour la sauvegarde. La lecture avec RECALL devra ensuite s'effectuer à la même vitesse.

L'utilisation de ces instructions est traitée in extenso et avec des exemples au § 2.4.2.

TRON-TROFF

Syntaxe: TRON
TROFF

Application: L'instruction TRON (TRace ON) fournit une aide précieuse à la mise au point des programmes: En effet, les numéros de toutes les lignes de programme exécutées après TRON seront affichés à l'écran en séquence, permettant ainsi de suivre le cheminement du programme et donc de détecter où et pourquoi une erreur se produit. TROFF (TRace OFF) permet d'annuler TRON.

REMARQUES:

— L'usage de TRON est impossible en mode direct, il faut donc l'incorporer à un endroit propice du programme à tester, et le supprimer ensuite une fois celui-ci au point. Par contre, TROFF peut s'utiliser indifféremment en modes direct ou indirect.

— TRON rend très rapidement l'écran illisible dans les programmes (ou parties de programmes) utilisant beaucoup l'affichage. Deux ou trois astuces permettent de contourner l'obstacle :

— Entrer et sortir alternativement du mode par des TRON et TROFF judicieusement disposés, de manière à éviter de surcharger l'écran.

— Inclure des points d'arrêt dans le programme (STOP).

— Utiliser des instructions PRINT @ dans le programme, afin de replacer le curseur entre les affichages de numéros de lignes, et effacer souvent l'écran.

— Inclure l'affichage des valeurs des principales variables, ce qui facilitera le "débogage" du programme.

WAIT

Syntaxe : WAIT N

Application : Suspend l'exécution du programme pour une durée égale à (N*Ø.Ø1) secondes. Donc WAIT 1 suspend l'exécution pendant 1/1ØØ^{e} de seconde, ou 1Ø ms.

REMARQUE : Certaines instructions, si elles s'exécutent de façon répétitive dans une boucle, ne peuvent agir correctement que si un WAIT de longueur appropriée est inclus dans la boucle. Il s'agit en particulier de INK, PAPER et de certaines instructions générant un son prédéfini (cf. 5^{e} partie).

1.5. Les fonctions du Basic

DEEK	Lit le contenu de deux adresses consécutives
FALSE-TRUE	Variables réservées.
FN	Exécution des fonctions définies par l'utilisateur.
FRE	Mémoire restant disponible.
HEX$	Conversion décimal-hexadécimal.
KEY$	Lecture fugitive du clavier.
MID$	Extraction d'une partie d'une chaîne.
POS	Position horizontale du curseur.

RND	Génération de nombres aléatoires.
SCRN	Code du caractère à la position du curseur.
SPC	Impression d'espaces.
USR	Appel de routines en langage-machine.

DEEK

Syntaxe: DEEK(A)

Application: Cette fonction, qui signifie "Double PEEK", nous permet de connaître le contenu de deux adresses-mémoire consécutives (A et A+1), il s'agit bien sûr du complément de l'instruction DOKE. L'adresse A contient l'octet de poids faible, et A+1 l'octet de poids fort. A doit être compris entre Ø et 65534.

REMARQUES: PRINT DEEK(A) est équivalent à:
PRINT PEEK(A)+PEEK(A+1)*256

Exemple:

```
PRINT DEEK(#A6)  retournera la valeur de HIMEM
```

FALSE et TRUE

Syntaxe: FALSE
TRUE

Application: Ce sont deux variables réservées du Basic, au même titre que PI. FALSE (FAUX) vaut Ø, et TRUE (VRAI) vaut −1. On les utilise dans les comparaisons avec IF, pour améliorer la lisibilité des programmes entre autres.

Exemple:

```
IF (A>2 AND B=>10)=TRUE THEN.....
```

FN

Syntaxe: FN<nom>(<variable>)

Application: Une fonction numérique préalablement définie grâce à DEF FN (cf. cette instruction), s'utilise ensuite comme n'importe quelle autre fonction mathématique standard (SIN ou TAN par exemple) grâce à la fonction FN.

***Exemple*:**

```
 10 ' Conversion Degrés -> Radians
 20 DEF FNDR(X)=(PI*X)/180
  .
  .
100 A=30
110 PRINT FNDR(A);SIN(FNDR(A))
```

Le résultat affiché sera Ø.523598775 Ø.5, nombres qui sont respectivement la conversion d'un angle de 3Ø° en radians, et le Sinus de 3Ø°.

FRE

Syntaxe: FRE(Ø)
ou FRE(" ")

Applications:

— Dans sa première syntaxe, cette fonction sert à connaître le nombre d'octets encore inoccupés en mémoire, pendant l'écriture d'un programme par exemple.

— Dans sa deuxième syntaxe, la fonction retourne aussi la quantité de mémoire inoccupée, mais en plus force une récupération de l'espace contenant des données inutiles ou supprimées (garbage collection).

REMARQUE: L'Atmos utilise une méthode de rangement des chaînes en mémoire qui a pour caractéristique de ne jamais effacer de données: Les nouvelles valeurs des variables sont rangées à côté des précédentes, même si certaines d'entre elles sont supprimées par le Basic en cours de route. Lorsque la totalité de la mémoire est occupée, un "nettoyage" est effectué (appelé "garbage collection"), supprimant toutes les données devenues inutiles. C'est ce nettoyage qu'on peut provoquer "manuellement" par l'usage de FRE(" ").

***Exemple*:**

```
10000 IF FRE(0)<200 THEN PRINT"PLUS DE MEMOIRE !"
10010 RETURN
```

Ce sous-programme peut être ajouté à un programme "gourmand" pour vous prévenir quand il n'y a presque plus de mémoire libre.

HEX$

Syntaxe: HEX$(<X>)

Application: Retourne une chaîne représentant la valeur hexadécimale de X.

REMARQUE: X ne peut être ni négatif, ni supérieur à 65535.

Exemple:

```
PRINT HEX$(65535) nous donne #FFFF
```

KEY$

Syntaxe: KEY$

Application: Scrute le clavier de façon fugitive: L'exécution du programme continue après que la fonction ait été rencontrée, qu'une touche ait été actionnée ou non à ce moment. La fonction doit donc être incluse dans des boucles pour lire efficacement le clavier. C'est la solution idéale pour les jeux, où le clavier doit être constamment surveillé sans pour cela empêcher d'autres actions de s'exécuter simultanément.

REMARQUE: Les habitués auront bien sûr reconnu là le INKEY$ des autres Basic, à peine déguisé. Voir aussi l'instruction GET au chapitre précédent.

Exemple:

```
  5 ' DEPLACEMENT D'UN MOBILE A L'ECRAN
  6 '
 10 CLS:X=10:Y=10:A$="<*>"
 20 XM=X:YM=Y
 30 X$=KEY$:A=ASC(X$+" ")
 40 IF A<8 OR A>11 THEN 30
 50 PLOT XM,YM,"   "
 60 IF A=8 THEN X=X-1
 70 IF X=9 THEN X=X+1
 80 IF A=10 THEN Y=Y+1
 90 IF A=11 THEN Y=Y-1
100 PLOT X,Y,A$:GOTO 20
```

— Les variables XM et YM servent à mémoriser l'ancienne position du mobile, pour l'effacer avant un déplacement.

— A la ligne 3Ø, la variable A prend le code ASCII de la dernière touche pressée. On ajoute systématiquement un espace à X$, car si X$ est une chaîne vide ASC(X$) génère un code d'erreur.

— Ligne 4Ø: Si aucune touche de déplacement n'a été pressée (codes 8 à 11), rien ne se passe.

— Ligne 5Ø: Dans le cas contraire, on efface d'abord le mobile à son ancienne position en y imprimant 3 espaces.

MID$

Syntaxe: MID$(A$,N[,M])

Application: Cette fonction très puissante permet d'extraire une partie d'une chaîne de caractères: La chaîne retournée contiendra M caractères de A$, en commençant par le Nième.

REMARQUES:

— Si M est omis, tous les caractères de A$ à droite du Nième seront pris.

— Si N est supérieur à la longueur de A$ ou inférieur à 1, une erreur sera provoquée.

Exemples:

```
10 A$="123456789"
20 PRINT MID$(A$,2,5)
```

Nous imprimera: 23456.

Voici un petit programme qui utilise toutes les fonctions de découpage de chaînes de manière amusante. Vous pourrez facilement l'incorporer comme sous-programme dans vos propres créations:

```
   5 REM CHENILLE LITTERALE
  10 CLS:GOSUB 1000
 100 A$=A1$+A2$+A3$+A4$:N=0:LA=LEN(A$)
 110 WAIT 10:PLOT 2,25,M$
 120 N=N+1:IF N=LA THEN N=1
 130 IF   N>=LA-38   THEN   M$=RIGHT$(A$,LA-N)+LEFT$(A$,38-(LA-
 N)):GOTO 110
 140 M$=MID$(A$,N,38):GOTO 110
   .
   .
1000 READ A1$,A2$,A3$,A4$:RETURN
1010 DATA" CETTE LONGUE PHRASE EST UNIQUEMENT DESTINEE A TESTER
LE PROGRAMME, "
1020 DATA"DONT  LA  SEULE  FONCTION EST DE FAIRE DEFILER LADITE
PHRASE DE DROITE "
1030 DATA"A  GAUCHE  SUR L'ECRAN, CREANT AINSI UNE ANIMATION DU
PLUS BEL EFFET: "
1040 DATA"IDEAL POUR LES EXPLICATIONS DANS VOS PROGRAMMES! **"
```

Explications:

Le principe du programme est assez simple: On crée une longue chaîne, puis on en extrait une sous-chaîne qui est affichée toujours au même endroit de l'écran. En décalant la sous-chaîne d'un cran vers la droite à chaque fois, on obtient l'effet de défilement désiré.

1ØØØ: Lecture de 4 chaînes dans des DATA; ces 4 chaînes formant une longue phrase sont ensuite ajoutées pour former A$ (ligne 1ØØ). Cette technique est indispensable, du fait que l'on ne peut mettre que 78 caractères par ligne de programme; n'oubliez pas cependant quand vous mettrez vos propres phrases qu'une chaîne ne peut contenir que 255 caractères en tout.

14Ø : Chargement de M$ avec un segment de longueur appropriée extrait de A$.

13Ø : S'il ne reste plus 38 caractères à prendre à la fin de A$, cette ligne raccorde la fin de A$ avec un bout de longueur appropriée pris à son début: De cette manière, M$ fait toujours 38 caractères de long.

12Ø : Lorsque N atteint la fin de A$, on repart au début.

11Ø : Introduction d'une petite pause pour améliorer la lisibilité, et impression de M$ à l'endroit désiré.

POS

Syntaxe: POS(Ø)

Application: Retourne la position horizontale du curseur, donc une valeur comprise entre Ø et 39.

REMARQUES:

Cette fonction est inutilisable en mode direct, car après un "RETURN" le curseur retourne à la première colonne de la ligne suivante, et on obtient toujours Ø.

— De même, elle ne fonctionne pas dans les déplacements "forcés", avec PRINT @ par exemple. Il existe cependant d'autres moyens de connaître la position du curseur, donnés au § 3.5.2.

RND

Syntaxe: RND(N)

Application: Cette fonction donne des résultats différents selon la valeur de N:

— Si N>=1, elle retourne un nombre pseudo-aléatoire compris entre Ø et 1 non inclus.

— Si N=Ø, elle renvoie toujours le nombre précédent.

— Si N<Ø, le nombre fourni est fonction de la valeur de N, et toujours le même pour des valeurs égales de N.

Exemple:

```
10 REM DE ELECTRONIQUE
20 PRINT INT(RND(1)*6+1)
30 GET X$:GOTO 20
```

SCRN

Syntaxe: SCRN(<colonne>,<ligne>)

Application: Utilisé dans les modes TEXT et LORES, retourne le code ASCII du caractère situé à la position de l'écran spécifiée.

Exemple:

```
IF SCRN(15,10)=42 THEN EXPLODE
```

SPC

Syntaxe: SPC(X)

Application: Utilisable uniquement avec PRINT et LPRINT, déplace de X colonnes vers la droite la position d'affichage.

REMARQUE: Dans certains cas, lorsque l'on veut par exemple effacer une zone limitée de l'écran, SPC peut remplacer avantageusement la fonction TAB.

Exemple:

```
PRINT SPC(10)"JE M'APPELLE"SPC(5)"ORIC"
```

USR

Syntaxe: USR(X)

Application: La variable en virgule flottante X est passée à une routine en langage-machine, dont l'adresse de début a été préalablement définie par DEF USR. La fonction USR retournera ensuite la valeur stockée par l'accumulateur.

Exemple:

```
10 DEF USR=#400
20 A=22.5
30 PRINT USR(A)
```

1.6. *Instructions et fonctions non-résidentes*

Malgré le fait que les principales défaillances de langage (en nombre assez limité, au demeurant) de son "grand frère" l'Oric-1 aient été corrigées sur l'Atmos, celui-ci n'est cependant pas encore parfait et certaines commandes disponibles sur les Basic très évolués lui font défaut. Nous avons essayé de sélectionner celles d'entre elles qui nous paraissaient les plus importantes, et dont la simulation par un petit programme reste dans des limites raisonnables. Sont également présentes certaines instructions existant sur l'Atmos, mais dont la syntaxe est moins évidente que celle du Basic "Standard".

Dans le cas des commandes simulées par programme, ceux-ci pourront être inclus comme sous-programmes dans vos propres réalisations. Le Basic Atmos permet de les ajouter à un programme présent en mémoire en les chargeant à partir d'une cassette, grâce à l'instruction CLOAD. C'est pour faciliter cette fusion que toutes ces routines ont été numérotées à partir de 6ØØØØ. La syntaxe à utiliser est: CLOAD[<Nom du sous-programme>],J [,S].

Pour plus de précisions, voyez aussi CLOAD aux § 1.4 et 2.4.2, ainsi que l'instruction MERGE dans ce chapitre.

1.6.1. Instructions

Voici la liste des commandes non disponibles sur Atmos, dont la simulation est traitée dans ce chapitre:

ERASE	Supprimer des tableaux pour les redimensionner.
MERGE	Fusionner des programmes.
MID$	Remplacer une partie d'une chaîne par une autre.
PRINT USING	Afficher des nombres sous un format précis.
RENUM	Renuméroter les lignes d'un programme.
SWAP	Échanger les valeurs de deux variables.
VERIFY	Vérifier qu'un programme est bien enregistré.

ERASE<liste de tableaux>

Application: Nous savons qu'une fois déclaré et dimensionné à l'aide de DIM, un tableau de variables (quel que soit son type, d'ailleurs), ne peut plus être redimensionné sous peine de provoquer une "REDIM'D ARRAY ERROR". Cette instruction très utile permet de contourner l'obstacle, en effaçant sélectivement un tableau sans toucher aux autres variables. Le tableau peut ensuite être redimensionné, comme s'il n'avait jamais existé.

Équivalent Atmos: Il n'existe aucun moyen d'effacer un seul tableau, mais une astuce utilisant les pointeurs-système nous permet d'effacer tous les tableaux, sans perturber les variables simples. Cela est parfaitement suffisant dans la grande majorité des cas. Il faut taper:

```
DOKE #A0,DEEK #9E
```

Le fonctionnement de ces pointeurs est expliqué au § 2.3.1.

MERGE<Nom de programme>

Application: Instruction indispensable, permettant de fusionner le programme résidant en mémoire avec un autre se trouvant sur cassette. Celui-ci est chargé, et "ajouté" à la suite du programme résident sans l'effacer. Ainsi, on peut se constituer des bibliothèques de sous-programmes qui seront incorporés aux divers programmes principaux sans qu'il soit besoin de les retaper à chaque fois.

Équivalent Atmos : Une syntaxe particulière de l'instruction CLOAD permet d'obtenir ce résultat. Une seule précaution à prendre : Le numéro de ligne le plus bas du programme fusionné doit être supérieur à la ligne la plus élevée du programme résident. Il faut écrire :

```
CLOAD[<Nom du programme appelé>],J [,S]
```

N'oubliez pas de spécifier la vitesse lente (,S), si celle-ci a été utilisée lors de la sauvegarde.

**MID$**(A$,N[,M])=B$

Application : Cette instruction, inverse de la fonction du même nom, permet de remplacer M caractères dans A$, à partir du Nième par les M premiers caractères de B$. Si M est omis, B$ est inséré en entier. Si B$ contient plus de caractères que A$ ne peut en recevoir, les caractères excédentaires seront ignorés.

Équivalent Atmos : Celui-ci possède bien la fonction MID$ permettant de lire une fraction d'une chaîne, mais pas l'instruction remplaçant une partie du contenu d'une chaîne par une autre. L'expression suivante doit être utilisée :

```
A$=LEFT$(A$,N-1)+LEFT$(B$,M)+RIGHT$(A$,LEN(A$)-(N+M-1))
```

PRINT USING<chaîne>;<liste de constantes ou variables>

Application : Permet d'afficher les nombres donnés comme arguments selon un format défini par <chaîne>. L'intérêt principal réside dans le fait que pour les tableaux de nombres, et surtout lorsque ceux-ci comportent des décimales, la lisibilité et la présentation sont grandement améliorées par l'affichage sur un format identique (même nombre de décimales, alignement sur la virgule).

Équivalent Atmos : Bien sûr, les vraies PRINT USING autorisent une grande souplesse dans le choix des formats d'impression, et traitent d'ailleurs aussi bien des chaînes que les expressions numériques. Néanmoins, le programme qui suit effectuera toutes les opérations d'arrondi exact et de formatage sur un nombre, et le retournera "prêt à imprimer" sous forme de chaîne de caractères. Celui-ci sera justifié à droite, avec un espace à droite

des décimales s'il est positif et le signe moins s'il est négatif, selon la norme de cette instruction. Si un nombre est trop grand pour le format, il sera remplacé par "****".

```
60000 '       PRINT USING
60005 '       F$=FORMAT  ("****.**")  N=NOMBRE A TRAITER
60007 '
60010 FOR  I=1  TO  LEN(F$):IF  MID$(F$,I,1)="."  THEN  P1=I-1:
P2=LEN(F$)-P1-1:GOTO 60020
60015 NEXT:P1=LEN(F$):P2=0
60020 N=INT((N*10↑P2)+.5)/10↑P2:ST$="**********************"
60030 N$=STR$(N):SN$=LEFT$(N$,1):N$=RIGHT$(N$,LEN(N$)-1):L=LEN
(N$)
60040 P=1:REPEAT
60050 P=P+1:UNTIL P>L OR MID$(N$,P,1)="."
60060 NE$=LEFT$(N$,P-1)
60070 IF P<=L THEN ND$=RIGHT$(N$,L-P) ELSE ND$=""
60080 IF LEN(ND$)>=P2 THEN 60110
60090 ND$=ND$+"0"
60100 GOTO 60080
60110 IF LEN(NE$)>=P1 THEN 60130
60120 NE$=" "+NE$:GOTO 60110
60130 IF ND$="" THEN N$=NE$+SN$:GOTO 60150
60140 N$=NE$+"."+ND$+SN$
60150 IF LEN(N$)>P1+P2+2 THEN N$=LEFT$(ST$,P1+P2+1)
60160 RETURN
```

Explication:

— Les variables que vous devez passer au programme sont N, qui est le nombre à traiter, et F$ qui détermine le format; il sert à illustrer les "positions" disponibles pour imprimer N, et peut contenir n'importe quels caractères: L'important est leur nombre avant et après le point décimal.

— Les lignes 6ØØ1Ø et 6ØØ15 lisent F$ pour en extraire P1 et P2, représentant respectivement le nombre de chiffres avant et après la virgule.

— La ligne 6ØØ2Ø arrondit N au nombre de décimales souhaitées.

— N est converti en chaîne (N$), puis le signe en est extrait et placé dans SN$.

— Les lignes 6ØØ4Ø à 6Ø13Ø chargent NE$ et ND$ avec les parties entière et décimale de N$, mises au format.

— Ensuite, le nombre est reconstitué, de manière différente s'il est entier (ligne 6Ø13Ø), ou s'il comporte des décimales (ligne 6Ø14Ø).

— Enfin, si N$ est plus long que le format demandé, il est remplacé par une chaîne de même longueur contenant des "*". Si vous ne désirez pas cette option, vous pouvez supprimer la ligne 6Ø15Ø et la deuxième moitié de 6ØØ2Ø.

RENUM <ancien N° début>,<nouveau N° début>,incrément

Application: Renumérote les lignes d'un programme. La renumérotation commence à <ancien N° début>, qui prend la valeur de <nouveau N° début>. Les nouveaux numéros sont séparés de <incrément>.

Équivalent Atmos: Notre programme fera plus de choses que le RENUM standard, puisqu'il vous permettra de choisir à la fois la ligne où commence la renumérotation et la ligne où elle s'arrête. Il a aussi l'avantage d'être court et rapide (environ 7 secondes pour 1ØØ lignes), mais attention: il ne change pas les numéros de ligne suivant les GOTO et GOSUB, qui devront être modifiés manuellement.

```
60000 '       RENUMEROTATION LIGNES
60010 '       SANS GOTO ET GOSUB
60015 '
60020 W=#501:CLS
60030 INPUT"Anciens Nos Debut, Fin ";DE,FI
60040 INPUT"Nouveau No Debut, Increment ";DD,IN
60050 REPEAT
60060 A=DEEK(W+2):IF A<DE THEN 60080
60070 DOKE W+2,DD:DD=DD+IN
60080 W=DEEK(W):UNTIL A=FI OR DEEK(W+2)=60000
60090 CLS:END
```

Explications:

Ce programme tire parti du fait qu'en mémoire, chaque numéro de ligne est précédé de deux octets contenant l'adresse où commence la ligne suivante. De ce fait, on peut sauter de début de ligne en début de ligne sans être obligé de lire tous les octets intermédiaires, et la vitesse dépend uniquement du nombre de lignes indépendamment de leur longueur.

— W contient les adresses à tester. #5Ø1 est la deuxième adresse de la zone programme, début de la première ligne.

— Ligne 6ØØ5Ø: A est chargé avec le numéro de la ligne "testée" (toujours situé aux adresses W+2 et W+3), et celui-ci est comparé à DE pour vérifier s'il n'est pas plus petit que la ligne où doit commencer la renumérotation.

— Si le numéro doit être modifié, c'est fait à la ligne 6ØØ7Ø. Puis le nouveau numéro est incrémenté.

— W prend la valeur de la première adresse de la ligne suivante, puis on vérifie que la dernière ligne à traiter n'a pas été atteinte (DE), ainsi que la ligne 6ØØØØ (pour éviter de renuméroter le programme RENUM lui-même !). Si ce n'est pas le cas, on retourne au début de la boucle.

Tout cela peut paraître très obscur, surtout si l'on n'est pas familiarisé avec l'organisation mémoire de l'Atmos. Pour mieux comprendre le principe de stockage des programmes en RAM, reportez-vous au § 2.3.1.

SWAP <variable 1>,<variable 2>

Application: Intervertit les valeurs des deux variables données comme argument. Instruction très utile pour les opérations de tri et de classement.

Équivalent Atmos: Une variable intermédiaire (X) est indispensable pour effectuer l'opération, où A et B sont les variables à intervertir:

X=A :A=B :B=X

Exemple: Tri Shell-Metzner.

Ce petit programme trie un tableau nommé A(), mais il peut aussi bien trier des tableaux de chaînes (classement alphabétique) ou d'entiers. La méthode proposée est l'une des plus rapides existantes, pour les tableaux de taille moyenne (jusqu'à 5ØØ éléments environ). A la sortie du sous-programme, le tableau A() est trié par ordre croissant, le premier élément étant le plus petit et le dernier le plus grand. La taille T du tableau à trier doit être fournie à la routine par le programme principal:

```
60000 '  TRI SHELL-METZNER
60005 '  A()=Tableau a trier, T=Taille de A()
60007 '
60010 P=T
60020 P=INT(P/2):IF P<1 THEN RETURN
60030 N=1:K=T-P
60040 M=N
60050 H=M+P
60060 IF A(M)<=A(H) THEN 60090
60070 X=A(M):A(M)=A(H):A(H)=X ' Instruction "SWAP"
60080 M=M-P:IF M<1 THEN 60090 ELSE 60050
60090 N=N+1:IF N>K THEN 60020 ELSE 60040
```

Nous n'entrerons pas dans les détails du fonctionnement de ce programme, dont le principe est assez complexe. Disons simplement qu'au début, le tableau est divisé en deux parties, chaque élément de la partie basse étant comparé à l'élément correspondant de la partie haute. On fait alors "monter" le plus petit des deux, et "descendre" le plus grand. En divisant successivement par deux le pas (P) séparant deux éléments, on arrive en fin de compte à un pas de 1, et un tableau totalement trié.

VERIFY<Nom de programme>

Application: Cette instruction, indispensable pour les "inquiets", sert à vérifier que la sauvegarde d'un programme sur bande s'est bien déroulée. Le programme en mémoire est comparé avec celui lu sur bande, et le nombre d'erreurs éventuelles détectées est affiché en fin d'opération.

Équivalent Atmos: Une syntaxe particulière de CLOAD effectue cette opération. Il s'agit de:

```
CLOAD [<Nom du programme>],V [,S]
```

— Si le nom du programme est omis, CLOAD vérifiera le premier programme rencontré sur la bande.

— Le programme lu n'est *pas* chargé, donc le programme en mémoire n'est absolument pas perturbé. Il est parfaitement possible de le sauvegarder une nouvelle fois.

— A la fin de l'opération, le message "XX Verify Errors" est affiché à l'écran. Si XX est supérieur à 5, tentez une nouvelle vérification et en cas de persistance des erreurs effectuez une nouvelle sauvegarde sur une autre cassette.

— N'oubliez pas l'option ",S" si le programme à vérifier a été sauvegardé en vitesse lente.

1.6.2. Fonctions

Le Basic de l'Atmos possède un très grand nombre de fonctions mathématiques, ainsi que toutes les fonctions standard de traitement des chaînes (il s'agit de l'un des rares Basic existants possédant à la fois LOG et LN).

Cependant, on a parfois besoin de certaines fonctions souvent fort simples mais non disponibles. Voici celles que nous avons sélectionnées:

ARRONDI EXACT	Convertit un nombre à l'entier le plus proche.
CONVERSION DEG/RAD	Convertit les degrés en radians et vice-versa.
DIVISION ENTIÈRE	Quotient entier de deux nombres.
DIVISION MODULO	Reste de la division entre deux nombres.
INPUT$	Saisie d'un nombre donné de caractères.
INSTR	Recherche d'une chaîne dans une autre.
SPACE$-STRING$	Chaîne de X espaces ou X caractères.

ARRONDI EXACT

Nous savons que la fonction INT ne fournit pas un arrondi exact, puisqu'elle arrondit toujours à l'entier inférieur. Voici la formule permettant d'obtenir l'arrondi pour X, D étant le nombre de décimales souhaité:

```
INT((X*10↑D)+.5)/10↑D
```

Si on veut convertir le nombre en entier (D=Ø), l'expression se réduit à INT(X+.5).

CONVERSION DEGRÉS/RADIANS ET RADIANS/DEGRÉS

Toutes les fonctions trigonométriques de l'Oric utilisent des angles exprimés en radians. Comme on a souvent besoin de traiter des angles en degrés, voici les formules de conversion:

```
Degrés/Radians          XR=XD*180/PI
Radians/Degrés          XD=PI*XR/180
```

Où XD est l'angle exprimé en degrés et XR en radians.

DIVISION ENTIÈRE

Il s'agit de l'opérateur retournant le quotient entier entre deux nombres. L'équivalent de la division entière (symbole A\ B) est:

```
INT(A/B)
```

DIVISION MODULO

Il s'agit du reste de la division entière entre deux nombres: (1Ø MOD 4)=2, (23 MOD 5)=3, etc... Qui n'a pas pesté contre les calculatrices (et les ordinateurs !) qui vous sortent toujours une ribambelle de décimales en résultat de la division la plus simple ? Voici la solution du problème (expression équivalente à (A MOD B):

```
A-INT(A/B)*B
```

INPUT$(N)

Application: Équivalent de l'instruction GET, avec la différence qu'au lieu d'un seul caractère, c'est une chaîne de N caractères qui est saisie au clavier.

Équivalent Atmos:

```
FOR I=1 TO N:GETX$:NEXT I
```

INSTR([N,]A$,B$)

Application: Recherche la chaîne B$ à l'intérieur de A$, à partir de la position N. La valeur retournée est Ø si B$ n'est pas trouvé, ou la position du premier caractère de B$ dans A$.

Équivalent Atmos: Un très court programme remplacera cette fonction indispensable, il pourra être incorporé comme sous-programme dans un ensemble plus important.

```
60000 '  FONCTION "INSTR"
60005 '  RECHERCHE B$ DANS A$ A PARTIR DE I
60007 '
60010 LA=LEN(A$):LB=LEN(B$):IF I=0 THEN I=1
60020 IF I>LA THEN RETURN
60030 FOR P=I TO LA
60040 IF B$=MID$(A$,P,LB) THEN RETURN
60050 NEXT P:P=0:RETURN
```

A la sortie du sous-programme, P contiendra la position de B$ dans A$, ou Ø si la recherche s'est avéré négative.

SPACE$**(N) et **STRING$(N,X$)

Application: STRING$ retourne une chaîne contenant N fois le premier caractère de X$. Quand à SPACE$, il retourne une chaîne composée de N espaces (code ASCII 32).

Équivalent Atmos: Il suffit de déclarer en début de programme une chaîne (S$ par ex.), contenant un certain nombre de fois le caractère désiré. La fonction LEFT$(S$,N) nous fournira ensuite la chaîne de longueur adéquate.

Exemple:

```
10 S$="----------------------------------------"
20
.
.
100 PRINT LEFT$(S$,10);"RESULTATS";LEFT$(S$,10)
110 PRINT LEFT$(S$,29)
```

Donnera:

```
----------RESULTATS----------
-----------------------------
```

2.

L'organisation mémoire et les périphériques

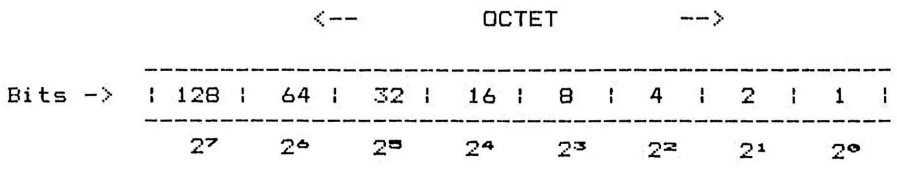
2.1. Généralités. Rappels

Pour bien comprendre la manière dont un ordinateur manipule les informations, il est important de connaître à la fois le type de données qu'il utilise, leur mode de stockage en mémoire et la manière dont celle-ci est structurée. Revoyons rapidement quelques principes de base :

1) L'unité d'information la plus élémentaire est le **BIT**, qui ne peut prendre que deux valeurs : Ø ou 1. Pour y gagner en vitesse et en efficacité, on regroupe ceux-ci en "mots" de 8 bits, appelés octets.

2) Un **Octet** peut prendre 256 (2^8) valeurs différentes, de Ø à 255 ; il constitue l'unité de stockage pour toutes les informations transitant par la mémoire, ainsi que la plus petite unité directement accessible à l'utilisateur. Certaines données, comme le code d'un caractère par exemple, peuvent se stocker sur un seul octet ; mais pour des valeurs plus grandes que 255, deux octets ou plus sont nécessaires.

3) Chacun des 8 bits constituant l'octet ne peut valoir que Ø ou 1, mais on attribue à chacun d'entre eux un "poids" différent, ce qui permet d'obtenir les 256 combinaisons citées plus haut. Ce principe est illustré par le schéma suivant:



Donc, pour obtenir la valeur totale d'un octet, on multiplie le poids de chacun des bits le constituant par sa valeur propre (Ø ou 1), et on additionne le tout: Si, par exemple, les Bits de "poids" 16, 8 et 1 sont à 1 et tous les autres à Ø, l'octet aura la valeur 16+8+1=25. On voit que grâce à cette méthode de codage, un octet peut effectivement prendre toutes les valeurs entières comprises entre Ø (tous les bits à Ø) et 255 (tous les bits à 1).

4) Pour savoir où est rangée chaque information et y accéder rapidement, l'ordinateur utilise des **Adresses**: Chaque octet est accessible par son adresse, qui permet de le définir par rapport à ses voisins.

On a très souvent besoin de stocker une adresse en mémoire; deux octets sont nécessaires dans ce cas, dans la mesure où sa valeur peut être comprise entre Ø à 65535 donc supérieure à 255. Un des deux octets, dit "de poids fort" contiendra le quotient entier de la division <nombre à coder>/256, et l'autre octet dit "de poids faible" le reste de cette opération. Ces deux octets seront ensuite placés dans deux adresses consécutives, celui de poids faible en premier.

Supposons par exemple que l'on veuille stocker le nombre 356Ø aux adresses 1ØØØ et 1ØØ1: Il faudra placer la partie entière de la division 356Ø/256, donc 13 en 1ØØ1, et le reste (232) en 1ØØØ.

Une instruction, POKE, est prévue pour charger un octet dans une adresse donnée, et DOKE évite tout calcul en plaçant un nombre codé sur deux octets dans deux adresses consécutives, conformément à la règle exposée ci-dessus (octet de poids faible en premier).

Le système hexadécimal

Notre système de comptage habituel est décimal, probablement parce que nous avons dix doigts: Nous disposons de 1Ø signes allant de Ø à 9, qui ont aussi un "poids" différent en fonction de leur position. Le chiffre le plus à droite est multiplié par $1Ø^0$ (1), celui à sa gauche par $1Ø^1$ (1Ø), le suivant par $1Ø^2$ (1ØØ), etc. Le total s'obtient par addition, comme pour un nombre binaire. Ainsi, 354 est égal à $3*1Ø^2+5*1Ø^1+4*1Ø^0$.

Comme la conversion du binaire au décimal est assez malaisée, on utilise en informatique le système **hexadécimal**, c'est-à-dire de base 16. On dispose donc de 15 signes, les 1Ø premiers étant Ø à 9 et les cinq autres A, B, C, D, E et F. A=1Ø, B=11, etc. jusqu'à F qui vaut 15 en décimal.

La raison d'utiliser ce système est très simple: Si l'on sépare un octet en 2 "mots" de 4 bits, chacun d'entre eux peut s'écrire sur un seul chiffre hexadécimal puisqu'il ne peut prendre que 16 (2^4) valeurs différentes. Un octet, qui s'exprime toujours par un nombre hexadécimal à deux chiffres, pourra être facilement converti par groupes de 4 bits: Dans le nombre binaire Ø111 1111 (127), le groupe de droite vaut 15, donc F, et le groupe de gauche vaut 7, qui reste 7 en base 16. Cet octet s'écrira donc "7F" en hexadécimal.

Le Basic de l'Atmos reconnaît un nombre comme hexadécimal s'il est précédé de "#", et convertit les nombres décimaux en base 16 grâce à la fonction HEX$.

Exemples:

Binaire	Décimal	Hexadécimal
00001000	8	8
00100100	36	24
11000101	197	C5
01101110	111	6F

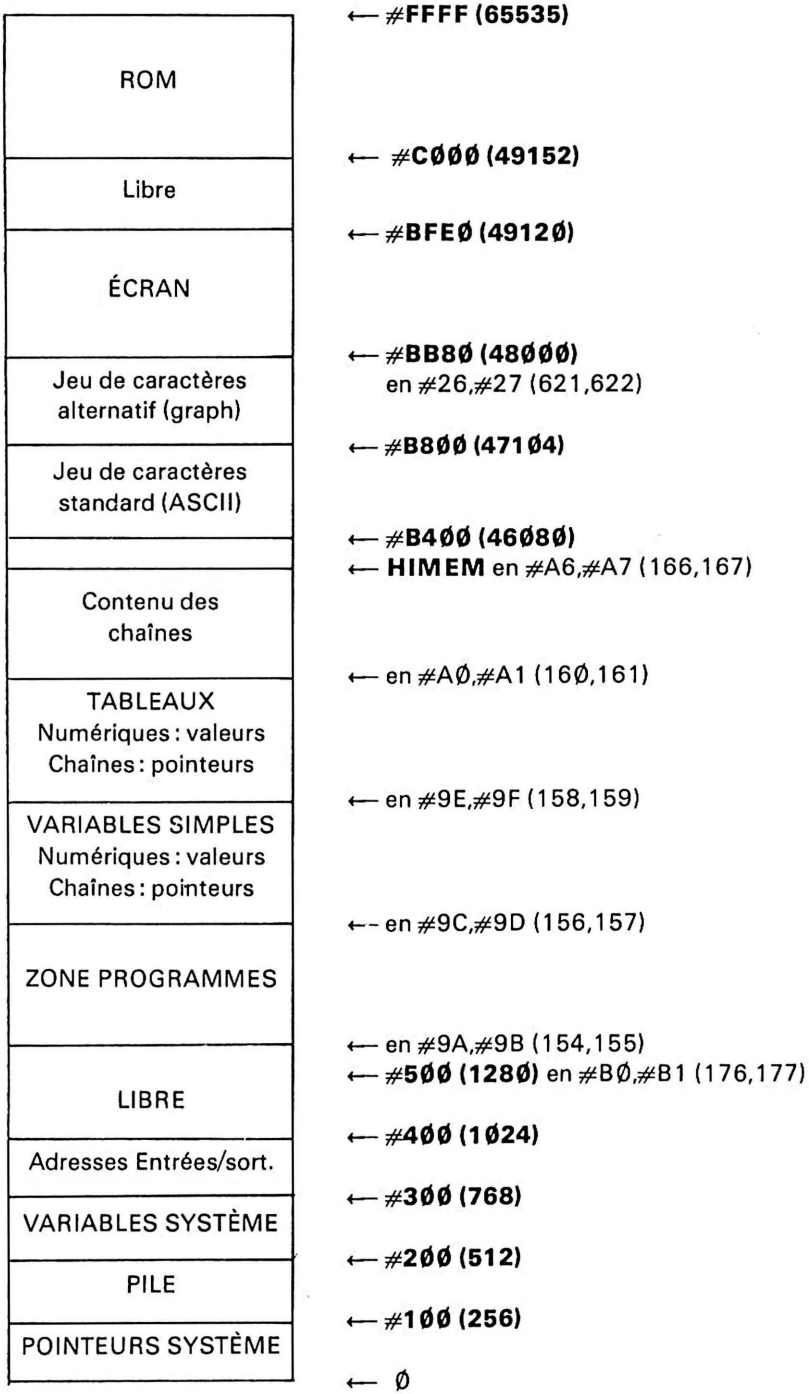
2.2. Le Memory-Map

Examinons maintenant la figure 1, qui est un "plan" de la mémoire vive, ou memory-map. On remarque que toutes les données indispensables au

Memory-Map

Version 48 K — Mode TEXT

#ØØØØ : Hexadécimal (ØØØØ) : Décimal



fonctionnement de l'Atmos y sont présentes : ROM contenant l'interpréteur Basic, écran, jeux de caractères, programmes éventuels, etc. La mémoire pourrait être comparée à un livre, divisé en chapitres et sous-chapitres, d'inégale importance et traitant de sujets différents. Chacun d'eux n'est susceptible de contenir qu'un type précis d'information. Quant à la zone située en bas de mémoire, entre les adresses Ø et #5ØØ, elle représente le "sommaire" du livre : Tous les octets situés entre ces deux valeurs constituent des "emplacements réservés", où sont stockées toutes les données susceptibles de varier en cours de fonctionnement.

En effet, si sur le papier les différentes parties de la mémoire semblent posséder des limites fixes, il n'en est rien en réalité (ce serait trop simple !). Certaines d'entre elles le sont bien sûr, comme celle contenant les variables système ou la ROM ; mais la plupart (celles séparées par des tirets sur le schéma) voient leur taille varier constamment, fluctuant au gré des besoins. Cela paraît évident pour la zone programme par exemple, dont la taille sera bien sûr fonction de l'importance de celui-ci. Pour connaître à tout moment la position de ces adresses variables, l'ordinateur stocke leur valeur dans d'autres adresses, qui elles sont fixes et connues ; ces dernières sont appelées les **pointeurs**.

Sur ce memory-map, les adresses fixes sont marquées en **gras**, et les nombres écrits normalement renvoient au pointeur contenant l'adresse considérée.

2.3. *Les différentes zones de la mémoire*

Dans ce chapitre sera donnée en détail la fonction de chaque partie spécialisée de la mémoire, la manière dont les différentes données qui la concernent y sont stockées ainsi que la liste des pointeurs ou variables-système s'y trouvant ou s'y référant.

REMARQUES :

— A la mise sous tension, toute la mémoire "utilisateur", c'est-à-dire la zone comprise entre #4ØØ et HIMEM est remplie du caractère "U" (ASCII 85). Ne vous étonnez donc pas de rencontrer souvent 85 si vous faites des PEEK dans cette zone, ou 21845 (85+85*256) si vous utilisez DEEK.

— Lorsqu'une seule adresse est donnée, c'est que son contenu est un nombre inférieur à 255, lu par PEEK et modifié par POKE. Dans le cas de deux adresses consécutives servant elles-mêmes à stocker une adresse, DEEK et DOKE seront bien entendu utilisés.

2.3.1. Les pointeurs-système Ø–# FF (Ø-255)

C'est là que l'on peut trouver tous les pointeurs renseignant sur les limites des zones programme et variables; leur lecture ou modification par le programmeur peut servir simplement à connaître l'espace occupé par un programme, ou à effectuer des opérations très sophistiquées comme l'effacement sélectif des variables et la récupération de programmes détruits.

#9A,#9B (154,155)

Fonction: Adresse du premier octet de la zone des programmes Basic, normalement initialisée à 1281.

Utilisation: Habituellement employée en conjonction avec l'adresse suivante, pour connaître le nombre exact d'octets occupés par un programme, hors variables.

REMARQUE: Cette adresse peut être déplacée, si par exemple vous désirez faire commencer votre programme plus loin. Mais n'oubliez pas de donner la valeur choisie +2 aux pointeurs suivants (#9C, #9E, #AØ), sinon des choses étranges se produiront lorsque vous taperez le programme ou entrerez les variables!

#9C,#9D (156,157)

Fonction: Fin de la zone programme, initialisée à 1283 à la mise sous tension.

Utilisation: DEEK(#9C)-DEEK(#9A) retournera l'espace mémoire, en octets, occupé par le programme résident.

Comment l'Atmos mémorise les lignes de programme:

Pour bien illustrer le principe de stockage des lignes, tapons au clavier ce petit programme:

```
10 REM DEMO
20 AB=10:C$="XXX"
30 PRINT AB;C$
```

Maintenant, effacez l'écran (CTRL L), et tapez en mode direct:

```
FOR N=1280 TO DEEK(#9C):PRINT N;PEEK(N);" ";:NEXT
```

Les adresses comprises entre 128Ø et 1324 s'afficheront à l'écran, chacune suivie de son contenu. Examinons celui-ci:

— 128Ø: Ø, indique le début de la 1re ligne.

— 1281,1282: Ces deux adresses contiennent 12 et 5, qui donnent 12+5*256=1292. Il s'agit en fait de l'adresse de début de la ligne suivante.

— 1283,1284: Codent le numéro de la ligne, ici 1Ø.

La suite représente le contenu de la ligne lui-même:

— 1285: 157 est le code de stockage de l'instruction REM.

— 1286-129Ø: Un espace (ASCII 32), suivi des codes ASCII des 4 lettres D, E, M et O.

— 1291: La ligne se termine par Ø.

— 1292,1293: C'est là que renvoie le début de la ligne précédente (1281,1282): Il s'agit bien du début de la deuxième ligne de programme. Cette fois-ci, on trouve les valeurs 31 et 5 donnant 1311, début de la troisième ligne.

— 1294,1295: N° de ligne (2Ø).

— 1296-13Ø9: On trouve ensuite tous les codes des caractères contenus dans la ligne. Remarquons aux adresses 1298 et 13Ø4 que le signe "=" n'est pas représenté par son code ASCII qui est 61, mais par son code d'opérateur (212).

— 131Ø: Fin de la deuxième ligne (Ø).

— 1311,1312: Début de la troisième ligne. Les valeurs 43 et 5 donnant 1323 fournissent l'adresse où commencerait la quatrième ligne de programme, si celle-ci existait. De cette manière, si vous ajoutiez une

ligne supplémentaire son adresse de début serait déjà stockée au début de la précédente.

— 1313,1314: N° de ligne (3Ø).

— 1315-1321 : Code de l'instruction PRINT (186), suivi des codes ASCII de tous les caractères composant la ligne.

— 1322: Fin de la troisième ligne (Ø).

— 1323,1324: Les deux octets sont mis à Ø, indiquant la fin du programme (ou l'absence de 4e ligne).

On retiendra trois points importants de ce qui précède:

1) Chaque instruction, fonction ou opérateur possède son propre code, valeur entière comprise entre 129 et 255 inclus; ce code est uniquement utilisé lors du stockage des lignes de programme en mémoire. Ce système autorise une très importante économie de mémoire, dans la mesure où chaque commande n'occupe qu'un seul octet. Si elles étaient mémorisées comme des mots, les commandes nécessiteraient autant d'octets qu'elles possèdent de caractères.

2) Chaque ligne commence par deux adresses codant l'adresse de début de la ligne suivante. C'est cette particularité qui a été utilisée dans le programme RENUM (cf. § 1.6), où le programme principal est renuméroté en sautant directement aux adresses contenant le N° de ligne.

3) Enfin, et ce quel que soit le programme en mémoire et sa longueur, les adresses 1281 et 1282 (début de la zone) contiendront une valeur nécessairement différente de Ø. Si on leur donne cette valeur en tapant DOKE 1281,Ø cela équivaut à faire un **NEW**, puisque l'ordinateur considèrera qu'il n'y a plus aucune ligne de programme après cette adresse. Heureusement et même si l'on utilise NEW (qui ne fait jamais que la même chose), le programme est toujours présent en mémoire tant que rien n'est venu le remplacer, et peut donc être récupéré. (Voir adresse #BØ, ci-après).

#9E,#9F (158,159)

Fonction: Fin de la zone affectée aux variables simples, initialisée à 1283.

Utilisation: Le début de cette zone coïncide avec la fin du programme (#9C), et ne nécessite donc pas de pointeur spécial. Si on élimine la zone des variables en tapant DOKE #9E,DEEK(#9C), on obtiendra le même effet qu'avec CLEAR mais **en conservant les tableaux**, qui sont placés ailleurs.

Mode de stockage des variables simples:

Celles-ci sont rangées au fur et à mesure de leur affectation, en débutant par les adresses les plus basses de la zone:

Les variables en virgule flottante commencent par deux octets contenant le code des deux caractères du nom, puis cinq octets représentant leur valeur. Total occupé: Sept octets.

Pour *les variables entières*, les deux octets du nom contiennent le code des caractères correspondants plus 128, et la valeur ne nécessite que deux octets puisqu'elle sera nécessairement comprise entre —32768 et +32767.

Exemple: Soit AB%, dont la valeur est 35ØØ. Elle sera codée sur quatre octets qui seront:

```
193   Code ASCII de A (65) plus 128
194   Code ASCII de B (66) plus 128
172   Reste de la division entière 3500/256
 13   Quotient entier de 3500/256
```

Pour *les variables de chaîne*, les choses se compliquent quelque peu, car les adresses les concernant ne constituent que des pointeurs, renvoyant à une autre zone où leur contenu se trouve physiquement:

— Le premier octet est le code du 1er caractère du nom.

— Le deuxième octet est le code du 2e caractère plus 128.

— L'octet suivant contient la longueur de la chaîne, en nombre de caractères (ce qui explique pourquoi les chaînes ne peuvent jamais dépasser 255 caractères !).

— Les deux octets suivants codent l'adresse où commence le stockage de la chaîne proprement dite, qui peut en fait se trouver n'importe où dans la zone "contenu des chaînes". Cette adresse combinée avec l'octet précédent donnant la longueur permet donc de toujours retrouver ce contenu avec précision.

— Enfin, deux octets toujours à Ø.

Donc, une chaîne occupera en mémoire un espace égal à sa longueur plus sept octets de pointeurs.

#AØ,#A1 (16Ø,161)

Fonction: Contient l'adresse de fin de la zone des tableaux. (Initialisée à 1283 à la mise sous tension).

Utilisation: Cette partie de la mémoire, qui commence là où se termine celle des variables simples, est réservée d'une part aux tableaux numériques et d'autre part aux pointeurs permettant de retrouver les données des tableaux de chaînes.

Stockage des tableaux numériques:

Espace occupé	*Données*
2 Octets	Codes ASCII des caractères du nom, ou codes plus 128 (tableaux d'entiers).
2 Octets	Nombre total d'octets occupés par le tableau.
1 Octet	Nombre de dimensions du tableau (D), déduit de l'instruction DIM le concernant.
2*D Octets	Nombre d'éléments par dimension (E), également fourni par DIM.
5*D*E Octets	Valeur de chaque élément du tableau, pour un tableau en virgule flottante, ou bien
2*D*E Octets	Valeur de chaque élément, pour un tableau d'entiers.

Nous voyons que les tableaux en virgule flottante sont très gourmands en mémoire, puisqu'ils occupent un nombre d'octets égal à: 5+2*<nombre dim.>+5*<nombre d'éléments/dim.>*<nombre dim.>. Ainsi, un tableau déclaré par DIM A(1Ø,1Ø) ne prendra pas moins de 525 octets "à vide"!

Stockage des pointeurs de tableaux de chaînes:

Espace occupé	*Données*
2 Octets	Nom du tableau, même que pour les chaînes simples (code du premier car. et code du deuxième car. plus 128).
2 Octets	Espace total pris par les pointeurs.

Espace occupé	*Données*
1 Octet	Nombre de dimensions du tableau (D).
2*D Octets	Nombre de caractères par dimension.
3*D Octets	Pointeurs permettant de retrouver toutes les chaînes du tableau, rangées dans la zone "Contenu des chaînes". — Le 1er octet donne la longueur de la chaîne. — Les deux autres, son adresse de début.

Redimensionner un tableau

Contrairement à certains autres Basic, celui de l'Atmos n'accepte pas de redimensionner (ou effacer) un tableau, toute tentative se soldant par un "REDIM'D ARRAY ERROR". Il existe cependant un moyen simple de le faire, si on peut accepter que tous les tableaux (numériques et chaînes) soient effacés en même temps, les variables simples n'étant pas touchées: Il suffit de faire coïncider les pointeurs de début (#9E,#9F) et de fin (#AØ,#A1) de la zone "tableaux":

```
DOKE #A0,DEEK(#9E)
```

Vous pouvez ensuite redéclarer les tableaux de votre choix par DIM sans aucun problème.

#A2,#A3 (162,163)

Fonction: Donne l'adresse la plus basse atteinte dans la zone "contenu des chaînes".

Utilisation: DEEK(#A2)-DEEK(#A6) fournit l'espace exact occupé par les chaînes de caractères. DEEK(#A2)-DEEK(#AØ) retournera l'espace total demeurant réellement disponible (équivalent à FRE(Ø).

REMARQUE: Les chaînes sont rangées dans la zone située entre HIMEM (voir ce pointeur) et le haut de la section "tableaux" (codée en #AØ,#A1) dans l'ordre de leur arrivée du haut vers le bas, c'est-à-dire en débutant juste au-dessous de HIMEM. C'est l'adresse atteinte par ce remplissage à un moment donné que nous fournissent #A2,#A3. Les zones "programme", "variables" et "pointeurs de chaînes", par contre, s'étendent du bas vers le haut au fur et à mesure des nouvelles entrées. C'est la

zone libre entre les deux adresses qui permet la création de nouvelles variables dans un programme ; une fois celle-ci remplie, le message "OUT OF MEMORY ERROR" est envoyé.

#A4,#A5 (164,165)

Fonction : Contient l'adresse où commence le stockage des caractères de la dernière chaîne entrée.

Utilisation : La lecture des adresses situées entre DEEK(#A4) et DEEK(#A2) retournera la totalité des caractères composant la dernière chaîne mémorisée, même si ses pointeurs ont été détruits (cas d'une chaîne effacée par un programme).

HIMEM #A6,#A7 (166,167)

Fonction : Définit à la fois l'adresse la plus haute accessible par le Basic, et le début de la zone de stockage du contenu des chaînes de caractères. (Chaînes simples et tableaux).

Utilisation : Cette adresse est la seule de la mémoire directement accessible par une instruction, c'est dire son importance. En fait, toute la mémoire au-delà de la valeur spécifiée ne sera plus affectée ni par NEW, ni par CLEAR et sera donc tout indiquée pour l'écriture de routines en langage-machine. Seul un RESET remettra tout à zéro et rétablira les limites normales.

#BØ,#B1 (176,177)

Fonction : Adresse la plus basse accessble par le Basic, ou LOMEM, initialisée à 128Ø.

Utilisation : Tout ce qui est situé en-dessous de l'adresse contenue dans #BØ,#B1 n'est affecté par aucune commande Basic, comme NEW ou CLEAR.

#B8,#B9 (184,185)

Fonction : Adresse de la dernière variable entrée (valeur pour un nombre, et pointeurs pour une chaîne).

Avant de passer à la suite il faut bien comprendre que tous les pointeurs que nous venons d'examiner sont manipulés par l'ordinateur pour toutes les opérations "globales" sur la mémoire.

En effet, tout effacement de zone par exemple est obtenu par simple modification des pointeurs définissant la partie à effacer; rien n'empêche le lecteur d'en faire autant en se servant des indications fournies et d'un peu de bon sens.

***Exemples*:**

Émulation de l'instruction NEW

Nous avons vu plus haut que chaque ligne de programme commençait par l'adresse de la ligne suivante, et que la fin du programme est indiquée par la mise à zéro des octets réservés à cet usage. Il suffit donc de mettre à Ø les deux premiers octets du programme, qui devraient normalement contenir l'adresse de la deuxième ligne, pour "effacer" celui-ci:

```
DOKE DEEK(#9A),0  ou plus simplement
DOKE 1281,0
```

Récupérer un programme détruit par NEW

Dans la mesure où cette instruction agit précisément comme décrit dans l'exemple précédent, voici la marche à suivre:

— Placer le début de la zone variables au-delà de la fin estimée de votre programme, par ex. DOKE #9C,128Ø+5ØØØ si celui-ci fait environ 4 Koctets.

— Lister à l'écran les octets correspondant à la première ligne de programme, en comptant large (un octet par caractère, étant donné qu'une commande n'en occupe qu'un):

FOR N=1281 TO 1281+<Nombre de car.>:PRINT N;PEEK(N);" ":NEXT

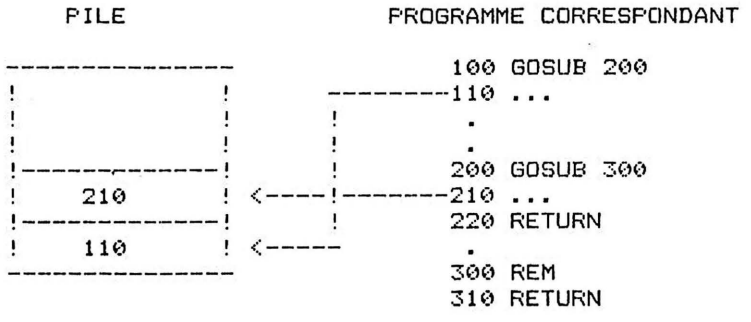
— En excluant 1281, 1282 qui sont de toutes façons à Ø et 1283, 84 qui représentent le numéro de ligne, repérer le premier octet ayant la valeur Ø: L'adresse suivante sera le début de la deuxième ligne du programme.

— Charger cette adresse dans 1281, 82 en tapant DOKE 1281,<adresse trouvée>: Votre programme est miraculeusement revenu. A présent, empressez-vous de le sauvegarder car tous les autres pointeurs (ceux des variables notamment) ont également été déplacés par NEW, et il serait malsain de continuer sur de telles bases!

2.3.2. La Pile # 1ØØ- #1FF (256-511)

Celle-ci est utilisée pour le stockage des adresses de retour des GOSUB, REPEAT-UNTIL et autres FOR-NEXT. En effet, lorsque par exemple un GOSUB est rencontré, l'adresse de début de la ligne suivante ou "adresse de retour" est immédiatement placée dans la pile; le programme saura ainsi où reprendre quand il arrivera à un RETURN; dès que cela est fait, l'adresse désormais inutile est extraite de la pile. En cas de GOSUB multiples ou de boucles FOR-NEXT imbriquées, les adresses de retour sont "empilées" les unes sur les autres, et extraites de telle sorte que la dernière arrivée est toujours la première à ressortir. La taille de la pile n'étant pas infinie, on comprend mieux pourquoi on n'a droit qu'à un nombre donné de boucles ou GOSUB imbriqués.

Exemple:



Lorsque la ligne 1ØØ est exécutée, l'adresse de retour (11Ø) est placée dans la pile. A la ligne 2ØØ, c'est au tour de 21Ø d'y entrer. Ainsi, à l'exécution du RETURN de la ligne 31Ø on retournera bien en 21Ø, puis ensuite en 11Ø.

2.3.3. Les variables-système #2ØØ-#2FF (512-767)

Véritable tableau de bord et de commandes interne de l'ordinateur, les variables-système sont de loin la zone la plus intéressante à explorer pour le programmeur, par la variété des interventions qu'elles autorisent par rapport au Basic, sur l'affichage et le clavier en particulier.

Avant d'aborder les plus importantes d'entre elles, rappelons que les variables-système mémorisent des données aussi diverses que le début et la fin de l'écran, le code de la dernière touche tapée ou le mode et la position du curseur. On peut toujours se renseigner sur la valeur en cours à l'aide de PEEK, mais la plupart du temps on peut également modifier celle-ci avec l'instruction POKE et c'est bien entendu cette dernière caractéristique la plus intéressante.

Les diverses variables étudiées sont regroupées en celles concernant l'écran et celles concernant le clavier plutôt que par ordre croissant, autorisant ainsi une consultation aisée.

LE CLAVIER

#2Ø8 (52Ø)

Fonction: Contient le code de la touche frappée au clavier.

Utilisation: La lecture de cette variable par PEEK fournit une alternative intéressante à la fonction KEY$, dans tous les cas où la détection rapide de l'appui sur une touche est primordiale (jeux en temps réel).

REMARQUES:

— Cette variable détecte l'appui sur les touches générant un caractère uniquement. Pour les touches de fonctions (SHIFT, CTRL, etc. voyez #2Ø9.

— Si aucune touche n'est actionnée, la valeur 56 est retournée.

— Attention, le code renvoyé n'est pas le code ASCII du caractère correspondant à la touche, mais un code interne définissant la touche. Les valeurs des touches qui vous intéressent devront être déduites par expérimentation, à l'aide d'un petit programme de la forme suivante:

```
10 CLS:PRINT CHR$(17)
20 PRINT@10,10;PEEK(#208)" "
30 GOTO 20
```

#**2Ø9** (521)

Fonction: Détecte l'appui sur une touche de fonctions. Les valeurs retournées sont les suivantes:

56 Aucune touche de fonction.
164 SHIFT gauche.
167 SHIFT droit.
162 CTRL
165 FUNCT

Utilisation: La lecture de cette adresse constitue le seul moyen de détecter l'usage des touches SHIFT et CTRL sans être tenu de tester le caractère généré. C'est également le seul moyen d'utiliser la touche FUNCT, qui ne génère aucun caractère. Le petit programme suivant vous donnera un exemple sur la manière d'exploiter cette variable:

```
10 ' TEST TOUCHES DE FONCTIONS
15 '
20 CLS:PRINT CHR$(17)CHR$(4)
30 A$="AUCUNE TOUCHE "
40 A=PEEK(#209)
60 IF A=162 THEN A$="<CTRL>          "
70 IF A=164 THEN A$="<SHIFT GAUCHE>"
80 IF A=167 THEN A$="<SHIFT DROITE>"
90 IF A=165 THEN A$="<FUNCT>         "
100 PRINT@4,9;CHR$(27)"JVOUS APPUYEZ SUR "A$
110 GOTO 30
```

#**2ØC** (524)

Fonction: Prend la valeur 127 en mode minuscules, et 255 en mode "blocage majuscules" (CAPS).

Utilisation: Très utile pour forcer le clavier dans le mode désiré sans tenir compte de son état précédent, ce que ne permet pas CTRL T qui agit en bascule.

#**2ØE** (526)

Fonction: Décompte le temps d'appui sur une touche, et déclenche l'auto-répétition si une certaine valeur est atteinte.

Utilisation: Au repos (aucune touche actionnée), l'octet a la valeur 32 (Bit 5 à 1). Si une touche est pressée:

— L'octet est décrémenté d'une unité à chaque scanning clavier;

— Lorsque la valeur 1 est atteinte, l'autorepeat est déclenché. Tant que celui-ci dure, seuls les Bits Ø, 1 et 3 pourront être à 1.

Concrètement, cela signifie qu'il est aisé de détecter l'appui sur une touche (octet<32), ainsi que la mise en route de l'autorepeat (octet<5). On peut aussi forcer la répétition dès qu'une touche est actionnée:

Exemple:

```
10 X$=KEY$
20 POKE 526,1:PRINT X$;
30 GOTO 10
```

— Pour toute autre valeur que 1 donnée comme argument à POKE 526, la répétition est bloquée.

— On peut ralentir la vitesse à volonté en insérant une instruction WAIT N avant GOTO 1Ø, avec 1<N<15.

Voyez aussi la variable suivante, #24E.

#**24E** (59Ø)

Fonction: Définit le temps entre l'appui sur une touche et le déclenchement de l'autorépétition. Sa valeur est de 1Ø à la mise sous tension.

Utilisation: Les applications possibles sont multiples, que vous trouviez que le curseur est trop lent, ou que vous vouliez que la répétition se déclenche dès l'appui sur une touche (dans un jeu, par ex.). Notez bien que cette variable ne modifie pas la *vitesse* de répétition (pour cela, voyez #24F), mais bien la vitesse de déclenchement de la répétition.

Il faut bien comprendre que cette adresse ne stocke pas des données fugitives comme #2ØE, mais une valeur qui est prise en compte tant qu'elle n'est pas modifiée par le programmeur ou réinitialisée par RESET.

#**24F** (591)

Fonction: Détermine la vitesse d'autorépétition du clavier.

Utilisation: La valeur normale est de 4 à la mise sous tension, on dispose donc de 3 valeurs seulement (1 est très très rapide...). Les adresses suivantes procurent un réglage plus "fin":

#3Ø6,#3Ø7 (774,775)

Fonction: Détermine le rapport de clignotement du curseur, et accessoirement la vitesse d'autorépétition du clavier.

Utilisation: La valeur normale est de 1ØØØØ sur les deux adresses, mais #3Ø6 étant l'octet de poids faible il pourra être ignoré en pratique, pour n'agir que sur #3Ø7. Sa valeur normale est alors de 39, valeur qu'il faudra diminuer (jusqu'à 1Ø environ) pour accélérer, et augmenter pour ralentir la vitesse de répétition. Cette adresse procure un contrôle beaucoup plus fin de cette vitesse, mais elle agit aussi malheureusement sur l'aspect du curseur.

L'ÉCRAN

REMARQUE: Les applications de toutes les adresses décrites dans cette rubrique sont développées en détail dans la troisième partie concernant l'écran.

#12,#13 (18,19)

Fonction: Contient l'adresse de début de la ligne courante (ligne où se trouve le curseur), dans la partie de la mémoire concernant le contenu de l'écran.

Utilisation: Du fait qu'il n'existe aucune fonction renseignant sur la position du curseur à l'écran, DEEK(#12) nous fournira la ligne où celui-ci se trouve en mémoire, comme PEEK(616) retourne la ligne du curseur à l'écran (cf. cette adresse, ci-après).

#268 (616)

Fonction: Retourne la ligne où se trouve le curseur à l'écran, de 1 à 27.

Utilisation: Cette adresse très importante, utilisée conjointement avec l'adresse suivante (#269), procure une souplesse d'utilisation supplémentaire à l'instruction PRINT @ en permettant de connaître à tout moment la position du curseur. Attention toutefois: Pour PEEK (616), la première ligne de l'écran accessible est la numéro 1, alors qu'il s'agit de la N° Ø pour PRINT @ .

#269 (617)

Fonction : Contient le numéro de la colonne où se trouve le curseur, de Ø à 39.

Utilisation : Contrairement à la fonction POS qui joue à peu près le même rôle, PEEK(617) suit les déplacements du curseur à l'aide de PRINT@.

Cette adresse peut aussi déplacer le curseur sur une colonne quelconque de la ligne en cours, en tapant POKE 617,<colonne>. Évidemment, l'essai en mode direct ne donnera aucun résultat puisque le curseur passe à la ligne suivante, colonne Ø une fois la commande exécutée. Pour voir son effet, il faut essayer une ligne du type POKE 617,2Ø:?"XXXX".

#26A (618)

Fonction : Contient toutes les données relatives au mode curseur, correspondant à certains codes CTRL du clavier. Voici celles qui ont pu être déterminées :

Bit	*Valeur*	*CTRL*	*Fonction*
1	Ø 1	Q	Curseur invisible Curseur visible
2	Ø 1	S	Blocage de l'affichage Déblocage de l'affichage
4			Néant
8	Ø 1	F	Rappel sonore au clavier Clavier muet
16	Ø 1		ESCAPE a été tapé
32	Ø 1	]	Affichage en 4Ø colonnes Affichage en 38 colonnes

Bit	*Valeur*	*CRTL*	*Fonction*
64	Ø 1	D	Affichage en simple hauteur Affichage en double hauteur
128			Néant

Utilisation: Tous les codes CTRL ayant des effets équivalents à ce qu'on peut obtenir avec POKE 618 agissant comme des bascules, il est souvent intéressant de pouvoir "forcer" tel ou tel mode indépendamment du mode précédent. A la mise sous tension #26A a la valeur 3, qui correspond bien à un curseur visible et clavier sonore. Voir aussi codage d'un octet, § 2.1.

REMARQUE: Le codage représenté dans le tableau est un codage "bit à bit", chaque bit de l'octet présent à l'adresse 618 étant suivant sa valeur (Ø ou 1) associé à un mode curseur. "1" est le bit de poids faible, et 128 celui de poids fort.

Exemple: POKE 618,74 démarre la double hauteur (64), rend le clavier muet (8), et garde l'affichage actif (2): 64+8+2=74. Le curseur disparaît (Bit de poids 1 à Ø, d'une manière générale toutes les valeurs impaires sont "curseur visible").

Les codes CTRL sont traités in extenso au § 3.2.1.

#26B et #26C (619-62Ø)

Fonction: Mémorisent respectivement l'attribut du fond et de l'affichage, pour tout l'écran. 619 contient l'octet placé dans toute la première colonne de l'écran, et 62Ø celui de la deuxième. Leur valeurs normales sont 23 et Ø (fond blanc, affichage noir).

Utilisation: Lorsqu'il s'agit de couleurs, PAPER et INK sont bien plus commodes, mais en pokant dans ces adresses on peut aussi bien faire clignoter l'écran, ou passer en caractères géants. Voir aussi § 3.6.2.

REMARQUE: L'effet de l'attribut donné à 619 ou 62Ø n'est visible qu'après un effacement de l'écran, avec CTRL L ou CLS.

Exemple:

```
POKE 618,67:POKE 619,10:INK 7:CLS
```

#27A,#27B (634,635)

Fonction: Stocke l'adresse de début de la première ligne de l'écran accessible au curseur. La valeur normale est de 48Ø4Ø, laissant la ligne Ø hors de portée.

Utilisation: Grâce à cette adresse ainsi qu'aux deux suivantes (voir ci-dessous), l'utilisateur peut définir quelle taille aura l'écran, et même la changer instantanément. La conversion ligne/adresse s'obtient par:

```
47960+<ligne>*40
```

Exemple: Pour "récupérer" la ligne Ø, taper DOKE # 26D,4796Ø.

#27C,#27D (636,637)

Fonction: Spécifie le nombre de caractères qui seront pris lorsque le scrolling intervient. La valeur est du nombre de lignes * 4Ø.

Utilisation: L'action de cette variable est assez délicate à expliquer, mais disons que si un nombre correspondant par exemple à 2Ø lignes y est chargé, lorsque le curseur arrivera en bas de l'écran les six dernières lignes ne défileront pas avec le reste: Elles demeureront fixes et la ligne les précédant sera répétée à la place.

Le seul intérêt réel de cette variable est que, utilisée conjointement avec #27A et #27E, elle permet de définir des "fenêtres d'affichage" à l'intérieur de l'écran.

#27E (638)

Fonction: Spécifie la dernière ligne de l'écran accessible au curseur, la valeur normale étant 27.

Utilisation: Adresse complémentaire des deux précédentes, elle permet d'interdire à l'affichage un nombre de lignes quelconque en bas d'écran. Le curseur "bute" sur la nouvelle ligne fixée, comme s'il s'agissait réellement de la dernière.

Exemple: Éteindre et rallumer l'Atmos, puis taper:

```
POKE 638,10:FOR N=1 TO 10:?:NEXT
```

L'utilisation pratique de ces trois dernières variables est expliquée et illustrée d'exemple au § 3.5.2.

2.3.4. Les autres zones de la mémoire

#3ØØ-#3FF (768-1Ø23)

Dans cette section sont situés tous les pointeurs utilisés par le système d'exploitation de cassettes (SEC) et par l'interface parallèle de l'imprimante (dont les fonctions sont vues au § 2.4.1), ainsi que pour les communications internes dans le détail desquelles nous n'entrerons pas.

La page 4 #4ØØ-#4FF (1Ø24-1279)

Cette zone est laissée libre, permettant à l'utilisateur de stocker ses propres routines en langage-machine, ou toutes sortes de données que l'on veut rendre inaccessibles à NEW, RUN ou CLEAR. Ces adresses sont employées dans certains des programmes qui suivent.

La mémoire utilisateur #5ØØ-#B3FF (128Ø-46Ø8Ø

Il s'agit d'une section qui totalise quelque 44 Koctets exploitables regroupant programmes et données, et dont seules les deux limites extrêmes sont fixes, les autres étant gérées par les pointeurs qui ont été traités au § 2.1.1.

Les jeux de caractères et l'écran #B4ØØ-#BFEØ (46Ø8Ø-4912Ø)

La zone située entre #B4ØØ et #BB7F regroupe le dessin de tous les caractères disponibles sur l'Atmos, chaque dessin de caractère étant codé sur 8 octets (8×8=64 point/caractère). Ceux-ci, ainsi que la manière de les modifier, sont vue au § 3.7.

Il en est de même pour la mémoire de l'écran texte, où chaque octet représente une position d'affichage, et située entre #BB8Ø et #BFEØ, examinée au § 3.1.

La ROM (mémoire morte)

Suivent une trentaine d'octets également disponibles pour le programmeur situés entre #BFEØ et #BFFF, et nous arrivons à la ROM. Ou plutôt, serait-il plus correct de dire "l'image de la ROM en RAM", car en effet cette dernière, tout comme les jeux de caractères et toutes les variables situées

entre Ø et #5ØØ est recopiée en RAM lors de la mise sous tension, laissant ainsi le champ libre à d'éventuelles modifications. Ces dernières (sur les jeux de caractères notamment) demeureront effectives jusqu'à l'extinction de la machine ou un RESET.

Simulation de RESET par programme

Deux routines, l'une située dans les variables système et l'autre en ROM, appelées par CALL<adresse> permettent d'obtenir ce résultat:

#F88F (63631) Effectue un RESET total, avec remise de toute la mémoire à l'état initial et perte du programme et des variables. Le logo Tangerine est affiché en fin d'opération, comme à la mise sous tension.

#247 (583) Équivalent de RESET, remet la mémoire à l'état initial sans perte des variables et du programme et sans affichage du logo.

2.4. *La gestion des périphériques*

Les quatre périphériques actuellement gérés par l'Atmos sont le clavier, l'écran, l'interface parallèle imprimante et le magnétophone à cassettes. Les différentes variables et instructions concernant le clavier ayant été vues plus haut et la troisième partie étant entièrement consacrée à l'écran, nous traiterons ici de l'imprimante et du magnétocassette.

2.4.1. L'Imprimante

L'interface disponible sur l'Atmos est du type Centronics parallèle, autorisant la connexion directe à un large éventail d'imprimantes et de tables traçantes sans adaptation particulière. Deux instructions du Basic sont dévolues à son exploitation:

LPRINT, qui agit de la même manière que PRINT en permettant l'envoi de tous les caractères ASCII vers l'imprimante, y compris les caractères de contrôle de cette dernière grâce à LPRINT CHR$(N), N étant compris entre Ø et 31.

LLIST, qui listera le programme sur l'imprimante et dont la syntaxe est identique à celle de LIST.

Remarquons que toutes les fonctions utilisables avec PRINT le sont aussi avec LPRINT, y compris celles concernant le placement horizontal du curseur telles que TAB ou SPC. Par contre, LPRINT @ est considéré comme une erreur de syntaxe.

Une variable-système, #256 (598) détermine le nombre de colonnes de l'imprimante, le caractère "Retour chariot" (CHR$ 13) étant envoyé automatiquement dès que ce nombre de caractères a été imprimé. Sa valeur est de 80 colonnes à la mise sous tension, et peut être aisément modifiée pour s'adapter à votre propre machine, ou pour une application particulière.

Pour des fonctions plus complexes utilisant l'imprimante, voir aussi § 3.4.1 (hardcopy texte) et § 4.4 (hardcopy graphique).

Transfert de zones-mémoire sur l'imprimante

Le programme suivant imprimera toutes les adresses situées entre des limites fixées par vous, suivies de leur contenu, et avec la meilleure présentation possible (Dump mémoire).

```
60000 '  DUMPS MEMOIRE
60005 '
60010 CLS:FO=7:?:?"DUMPS MEMOIRE SUR IMPRIMANTE":?:?
60020 INPUT"ADRESSE DEBUT ";AD:?
60030 INPUT"ADRESSE FIN   ";AF
60040 FIN=AF-AD
60050 ?:INPUT"DESCRIPTIF DE LA ZONE ";M$
60060 LPRINT"ORIC ATMOS -- DUMP HEXA ZONE "M$" "HEX$(AD)" - "
HEX$(AF)
60070 LPRINT:LPRINT:LPRINT
60080 FOR I=0 TO FIN
60090 A$=HEX$(PEEK(AD+I)):B$=HEX$(AD+I)
60100 LPRINT B$"="A$;SPC(10-LEN(A$+B$));
60110 IF(I+1)/FO=INT((I+1)/FO)THEN LPRINT
60120 NEXT:LPRINT
60130 RETURN
```

Explications :

— La variable FO, attribuée à la ligne 60010, correspond au format correct pour une imprimante 80 colonnes (impression de 7 adresses par ligne). Vous pourrez facilement modifier sa valeur pour qu'elle corresponde à votre modèle d'imprimante.

— AD est l'adresse de début, AF l'adresse de fin; A$ est une chaîne formattée (4 caractères), contenant l'octet à imprimer, alors que B$ contient l'adresse: Tous les nombres sont ainsi alignés en colonnes.

L'imprimante-table traçante Oric MCP-4Ø

Cette remarquable petite machine est en fait une table traçante, pouvant également fonctionner en mode "imprimante". Nous n'entrerons pas dans le détail de son exploitation, mais la liste et la syntaxe de ses diverses fonctions données ci-après devrait vous aider à vous en faire une idée si vous ne la possédez pas, et à l'utiliser si vous l'avez.

Les diverses fonctions sont obtenues en envoyant un ou plusieurs caractères à l'imprimante, à l'aide de l'instruction LPRINT. Deux modes sont disponibles, le mode imprimante dans lequel les caractères à imprimer sont envoyés exactement comme à l'écran, et le mode "table traçante" qui fonctionne un peu comme le graphique haute-résolution de l'Atmos.

L'appareil possède une tête rotative à quatre couleurs, et est capable d'effectuer des tracés avec une précision de Ø,2 mm, ou d'imprimer du texte à la vitesse de 12 caractères/seconde.

1) *Mode imprimante*

Codes de contrôle:

Ces codes déterminent certaines actions de la part de l'imprimante, et sont obtenus par LPRINT CHR$(⩽code⩾):

17 Sélection du mode "imprimante"
18 Sélection du mode "table traçante"

Les codes suivants sont uniquement valides en mode imprimante:

8 Recule d'un caractère (backspace)
1Ø Saut de ligne
11 Saut de ligne négatif
13 Retour chariot
29 Passe à la couleur suivante (rotation de la tête).

REMARQUE: La taille des caractères ne peut pas être modifiée en mode texte, cela doit être effectué préalablement en mode table traçante.

2) *Le mode table traçante*

L'imprimante est capable de tracer dans un repère orthonormé à deux dimensions, comportant 48Ø points en abscisses et un nombre illimité en ordonnées. Dans ce qui suit, x est l'abscisse, y l'ordonnée et n, m, c des paramètres quelconques. Une commandes est définie par une lettre, puis un ou plusieurs paramètres séparés par des ",":

```
LPRINT "<lettre>"<param.1>","<param.2>"," (...)
```

Commandes générales

"A" Sélectionne le mode "Texte".

"C" c Sélectionne la couleur (c), valeurs autorisées Ø-3.

"I" Situe l'origine des axes au point actuel de la tête. Par défaut, l'origine des axes est placé à l'endroit où la tête se trouve à la mise sous tension.

"L" n Sélectionne le type de trait utilisé dans les instructions de dessin, un peu comme l'instruction "PATTERN" en graphique haute-résolution (voir 5e partie). Les valeurs autorisées pour (n) sont Ø-12.

Déplacer la tête d'impression

"H" Replace la tête à l'origine du repère (point Ø,Ø).

"M" x,y Déplace la tête au point de coordonnées (x,y).

"R" x,y Déplace la tête de (x) points en abscisses et (y) points en ordonnées (mouvement relatif).

Commandes de dessin

"D" x,y Trace une droite entre la position de la tête et le point (x,y).

"J" x,y Trace une droite entre la position actuelle de la tête et le point situé (x,y) points plus loin (tracé relatif).

"X" a,p,n Trace un axe du repère de coordonnées, la longueur de l'axe dépendant de la longueur de chaque pas et de leur nombre:
- (a) choisit l'axe à tracer, 1 pour x et Ø pour y.
- (p) détermine la longueur de chaque pas.
- (n) détermine le nombre de pas.

Le Texte en mode "table traçante"

"Q" n Détermine la direction de l'impression:
- Ø imprime horizontalement.
- 1 imprime verticalement de haut en bas.

— 2 imprime horizontalement à l'envers.
— 3 imprime verticalement de bas en haut.

"S" n Sélectionne la taille des caractères, valeurs autorisées Ø-63. La taille Ø correspond à 8Ø caractères par ligne, la taille 63 à un seul.

"P" A$ Imprime la chaîne A$.

Exemples:

1) Tracer les axes en rouge:

```
10 LPRINT CHR$(18)"I"
20 LPRINT"C3"
30 LPRINT"X1,25,10":LPRINT"H"
40 LPRINT"X0,25,10":LPRINT"C0"
```

2) Tracer un cercle:

```
10 LPRINT CHR$(18)"I"
20 LPRINT"M240,-240"
30 FOR N=0 TO 2*PI STEP PI/120
40 X=100*COS(N):Y=100*SIN(N)
50 LPRINT"D"X","Y
60 NEXT
```

2.4.2. Le magnétophone à cassettes

Quatre instructions Basic assurent sa gestion: CSAVE et CLOAD, affectées au transfert de programmes (mais dont les applications sont beaucoup plus nombreuses qu'il ne peut y paraître à première vue) et STORE-RECALL qui assurent le transfert de variables. Les utilisations standard de CSAVE-CLOAD pour la sauvegarde et lecture de programmes étant traitées au § 1.4, nous nous intéressons plus particulièrement à la sauvegarde de zones mémoire et de tableaux de variables.

CSAVE

Syntaxe: CSAVE"<Nom>",A<adresse début>,E<adresse fin> [,S][,AUTO]

Fonction: Sauvegarde toute la zone-mémoire comprise entre <adresse début> et <adresse fin>, ces deux dernières y compris, et sous le nom spécifié.

REMARQUES:

— Le nom peut être une constante ou une variable chaîne, de 16 caractères au maximum.

— L'option AUTO ne devrait être utilisée que si la zone sauvegardée contient une routine exécutable en langage-machine. S demande la sauvegarde lente (3ØØ baud, environ 4Ø octets/s.), l'omission impliquant une sauvegarde rapide (24ØØ baud, ou 3ØØ octets/seconde).

— <adresse début> et <adresse fin> peuvent être des constantes décimales ou hexadécimales, des variables ou des expressions numériques.

— Pendant la sauvegarde, le message "Saving <Nom du fichier>" s'affiche en première ligne de l'écran, suivi de la lettre "B" ou "C" (code) selon qu'il s'agit d'un programme Basic ou d'une zone-mémoire.

CLOAD

Syntaxes
1: CLOAD ["<Nom>"] [,S]
2: CLOAD ["<Nom>"],J [,S]
3: CLOAD ["<Nom>"],V <,S]

Fonctions: Selon la syntaxe employée, effectue les fonctions suivantes:

1) Charge le programme Basic ou la zone mémoire portant le nom spécifié, ou la première rencontrée si celui-ci est omis. S'il s'agit d'un programme Basic, le programme résident est effacé et remplacé par le nouveau. Le programme résident n'est pas affecté si on charge une zone-mémoire ne le recouvrant pas.

2) L'option J ("join") permet de charger un programme Basic, à la suite du programme résident et sans affecter ce dernier. Le programme résultant ne tournera que si les numéros de lignes du programme appelé sont différents, et plus élevés que ceux du précédent. Sinon, il est parfaitement possible de se retrouver avec plusieurs lignes ayant le même numéro.

3) L'option V ("verify") permet de contrôler le bon déroulement d'une sauvegarde, en comparant le fichier lu sur cassette au programme résidant en mémoire. Celui-ci n'est pas affecté par l'opération, de telle sorte

qu'il est parfaitement possible de re-sauvegarder si des erreurs sont détectées.

REMARQUES:

— L'instruction utilisée sans paramètres (CLOAD"") charge en mémoire le premier fichier rencontré sur la cassette, quel que soit son type.

— CLOAD fait très bien la différence entre un programme Basic et une zone-mémoire, cette dernière étant rechargée à l'emplacement correct (défini lors de sa sauvegarde). Le paramètre <Nom> doit être écrit de la même manière que lors de la sauvegarde (espaces compris) pour être retrouvé par l'instruction. Notez qu'il peut aussi bien être une constante qu'une variable-chaîne.

— Dès que CLOAD est exécuté, le message "Searching..." s'affiche en première ligne de l'écran. Si le premier fichier rencontré n'est pas celui demandé, le message "Found <Nom du fichier trouvé>" est affiché, suivi d'une lettre qui en definit le type: B pour un programme Basic, C (code) pour une zone-mémoire. Ce message est remplacé par "Loading <Nom du fichier> <type>" lorsque le fichier correct a été trouvé.

  Cette caractéristique intéressante signifie que vous pouvez facilement repérer les fichiers enregistrés sur une cassette, en spécifiant un nom inexistant après CLOAD: Ainsi, toute la cassette sera parcourue et les noms des fichiers successifs rencontrés seront affichés après "Found".

— Contrairement à ce qui était le cas sur l'Oric-1, on ne retourne pas en mode direct après un CLOAD si celui-ci est effectué par programme. Il est donc possible de charger des jeux de caractères, des écrans ou même des routines en langage-machine à partir d'un programme Basic.

— L'option "V" (verify) affiche le message "Verifying <Nom du fichier> <type>" en première ligne, et sitôt la comparaison terminée le message "XX Verify errors" est envoyé. XX est le nombre d'erreurs détectées, et il est prudent de recommencer la sauvegarde si ce nombre dépasse 5.

— Lors de la sauvegarde par CSAVE, l'Atmos envoie une fréquence pendant quelques secondes au magnétocassette, avant les données proprement dites, afin de stabiliser l'enregistrement automatique. Toutefois, sur certains magnétophones, cette fréquence est mal relue par CLOAD et une erreur est provoquée. Le programme est correctement chargé, mais ne démarre pas automatiquement même si l'option

AUTO a été utilisée lors de la sauvegarde. L'ordinateur reste en mode direct et le message "Errors found" est affiché à l'écran. Une routine en langage-machine évitant ce problème est fournie à la page 222 du manuel Atmos.

Exemples: Il est parfaitement possible de sauvegarder et de recharger des zones en mode direct:

Sauvegarde de l'écran-texte

```
CSAVE"<Nom>",A#BB80,E#BFE0  et relecture par
CLOAD"<Nom>
```

Sauvegarde d'un jeu de caractères:

```
CSAVE"<Nom>",A#B400,E#BB7F  relu par
CLOAD"<Nom>"
```

Voir aussi les diverses applications de ces méthodes dans les programmes donnés plus loin, CARGEN et MULTIGRAF.

La sauvegarde et le chargement de variables sur cassette.

La possibilité de stocker et recharger des données sur un support, dont la quantité totale dépasse la capacité de la mémoire centrale, est la base même de toute application informatique dépassant le stade des jeux ou de l'éducation.

Deux instructions ont été prévues sur l'Atmos, **STORE** et **RECALL** qui effectuent ces opérations sur cassette. Les données à transférer doivent être constituées en tableaux, les trois types existants étant acceptés: Virgule flottante, entiers ou chaînes de caractères.

STORE

Syntaxe: STORE A,"<Nom du fichier>" [,S]
ou STORE A%,"<Nom du fichier>" [,S]
ou STORE A$,"<Nom du fichier>" [,S]

Fonction: Sauvegarde sur cassette le tableau A(), A%(), ou A$(). Les principes d'utilisation sont les mêmes que pour CSAVE.

REMARQUES:

— Le nom du tableau peut être quelconque, suivant les règles d'appellation des variables. De même, il peut avoir n'importe quelle taille et un nombre quelconque de dimensions.

Attention: Le nom du fichier ne peut *pas* être une variable. Si c'est le cas, la syntaxe est acceptée et le tableau sauvegardé, mais celui-ci ne pourra plus être rechargé par la suite. Il faut donc impérativement employer une constante de chaîne entre guillemets, de 16 caractères au maximum. Cette limitation réduit quelque peu la souplesse de l'instruction, et du même coup celle de RECALL.

— Lors de la sauvegarde, le message "Saving.. <Nom du fichier> <Type>" apparaît à la première ligne. Le <type> est une lettre, "R" définissant un tableau de nombres réels en virgule flottante, "I" (pour "Integer") un tableau d'entiers, et "S" (pour "String") un tableau de chaînes.

RECALL

Syntaxe: RECALL A,"<Nom du fichier>" [S]
ou RECALL A%,"<Nom du fichier>" [S]
ou RECALL A$,"<Nom du fichier>" [S]

Fonction: Recharge, dans un tableau de type adéquat et préalablement dimensionné, les données sauvegardées à l'aide de STORE.

REMARQUES:

— Le tableau qui recevra les données doit bien sûr être du même type que celui sauvegardé par STORE, mais il doit aussi avoir le même nombre de dimensions, et une taille égale ou supérieure. Par contre, il n'a pas besoin d'avoir le même nom: On peut très bien sauvegarder A$() par STORE, et recharger les données dans B$().

— RECALL accepte une variable comme nom de fichier, contrairement à STORE.

— Ne pas oublier d'utiliser l'option "S" si la vitesse lente a été spécifiée pour STORE.

— Lorsque l'instruction est exécutée, la mention "Searching.." est affichée, puis "Loading.. <Nom du fichier> <Type>". Si le premier

fichier trouvé n'est pas celui spécifié, ou n'est pas du même type, la mention " Found.. <Nom du fichier trouvé> <Type>" apparaît. Si l'on a oublié le nom ou le type d'un fichier, il sera facile de taper RECALL avec un nom quelconque, puis de lire la cassette et de noter les nom et type des différents fichiers apparaîssant derrière " Found..".

Exemples:

1) Charger un tableau avec des nombres aléatoires, le sauvegarder puis le relire.

```
 10 ' ESSAI STORE/RECALL
 20 '
 30 DIM A%(15):CLS
 40 FOR N=1 TO 15
 50 A%(N)=RND(1)*20
 60 NEXT
 70 PRINT@4,10;"METTEZ LE K7 EN RECORD ET PRESSEZ"SPC(16)"UNE
 TOUCHE"
 80 STORE A%,"FICHIER"
 90 CLEAR ' On efface le tableau
100 CLS:PRINT@4,10;"REMBOBINEZ ET RENVOYEZ LA BANDE"
110 DIM A%(15)
120 RECALL A%,"FICHIER"
130 FOR N=1 TO 15
140 ? A%(N)
150 NEXT
```

Explications:

— On crée le tableau A%, puis on y place 15 nombres aléatoires (lignes 1Ø à 6Ø).

— Le tableau est sauvegardé à la ligne 8Ø, puis on efface toutes les variables (instruction CLEAR, ligne 9Ø).

— On redimensionne le tableau à la même taille qu'auparavant (ligne 11Ø).

— Enfin, on affiche tous les nombres aléatoires que contiennent les 15 éléments du tableau.

2) Un petit fichier d'adresses

Ce programme procure 4 fonctions: Création d'une fiche, Visualisation d'une fiche, Sauvegarde et chargement d'un fichier.

C'est évidemment une gestion de fichier assez élémentaire (en particulier, les fonctions modification/destruction de fiche et visualisation de tout le fichier en sont absentes), mais le programme est parfaitement utilisable tel quel et il est facile de le modifier et de l'améliorer.

```
  10 ' FICHIER D'ADRESSES
  15 '
  20 E$=CHR$(27):L=1:DIM F$(600)
 100 GOTO 8000
 110 '
1000 ' CREATION
1005 '
1010 CLS:PRINT:PRINT"CREATION FICHE No"INT(L/4+1):?:?
1020 INPUT"NOM ";F$(L):L=L+1:PRINT
1030 INPUT"Prenom ";F$(L):L=L+1:PRINT
1040 INPUT"Adresse ";F$(L):L=L+1:PRINT
1050 INPUT"Teleph. ";F$(L):L=L+1:PRINT:PRINT
1060 INPUT"VALIDATION (O/N)";R$:IF R$="N" THEN L=L-4:GOTO 1000
ELSE RETURN
1990 '
2000 ' VISUALISATION
2005 '
2010 IF L<2 THEN RETURN
2015 CLS:PRINT:PRINT"VISUALISATION"@2,6;
2020 PRINT@2,6;:INPUT"NOM RECHERCHE";N$
2030 N=1
2040 IF LEFT$(F$(N),LEN(N$))=N$ THEN 2100
2050 N=N+4:IF N>L THEN PRINT@2,15;"FICHE NON TROUVEE":GOTO 2200
ELSE 2040
2100 CLS:PRINT:PRINT"FICHE No"INT(N/4+1);@2,6;
2110 PRINT"NOM:":PRINT F$(N)" "F$(N+1):PRINT
2120 PRINT"ADRESSE:":PRINT F$(N+2):PRINT
2130 PRINT"TELEPHONE:":PRINT F$(N+3)
2200 PRINT@2,20;:INPUT"UNE AUTRE FICHE (O/N) ";R$:IF R$="O" THEN
GOTO 2000 ELSE RETURN
2990 '
3000 ' SAUVEGARDE
3005 '
3010 IF L<2 THEN RETURN
3020 CLS:PRINT@4,10;"METTEZ LE K7 EN RECORD ET PRESSEZ"SPC(16)
"UNE TOUCHE"
3030 GET X$
3040 F$(0)=STR$(L)
3050 STORE F$,"ADRESSES"
3060 RETURN
3090 '
3100 ' LECTURE
3105 '
3110 CLS:PRINT@6,10;"METTEZ LE K7 EN LECTURE"
3120 RECALL F$,"ADRESSES"
3130 L=VAL(F$(0))
```

```
3140 RETURN
7990 '
8000 ' MENU
8005 '
8010 CLS:PRINT CHR$(4)@14,5;E$"JMENU"CHR$(4)@6,10;
8020 PRINT"1  CREER UNE FICHE"
8030 PRINT TAB(6)"2  VISUALISER UNE FICHE"
8040 PRINT TAB(6)"3  SAUVEGARDER LE FICHIER"
8050 PRINT TAB(6)"4  CHARGER UN FICHIER"
8060 PRINT:PRINT:PRINT:INPUT"VOTRE OPTION ";R
8070 ON R GOSUB 1000,2000,3000,3100
8080 GOTO 8000
```

Comme on peut le voir clairement d'après le listing, quatre modules remplissant les fonctions principales ont été utilisés, plus le module "Menu" qui les appelle selon les besoins.

La structure du fichier est assez simple: Chaque fiche occupe 4 éléments du tableau F$(), le premier est le nom, le deuxième le prénom, le troisième l'adresse et le dernier le téléphone. Pour accéder à une fiche, il suffit de démarrer à 1 puis d'ajouter 4 à chaque fois pour trouver le nom suivant.

Voici les variables utilisées:

F$(6ØØ)	Tableau contenant les fiches.
R$	Réponse de l'utilisateur (Oui/Non).
N$	Nom de la fiche recherchée.
L	Nombre d'éléments de F$ utilisés. Le nombre de fiches s'obtient par L/4+1.
R	Réponse de l'utilisateur dans le menu.
N	Compteur de boucles.

Explications:

1ØØØ-1Ø6Ø: Module création. Chaque rubrique est saisie dans un élément de F$, L étant incrémenté de 1 à chaque fois.

1Ø6Ø : L'utilisateur a la possibilité de saisir la fiche à nouveau s'il a commis une erreur (attention, INPUT n'accepte pas les virgules dans les chaînes de caractères !)

2ØØØ-22ØØ: Module de visualisation. La recherche d'une fiche se fait sur le nom: La variable N démarre à 1, puis compare F$(N) avec N$. S'ils sont différents, on ajoute 4 à N et on recommence.

2Ø5Ø : Si N dépasse L, c'est que la fiche n'existe pas: Un message approprié est affiché.

21ØØ-213Ø: Les 4 éléments de F$ composant la fiche sont affichés.

22ØØ : Que la fiche ait été trouvée ou non, on demande si l'utilisateur veut continuer ou retourner au menu.

3ØØØ-314Ø: Modules Sauvegarde et Relecture du tableau F$. L'élément F$(Ø), non utilisé pour le fichier, est chargé avec la valeur de L avant la sauvegarde (ligne 3Ø4Ø), puis reconverti en variable numérique après relecture (ligne 313Ø): Ainsi, la taille du fichier est sauvegardée avec lui.

8ØØØ-8Ø8Ø: Menu principal. Remarquez la ligne 8Ø7Ø, qui effectue le branchement vers les différents modules en fonction de la valeur de R.

3

L'écran et les jeux de caractères

Cette section est consacrée à l'écran en modes TEXT et LORES, ainsi qu'à tous les codes, commandes et fonctions correspondants; l'écran haute-résolution graphique est traité en quatrième partie.

3.1. La représentation de l'écran en mémoire

L'écran, tel que nous le voyons sur un téléviseur, est constitué comme un rectangle divisé en 28 lignes de 4Ø colonnes, comprenant un total de 28*4Ø=112Ø cases. Un caractère ne pourra s'imprimer que dans une de ces "positions d'affichage", qui sera définie par son numéro de ligne et colonne.

Or, il faut savoir que l'écran est entièrement représenté en mémoire, chaque case de celui-ci occupant un octet. Si on place par POKE le code

ASCII d'un caractère dans une des adresses de cette zone-mémoire, le caractère s'affichera à la position correspondante sur l'écran. Inversement, on pourra lire le contenu d'une position d'affichage par PEEK. Rappelons en effet que chaque caractère du jeu possède un code compris entre 32 et 126, que l'on nomme "code ASCII" et dont la liste complète est donnée à l'Appendice 1.

La mémoire d'écran va de l'adresse #BB8Ø (48ØØØ) qui représente la première case en haut à gauche, à l'adresse #BFEØ (4912Ø), dernière case en bas à droite. Nous avons bien 4912Ø–48ØØØ=112Ø octets.

Les adresses sont associées aux lignes/colonnes selon le schéma suivant:

Colonne	0	1 + 2	3		37	38	39
Ligne 0	48000	48001+48002	48003		48037	48038	48039
1	48040	48041+48042	48043		48077	48078	48079
2	48080	48081+48082	48083		48117	48118	48119
25	49000	49001+49002	49003		49037	49038	49039
26	49040	49041+49042	49043		49077	49078	49079
27	49080	49081+49082	49083		49117	49118	49119

Nous voyons que toutes les adresses se suivent de haut en bas et de gauche à droite. Deux formules simples permettent de convertir des coordonnées en adresse et vice-versa (A=Adresse, L=Ligne et C=Colonne), considérant que la première ligne et colonne ont le N° Ø:

```
A=48000+C+L*40
L=INT((A-48000)/40)
C=A-INT(A/40)*40
```

3.1.1. Ligne et colonnes réservées

Vous aves certainement remarqué que sur le schéma, la première ligne (N° Ø) et les deux colonnes de gauche sont séparées des autres par des "+": Ces positions sont en effet "interdites" à l'affichage en temps normal, la première ligne étant utilisée pour les messages de l'ordinateur (notamment au cours des transferts de fichiers vers le K7. On peut toutefois afficher des caractères dans cette ligne en les plaçant un à un avec POKE, mais un moyen beaucoup plus élégant (et rapide !) de le faire est donné au § 3.5.2.

Quant aux deux colonnes de gauche, elles servent à stocker deux codes: La première est dévolue au code déterminant la couleur du fond pour toute la ligne, alors que celui placé dans la deuxième détermine la couleur des caractères affichés, toujours pour toute la ligne. Vous trouverez la liste complète de ces codes au § 3.4.

3.2. Ecrire avec PRINT. Les codes de contrôle

La "position d'affichage", dont nous avons parlé précédemment, est constamment matérialisée à l'écran par un carré noir clignotant: C'est le **curseur**, qui nous indique à quel endroit s'imprimera le prochain caractère. L'instruction PRINT est un des moyens d'effectuer cette impression.

Cette instruction, qui peut être remplacée par ? (point d'interrogation) nous permet bien sûr d'écrire toutes sortes de caractères "visibles" à l'écran, soit en les mentionnant directement, par exemple PRINT"Bonjour", soit un par un en les appelant par leur code, à l'aide de la fonction CHR$:

PRINT CHR$(65) imprimera un "A" (code 65) à la position du curseur. Évidemment, CHR$ n'est que de peu d'utilité pour écrire des caractères "visibles", comme les lettres ou nombres. Mais il en est autrement lorsqu'il s'agit d'imprimer des **codes de contrôle**.

3.2.1. Les codes de contrôle

Nous avons vu précédemment que les codes du jeu de caractères étaient compris entre 32 et 126. Mais les valeurs de Ø à 31 sont également utilisées, simplement elles ne génèrent pas un caractère visible mais une **fonction**, qui concerne en général l'écran ou le clavier. Ces fonctions sont appelées **codes de contrôle** car on les obtient en mode direct en actionnant simultanément la touche CTRL (CONTROL) et une lettre de l'alphabet.

La correspondance entre le mode direct et le mode programme (code à utiliser avec PRINT CHR$) est simple :

```
Code CTRL=Code ASCII-64
```

Donc, CTRL A = CHR$(1), CTRL B = CHR$(2), etc...

Vous trouverez ci-dessous la liste complète des codes CTRL, en mode direct et mode programme. Souvent, les codes les plus importants sont générés par une touche spécialisée du clavier ; dans ce cas, cette touche est rappelée en-dessous du code correspondant. D'autre part, il est normal que certains caractères soient omis car tous ne sont pas nécessairement utilisés.

Modes		*Fonction*
Prog	*Direct (CTRL)*	
CHR$(1)	CTRL A	Incrémente d'un caractère le buffer de ligne (voir éditeur, § 3.2.2).
CHR$(3)	CTRL C	Interrompt l'exécution du programme (BREAK).
CHR$(4)	CTRL D	Démarre/arrête le dédoublement de l'impression (cf. § 3.3).
CHR$(6)	CTRL F	Démarre/arrête la répétition sonore du clavier (bascule).
CHR$(7)	CTRL G	Sonnerie clavier.
CHR$(8)	CTRL H ou ←	Déplace le curseur d'une colonne vers la gauche.

Modes *Prog*	*Direct (CTRL)*	*Fonction*
CHR$(9)	CTRL I ou →	Déplace le curseur d'une colonne vers la droite.
CHR$(1Ø)	CTRL J ou ↓	Déplace le curseur à la ligne suivante, même colonne.
CHR$(11)	CTRL K ou ↑	Déplace le curseur à la ligne précédente, même colonne.
CHR$(12)	CTRL L	Effacement de l'écran (également CLS en mode programme).
CHR$(13)	CTRL M ou RETURN	Retour chariot (valide de la ligne en cours).
CHR$(14)	CTRL N	Efface de l'écran la ligne où se trouve le curseur.
CHR$(15)	CTRL O	Bloque/débloque l'affichage, mais pas les entrées au clavier (bascule).
CHR$(17)	CTRL Q	Rend le curseur visible/invisible (bascule).
CHR$(19)	CTRL S	Bloque/débloque l'affichage *et* le clavier (bascule).
CHR$(2Ø)	CTRL T	Majuscules/Minuscules (bascule).
CHR$(24)	CTRL X	Annule la ligne en cours de saisie (voir éditeur, § 3.2.2).
CHR$(27)	CTRL [ou ESC	Caractère Escape (cf. § 3.3).
CHR$(29)	CTRL]	Déblocage/blocage des deux colonnes de gauche (résultat visible : affichage en vidéo inverse).
CHR$(3Ø)		Place le curseur à la première ligne, première colonne de l'écran (fonction HOME — accessible par programme uniquement).
CHR$(127)	DEL	Efface de l'écran et du buffer de ligne le dernier caractère entré, et déplace le curseur d'une colonne vers la gauche.

Déplacements du curseur à l'aide de PRINT

La commande PRINT à elle seule autorise toutes les opérations de positionnement du curseur, des plus simples aux plus complexes. Les plus simples sont les déplacements case par case, à l'aide des codes de 8 à 11 correspondant aux quatre flèches de l'écran, et les plus complexes s'obtiennent à l'aide de PRINT @ :

Exemples :

1) Déplacements par les codes de contrôle.

```
FOR N=1 TO 5:?CHR$(11);:NEXT
```

Place le curseur 5 lignes plus haut;

```
FOR N=1 TO 4:?CHR$(10)CHR$(8);:NEXT
```

Déplace le curseur de 4 colonnes à gauche et 4 lignes plus bas.

2) Déplacements par PRINT @

```
10 CLS:L=1:A$="ATMOS"
20 FOR C=29 TO 5 STEP-1
30 PRINT@34-C,L;A$;@C,L;A$
30 L=L+1
40 NEXT:GET X$
```

3.2.2. L'éditeur de ligne

L'ensemble des commandes permettant d'écrire, puis de corriger un programme sont regroupées sous le nom d'"éditeur". On ne peut pas dire que l'Atmos nous gâte de ce côté-là, puisque seules deux commandes et une instruction sont prévues. Mais à elles seules, elles autorisent toutes les opérations nécessaires.

L'éditeur du Basic est appelé un éditeur de ligne, car il ne prend en compte qu'une ligne à la fois par opposition à un éditeur d'écran capable de prendre en compte des groupes entiers de lignes ou même tout un programme. Par contre, il gère l'affichage beaucoup plus rapidement qu'un éditeur d'écran. Son principe est le suivant:

— Chaque caractère frappé au clavier est affiché à l'écran, et en même temps placé dans une mémoire-tampon appelée "**buffer**".

— Les **quatre flèches** de déplacement, si elles ont un effet sur l'écran, n'en ont aucun sur le buffer.

— La touche **DEL** élimine du buffer le dernier caractère entré. Elle aura le même effet sur l'écran *uniquement si le curseur n'a pas été déplacé entre-temps avec les flèches!* Prudence, donc.

— La touche **RETURN** vide le buffer, en "enregistrant" la ligne qui s'y trouve : Exécution en mode direct si la ligne est correcte sinon émission d'un message d'erreur, ou validation de celle-ci en tant que ligne de programme *si le premier caractère du buffer est un nombre.*

Si vous avez suivi jusqu'ici, vous n'aurez pas de problèmes pour la suite !

Modification de lignes existantes

Deux codes CTRL sont disponibles à cet effet :

CTRL A : Le fait de passer sur un caractère affiché à l'écran avec CTRL A recopie celui-ci dans le buffer ; on peut ainsi recopier des lignes entières en passant dessus, sans avoir à les retaper. Évidemment, on peut recopier une ligne jusqu'à un certain point, taper ensuite au clavier des caractères à modifier, puis continuer à recopier cette ligne ou une autre, à partir de n'importe quel endroit. N'oublions pas que les flèches, si elles déplacent le curseur, n'ont *aucun effet* sur le buffer.

CTRL X : Affiche un " \ " (backslash) à la position du curseur, et vide le buffer. Le contenu de celui-ci n'est alors pas pris en compte, contrairement à ce qui se passe avec RETURN.

Exemples :

Pour recopier une ligne de programme en changeant son numéro :

— Taper EDIT<Numéro de la ligne>.

— Le curseur est placé juste avant le numéro. Retaper celui-ci, et s'il est plus long que l'ancien monter d'une ligne avec ↑, puis taper les chiffres restants.

— Placer le curseur avec les flèches sur le premier caractère de la ligne.

— Copier toute la ligne, en faisant passer le curseur dessus avec CTRL A.

— Valider avec RETURN.

Pour insérer dans une ligne de programme:

— Lister la ligne avec EDIT<Numéro de ligne>

— Faire passer le curseur sur la ligne avec CTRL A, jusqu'à amener le curseur sur le caractère *suivant le point d'insertion.*

— Faire monter le curseur d'une ligne avec ↑, puis taper le texte à insérer.

— Revenir sur le caractère suivant le point d'insertion avec les flèches.

— Copier la ligne jusqu'à la fin avec CTRL A, puis valider par RETURN.

Ces deux exemples devraient vous permettre de faire toutes sortes de modifications dans vos programmes, et même dans des lignes en mode direct. Rappelez-vous que:

— Une ligne est éliminée d'un programme simplement en tapant son numéro.

— Il faut faire attention lorsque vous tapez des nombres à l'écran: Ils seront considérés comme des numéros de ligne s'ils sont suivis de RETURN, et en effaceront éventuellement de votre programme.

— Tous les codes CTRL sont ignorés par le buffer, excepté bien sûr CTRL A et CTRL X: Vous pouvez non seulement vous déplacer sans risques, mais aussi passer en minuscules ou effacer de l'écran une ligne qui vous gène.

— Lorsque l'écran est vide, il est rempli d'espaces. Quand vous tapez CTRL A sur une ligne vide, vous ne faites pas "rien", vous mettez des espaces (code ASCII 32) dans le buffer!

3.3. Utilisation de "ESCAPE"

Déplacez le curseur de deux ou trois colonnes vers la droite, puis tapez la touche ESC; ensuite, tapez la lettre "L", puis un peu de texte: Vous voyez l'effet à l'écran d'un code ESCAPE, en l'occurrence ESC L. Ces codes fonctionnent de la manière suivante:

Le caractère "Escape" (code ASCII 27) indique à l'Atmos que le caractère suivant ne doit pas être affiché, mais considéré comme un **attribut**; cet attribut sera valide sur le restant de la ligne où il se trouve, mais pas sur les suivantes. Il existe toutes sortes d'attributs, déterminant la couleur du fond et des caractères, leur clignotement, le passage en double taille ou le changement de jeu. Vous trouverez la liste complète de ces attributs au § 3.4.

Nous avons vu qu'il est extrêmement simple d'utiliser Escape en mode direct; ce n'est pas plus compliqué par programme, il suffit de taper:

```
        PRINT CHR$(27)"<caractère>"
ou bien PRINT CHR$(27)CHR$(<code caractère>)
```

Exemples:

Tapez la séquence de caractères suivante ("ESC" est bien sûr la touche ESC):

```
ESC Q ESC C JAUNE/ROUGE ESC L FLASH
```

Maintenant, tapez ce programme puis exécutez-le:

```
10 E$=CHR$(27)
20 ?E$"Q"E$"CJAUNE/ROUGE"E$"LFLASH"
```

— Vous constaterez que l'effet est identique à celui de la ligne précédente en mode direct.

— A la ligne 1Ø, on a mis le caractère Escape dans la variable E$, pour s'éviter de taper trois fois de suite CHR$(27).

— Les lettres "C" et "L", incluses dans les messages, ne sont pas considérées comme des caractères à imprimer mais comme des attributs, car elles suivent immédiatement le caractère Escape.

Utilisation de la double hauteur

Vous aurez certainement remarqué, en essayant les codes CTRL, que CTRL D provoquait un dédoublement de l'impression sur deux lignes. D'autre part, ESC J imprime un caractère bizarre qui en fait est la moitié (inférieure ou supérieure, selon la ligne où l'on se trouve) d'un caractère complet en double hauteur. C'est la réunion de ces deux fonctions qui permettra l'affichage de caractères en double taille. Mais attention: Les lignes impaires ne peuvent contenir que la moitié supérieure, et les lignes

paires que la moitié inférieure d un caractère (pour PRINT @ , la première ligne d'écran a le N° Ø).

Exemple :

```
10 E$=CHR$(27):CLS:?CHR$(4)
20 ?E$"JDOUBLE  "E$"NDOUBLE FLASH"
30 ?CHR$(4)
```

REMARQUE: CTRL D ou CHR$(4) agissant comme une bascule, il faut être sûr que la double taille n'est pas déjà en fonction quand on l'utilise. Il est cependant facile de forcer la double ou simple taille avec POKE 618 (cf. § 2.3.3); dans ce cas, les lignes 1Ø et 3Ø deviendraient:

```
10 E$=CHR$(27):CLS:?:POKE 618,67
```

et

```
30 POKE 618,3
```

3.3.1. La Hardcopy texte

Une "hardcopy" est une copie exacte sur imprimante de ce qui apparaît à l'écran. Le programme qui suit exécute cette fonction pour l'écran texte (Un programme analogue effectuant la hardcopy graphique est donné au § 4.4.), et il est numéroté à partir de 6ØØØØ afin d'être inclus comme sous-programme dans vos applications personnelles. Ne comprenant aucun code spécifique, cette routine fonctionne sur toutes les imprimantes.

```
60000 '       HARDCOPY TEXTE
60002 '
60010 AD=48000:AF=49080:LPRINT TAB(20)
60020 FOR N=AD TO AF STEP 40
60030 FOR A=0TO 39
60040 D=PEEK(A+N):IF D<32 OR D>126 THEN D=32
60050 LPRINT CHR$(D);
60060 NEXT A:LPRINT:LPRINT TAB(20);
60070 NEXT N:LPRINT
```

Explications :

6ØØ1Ø: Fixe les limites de la zone à copier (écran), et centre celle-ci sur une imprimante 8Ø col.

6ØØ4Ø: Lit le contenu de chaque adresse-écran, et remplace les attributs non imprimables par des espaces (ASCII 32).

6ØØ5Ø: Envoie le code du caractère lu à l'imprimante.

6ØØ6Ø: Saut de ligne tous les 4Ø caractères, et recentrage de la ligne suivante.

Notez que les commandes LPRINT TAB(X) ne présentent d'intérêt que dans le cas d'une imprimante 8Ø colonnes, pour le centrage du texte qui n'en fait que 4Ø. Elles devront bien sûr être supprimées si vous utilisez un modèle 4Ø colonnes.

3.4. *L'instruction PLOT. Les attributs d'écran*

Cette instruction, dont l'effet surprend un peu ceux qui sont habitués à d'autres machines, est néanmoins extrêmement puisante:

— Elle permet d'afficher à l'écran des variables ou constantes de chaînes, ainsi que des attributs ou des caractères isolés appelés par leur code (comme PRINT@).

— Elles ne permet *pas* d'imprimer des variables ou constantes numériques ou des codes CTRL, et ne *déplace pas le curseur* (contrairement à PRINT@).

— L'affichage de nombres reste néanmoins possible, à condition de les convertir en chaîne au préalable à l'aide de STR$: Si on veut imprimer en colonne 5, ligne 1Ø la variable N, nous devront écrire: PLOT 5,1Ø,STR$(N).

— On peut imprimer un caractère du jeu ASCII en vidéo inverse en écrivant:

```
PLOT<Col.>,<Ligne>,<Code Car.>+128.
```

(Le jeu ASCII complet avec ses codes est donné à l'Appendice 1).

Exemples:

Tapez au clavier: PLOT 1Ø,1Ø,22

Et le petit programme suivant:

```
10 CLS:?
20 ?"JAUNE/ROUGE"SPC(2)"FLASH"
30 PLOT 0,1,17:PLOT 1,1,3:PLOT 13,1,12
```

Comme on peut le voir dans cet exemple, PLOT sert non seulement à afficher des caractères mais aussi, comme ESC, à placer des attributs à l'écran. En fait, chaque caractère considéré comme attribut après ESC possède un équivalent numérique utilisable par PLOT.

Il ne faut *absolument pas confondre ces attributs avec les codes CTRL, qui ont pourtant les mêmes nombres* (Ø à 31). Ceux-ci s'emploient avec PRINT, et correspondent à un standard international (ASCII), alors que les attributs sont des codes internes à l'Atmos affichés par PLOT ou POKE.

Attributs d'écran

Pour chacun d'eux nous donnons la valeur à donner derrière PLOT ou POKE, la séquence ESCAPE correspondante et la fonction exacte:

Couleurs de l'affichage

Ø	ESC	Noir
1	ESC A	Rouge
2	ESC B	Vert
3	ESC C	Jaune
4	ESC D	Bleu
5	ESC E	Magenta
6	ESC F	Cyan
7	ESC G	Blanc

Taille et jeu de caractères

8	ESC H	Taille normale, jeu normal (ASCII)
9	ESC I	Taille normale, jeu graphique
1Ø	ESC J	Double taille, jeu normal
11	ESC K	Double taille, jeu graphique

Clignotement

12	ESC L	Clignotant, taille normale, jeu normal
13	ESC M	Clignotant, taille normale, jeu graphique
14	ESC N	Clignotant, Double taille, jeu normal
15	ESC O	Clignotant, Double taille, jeu graphique

Couleur du fond

16	ESC P	Noir
17	ESC Q	Rouge
18	ESC R	Vert
19	ESC S	Jaune
2Ø	ESC T	Bleu
21	ESC U	Magenta
22	ESC V	Cyan
23	ESC W	Blanc

— On peut constater que les codes sont organisés de façon logique, et que les couleurs du fond correspondant à celles de l'affichage plus 16.

— En ce qui concerne les caractères double taille, l'utilisation de PLOT est assez malaisée: comme ceux-ci occupent deux lignes à l'écran, il faudrait placer le code par PLOT sur les deux lignes concernées pour obtenir le résultat souhaité ! Escape et PRINT @ sont, dans ce cas, beaucoup plus pratiques d'emploi.

N'oublions pas enfin que PLOT peut aussi servir à écrire n'importe quelle chaîne de caractères n'importe où à l'écran (une à la fois cependant) et se révèle donc très utile pour imprimer des messages ou animer des objets rapidement sans déplacer le curseur.

3.4.1. Le programme "Tennis"

Le programme suivant montre comment on peut exploiter les particularités de PLOT dans un jeu en temps réel, en l'occurrence un jeu de tennis à un joueur rappelant les ancêtres (déjà !) de nos actuels jeux de café.

Peu d'explications sont nécessaires: On peut choisir la taille du terrain, la barre d'espacement envoie une balle, la raquette se déplace avec les flèches ↓ ↑ et la touche R permet de changer la taille du terrain en cours de partie.

```
10 '      TENNIS
12 '
50 R=1:X=1:Y=1:FX=1:FY=1:AX=8:S=0
100 GOSUB 9000
110 GOSUB 6000
120 GOSUB 8000
130 GOSUB 7000
140 POKE 618,10
150 PLOT4,1,"SCORE: 0    ":PLOTX,Y,64
998 '
1000 '      BOUCLE PRINCIPALE
1002 '
1100 X$=KEY$:IF X$<>"" THEN AX=ASC(X$)
1110 ON AX-7 GOTO 1150,1150,1130,1140
1120 GOTO 1150
1130 IF R<23 THEN PLOT2,R,96:R=R+1:PLOT2,R,126:PLOT2,R+1,126
1135 GOTO 1150
1140 IF R>3 THEN PLOT2,R+1,96:R=R-1:PLOT2,R,126:PLOT2,R+1,126
1150 IF Y<4 THEN FY=1:SHOOT:PLAY0,0,0,0
1160 IF Y>23 THEN FY=-1:SHOOT:PLAY0,0,0,0
1170 IF X>T THEN FX=-1:SHOOT:PLAY0,0,0,0
1200 IF X>3 THEN 1300
1210 IF X<3 THEN 1500
1220 IF Y=R THEN FX=1:FY=-1:SHOOT:PLAY0,0,0,0:S=S+1:IF Y=3 THEN FY=1
1230 IF Y=R+1 THEN FX=1:FY=1:SHOOT:PLAY0,0,0,0:S=S+1:IF Y=24 THEN FY=-
1240 PLOT10,1,STR$(S)
1300 PLOTX,Y,32:X=X+FX:Y=Y+FY:PLOTX,Y,64
1310 GOTO 1100
1498 '
1500 '      BALLE PERDUE
1502 '
1510 ZAP:S=0:AX=32:?@2,26;"<BARRE> ou <R>=CHANGEMT DE TERRAIN";
1520 GET X$:IF X$="R" THEN RUN
1530 IF X$<>CHR$(32) THEN 1520
1535 PRINT@2,26;CHR$(14);
1540 IF Y=R OR Y=R+1 THEN PLOT2,Y,126 ELSE PLOT2,Y,96
1550 GOTO 130
5998 '
6000 '      CHOIX DU TERRAIN
6002 '
6010 CLS:POKE 618,67
6020 PRINT@12,5;CHR$(27)"JTENNIS"
6030 POKE 618,3:?@2,11;:INPUT"TAILLE DU TERRAIN (0 A 14) ";T
6040 IF T<0 OR T>14 THEN 6000
6050 T=T+20:RETURN
6998 '
7000 '      SERVICE BALLE
7002 '
7010 IF RND(1)<.5 THEN FY=-1 ELSE FY=1
7020 X=T+1:FX=-1
7030 Y=INT(RND(1)*21)+3:RETURN
7998 '
```

```
8000 '      DESSIN TERRAIN
8002 '
8005 CLS
8010 FOR N=48122 TO 49042 STEP 40
8020 POKE N,96:POKE N+T+1,128:NEXT
8030 R=13:PLOT2,13,126:PLOT2,14,126
8040 AD=48122:Z=0
8050 FOR N=0 TO T STEP 2
8060 DOKE AD+N,32896:NEXT
8070 IF Z=0 THEN Z=1:AD=49042:GOTO 8050
8080 RETURN
8998 '
9000 '      INITIALISATIONS
9002 '
9010 READ AD:IF AD=0 THEN 9100
9020 FOR N=0 TO 7
9030 READ DA:POKE AD+N,DA
9040 NEXT
9050 GOTO 9010
9100 RETURN
9500 DATA 46592,12,30,63,63,63,30,12,0
9510 DATA 47088,31,31,31,31,31,31,31,31
9520 DATA 46848,0,4,4,4,4,4,4,0
9600 DATA 0
```

Explications :

5Ø : Attribue le plus tôt possible des variables importantes pour la vitesse d'exécution :
R=Ordonnée de la raquette
X=Abscisse de la balle
Y=Ordonnée de la balle
FX=Valeur à ajouter à l'abscisse de la balle (1 ou −1)
FY=Valeur à ajouter à l'ordonnée de la balle (1 ou −1)
S=Score

11Ø-13Ø : Appel des sous-programmes de définition des caractères spéciaux, saisie de la taille du terrain, dessin du terrain et service.

9ØØØ-96ØØ : Dessin de la balle (code 64), de la raquette (code 126) et du pointillé servant de fond à la raquette (96). Cf. § 3.7.3.

6ØØØ-6Ø5Ø : En-tête du jeu et saisie de la taille du terrain (T).

8ØØØ-88Ø : Tracé du terrain. Le caractère 128 (carré noir) est placé par POKE et PLOT aux emplacements corrects.

7ØØØ-7Ø4Ø: Service aléatoire. On détermine d'abord un sens aléatoire pour la balle (7Ø2Ø), puis une ordonnée aléatoire pour son apparition.

11ØØ : Saisie de la touche pressée.

113Ø-114Ø: Gestion des mouvements de la raquette.

115Ø-117Ø: Détermination du sens de rebond de la balle, en fonction de son angle d'impact.

12ØØ : Teste si la balle atteint l'abscisse de la raquette.

122Ø-123Ø: Balle rattrapée; gestion de son rebond sur la raquette.

124Ø : Affichage du nouveau score.

13ØØ : Efface la balle à ses anciennes coordonnées, et l'affiche aux nouvelles.

131Ø : Retour dans la boucle.

3.5. Extensions au Basic. Le langage-machine

Dans les chapitres précédents, nous avons passé en revue toutes les commandes et fonctions offertes par le Basic pour la gestion de l'affichage, en rapport avec PRINT et PLOT. On peut cependant aller nettement plus loin dans les possibilités et le confort d'utilisation, en exploitant notamment les variables-système liées à l'écran.

3.5.1. Ecrire et lire à l'écran avec POKE et PEEK

Nous avons vu au § 3.1 que le stockage de tout le contenu de l'écran en mémoire permettait d'y afficher n'importe quel caractère avec POKE, en tapant POKE<Adresse>,<Code Car.> (les codes ASCII de tous les caractères se trouvent à l'appendice 1).

En général, il est beaucoup plus pratique d'utiliser PLOT ou PRINT @ qui ont la même fonction en étant plus "parlants", dans la mesure où la position d'affichage y est donnée non pas par une adresse mais par des numéros de ligne et colonne.

L'usage de POKE se révèle cependant indispensable dans le cas où l'on veut accéder à la première ligne de l'écran, interdite à PRINT et PLOT. Il suffira d'écrire:

```
10 M$="<Message à imprimer>"
20 FOR N=1 TO LEN(M$)
30 POKE 48001+N+<Col.début>,ASC(MID$(M$,N,1))
40 NEXT N
```

Toute chaîne de caractères attribuée à M$ à la ligne 1Ø sera imprimée sur la première ligne de l'écran, à partir de <Col. début>. Une autre méthode, encore plus rapide, est donnée au § 3.5.2.

En ce qui concerne les codes fournis comme arguments à POKE, tous ceux compris entre 32 et 128 sont considérés comme des caractères du jeu ASCII, et ceux compris entre Ø et 31 comme des *attributs*: les règles sont identiques à celles en vigueur pour PLOT, et exposées plus haut. De même, un caractère sera imprimé en vidéo inverse si on POKE son code +128.

Enfin, en lisant un code dans une adresse de l'écran, PEEK établit la même correspondance avec la fonction SCRN que POKE avec PLOT. Rappelons que SCRN ne permet pas plus de lire le contenu de la première ligne que PLOT ne permet d'y écrire, PEEK sera donc indispensable dans ce cas.

3.5.2. L'utilisation des variables-système

Modification de la taille de l'écran

Quatre variables-système déterminent entièrement les limites haute et basse de la zone d'évolution du curseur:

— *L'adresse de début de la première ligne de l'écran* est contenue dans #27A,#27B (634,635). Normalement, cette valeur est de 48Ø4Ø, laissant la première ligne (qui commence à 48ØØØ) hors d'atteinte du curseur. Pour récupérer celle-ci, il suffira donc de taper DOKE 634,48ØØØ. Effacez l'écran, et voyez où se place le curseur!

Maintenant faites un RESET, écrivez un peu de texte à l'écran puis entrez la ligne suivante en mode direct:

```
DOKE634,48000:?CHR$(30)"MESSAGE INAMOVIBLE!":DOKE634,48040
```

Vous constatez que votre message, qui ne peut plus être effacé, s'est affiché considérablement plus vite qu'avec POKE, et en utilisant tout bêtement PRINT. Rappelons que CHR$(3Ø) place le curseur au coin supérieur gauche de l'écran, sans effacer celui-ci.

Vous pouvez en fait interdire au curseur un nombre quelconque de lignes, avec la formule:

```
DOKE 634,<Nombre de lignes interdites>*40+48000
```

PRINT @, PLOT et POKE permettront d'aller encore dans ces lignes, mais celles-ci ne seront pas du tout affectées par CLS, CTRL L, INK ou PAPER. Évidemment, rien n'est plus simple que de redimensionner l'écran à sa taille maximale, l'effacer puis redéfinir ses limites tout de suite après: Le tout est extrêmemnt rapide.

— *La dernière ligne de l'écran* est déterminée par #27E (638). Cette valeur, qui est normalement de 27, est elle aussi modifiable par l'utilisateur selon la formule:

```
POKE 638,<N° de la ligne inférieure>
```

```
PRINT@2,24;"MESSAGE INAMOVIBLE"@2,10;:POKE 638,20
```

En effaçant l'écran, vous constatez que le message n'est pas affecté. Mais pour être pleinement effective, notamment en cas de scrolling, cette variable doit être utilisée en conjonction avec la suivante:

— #27C,#27D (636,237) définit *combien de lignes de l'écran seront affectées par le scrolling.* (Défilement de l'affichage vers le haut quand le curseur atteint la dernière ligne "accessible".

Ainsi, si l'on protège une fenêtre en bas d'écran avec #27E, il faut aussi protéger le contenu de cette fenêtre contre le scrolling, par la formule:

```
DOKE 636,<nombres de lignes devant scroller>*40
```

Si l'on veut par exemple protéger 5 lignes en bas d'écran, on écrira:

```
POKE 638,21:DOKE 636,800
```

Plus qu'un long discours, le petit programme suivant vous donnera un aperçu des possibilités offertes:

```
10 CLS:PLOT3,20,"VOUS NE POUVEZ PAS ECRIRE ICI"
20 POKE 638,15:DOKE 636,560
30 FOR N=1 TO 20
40 PRINT:PRINT SPC(10)"TEXTE.....":WAIT15
50 PRINT SPC(10)"A IMPRIMER":WAIT15:NEXT
60 CLS:PLOT6,22,"NI EFFACER D'AILLEURS!"
```

Pour terminer, il faut savoir que:

— PRINT@, PLOT et POKE peuvent toujours accéder aux lignes hors limites en bas de l'écran.

— Si vous changez à la fois les limites haute et basse, commencez par le bas, ou alors comptez le nombre de lignes données comme argument à POKE 638 à partir de la *nouvelle* limite supérieure.

— Rien ne vous oblige à donner une adresse multiple de 40 (48000, 48040, 48120, etc) comme argument à DOKE 634. Mais dans ce cas, considérez-vous comme seul responsable de ce que vous verrez à l'écran! Même remarque pour des nombres supérieurs à 27 fournis à POKE 638.

Le contrôle de la position du curseur

Les deux variables #268 (616) et #269 (617) contiennent respectivement la ligne et la colonne où se trouve le curseur. Elles sont très utiles pour "suivre" les déplacements du curseur, si l'on veut par exemple reculer d'un certain nombre de colonnes sur la ligne où l'on se trouve:

```
PEEK (616) renvoie la ligne courante+1.
PEEK (617) renvoie la colonne où se trouve le curseur.
```

La position exacte du curseur sera donc renvoyée par:

```
PRINT PEEK(617)PEEK(616)-1
```

Une autre variable gère également la ligne où se trouve le curseur: Il s'agit de #12,#13 (18,19), contenant l'adresse de début de Numéro de ligne où se trouve le curseur.

3.6. *Modes TEXTE et LORES*

3.6.1. Le mode LORES

Celui-ci ne présente que peu de différences avec le mode Texte normal, les deux options proposées sont:

LORES Ø, qui est identique à TEXT avec un affichage en vidéo inverse, et utilise le jeu de caractères normal.

LORES 1, comme ci-dessus mais utilisant le deuxième jeu de caractères (mosaïques graphiques) sur tout l'écran.

REMARQUE:

— Ces deux fonctions sont obtenues en plaçant le code du jeu désiré (8 ou 9) dans la première colonne réservée, puis de remplir la deuxième colonne avec le code "fond noir". Le fait de fournir un code de couleur de fond à la deuxième colonne (normalement réservée à la couleur des caractères) provoque l'affichage en blanc sur la couleur choisie.

— En conséquence de cela, il est impossible de changer les couleurs de l'écran LORES par PAPER ou INK, ceux-ci agissant précisément sur des deux premières colonnes.

— Les couleurs valides en mode texte sont mémorisées dans **#26B** (619) pour le fond et **#26C** (62Ø) pour l'affichage. Si vous effacez l'écran en mode LORES, ces valeurs seront restituées ainsi que le jeu de caractères standard, vous faisant retourner au mode normal. Heureusement, il existe un moyen très simple de résoudre ces problèmes:

3.6.2. Choix d'attributs pour tout l'écran

Les deux variables que nous avons mentionnées (619,62Ø) offrent en quelque sorte un LORES "à la carte": On peut choisir n'importe quelle couleur de fond ou de premier plan, ainsi que n'importe quel autre attribut d'affichage instantanément valide sur *tout l'écran*: Caractères double taille,

clignotants, changement de jeu, etc... (La liste complète des attributs est donnée au § 3.4). Il suffit de taper:

```
POKE619,<Attribut1>:POKE620,<Attribut2>
```

Et de respecter les quelques consignes suivantes:

— L'astuce consiste à ne *pas* placer l'attribut correct dans la colonne adéquate (couleur fond dans 619 et premier plan dans 62Ø). Si c'était le cas, PAPER et INK sont bien sûr plus intéressants.

— Les nouvelles valeurs entrent en vigueur après effacement de l'écran. Le fait d'effacer l'écran ensuite respectera ces valeurs, contrairement à ce qui se passe en LORES.

— Étant donné que l'attribution aux colonnes réservées d'un code qui ne leur est pas destiné rend le fond noir et les caractères blancs, l'affichage se fera toujours en clair sur sombre.

Exemples:

```
Simulation de LORES 1:
POKE619,9:POKE620,16:CLS

Blanc sur rouge, caractères géants:
POKE619,10:POKE620,17:CLS  puis  CTRL D

Rouge sur noir, clignotant:
POKE619,1:POKE620,12:CLS
```

3.7. Les jeux de caractères

3.7.1. Généralités

Deux jeux de caractères sont disponibles sur l'Atmos:

— Le jeu standard comprenant les caractères alphanumériques et les ponctuations, qui correspond à la norme Américaine ASCII. Les codes de ces caractères vont de 32 ("Espace") à 126 (zone grisée). La liste du jeu ASCII avec tous ses codes se trouve à l'Appendice 1.

— Le deuxième jeu, composé de mosaïques graphiques, comprenant 64 caractères dont les codes vont de 32 à 95. Ces caractères s'obtiennent par l'écriture d'attributs appropriés à l'écran (voir plus haut).

3.7.2. Principes de codage

Un caractère est dessiné dans une matrice de 8×8, chacun des points de la matrice pouvant être "allumé" (noir) ou "éteint" (blanc). Le dessin de tous les caractères des deux jeux est stocké en mémoire, un point étant représenté par un Bit: Il faut donc un octet pour représenter 8 points ou une ligne horizontale de la matrice, et 8 octets pour stocker un caractère complet en mémoire. Ces 8 octets sont placés dans huit adresses consécutives, l'ensemble de ceux-ci constituant deux zones: Entre les adresses #B4ØØ (46Ø8Ø) et #B7FF (471Ø3) pour le jeu standard et #B8ØØ (471Ø4)-#BB57 (47959) pour le deuxième jeu.

Étant donné que ces zones sont en mémoire vive, il en découle que n'importe quel caractère peut être entièrement redessiné par l'utilisateur en fonction de ses besoins, en modifiant la valeur des adresses le définissant: Minuscules accentuées, invaders, etc.

La première des 8 adresses codant un caractère donné est trouvée par la formule:

<Adresse>=46Ø8Ø+8*<code ASCII du car.> pour le jeu normal, et

<Adresse>=471Ø4+8*<code ASCII du car.> pour le jeu graphique.

L'exemple suivant montre comment un caractère est stocké dans ces adresses:

Matrice	*Adresses*	*Valeur décimale*	*Valeur binaire*
Bits 32 16 8 4 2 1			
..*...	466ØØ	8	ØØ ØØ1ØØØ
.*.*..	466Ø1	2Ø	ØØ Ø1Ø1ØØ
....	466Ø2	34	ØØ 1ØØØ1Ø
....	466Ø3	34	ØØ 1ØØØ1Ø
*.***.	466Ø4	62	ØØ 11111Ø
....	466Ø5	34	ØØ 1ØØØ1Ø
....	466Ø6	34	ØØ 1ØØØ1Ø
.....	466Ø7	Ø	ØØ ØØØØØØ

Vous remarquerez que les bits de poids 64 et 128 sont inutilisés, ils ont d'ailleurs été omis de la matrice pour plus de clarté. En fait, seuls les 6 premiers bits d'un octet sont utilisés pour le dessin, la matrice fait donc 8×6 cases. Si vous avez du mal à comprendre comment le contenu de la matrice est codé, revoyez aussi les principes du système binaire au § 2.1.

3.7.3. Comment créer un caractère

Pour y arriver facilement, commencez par suivre la méthode ci-dessous:

1) Dessinez sur une feuille de papier un damier de 6 colonnes et 8 lignes.
2) Au-dessus de chaque colonne, marquez la valeur de chaque Bit, comme pour la matrice du "A" ci-dessus (32,16,8,4,2 et 1).
3) Dessinez votre caractère en noircissant les cases appropriées.
4) A droite à côté de chaque ligne, notez le nombre résultant de l'addition des valeurs de toutes les cases noires de la ligne, cette valeur étant notée au-dessus de chaque colonne. (Une case blanche vaut Ø).
5) Choisissez le caractère à remplacer par le vôtre, puis repérez son code ou demandez-le à l'Atmos en tapant: PRINT ASC("<Caractère>").
6) Trouvez la première adresse des 8 codant le caractèère avec <Adresse>=46Ø8Ø+<code ASCII>*8.
7) Placer la valeur marquée à côté de la première ligne du damier dans l'adresse trouvée, puis la valeur à côté de la deuxième ligne dans l'adresse trouvée +1, etc. jusqu'à la ligne 8 dans <Adresse>+7: POKE<Adresse>,<Valeur>.

Voilà ! votre caractère est redessiné, utilisez-le pour vérifier. Tout cela a l'air assez long, mais en fait l'habitude vient très vite.

Si vous aimez le calcul mental, voici une autre méthode qui économise des instructions, donc du temps et de la place. Nous avons vu qu'il fallait POKER 8 nombres dans 8 adresses. Mais pourquoi ne pas le faire sur 4 "couples" d'adresses avec DOKE?

Soient les 8 adresses AØ à A7 et les 8 valeurs V1 à V8, il faut écrire:

```
DOKE A0,V1+V2*256:DOKE A2,V3+V4*256:DOKE A4,V5+V6*256:DOKE A6,V7+V8*256
```

Exemple :

Création d'un "invader" à la place du caractère " @ " (Code ASCII 64).

Matrice	*Adresses*	*Valeur décimale*	*Valeur binaire*
Bits 32 16 8 4 2 1			
. . * * . . . * * * * . * . * * . * * * * * * * . * * * * . . . * * . . . * * * * . * * . . * *	46592 46593 46594 46595 46596 46597 46598 46599	12 30 45 63 30 12 30 51	ØØ ØØ11ØØ ØØ Ø1111Ø ØØ 1Ø11Ø1 ØØ 111111 ØØ Ø1111Ø ØØ ØØ11ØØ ØØ Ø1111Ø ØØ 11ØØ11

Ces valeurs seront entrées à l'aide d'une des deux séquences suivantes:

1)

```
10 POKE 46592,12:POKE 46593,30:POKE 46594,45:POKE 46595,63
20 POKE 46596,30:POKE 46597,12:POKE 46598,30:POKE 46599,51
```

2)

```
10 DOKE 46592,7692:DOKE 46594,16173:DOKE 46596,3102:
   DOKE 46598,13086
```

Voici également un exemple de sous-programme, qui lit les valeurs des caractères à redessiner (adresse de début et octets) dans des DATA, puis effectue la transformation à l'aide de POKE. Cette méthode, très souple (on peut rajouter autant de lignes de DATA que l'on veut) est utilisée dans tous les programmes d'applications présentés par la suite:

```
60000 ' SOUS-PROG REDEFINITION CARACTERES
60005 '
60010 READ AD IF AD=0 THEN RETURN
60020 FOR N=0 TO 7
60030 READ OC:POKE AD+N,OC
60040 NEXT:GOTO 60010
60100 DATA 46592,12,30,45,63,30,12,30,51 ' @
```

```
60110 DATA 46808,30,32,38,42,38,16,8,7    ' [
60120 DATA 46824,62,1,25,21,25,2,4,56     ' ]
60130 DATA 46840,8,28,28,28,28,28,62,54   ' £
60200 DATA 0
```

Le premier DATA lu est une adresse (AD), la boucle des lignes 6ØØ2Ø–6ØØ4Ø place les huit valeurs lues à sa suite (OC) dans AD+N, N valant de Ø à 7. La lecture ne se termine que si AD=Ø, ce qui rend la boucle complètement indépendante du nombre total de lignes de DATA. Il suffit juste de veiller à ce que chaque adresse soit suivie de 8 valeurs, et que le dernier DATA soit un Ø. Dans l'exemple, les caractères redéfinis sont: @ , [,] et £.

Il faut bien comprendre que les DATA ne sont donnés qu'à titre d'exemple, ceux-ci pouvant aussi bien être situés n'importe où ailleurs dans le programme. La partie réellement active du sous-programme est constituée par les lignes 6ØØ1Ø à 6ØØ4Ø.

3.8. Les programmes d'applications: CARGEN et GENETEX

Ces deux programmes reprennent à eux deux quasiment toutes les commandes de gestion de l'écran, de la mémoire ou des périphériques qui ont été vues jusqu'à présent. CARGEN est directement en rapport avec le chapitre précédent, quant à GENETEX il vous permettra, comme son nom l'indique de faire du "traitement de textes".

3.8.1. Le programme "CARGEN"

Les procédures de modifications de caractères exposées ci-contre sont idéales lorsque vous n'avez à en redessiner que quelques-uns pour une application particulière. Mais si vous voulez reconfigurer un jeu complet, puis le sauvegarder sur cassette pour réutilisation ultérieure dans un programme Basic, voici ce que vous pourrez faire avec le programme CARGEN:

1) Afficher sur l'écran le jeu complet, ainsi qu'une matrice agrandie.

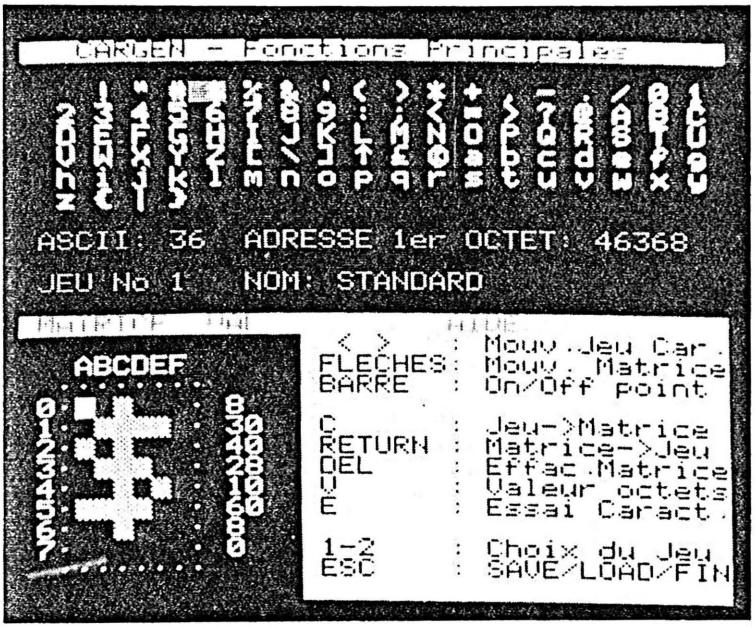
2) Choisir un caractère du jeu en vous déplaçant dans celui-ci, avec un curseur spécial de couleur rouge.

3) Transférer le caractère choisi dans la matrice, le modifier en dessinant avec les 4 flèches, puis replacer le caractère n'importe où dans le jeu.

4) Transférer des caractères d'un emplacement à l'autre dans le jeu.

5) Afficher une zone spéciale de test, où vous pourrez essayer les caractères créés (cas de figures composées de plusieurs caractères, par exemple).

6) Sauvegarder et charger un jeu sur cassette.



Écran principal du programme

CARGEN possède en outre certaines fonctions très intéressantes pour ceux qui dessinent des caractères par POKE ou DOKE dans leurs propres programmes. En effet, le code ASCII et l'adresse en mémoire sont affichés

pour chaque caractère pointé par le curseur rouge dans le jeu. D'autre part, on peut faire apparaître à côté de chaque ligne de la matrice la valeur décimale du dessin de cette ligne (à l'aide de la commande V). Il devient alors très simple de tester vos caractères sur CARGEN, de relever l'adresse et la valeur des 8 octets pour chacun d'eux, et ensuite les modifier dans vos programmes à l'aide des valeurs relevées, suivant les méthodes décrites au § 3.7.3.

Commandes disponibles en mode normal:

<>	Déplacements dans le jeu de caractères (mouvements du curseur rouge).
1–2	Choix du jeu (1=Jeu standard, 2=Jeu graphique).
C	Copier dans la matrice le caractère pointé par le curseur rouge.
V	Calcule la valeur des octets composant un caractère que vous venez de dessiner. Ces données, couplées avec l'adresse de début fournie sur fond rouge, vous permettent de modifier des caractères ponctuellement dans vos propres programmes.
E	Bascule en mode "Test caractères", affiche la zone de test et les nouvelles commandes en bas d'écran.
RETURN	Copie à la position du jeu pointée par le curseur rouge le caractère se trouvant dans la matrice.
Flèches	Déplacements dans la matrice (mouvements du curseur normal).
BARRE	(Barre d'espacement): Trace/Efface un point de la matrice (bascule).
DEL	Efface la matrice.
ESC	Permet d'accéder aux fonctions de Sauvegarde et Chargement d'un jeu, et abandon du programme.

Commandes disponibles en mode "Test caractères":

<>	Déplacements dans le jeu de caractères (mouvements du curseur rouge).
1–2	Choix du jeu (1=Jeu standard, 2=Jeu graphique).

BARRE (Barre d'espacement): Écrit, à la position du curseur normal dans la zone de test, le caractère pointé dans le jeu par le curseur rouge.

Flèches Déplacements dans la zone de test (mouvements du curseur normal).

DEL Efface le caractère sous le curseur normal, dans la zone de test.

ESC Retour en mode "normal" (fin du mode test).

```
1 ' ****************************
2 ' *       CARGEN V2.0        *
3 ' *                          *
4 ' * Generateur de caracteres *
5 ' * Pour ORIC ATMOS  48K     *
6 ' ****************************
7 '
8 '
10 '
40 CLS:PAPER0:INK7
60 GOSUB9100  ' Initialisations
70 EN$="  CARGEN - Fonctions Principales      ":EP$=EN$
80 PRINTCHR$(17);
85 '
90 GOSUB9000  ' Affichage en-tete
100 GOSUB8000 ' Affichage du jeu
105 GOSUB8200 ' Affichage Matrice
110 GOSUB8500 ' Aides
200 GOSUB7000 ' Gestion curseurs
210 GOTO3000  ' Sauveg. jeu/fin prog
220 '
1000 MS$="JEU No "+RIGHT$(STR$(J),1)+"   NOM: "+NJ$
1010 PLOT3,10,MS$
1020 RETURN
1998 '
2000 '  LECTURE/ECRITURE EN MEMOIRE
2005 '
2010 PRINTCHR$(17);
2020 IFAS=13ORAS=86THEN2500
2098 '
2100 '  Lecture d'un caractere
2110 '
2120 NO(1)=127:NO(0)=32
2130 FORM=0TO280STEP40
2140 AW=AA+M/40
2145 Q1=PEEK(AW)/2:R=PEEK(AW)-(INT(Q1)*2)
2150 FORN=5TO0STEP-1
2170 IFQ1=.5THENPOKE48685+M+N,127:Q1=0ELSEPOKE48685+M+N,NO(R)
2175 R=Q1-(INT(Q1/2)*2):Q1=Q1/2
2180 NEXTN
2185 PLOT12,16+M/40,"   ":PLOT12,16+M/40,STR$(PEEK(AW))
2188 PLAY1,0,1,900
2190 NEXTM
2200 PRINTCHR$(17);:RETURN
2498 '
2500 '  Ecriture d'un caractere
2502 '
```

```
2505 MUSIC1,2,5,0
2510 FORM=0TO280STEP40
2520 AW=AA+M/40:T=0
2540 FORN=0TO5
2550 IFPEEK(48685+M+(5-N))=127THENT=T+2^N
2560 NEXTN
2570 IFAS=13THENPOKEAW,T
2580 PLOT12,16+M/40,"   ":PLOT12,16+M/40,STR$(T)
2585 PLAY1,0,1,1200
2590 NEXTM
2600 PRINTCHR$(17);:ZAP:RETURN
2998 '
3000 '  SAVE/LOAD JEU, FIN PROGRAMME
3010 '
3020 CLS:EN$="  CARGEN - Fonctions Utilitaires    ":GOSUB9000
3030 INK6
3040 PRINT@2,6;"<S>ave: Sauvegarde sur cassette"
3050 PRINT:PRINT"<L>oad: Chargement"
3060 PRINT:PRINT"<R>etour au programme"
3070 PRINT:PRINT"<F>in du programme"
3100 GETX$
3110 IFX$="S"THEN3200
3120 IFX$="L"THEN3500
3130 IFX$<>"F"THEN3140
3135 EN$=CHR$(14):GOSUB9000:DOKE48000,1808:CLS::DOKE(#9C),1283
3140 IFX$="R"THENCLS:INK7:GOTO70
3190 GOTO3100
3198 '
3200 '  Save
3210 '
3240 PRINT:PRINTEA$"Nom du jeu ";E$"W";:INPUTNJ$
3250 PRINTCHR$(4);CHR$(17);
3260 PRINT:PRINT:PRINTE$"N"E$"A   PREPAREZ LA CASSETTE":PRINT
3270 PRINTE$"N"E$"A   ET PRESSEZ LA <BARRE>"
3280 PRINTCHR$(4)
3290 GETX$:IFX$<>" "THEN3000
3300 FORN=1TO5:PRINTCHR$(11);CHR$(14);:NEXTN
3310 PRINTCHR$(17);
3315 CSAVENJ$,A#B500,E#BB7F
3320 CLS:GOTO3000
3498 '
3500 '  Load
3510 '
3520 CLS:PRINT:PRINT:PRINTCHR$(4)
3530 PRINTE$"N"E$"A    CHARGEMENT"
3540 PRINTCHR$(4)@2,10;
3550 PRINT" "EA$"Nom du jeu ";E$"W";:INPUTNJ$
3560 CLOADNJ$:GOTO70
5998 '
6000 '  ESSAI DES CARACTERES CREES
6002 '
6005 EN$="  CARGEN - Essai Caracteres          ":GOSUB 9000:PRINT@2,12;CHR$(17)
6010 FORN=1TO13
6020 PRINT"                "
6030 NEXT
6040 PRINT@3,15;
6050 PLOT3,12,"ZONE DE TEST  ":PRINTCHR$(17);
6060 PLOTKK,13,"< >     : Mouv.Jeu Car."
6070 PLOTKK,14,"FLECHES: Mouv. Curseur"
6080 PLOTKK,23,"1-2     : Chgt de Jeu  "
6085 PLOTKK,15,"                       "
6090 PLOTKK,16,"BARRE   : Ecrit sous le"
```

```
 6100 PLOTKK,17," curseur le caractere"
 6110 PLOTKK,18," pointe dans le jeu.  "
 6115 PLOTKK,19,"                      "
 6120 PLOTKK,20,"DEL    : Efface le Ca-"
 6130 PLOTKK,21," ractere sous le Curs."
 6150 PLOTKK,24,"ESC    : Retour Progr."
 6180 GOSUB6500
 6190 XE=0:YE=0
 6200 GETX$:AS=ASC(X$)
 6210 IFAS=27THEN6600
 6220 IFAS=44THENGOSUB7700
 6230 IFAS=46THENGOSUB7750
 6240 IFAS=8ANDXE>0THENXE=XE-1:PRINTX$;:GOTO6200
 6250 IFAS=9ANDXE<10THENXE=XE+1:PRINTX$;:GOTO6200
 6260 IFAS=10ANDYE<10THENYE=YE+1:PRINTX$;:GOTO6200
 6270 IFAS=11ANDYE>0THENYE=YE-1:PRINTX$;:GOTO6200
 6280 IFAS=32THENGOSUB6450
 6290 IFAS=49ORAS=50THENGOSUB7800:GOSUB6500
 6300 IFAS=127THENPRINT" "CHR$(8);
 6400 GOTO6200
 6450 PRINTCHR$(PO+32);:IFXE<10THENXE=XE+1:GOTO6470
 6460 IFYE<10THENPRINTCHR$(17):POKE617,3:PRINTCHR$(17);:XE=0:YE=YE+1ELSEPRINTCHR
$(8);
 6470 RETURN
 6500 FORN=0TO11
 6510 PLOT1,14+N,A(J):PLOT14,14+N,8
 6520 NEXT:RETURN
 6600 FORN=14TO25
 6610 PLOT1,N,8
 6620 NEXT
 6630 EN$=EP$:PRINTCHR$(17);:GOSUB9000:GOTO105
 6998 '
 7000 '  GESTION DES CURSEURS+
 7010 '  BOUCLE PRINCIPALE DE SAISIE
 7020 '
 7090 GOSUB7900
 7100 GETX$:AS=ASC(X$)
 7110 IFAS=49ORAS=50THENGOSUB7800
 7150 IFAS=8THENGOTO7500
 7160 IFAS=9THENGOTO7550
 7170 IFAS=10THENGOTO7600
 7180 IFAS=11THENGOTO7650
 7190 IFAS=69THEN6000
 7200 IFAS=127THEN7300
 7210 IFAS=27THENRETURN
 7220 IFAS=44THENGOSUB7700
 7230 IFAS=46THENGOSUB7750
 7240 IFAS=67ORAS=13ORAS=86THENGOSUB2000
 7245 IFAS=32THENGOSUB7400
 7250 GOTO7100
 7300 FORM=0TO280STEP40
 7310 FORN=0TO5
 7320 POKE48685+N+M,32:PLOT13,16+M/40,"0   "
 7330 NEXTN:NEXTM
 7340 GOTO7100
 7398 '
 7400 '  Tracer/effacer un point
 7402 '
 7410 LA=48685+YM+XM:PRINTCHR$(17);
 7420 IFPEEK(LA)=127THENPOKELA,32ELSEPOKELA,127
 7430 PRINTCHR$(17);:RETURN
 7498 '
```

```
7500 '  Deplacements dans la matrice
7502 '
7510 IFXM>0THENXM=XM-1:PRINTX$;
7520 GOTO7100
7550 IFXM<5THENXM=XM+1:PRINTX$;
7560 GOTO7100
7600 IFYM<280THENYM=YM+40:PRINTX$;
7640 GOTO7100
7650 IFYM>0THENYM=YM-40:PRINTX$;
7690 GOTO7100
7698 '
7700 '  Deplacements dans le jeu
7702 '
7710 IFXJ>3THENXJ=XJ-2:GOTO7730ELSEPOKEAD+XJ+YJ,16:XJ=37
7720 IFYJ=0THENYJ=JJ(J,2):XJ=JJ(J,1)ELSE YJ=YJ-40
7730 GOSUB7900:RETURN
7750 POKEAD+XJ+YJ,16
7755 IFXJ<37ANDXJ+YJ<JJ(J,1)+JJ(J,2)THENXJ=XJ+2:GOTO7770ELSEXJ=3
7760 IFYJ<JJ(J,2)THENYJ=YJ+40ELSEYJ=0
7770 GOSUB7900:RETURN
7798 '
7800 '  Changement de jeu
7802 '
7810 IFVAL(X$)<>JTHENJ=VAL(X$)ELSERETURN
7820 FORN=48042TO48282STEP40
7830 POKEN,A(J)
7840 NEXTN
7845 IFXJ+YJ<=JJ(J,1)+JJ(J,2)THEN7850
7847 POKEAD+XJ+YJ,23:XJ=JJ(J,1):YJ=JJ(J,2):GOSUB7900
7850 GOSUB1000:GOSUB7900:RETURN
7898 '
7900 '  Mise a jour de l'affichage
7902 '
7920 POKEAD+XJ+YJ+2,16
7930 POKEAD+XJ+YJ,17
7940 PO=((XJ-3)/2)+YJ/40*18:PO$=STR$(PO+32)
7945 AA=DB(J)+PO*8
7950 PLOT9,8,PO$+" ":PLOT32,8,STR$(AA)
7960 RETURN
7965 '
7970 '  8000->FIN: SOUS-PROGRAMMES
7980 '  DE GESTION ECRAN ET VARIABLES
7998 '
8000 '  AFFICHAGE DES JEUX DE CARACT.
8002 '
8010 PRINTCHR$(30):PRINT" ";
8020 FORN=0TO72STEP18
8030 FORI=32+NTO32+17+N
8040 PRINT" "CHR$(I);
8050 NEXTI
8060 PRINT"  ";
8070 NEXTN
8080 PRINT" z { ¦ }"
8090 RETURN
8198 '
8200 ' DESSIN DE LA MATRICE
8202 '
8205 PRINT@2,8;E$"A":PRINT:PRINTE$"B":PRINT
8210 G$=CHR$(126):G1$="  "
8215 GOSUB1000
8220 FORI=1TO8:G1$=G1$+G$:NEXT:PLOT1,12,1
8222 PLOT3,8,"ASCII:      ADRESSE 1er OCTET:      "
```

```
8225 PRINTE$"VMATRICE  VAL.":PRINTSPC(14)EA$
8226 PRINT"   ABCDEF"SPC(5)EA$:PRINTG1$SPC(4)EA$
8230 FORI=0TO7:PRINTSTR$(I)G$"      "G$" 0  "EA$
8240 NEXTI:PRINTG1$SPC(4)EA$:PRINTSPC(14)EA$;
8250 XM=0:YM=0
8260 RETURN
8498 '
8500 '  AFFICHAGE DES AIDES
8502 '
8505 KK=18
8510 PLOTKK,12,"        AIDE"
8520 PLOTKK,14,"FLECHES: Mouv. Matrice"
8530 PLOTKK,13," < >    : Mouv.Jeu Car."
8540 PLOTKK,15,"BARRE   : On/Off point"
8545 PLOTKK,16,"                      "
8550 PLOTKK,18,"RETURN : Matrice->Jeu"
8560 PLOTKK,17,"C      : Jeu->Matrice"
8570 PLOTKK,19,"DEL    : Effac.Matrice"
8572 PLOTKK,20,"V      : Valeur octets"
8574 PLOTKK,21,"E      : Essai Caract."
8575 PLOTKK,23,"1-2    : Choix du Jeu"
8580 PLOTKK,24,"ESC    : SAVE/LOAD/FIN"
8590 PRINT@5,16;CHR$(17);:RETURN
8998 '
9000 '  GESTION DE L'EN-TETE
9002 '
9005 DOKE634,48000:POKE48001,0:PRINTCHR$(30)E$"S"EN$:DOKE634,48040
9010 RETURN
9098 '
9100 '  INITIALISATIONS
9102 '
9110 FORI=1TO2:READA(I),JJ(I,1),JJ(I,2),DB(I):NEXTI
9120 READ NJ$,J
9140 E$=CHR$(27):EA$=E$+"@"+E$+"U"
9160 IFPEEK(47096)=63THENFORN=0TO7ELSEGOTO9185
9170 POKE47096+N,PEEK(47088+N):POKE47088+N,0
9180 NEXT:POKE47091,8
9185 AD=#BBD0:CH=#B500:XJ=3:YJ=0
9190 RETURN
9200 DATA 8,9,200,#B500,9,17,160,#B900
9210 DATA STANDARD,1
```

Structure et explications du programme

"CARGEN" est construit de façon modulaire, cinq sous-programmes servent à sa mise en route et à la constitution de l'écran, et huit autres gèrent l'ensemble des fonctions disponibles; l'ensemble occupe environ 4,3 Koctets en mémoire. Dans les explications, nous respecterons l'ordre d'exécution du programme plutôt que celui des numéros de ligne.

6Ø : Appel du sous-programme 91ØØ où sont effectuées les initialisations de variables et la création des 2 caractères spéciaux utilisés par le programme.

Le tableau A() contient des codes à afficher pour le changement de jeu, et le tableau JJ() les pointeurs servant à l'affichage de ces jeux. E$ est le caractère ESCAPE ou CHR$(27).

* 9Ø-2ØØ : Appel des routines de démarrage et de constitution de l'écran.

9ØØØ : Gère l'en-tête affiché en première ligne d'écran.

8ØØØ : Ces trois sous-programmes affichent à l'écran tout ce qui est nécessaire au programme. Le sous-programme 8ØØØ imprimant le jeu n'est appelé qu'une fois, le changement de jeu s'obtenant par l'impression des codes adéquats dans la troisième colonne de l'écran.

* 7ØØØ-725Ø: Boucle principale du programme; le clavier est lu, et en fonction de la touche appuyée on se branche sur les routines suivantes:

74ØØ : Trace ou efface un point de la matrice agrandie, en fonction de l'état précédent de ce point.

75ØØ : Gère les déplacements du curseur à l'intérieur de la matrice à l'aide des 4 flèches; YM est l'ordonnée et XM l'abscisse du curseur à l'intérieur de la grille.

77ØØ : Gère les déplacements du curseur spécial (qui est en fait une zone colorée) dans le jeu de caractères; XJ et YJ sont l'abscisse et l'ordonnée de ce curseur.

78ØØ : Opère le changement de jeu, par simple modification des codes placés au début de chaque ligne où celui-ci est affiché. NJ contient le N° du jeu (1 ou 2), et le tableau A() le code à imprimer. Le tableau JJ() contient les limites d'affichage (n'oublions pas qu'il n'y a que 64 mosaïques graphiques).

2ØØØ : Calcul de l'adresse de début (AA) du caractère pointé par le "curseur jeu", en fonction de la position de celui-ci (PO, lui-même calculé en fonction de XJ et YJ) et du jeu affiché (J=1 ou 2).

21ØØ : Lit les octets constituant le caractère pointé par le "curseur jeu" dans les adresses correspondantes, puis les convertit en binaire afin de dessiner chaque ligne de la matrice. AW est l'adresse de début du caractère.

2ØØØ : Effectue l'opération inverse : Lit chaque ligne de la matrice, la convertit en décimal, puis place chaque nombre aux adresses du caractère pointé par le "curseur jeu".

6ØØØ : Crée une zone de test pour les caractères à la place de la matrice, où le curseur peut se mouvoir librement. On y imprime le caractère pointé par le curseur jeu en pressant la barre d'espacement.

3ØØØ : Fonctions annexes, Sauvegarde et Chargement d'un jeu de caractères sur cassette.

Note: De nombreuses remarques (') ont été incluses dans le programme, dans un souci de clarté. Vous pouvez évidemment supprimer purement et simplement les lignes du type 1Ø98, 2ØØ2, etc. pour vous éviter du travail de frappe, mais conservez les Numéros de ligne "ronds": 2ØØØ, 3ØØØ, 8ØØØ... car de nombreux GOTO et GOSUB y font référence. N'oubliez pas toutefois que plus vos programme seront documentés, plus ils seront lisibles par la suite.

3.8.2. Le programme "GENETEX"

Programme de traitement de textes articulé autour d'un menu principal proposant huit options différentes, GENETEX est également conçu de façon modulaire: Chaque option du menu constitue pratiquement un programme indépendant exécutant une tâche précise. Son occupation mémoire est de 9 Koctets environ. Le Menu principal vous donne le choix entre:

Écrire un **Nouveau texte**

Demande la longueur de ligne désirée (jusqu'à 76 colonnes), et passe en mode "saisie"; vous pouvez taper du texte "au kilomètre", c'est-à-dire sans vous soucier des passages à la ligne. Les mots trop longs ne sont pas coupés en fin de ligne, et le texte est mémorisé dans un tableau de chaînes au fur et à mesure de son entrée; celui-ci peut faire jusqu'à 6ØØ lignes de long.

Ajouter du texte

Affiche les cinq dernières lignes du texte en mémoire, et passe au mode "saisie".

Visualiser le texte

Permet de parcourir celui-ci rapidement en affichant les numéros de ligne.

Modifier le texte

De loin l'option offrant le plus de possibilités, elle autorise:

— L'insertion d'un nombre quelconque de lignes blanches.
— La réécriture totale ou partielle d'une ligne.
— La destruction totale ou partielle d'une ligne.
— Le réalignement d'un paragraphe sur les marges choisies, après sa modification.

Imprimer le texte

Procède à l'impression sur papier du texte en mémoire, en gérant le centrage, l'interligne, le nombre de copies, les sauts de page, etc...

Sauvegarde et relecture du texte sur K7

Il serait évidemment gênant de perdre un texte sitôt l'ordinateur éteint; ces deux options autorisent la sauvegarde du tableau sur cassette ainsi que son rechargement par la suite.

Chargement d'un jeu de caractères

Cette option opèrera le chargement d'un jeu de caractères personnel élaboré par CARGEN, par exemple un AZERTY accentué pour une présentation améliorée.

REMARQUE: En ce qui concerne les REM, ce que nous avons dit à propos de CARGEN reste entièrement valable.

```
1 '     ************************
2 '     *     GENETEX V2.1     *
3 '     *                      *
4 '     * TRAITEMENT DE TEXTE *
5 '     * POUR ORIC ATMOS      *
6 '     ************************
7 '
8 '
10 GOSUB9900
20 CLS:PAPER0:INK7
40 GRAB:HIMEM46000
90 '
100 GOSUB9000 ' Initialisations
120 GOTO8000  ' Menu principal
130 '
995 '
1000 '  NOUVEAU TEXTE
1005 '
1010 M$=E$+"B"+"NOUVEAU TEXTE":GOSUB8800
1020 CLS:INK6
1030 PRINT@2,6;E$"AATTENTION:"E$"FSi un texte se trouve en"
1040 PRINT"  memoire, il sera efface.":PRINT
1050 PRINT"  <RETURN>= suite":PRINT"  <ESC>   = retour menu":PRINT
1060 GETX$
1070 IFX$=E$THEN RETURN
1080 IFX$<>CHR$(13)THEN1060
1090 GOSUB9900
1100 PRINT"Longueur des lignes, en nombre de"
1110 INPUT"caracteres (Max.=76)";LL
1130 IFLL>76ORLL<10THEN1110
1140 IFLL>39THENP1=2ELSEP1=1
1150 GOSUB7000:CLS
1235 '
1240 '  BOUCLE PRINCIPALE DE SAISIE
1245 '
1250 T$="":C=1:L=1:LS=LL-4:P2=1
1270 POKE#20C,127
1300 C$=KEY$:IFC$=""THEN1300
1320 AX=ASC(C$)
1325 IFAX=32ORAX=45THENC1=C:HT=PEEK(617)+1:VT=PEEK(616)-1:GOTO1400
1330 IFAX=13THEN1600
1340 IFAX=127ANDC>1THENPRINTC$;:C=C-1:T$=LEFT$(T$,C-1):GOTO1300
1350 IFAX=27THENT$(L)=T$:RETURN
1360 IFAX=20THENPRINTC$;
1390 IFAX<32ORAX=127THEN1300
1400 IFC=LSTHENPING
1420 IFC=LLTHEN1500
1430 PRINTC$;
1440 T$=T$+C$:C=C+1
1450 GOTO1300
1495 '
1500 '  Saut de ligne auto
1505 '
1510 IFAX=32ORAX=45ORC1=0THENC$=CHR$(13):GOTO1640
1520 IFC1<39ANDP1=2THENPRINTCHR$(11);
1550 PLOTHT,VT,0
1560 T$(L)=LEFT$(T$,C1):L=L+1:T$=RIGHT$(T$,C-C1-1):PRINTCHR$(13)
1570 PRINTT$;:C=LEN(T$)+1:C1=0
1580 GOTO1400
1595 '
1600 '  Saut de ligne manuel
```

```
1605 '
1610 T$=T$+PA$:PRINTPA$;
1640 T$(L)=T$:L=L+1:T$="":C=1:C1=0:PRINTC$
1650 GOTO1300
1895 '
1900 '  AJOUTER AU TEXTE
1905 '
1910 M$=E$+"BAJOUTER AU TEXTE":GOSUB8800:INK6:CLS
1920 GOSUB7000:CLS
1930 IFL=0THENPOP:GOTO8000
1940 IFL>10THENL1=L-10ELSEL1=1
1950 FORN=L1TOL-1
1960 PRINTT$(N)
1970 NEXT
1980 PRINTT$(L);:C=1
1990 RETURN
1995 '
2000 '  MODIFICATIONS
2005 '
2010 M$=E$+"CMODIFICATIONS":GOSUB8800:CLS:INK6
2020 GOSUB8700
2030 GOSUB7200
2050 GOSUB2900
2060 T$="":C=1
2070 POKE#20C,127
2100 PLOT2,18,"LIGNE No"+STR$(DL)+"  "
2110 X$=KEY$:IFX$=""THEN2110ELSEAX=ASC(X$)
2120 IFAX=12THEN2300     'Insert ligne
2130 IFAX=4 THEN2400     'Destr. ligne
2140 IFAX=18THEN2500     'Restr. parag
2150 IFX$=E$THENRETURN:  'Retour Menu
2155 IFAX=20THENPRINTX$;
2157 IFAX=5THENT$(DL)=LEFT$(T$(DL),C+(C>1)):GOTO2050
2160 IFAX=10ANDDL<LTHENDM=DL:DL=DL+1:GOTO2050
2165 IFAX=9ANDC<LLTHEN2250
2170 IFAX=11ANDDL>1THENDM=DL:DL=DL-1:GOTO2050
2180 IFAX=127ANDC>1THENPRINTX$;:C=C-1:T$=LEFT$(T$,C-1):GOTO2110
2190 IFAX=13THENT$(DL)=T$+PA$:C=1:GOTO2050
2195 IFAX<32THEN2110
2200 IFC=LLTHEN2110
2220 PRINTX$;:T$=T$+X$:C=C+1
2230 GOTO2110
2250 T$=T$+MID$(T$(DL),C,1)
2270 PRINTX$;:IFC=38THENPRINT@2,PO-1;
2280 C=C+1:GOTO2110
2295 '
2300 '  Insert lignes
2305 '
2310 CLS:PRINT:PRINT:PRINT
2320 INPUT"Combien de lignes a inserer ";IN:IN=IN+1
2330 IFIN+L>600THENPRINT:PRINT"  PLUS DE PLACE !":PRINT:GOTO2320
2340 FORN=(L+IN)TO(DL+IN)STEP-1
2350 T$(N)=T$(N-IN)
2360 NEXT
2370 FORN=DLTODL+IN-1
2380 T$(N)=PA$
2390 NEXT:L=L+IN
2395 '
2400 '  Destruction ligne
2405 '
2410 FORN=DLTOL-1
2420 T$(N)=T$(N+1)
2430 NEXT:L=L-1
```

```
2440 C=1:GOTO2050
2495 '
2500 '  Refonte paragraphe
2505 '
2510 U=0:CLS:PRINT:PRINT:PRINTE$"L         REFONTE EN COURS"
2520 M=DL
2530 N=LEN(T$(M))+2:A=0:T1=LEN(T$(M+1))
2540 REPEAT
2550 N=N-1
2560 UNTILMID$(T$(M),N,1)<>" "ORN=1
2570 PL=(LL-N)
2580 C=0
2590 REPEAT
2600 C=C+1:IF MID$(T$(M+1),C,1)=" "THENA=C
2605 IF MID$(T$(M+1),C,1)=PA$THENA=C+1
2610 UNTILC>PL
2620 IFA=0THEN2650
2630 T$(M)=LEFT$(T$(M),N)+" "+LEFT$(T$(M+1),A-1)
2635 IFA>T1THENA=T1
2640 T$(M+1)=RIGHT$(T$(M+1),T1-A)
2650 M=M+1
2660 IFRIGHT$(T$(M-1),1)=PA$THENC=1:T$(M)=PA$:GOTO2680
2665 IF RIGHT$(T$(M),1)=PA$THENC=1:GOTO2680
2670 GOTO2530
2680 IFUTHEN2050ELSEU=-1:GOTO2520
2895 '
2900 '  Affichage lignes
2902 '
2903 IFC=1THEN2910
2904 IFC<=LEN(T$(DM))THENTD$=RIGHT$(T$(DM),(LEN(T$(DM))-C)+1)ELSETD$=""
2906 T$(DM)=T$+TD$
2910 CLS:PRINT:IFDL-8/P1<1THENMI=0ELSEMI=DL-6/P1
2915 IF(DL+4/P1)<LTHENMA=(DL+6/P1)ELSEMA=L
2920 FORN=MITOMA:IFN=DLTHENPO=PEEK(616)
2930 PRINTT$(N)
2940 NEXT
2950 PRINT@2,PO-1;
2990 DM=DL:RETURN
2995 '
3000 '  IMPRESSION PAPIER
3005 '
3010 M$=E$+"CIMPRESSION":INK5
3020 GOSUB8800:CLS
3085 '
3090 '  Saisie des parametres
3095 '
3100 PRINT@2,5;"Marge (Max="79-LL") Centre=100 ";
3110 PRINT"Marge (Max="67-LL") Centre=100 ";
3120 INPUT MG
3124 IFMG=100THENMG=(80-LL)/2
3126 IFMG>79-LLTHENMG=80-LL
3130 PRINT:INPUT"Simple/double interligne (1/2) ";IT
3140 PRINT:INPUT"Ligne debut ";L1
3150 PRINT:INPUT"Ligne fin   ";L2
3155 IFL2>LTHENL2=L
3160 PRINT:INPUT"Lignes/page ";PP
3170 PRINT:INPUT"Nombre de copies ";NC
3185 '
3190 '  Impression
3195 '
3200 GOSUB7300:CLS
3210 FORM=1TO NC
3220 GOSUB 3350
```

```
3230 N=L1
3240 REPEAT
3250 IFRIGHT$(T$(N),1)=PA$THENP$=LEFT$(T$(N),LEN(T$(N))-1)ELSEP$=T$(N)
3260 PRINTP$:LPRINTSPC(MG)P$
3265 X$=KEY$:IFX$=E$THENN=L2+1:M=NC+1
3270 IFIT=2THENLPRINT
3280 IFN/PP/IT=INT(N/PP/IT)THENLPRINTCHR$(12);:GOSUB3350
3290 N=N+1
3300 UNTIL N>L2
3310 LPRINTCHR$(12);
3320 NEXT M
3330 RETURN
3350 FORI=1TO(65-PP)/2:LPRINT:NEXT:RETURN
3995 '
4000 '  VISUALISATION
4005 '
4010 M$=E$+"CVISUALISATION":GOSUB8800:CLS:INK6
4020 GOSUB 8700
4100 GOSUB7100
4105 CLS:PRINT
4110 IFDL<1ORDL>LTHENDL=1
4120 N=DL
4130 REPEAT
4140 PRINTT$(N)
4150 N=N+1
4170 UNTILN=DL+20/P1ORN=L+1
4190 PLOT2,22,"LIGNES "+STR$(DL)+"-"+STR$(N)+"   "
4200 GETX$
4210 IFX$=CHR$(10)THENDL=DL+16/P1:GOTO4105
4220 IFX$=CHR$(11)THENDL=DL-16/P1:GOTO4105
4230 IFX$=E$THENRETURN
4240 GOTO4190
4995 '
5000 '  SAUVEG/LECTURE SUR K7
5005 '
5010 CLS:INK2:IFX$="L"THEN5200
5020 IFL=0THEN POP:GOTO8000
5023 '
5025 '  Sauvegarde
5027 '
5030 T$(0)=RIGHT$(STR$(L),LEN(STR$(L))-1)+RIGHT$(STR$(LL),2)+RIGHT$(STR$(P1),1)
5040 M$=E$+"ESAUVEGARDE":GOSUB8800:CLS
5060 PRINT@2,9;CHR$(4)E$"J METTEZ LE K7 EN ENREGISTREMENT":PRINTCHR$(4):PRINT
5070 PRINT"         ET PRESSEZ UNE TOUCHE"
5080 GETX$
5090 STORE T$,"TEXTE"
5100 RETURN
5195 '
5200 '  Chargement
5205 '
5210 M$=E$+"ECHARGEMENT":GOSUB8800:CLS
5250 PRINT@2,9;CHR$(4)E$"J      DEMARREZ LA CASSETTE":PRINTCHR$(4):PRINT
5255 GOSUB9900
5260 RECALL T$,"TEXTE"
5270 N=LEN(T$(0))-3
5280 L=VAL(LEFT$(T$(0),N)):LL=VAL(MID$(T$(0),N+1,2)):P1=VAL(RIGHT$(T$(0),1))
5290 LS=LL-4:T$="":C=1:P2=1:T$(0)=""
5300 RETURN
6985 '
6990 '  MESSAGES EN BAS D'ECRAN
6995 '
6997 '
7000 '  Saisie
```

```
7005 '
7010 DOKE636,800:POKE638,20:DOKE48880,5376
7020 ZX=23:EN=2:GOSUB7500
7030 PRINT@2,21;"<MARGE"SPC(LL-7)">"E$"P"
7040 IFLL>39THENDOKE48920,22
7050 PRINT@2,23;"-RETURN  fin de paragraphe"
7060 PRINT"-DEL     efface le caractere precedent";
7070 PRINT"-ESCAPE  retour menu"
7080 RETURN
7095 '
7100 '  Visu
7105 '
7110 DOKE48920,21
7120 ZX=24:EN=3:GOSUB7500
7130 PRINT@2,24;"<^> ou <"CHR$(126)">   Deplacement"
7140 PRINT"<ESC>     Retour Menu"
7150 DOKE636,840:POKE638,22:RETURN
7195 '
7200 '  Modif
7205 '
7210 DOKE48760,21:ZX=20:EN=3:GOSUB7500
7220 PRINT@2,20;"<^><"CHR$(126)">   Balayage ecran"
7230 PRINT"-> <DEL> Balayage ligne"
7240 PRINT"<CTRL E> Effacement ->Fin de ligne"
7250 PRINT"<CTRL L> Insertion lignes vides"
7260 PRINT"<CTRL D> Destruction ligne"
7270 PRINT"<CTRL R> Restructuration paragraphe"
7280 PRINT"<ESC>    Retour Menu";
7290 DOKE636,720:POKE638,18:RETURN
7295 '
7300 '  Impression
7305 '
7310 DOKE48960,21:PRINT@2,23;"       ALLUMEZ L'IMPRIMANTE":PRINT
7320 PRINT"<ESC>: RETOUR MENU"
7330 DOKE636,880:POKE638,22
7340 RETURN
7500 FORN=ZXTO26:PLOT0,N,EN:NEXT
7510 RETURN
7990 '
7995 '
8000 '  MENU PRINCIPAL
8005 '
8010 M$=M1$:GOSUB8800:INK7
8020 POKE#20C,255:DOKE#27C,1040:POKE#27E,27:CLS
8100 PRINT@2,5;CHR$(4)E$"J"E$"A   MENU PRINCIPAL"
8110 PRINTCHR$(4):PRINT:PRINT
8120 PRINTE$"F <N>ouveau texte":PRINT
8130 PRINTE$"C <A>jouter au texte en memoire"
8140 PRINTE$"C <V>isualiser le texte en memoire"
8150 PRINTE$"C <M>odifier le texte en memoire"
8160 PRINTE$"C <I>mprimer le texte en memoire":PRINT
8170 PRINTE$"E <S>auvegarder le texte sur K7"
8180 PRINTE$"E <L>ire un texte sur K7"
8190 PRINTE$"E <C>harger un jeu de caracteres"
8200 PRINT:PRINTE$"D  VOTRE CHOIX ?";:GETX$
8210 IFX$="N" THEN GOSUB1000
8220 IFX$="C" THEN GOSUB8500
8230 IFX$="L"ORX$="S"THENGOSUB5000
8240 IFL=0THEN8000
8260 IFX$="M" THEN GOSUB2000
8270 IFX$="I" THEN GOSUB3000
8280 IFX$="V" THEN GOSUB4000
8290 IFX$="A" THEN GOSUB1900:GOSUB1270
```

```
8300 GOTO8000
8495 '
8500 '  CHARGER UN JEU DE CARACTERES
8505 '
8510 M$=E$+"ELOAD JEU":GOSUB 8800:CLS:INK2
8520 PRINT@4,6;
8550 INPUT"NOM DU JEU DE CHR ";NJ$
8560 CLOADNJ$
8570 RESTORE:GOSUB 9050:RETURN
8700 '
8720 PRINT@2,5;"Ligne de debut (MAX="L") ";:INPUT DL
8730 CLS:RETURN
8800 DOKE48000,528:DOKE634,48000:PRINTCHR$(30)CHR$(14)M$:DOKE634,48040
8820 RETURN
8985 '
8990 '  Ecriture dans la ligne 0
8995 '
9000 M1$=" "+CHR$(27)+"BGENETEX - TRAITEMENT DE TEXTES  ":M$=M1$:GOSUB 8800
9030 ES$="                                              "
9040 E$=CHR$(27):PA$=CHR$(126)
9045 '
9050 '  Creation caractere
9055 '
9300 READAD
9310 IFAD=0THEN9800
9320 FORN=0TO7
9330 READDA:POKEAD+N,DA
9340 NEXT
9350 GOTO9300
9700 DATA 47088,8,8,8,8,42,28,8,0
9710 DATA 0
9800 RETURN
9895 '
9900 '  Vidage variables
9905 '
9910 DOKE#A0,DEEK(#9E)
9920 DIMT$(600)
9930 DIMI$(15)
9960 RETURN
```

Fonctionnement du programme

1ØØ-12Ø : Appel des sous-programmes de mise en route:

Mise en route

9ØØØ : Initialisations: M1$ est le message placé en en-tête sur la ligne réservée. Création du caractère "flèche vers le bas", qui indique les fins de paragraphe et se trouve au code 126 (PA$).

88ØØ : Impression sur la ligne réservée du message contenu dans M$.

8ØØØ-83ØØ : Menu principal. Celui-ci effectue l'aiguillage vers les divers sous-programmes, et il faut y retourner pour passer d'une fonction à une autre. Le clavier est mis automatiquement en majuscules pour la saisie des commandes.

Nouveau texte

1ØØØ-115Ø : Nouveau texte; saisie et initialisation de ses pointeurs et variables:

- LL : Longueur des lignes.
- T$: Ligne en cours de frappe.
- T$() : Tableau, contenant l'ensemble des lignes du texte.
- C : Numéro de colonne du caractère en cours de frappe à l'intérieur de la ligne.
- L : Longueur du texte déjà tapé, ou N° de la dernière ligne.
- LS : N° de colonne où intervient la sonnerie.
- P1 : Pointeur déterminant si une ligne de texte est plus ou moins longue qu'une ligne d'écran (LL>38 ou LL<38). Utilisé par les sous-programmes "Ajouter du texte" et "Modifications".

127Ø : Passage automatique du clavier en minuscules pour la frappe du texte.

13ØØ-139Ø : Boucle principale en mode "saisie". C$ est le caractère tapé.

14ØØ-145Ø : Gère l'impression du caractère, l'ajoute à T$, incrémente C et actionne la sonnerie trois colonnes avant la fin de ligne.

15ØØ-158Ø : Si le dernier caractère de la ligne n'est pas un espace le mot est considéré comme trop long; il est alors soustrait de T$. La ligne obtenue est chargée dans T$(L), et le bout de mot extrait dans T$. Le curseur passe à la ligne suivante, où le nouveau T$ est imprimé. Enfin, le curseur se place à la fin de celui-ci, position d'affichage du prochain caractère.

16ØØ-165Ø : Passage manuel à la ligne suivante (fin de paragraphe), et impression du symbole de fin de paragraphe.

Ajout de texte

19ØØ-199Ø : Affiche les dix dernières lignes du texte, et va à la section "saisie", comme pour un nouveau texte (127Ø-); Le branchement s'effectue à partir du menu (829Ø).

Modifications

2Ø2Ø-2Ø3Ø : Impression de l'en-tête et du texte d'aide.

2Ø5Ø : Saisie du numéro de la ligne à corriger (87ØØ), puis appel de la routine d'affichage de celle-ci avec les cinq précédentes et suivantes (29ØØ-299Ø). Le curseur peut ensuite être déplacé dans le texte et dans les quatre directions avec les flèches.

DL : Numéro de la ligne où se trouve le curseur.
P1,MA,MI : Pointeurs gérant l'affichage des lignes plus longues qu'une ligne d'écran.
PO : Position verticale du curseur à l'écran.
DM : Mise en mémoire du N° de la ligne en cours, pour les déplacements verticaux.

21ØØ-2195 : Boucle principale de saisie du texte modifié et des caractères de contrôle des fonctions spéciales.

22ØØ-223Ø : Saisie-impression d'un caractère.

225Ø-228Ø : Gestion des déplacements du curseur dans la ligne.

23ØØ-2395 : Insertion de IN lignes vides.

24ØØ-244Ø : Destruction de la ligne où se trouve le curseur.

25ØØ-268Ø : Si des mots ont été supprimés sur des lignes, reformate le paragraphe de manière à ce que chaque ligne contienne le maximum de mots possible.

29ØØ-299Ø : Ce sous-programme réaffiche l'écran chaque fois qu'on change de ligne, permettant de tenir compte des modifications survenues.

Impression

3ØØØ-317Ø : Saisie des paramètres d'impression.

MG : Marge gauche.
IT : Simple ou double interligne (1 ou 2).
L1 : Première ligne de la partie du texte à imprimer.
L2 : Dernière ligne à imprimer (Si L2>longueur du texte, celui-ci est imprimé jusqu'à la fin).
PP : Nombre de lignes par page.
NC : Nombre de copies à effectuer.

32ØØ-333Ø : Boucle d'impression.
Note : Cette routine fonctionne avec tous les types d'imprimantes.

P$: Ligne à imprimer.
N : Numéro de la ligne à imprimer dans T$().

325Ø : Suppression du caractère PA$ (fin de paragraphe-flèche vers le bas) avant impression.

328Ø : Commande le changement de page lorsque le nombre correct de lignes y a été imprimé. CHR$(12) est le code de saut de page sur la plupart des imprimantes.

335Ø : Routines imprimant des lignes blanches en haut de chaque page, pour y centrer le texte en fonction de PP. Si vous utilisez du papier de 12 pouces, changez le (65-PP) en (7Ø-PP).

Visualisation du texte en mémoire

4ØØØ-424Ø: Affiche le texte par blocs de 2Ø lignes (ou dix lignes si elles sont plus longues que 38 car.), avec leurs numéros de début et de fin.

Sauvegarde, Chargement d'un texte

5Ø3Ø : Charge T$(Ø) avec les paramètres du texte, sa longueur (L), la longueur des lignes (LL) et le pointeur P1.

5ØØØ-5Ø9Ø: Effectue la sauvegarde du tableau T$ en utilisant l'instruction STORE.

52ØØ-526Ø: Procède au chargement de T$() en mémoire à partir du magnétocassette.

528Ø-529Ø: Reconvertit en variables les paramètres constituant T$(Ø).

Textes d'aide

7ØØØ-733Ø: Affichage de petits textes d'aide (4-5 lignes) dans une zone réservée en bas d'écran. Il y a un texte par sous-programme.

88ØØ : Dimensionne l'écran pour récupérer la ligne du haut, y imprime l'en-tête contenue dans M$ (différente pour chaque sous-programme), puis le remet à sa taille normale.

Chargement d'un jeu de caractères

85ØØ-856Ø: Effectue le chargement d'un jeu de caractères enregistré sur K7, en particuliers les jeux élaborés par CARGEN.

4

Le graphique haute- résolution

4.1. Introduction

En plus des modes TEXT et LORES qui ne permettent de dessiner qu'avec des caractères déjà formés, l'Atmos dispose d'un mode graphique haute-résolution de 24Ø par 2ØØ points adressables individuellement. Tout l'écran est accessible, même les deux colonnes et la ligne normalement interdites en mode texte.

Les huit couleurs restent toujours disponibles, mais on ne peut pas définir séparément la couleur de chaque point : Celle-ci est obtenue par un attribut placé dans une "zone" rectangulaire, dont la dimension minimale est de 1 point de haut par 6 points de long. L'attribut choisi est actif pour tout ce qui s'affiche à droite de cette zone, rappelant ainsi les codes ESCAPE en mode texte (voir § 3.3.-3.4) ; les valeurs de ces attributs sont d'ailleurs exactement les mêmes.

Nous verrons dans cette section comment exploiter toutes les commandes du graphique, mais aussi comment créer celles qui n'existent pas.

4.2. *L'organisation-mémoire du graphique*

Nous avons vu qu'en mode texte, chaque case de l'écran occupait un octet. Cette méthode serait inapplicable en graphique s'il en était de même pour chaque point: Il ne faudrait pas moins de 24Ø×2ØØ=48ØØØ octets pour stocker l'écran, c'est-à-dire presque la totalité de la mémoire ! Un autre principe est donc utilisé, qui consiste à grouper les points horizontalement, six d'entre eux constituant un octet. De cette manière, l'écran "tient" dans 2ØØ×24Ø=8 K., ce qui est déjà beaucoup plus raisonnable.

Le dessin déterminé par un octet est l'image exacte de sa valeur binaire, un point "allumé" (prenant la couleur du 1er plan) valant 1 et un point "éteint" (couleur du fond) zéro. La valeur décimale de chaque octet est calculée selon le principe déjà vu au § 3.7.2, utilisé pour le codage des caractères.

Nous savons qu'un octet est constitué de 8 bits, mais l'Atmos n'en utilise que 6 pour coder le graphique; les deux bits inutilisés (bits 5 et 6, de poids 32 et 64) de l'octet déterminent si celui-ci doit être interprété comme un dessin ou comme un attribut: Si les deux sont à Ø il s'agit d'un attribut, si l'un des deux est à un c'est un dessin. Voyez aussi les commandes FILL et PATTERN, § 4.3.1.

Adresses écran

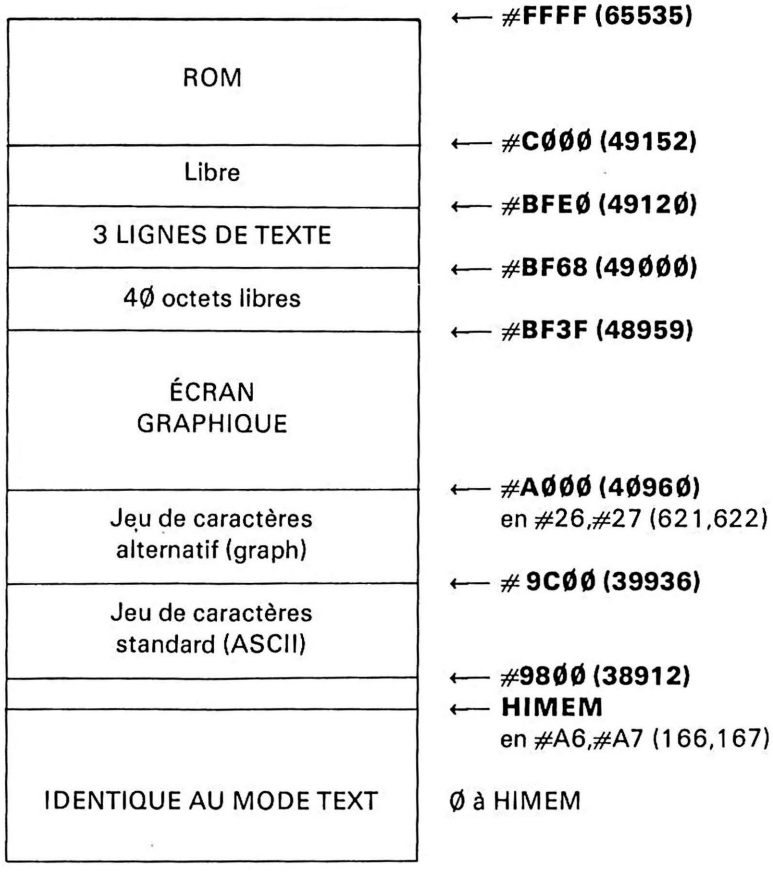
Abscisse→ Ø 1 2 3 4 5 6 7 8 9 10 11		226 – 232233 – 239	
Ordonnée			
Ø	4Ø96Ø	4Ø961	4Ø998	4Ø999
199	4892Ø	48921	48958	48959

On trouvera l'adresse-mémoire de l'octet codant un groupe de points par la formule suivante, où Y est l'ordonnée du groupe et X l'abscisse de n'importe lequel des six points :

```
<Adresse>=40960+Y*40+X/6
```

4.2.1. Le Memory-map en haute-résolution

L'écran graphique occupant environ quatre fois plus de place en mémoire que l'écran-texte, le memory-map s'en trouve légèrement modifié :



Vous remarquerez que 3 lignes de texte sont disponibles en bas d'écran pour des commentaires ou explications, ou pour lister des lignes de votre programme. Celles-ci ne sont pas complètement jointives avec la zone haute-résolution, une quarantaine d'octets restant libres. Leur adresse de début est #BF68 (49ØØØ), toute la zone comprise entre celle-ci et la fin de l'écran est gérée comme en mode TEXT: Scrolling en bas d'écran, effacement par CLS etc...

En pratique, nous n'agirons pas sur les jeux de caractères en mode graphique (c'est beaucoup plus simple en TEXT), et les limites de l'écran n'ont besoin d'être connues que si on veut le sauvegarder sur cassette ou en faire une copie sur imprimante (hardcopy), comme le fait le programme du § 4.4.

Pour sauvegarder l'écran sur cassette, il faut être en mode HIRES puis taper:

```
CSAVE"<Nom>",A#A000,E#BFE0 [,S]
```

Et pour le recharger:

```
CLOAD "[<Nom>]" [,S]
```

4.2.2. Les variables-système liées au graphique

La plupart des paramètres importants du mode HIRES sont disponibles en permanence dans des variables-système, autorisant un grand nombre d'interventions "hors-Basic", et même certaines fonctions que celui-ci ne permet pas.

#**213** (531)

Fonction: Stocke la valeur définissant le type de trait utilisé par DRAW et CIRCLE (voir PATTERN, § 4.3.1)

#**219** (537)

Fonction: Retourne l'abscisse (X) du curseur graphique, lequel est déplacé par les diverses commandes de dessin (DRAW, etc.).

#**21A** (538)

Fonction: Retourne l'ordonnée (Y) du curseur graphique.

Les deux variables ci-dessus sont très importantes, car elles représentent le seul moyen de connaître la position du curseur graphique. Celle-ci sera donnée par:

```
PRINT PEEK(537);PEEK(538)
```

#21F (543)

Fonction: Cette variable prend la valeur 1 en mode HIRES, et Ø en modes TEXT et LORES.

D'autre part, toutes les coordonnées passées au Basic lors de l'exécution d'une commande graphique (abscisse-ordonnée, couleur, etc.) transitent par une zone de variables-systèmes débutant à **#2EØ** (736). Cette variable est nommée PARAMS. Il est ainsi possible d'effectuer des CALL vers des routines du Basic, après avoir placé les paramètres requis par la routine dans cette zone. A titre d'exemple, la fonction PAPER s'obtient en mettant l'argument (N° de couleur) dans PARAMS+1 (737), puis en exécutant un CALL à l'adresse #F2Ø4: POKE 737,3:CALL#F2Ø4 est équivalent à PAPER 3. Les adresses de ces routines sont répertorisées à l'Appendice 9 du Manuel Atmos.

4.3. Les commandes graphiques du Basic

Pour utiliser à fond toutes les possibilités de la haute résolution, il faut savoir que:

— Les coordonnées de l'écran vont de gauche à droite et de haut en bas, donc le point Ø,Ø se trouve dans le coin supérieur gauche.

— Des coordonnées hors limites (Ø-239 et Ø-199) fournies aux commandes graphiques provoqueront un message d'erreur.

— Si les couleurs de l'écran-texte ont été modifiées par INK et PAPER *avant* d'appeler la haute résolution, celles-ci resteront effectives pour les trois lignes de texte.

— Par contre, si INK et PAPER sont utilisés en haute-résolution, ils agiront sur l'écran graphique et non plus sur la fenêtre-texte.

— Toutes les commandes applicables au mode texte fonctionnent sur cette fenêtre, sauf PLOT et PRINT @ qui provoquent un message d'erreur.

— Par la suite, nous utiliserons constamment la notion de "curseur graphique" que nous appellerons simplement curseur. Celui-ci détermine l'endroit de l'écran où se fera le prochain tracé, mais n'est pas forcément visible comme le curseur texte. Quand on passe en mode haute-résolution, le curseur est placé aux coordonnées Ø,Ø (coin supérieur gauche).

— Les instructions graphiques, en dehors de leurs arguments spécifiques, utilisent toutes un paramètre définissant la "couleur" dans laquelle s'effectue le tracé que nous appellerons C, et qui peut prendre les valeurs suivantes:

Ø: Dessine avec la couleur du fond (donc efface les points déjà tracés).

1: Dessine dans la couleur définie pour le premier plan.

2: Inverse la couleur des points sur lesquels passe le curseur (Agit comme Ø sur un point déjà tracé et comme 1 s'il n'y a rien). Donc, ne perturbe pas l'affichage existant.

3: Ne fait rien (aucun point tracé ni effacé).

4.3.1. Instructions

Commandes d'intérêt général

HIRES

Syntaxe: HIRES

Application: Démarre le mode haute-résolution si on est en mode texte, et efface l'écran graphique si on est déjà dans ce mode. Précisons que ce dernier n'est pas affecté par CLS ou CTRL L, qui agiront sur les trois lignes de texte.

TEXT ou LORES N

Syntaxe: TEXT ou LORES N avec N=Ø, N=1

Application: Retourne au mode texte.

Commandes de dessin

CURSET

Syntaxe: CURSET <Abscisse>,<Ordonnée>,<C>

Application: Place le curseur au point spécifié. Cette commande est l'équivalent de PLOT ou PSET dans d'autres Basic.

Exemple: CURSET 12Ø,1ØØ,1 place le curseur au centre de l'écran.

CURMOV

Syntaxe: CURMOV <X>,<Y>,C

Application: Ajoute X et Y aux coordonnées courantes du curseur, et place celui-ci à la position obtenue (déplacement relatif).

DRAW

Syntaxe: DRAW <X>,<Y>,C

Application: Trace une droite entre la position du curseur et le point obtenu en y ajoutant X et Y (déplacement relatif).

REMARQUE: Si la deuxième extrémité de la droite se trouve plus haut ou à gauche de la position du curseur, X et Y peuvent bien sûr avoir une valeur négative.

CIRCLE

Syntaxe: CIRCLE <Rayon>,C

Application: Trace un cercle du rayon spécifié, centré sur la position du curseur.

REMARQUES:

— Vous devrez veiller à ce que le cercle ne dépasse pas les limites de l'écran, ce qui provoquerait une "ILLEGAL QUANTITY ERROR".

— Si vous utilisez la valeur 2 pour C, tous les points du cercle ne seront pas tracés: L'approximation de calcul ayant pour effet d'écrire deux fois de suite certains points, ceux-ci sont effacés au deuxième passage.

Exemple: Dessin d'un cercle plein (boule).

```
10 CURSET 120,100,3
20 FOR N=1 TO 50
30 CIRCLE N,1
40 NEXT N
```

CHAR

Syntaxe: CHAR <Code Ascii>,<Jeu>,C

Application: Trace à la position du curseur le caractère de code spécifié. Si <Jeu>=Ø, le caractère est pris dans le jeu standard et si <Jeu>=1 c'est le jeu semi-graphique qui est utilisé.

REMARQUES:

— La position du curseur définit le coin supérieur gauche de la matrice du caractère.

— Après écriture le curseur demeure au même endroit, ses déplacements doivent donc être entièrement gérés par l'utilisateur.

— Le seul moyen d'effacer un caractère est de le réécrire au même endroit, en donnant à C la valeur Ø ou 2.

Exemple:

Ce petit programme écrit le message contenu dans M$ au centre de l'écran:

```
10 HIRES:M$="ECRITURE HAUTE RESOLUTION"
20 CURSET (240-LEN(M$)*6)/2,90,3
30 FOR N=1 TO LEN(M$)
40 CHAR ASC(MID$(M$,N,1)),0,1
50 CURMOV 6,0,3
60 NEXT N
```

— La ligne 2Ø centre la phrase en fonction de sa longueur.

— La ligne 4Ø prend le code de chaque caractère extrait de M$, et le donne comme argument à CHAR.

— La ligne 5Ø fait avancer le curseur de 6 points (largeur d'un caractère) après chaque écriture.

Définition d'une zone

FILL

Syntaxe: FILL <hauteur>,<largeur>,<N>

Application: Remplit une zone dont le coin supérieur gauche est marqué par la position du curseur, de deux manières différentes selon la valeur donnée à N:

1) Si N est compris entre Ø et 31, il est considéré comme un attribut exactement comme pour PLOT (cf. 3.4); les mêmes valeurs sont d'ailleurs applicables. L'attribut est alors en vigueur sur un rectangle situé à droite de la zone ainsi créée, celle-ci étant le côté gauche du rectangle.
2) Si N est compris entre 32 et 127, il est considéré comme un motif correspondant à son équivalent binaire sur six Bits. Un des deux Bits 5 ou 6, respectivement de poids 32 et 64, doit obligatoirement être à 1.

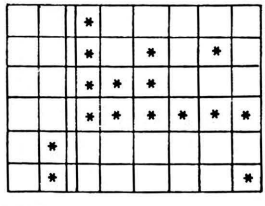
REMARQUES:

— La hauteur est définie en nombre de points, mais la largeur est définie en *nombre de cellules de 6 points*. Elle ne pourra donc avoir qu'une valeur comprise entre 1 et 4Ø (24Ø/6).

— Après exécution de FILL, le curseur est déplacé au coin inférieur gauche de la zone définie et ce indépendamment de sa largeur, comme si on avait exécuté CURMOV Ø,<hauteur>,C.

— Lorsque FILL définit un attribut, il est impossible de dessiner ensuite dans les cellules qui contiennent l'attribut.

Le tableau suivant montre la correspondance existant entre le tracé apparaissant effectivement à l'écran, et les valeurs décimale et binaire de N:

Cellule obtenue par FILL 1,1,<N>	*Valeur décimale de N*	*Équivalent binaire*
Bits ⟶ 7 6 **5 4 3 2 1 Ø**		
	32	ØØ **1ØØØØØ**
	42	ØØ **1Ø1Ø1Ø**
	56	ØØ **111ØØØ**
	63	ØØ **111111**
	64	Ø1 **ØØØØØØ**
	65	Ø1 **ØØØØØ1**

Les Bits marqués en **gras** sont ceux effectivement pris en compte pour l'élaboration du motif. Nous constatons que dans tous les cas, soit le Bit 5 soit le Bit 6 est à 1 : Comme il est impossible de donner à N une valeur inférieure à 32 qui le définirait comme un attribut, on met à 1 le Bit 6 qui n'entre pas dans le dessin de la cellule elle-même ; cela revient en fait à ajouter 64 à la valeur décimale obtenue par les 6 premiers Bits de l'octet.

Rappelons que le poids (ou valeur propre) d'un bit est trouvé en élevant 2 à la puissance de son N° d'ordre : Le Bit 7 vaut $2^7=128$, le Bit 5 vaut $2^5=32$, etc... (cf. § 2.1).

Exemples :

Programme créant une zone de 12Ø points de large par 1ØØ points de haut au centre de l'écran, où l'affichage se fait en jaune clignotant sur bleu :

```
10 HIRES
20 CURSET 60,50,3:FILL 100,1,20
30 CURSET 66,50,3:FILL 100,1,3
40 CURSET 180,50,3:FILL 100,1,7
50 CURSET 186,50,3:FILL 100,1,16
60 CURSET 120,100,3:FOR N=1 TO 30 STEP 3:CIRCLE N,1:NEXT
70 WAIT 40:CURSET 54,50,3:FILL 100,1,12
```

— Lignes 2Ø à 5Ø : Les attributs de couleur sont placés aux colonnes 6Ø et 66, puis remis aux valeurs d'origine aux colonnes 174 et 18Ø.

— Ligne 6Ø, création d'un dessin.

— Ligne 7Ø, l'attribut "clignotant" (12) est mis à la colonne 54.

Pour dessiner dans la zone précédente un motif "3 points 2 blancs un point", remplacer la ligne 6Ø par :

```
6Ø CURSET 8Ø,66,3 :FILL 68,15,57
```

Le dessin de valeur décimale 57 (111ØØ1) remplit une zone de 68 points de haut par 15*6=9Ø points de large.

Définition du trait

PATTERN

Syntaxe : PATTERN <N>

Application: Sélectionne le type de trait utilisé par DRAW et CIRCLE, qui est fonction de l'équivalent binaire de N sur 8 Bits. N est donc compris entre Ø et 255. La valeur à la mise sous tension ou à l'appel du mode graphique par HIRES est de 255 (trait plein).

REMARQUE: On trouve l'équivalent décimal du type de trait désiré exactement de la même façon que pour FILL, avec la différence qu'il est constitué de 8 bits au lieu de 6.

Notez aussi que N est remis à 255 chaque fois qu'un HIRES (effacement d'écran) est exécuté.

Exemple:

Le programme suivant vous permet d'expérimenter tous les types de traits possibles sur le dessin d'un rectangle. L'appui sur les flèches "haut" ou "bas" augmentera ou diminuera la valeur servant d'argument à PATTERN.

Exemple:

```
10 HIRES:PATTERN N
20 ?:?"MOTIF NO"N
30 CURSET  60,60,3:DRAW 120,0,1:DRAW 0,80,1:DRAW-120,0,1:DRAW
   0,-80,1
40 GET X$
50 IF X$=CHR$(10) AND N>0 THEN N=N-1
60 IF X$=CHR$(11) AND N<255 THEN N=N+1
70 GOTO 10
```

4.3.2. Fonction

POINT

Syntaxe: POINT(<abscisse>,<ordonnée>)

Application: Seule fonction spécifique du graphique, elle retourne 1 si le point aux coordonnées fournies est dans la couleur du 1er plan, et Ø s'il est dans la couleur du fond.

REMARQUE: N'oublions pas aussi les variables-système #219 et #21A, qui peuvent s'assimiler à une fonction dans la mesure où elles retournent respectivement l'abscisse et l'ordonnée du curseur graphique: PRINT PEEK(#219);PEEK(21A).

4.4. *Quelques programmes*

Déplacer le curseur et dessiner dans 8 directions

Ce programme autorise le déplacement du curseur dans les quatre directions principales avec les flèches, et dans les 4 diagonales par combinaison des flèches et de la touche SHIFT. On peut écrire ou effacer en pressant les touches "1" ou "Ø", et effacer l'écran avec DEL:

```
  5 '     PAPIER & CRAYON
  7 '
 10 HIRES:HIMEM 9000
 20 X=100:Y=100:T=0
 30 ?:?"TRAIT=0"
 40 CURSET XM,YM,T:CURSET X,Y,1
 50 X$=KEY$:IF X$="" THEN 50
 60 XM=X:YM=Y
 70 IF PEEK(521)=56 THEN 130
 80 IF X$=CHR$(8) AND X>0 AND Y>0 THEN X=X-1:Y=Y-1
 90 IF X$=CHR$(10) AND X>0 AND Y<199 THEN X=X-1:Y=Y+1
100 IF X$=CHR$(11) AND X<239 AND Y>0 THEN X=X+1:Y=Y-1
110 IF X$=CHR$(9) AND X<239 AND Y<199 THEN X=X+1:Y=Y+1
120 GOTO 40
130 IF X$=CHR$(8) AND X>0 THEN X=X-1
140 IF X$=CHR$(10) AND Y<199 THEN Y=Y+1
150 IF X$=CHR$(11) AND Y>0 THEN Y=Y-1
160 IF X$=CHR$(9) AND X<239 THEN X=X+1
170 IF X$="0" OR X$="1" THEN T=VAL(X$):CLS:?:?"TRAIT="T
180 IF X$=CHR$(127) THEN RUN
190 GOTO 40
```

Explications du programme:

2Ø : X et Y sont les coordonnées du curseur, T est le paramètre de couleur (1=écrit, Ø=efface).

4Ø : Cette ligne déplace le curseur, tout en l'effaçant le cas échéant de son ancienne position si T=Ø.

5Ø : Mise en mémoire dans XM et YM des valeurs de X et Y, avant leur modification éventuelle.

7Ø : Teste si la touche SHIFT est appuyée (pour l'explication de cette variable-système, voir § 2.3.3).

8Ø-11Ø: Gestion des déplacements en diagonale; incrémente ou décrémente X et Y en fonction de la touche appuyée en même temps que SHIFT, et en veillant à ne pas dépasser les limites de l'écran.

13Ø-16Ø: Gère les déplacements simples (haut, bas, droite, gauche).

17Ø : Change le mode curseur (écrit-efface), et l'affiche dans la fenêtre-texte.

18Ø : Touche DEL, effacement écran.

Dessiner un rectangle

```
10 HIRES:C1=100:C2=50:X=60:Y=60
20 CURSET X,Y,3
30 DRAW C1,0,2:DRAW 0,C2,2:DRAW-C1,0,2:DRAW 0,-C2,2
```

1Ø : Définition de la taille des côtés (C1 et C2) et de la position du coin supérieur gauche.

Dessiner un rectangle plein

```
10 HIRES:C1=100:C2=50:X=60:Y=60
20 CURSET X,Y,3:FOR N=1 TO C2
30 DRAW C1,0,2:CURMOV-C1,1,3
40 NEXT
```

Faites tourner ce programme, et ajoutez ces deux lignes:

```
15 CURSET 100,100,3:CIRCLE 50,1
50 IF X=40 THEN END ELSE X=40:Y=80:GOTO 20
```

Cela vous donne un aperçu des possibilités du graphique!

Dessiner une ellipse

```
5 '      ELLIPSE
7 '
10 INPUT"Rayons horizontal, vertical";R1,R2
20 INPUT"Coordonnées du centre (X,Y)";X,Y
30 HIRES
40 S=PI/(R1+R2)/1.8
50 FOR I=0 TO 2*PI STEP S
60 X1=SIN(I)*R1+X
70 Y1=COS(I)*R2+Y
80 CURSET X,Y,1
90 NEXT I
```

4Ø : Définit le pas séparant chaque point tracé. Si vous voulez un tracé plus fin, augmentez la valeur 1.8, sinon la diminuer.

6Ø-7Ø: Calcul de l'abscisse et de l'ordonnée de chaque point de l'ellipse, I étant exprimé en radians. Les deux rayons fournissent un coefficient différent pour chaque coordonnée, déterminant l'aspect elliptique.

"END"

Ce petit programme amusant dessine le mot "END" en caractères énormes, et en utilisant à fond la commande DRAW:

```
 10 '      PROGRAMME END
 20 '
100 HIRES
110 CURSET 16,170,3
115 '    "E"
120 DRAW 0,-120,1:DRAW 60,0,1:DRAW 0,16,1:DRAW-44,0,1:DRAW
    0,36,1:DRAW 30,0,1
130 DRAW 0,16,1:DRAW-30,0,1:DRAW 0,36,1:DRAW 44,0,1:DRAW
    0,16,1:DRAW-60,0,1
150 '    "N"
160 CURMOV 75,0,3:DRAW 0,-120,1:DRAW 16,0,1:DRAW 28,80,1:DRAW
    0,-80,1:DRAW 16,0,1
170 DRAW 0,120,1:DRAW-16,0,1:DRAW-28,-80,1:DRAW 0,80,1:DRAW -
    16,0,1
190 '    "D"
200 CURMOV 75,0,1
210 DRAW 0,-120,1:DRAW 45,0,1:DRAW 10,5,1:DRAW 5,10,1:DRAW
    0,90,1:DRAW-5,10,1
220 DRAW-10,5,1:DRAW-45,0,1:CURMOV 16,-16,3:DRAW 0,-88,1:DRAW
    24,0,1:DRAW 5,2,1
230 DRAW 2,5,1:DRAW 0,74,1:DRAW-2,5,1:DRAW-5,2,1:DRAW-24,0,1
```

Aucune explication n'est nécessaire, nous remarquerons simplement que les arrondis de la lettre "D" sont obtenus par approximation, en traçant des petits segments de droite.

La Hardcopy graphique

Nous avons déjà proposé un programme effectuant la copie de l'écran-texte sur imprimante, au § 3.3.1. Cette fois-ci il s'agit d'obtenir une copie du graphique, opération légèrement plus complexe. La méthode de mémorisation de l'écran haute-résolution ayant été expliquée au § 4.2, nous ne

reviendrons pas dessus. Il nous suffira de savoir que la plupart des imprimantes utilisent le même procédé pour l'impression point par point, à savoir qu'un octet envoyé à l'imprimante code toujours un groupe de points.

Deux difficultés restent cependant à résoudre :

1) Sur l'écran, les octets sont disposés "horizontalement", alors que sur l'imprimante ils le sont "verticalement", le bit de poids faible étant en bas. La solution est assez simple, il suffira d'imprimer le graphique "couché", c'est-à-dire incliné de 90°. L'écran sera lu de haut en bas mais de droite à gauche, en commençant par le coin supérieur droit.

2) Les imprimantes utilisent les 8 bits de l'octet pour le codage, l'Atmos seulement 6. A chaque ligne imprimée (en fait, une colonne de l'écran), le saut de ligne sur l'imprimante devra correspondre à six points au lieu de 8.

Le programme a été écrit pour une imprimante EPSON 8Ø série III, voici un résumé des différents séquences de codes qui lui sont propres et que vous pourrez éventuellement adapter pour votre machine (ESC est le caractère ESCAPE ou CHR$(27) :

— ESC;"A";CHR$(N) : Définit un espacement de N points entre deux lignes.

— ESC;"D";CHR$(N);CHR$(Ø) : N est la colonne où commence l'impression. Cette commande sert à centrer le dessin, et n'est pas indispensable.

— CHR$(9) : Effectue le centrage à la position définie par la commande précédente, et peut également être omis.

— ESC;"K";CHR$(N);CHR$(Ø) : "Escape K" passe en mode d'impression point par point. Le nombre N renseigne l'imprimante sur le nombre de points à imprimer sur une ligne (ici 2ØØ, puisque l'écran est copié de haut en bas).

— CHR$(1Ø) : Code de passage à la ligne.

```
 5 '       HARDCOPY GRAPHIQUE
 7 '
10 E$=CHR$(27)
20 LPRINT E$;"A";CHR$(6);
30 LPRINT E$;"D";CHR$(23);CHR$(0);
40 FOR I=#A000 TO #A027
50 LPRINT CHR$(9);E$;"K";CHR$(200);CHR$(0);
```

```
 60 FOR J=7960 TO 0 STEP -40
 70 DA=PEEK(I+J)-64:IF DA<0 THEN DA=0
 80 PRINT
 90 LPRINT CHR$(DA);
100 NEXT J
110 LPRINT CHR$(10);
120 NEXT I
```

Explications:

1Ø : Le caractère Escape est attribué à la variable E$.

4Ø : Démarrage d'une boucle de 4Ø pas (une par ligne).

5Ø : Initialisation de l'imprimante avant lecture-impression de la ligne.

6Ø : Démarrage d'une boucle de 24Ø pas, nombre d'octets "verticaux" sur chaque ligne.

7Ø : Lecture de l'adresse définie par I+J et chargée dans DA, qui est mis à Ø s'il s'agit d'un attribut (non imprimable).

8Ø : Le PRINT est indispensable pour le bon fonctionnement du programme.

9Ø-12Ø: Envoi de l'octet lu à l'imprimante, saut de ligne à la fin de la boucle interne et suite de la boucle externe.

4.5. *Dessin assisté par ordinateur (D.A.O.): le programme MULTIGRAF*

Ce programme, utilisant à fond les diverses commandes graphiques de l'Atmos, vous permettra de vous servir de l'écran comme d'une tablette de dessin. Le tracé de droites et de diverses figures géométriques y est simplifié, ainsi que l'usage des divers attributs disponibles (couleur, etc.).

Son principe d'utilisation est très simple:

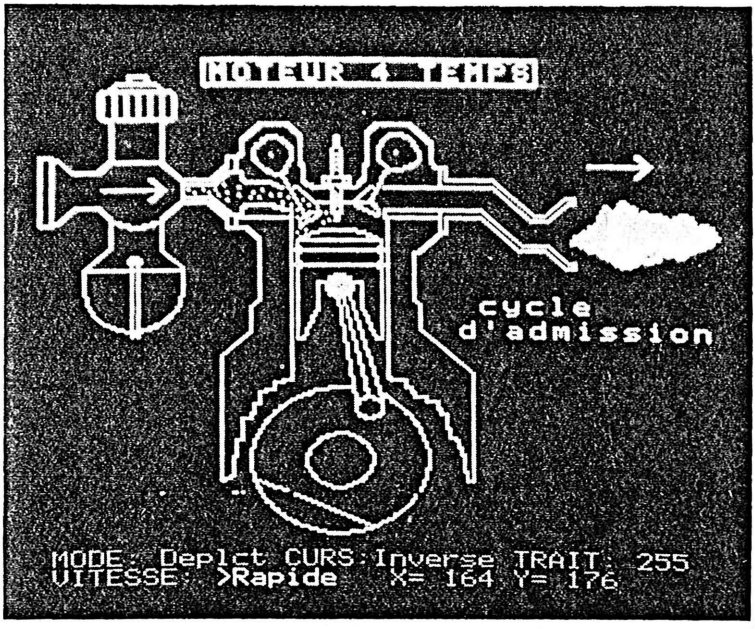
— Les déplacements élémentaires du curseur, haut bas et droite gauche, sont obtenus avec les quatre flèches.

— Les déplacements en diagonale, en pressant la touche SHIFT en même temps que l'une des quatre flèches.

— Deux vitesses de déplacement du curseur sont possibles, la vitesse lente étant point par point et la rapide par dix points.

— Le curseur peut écrire, effacer ou ne pas modifier les points sur lesquels il passe.

— Plusieurs commandes utilisent deux paramètres: Pour une droite par exemple, il faut "pointer" les deux extrémités. En fait, il suffit de placer le curseur à une des deux extrémités et de presser la lettre correspondant à la commande; ensuite, on déplace le curseur à la deuxième extrémité et on rappuie sur la lettre-clé ce qui a pour effet d'exécuter la fonction demandée.



Un dessin réalisé avec MULTIGRAPH

Commandes disponibles

→	Curseur à droite.
←	Curseur à gauche.
↑	Curseur vers le haut.
↓	Curseur vers le bas.
→ + **SHIFT**	Curseur à droite vers le bas.
← + **SHIFT**	Curseur à gauche vers le haut.
+ **SHIFT**	Curseur à droite vers le haut.
+ **SHIFT**	Curseur à gauche vers le bas.
<	Sélectionne la vitesse de déplacement lente du curseur.
>	Sélectionne la vitesse rapide.
Ø,1,2	Choix du mode curseur, Ø=Efface 1=Écrit et 2=Inverse.
<**A**>ide	Affiche la liste des commandes acceptées par le programme.
<**D**>roite	Trace une droite entre les deux points marqués par le curseur.
<**R**>ectangle	Dessine un rectangle dont les angles opposés sont les points r-marqués par le curseur.
<**C**>ercle	Dessine un cercle, dont le premier point marqué est le centre et le deuxième un point quelconque de la périphérie.
<**B**>oule	Trace un cercle plein, avec les mêmes paramètres que pour un cercle normal.
<**Z**>one	Le premier appui sur "Z" marque un angle; le deuxième marque l'angle opposé, et le programme vous demande quel attribut vous désirez affecter à cette zone (couleur, motif, etc... voir § 4.3.1).
<**T**>rait	Choix du type de trait utilisé pour les dessins. Voir instruction PATTERN, § 4.3.1.
<**E**>crit	Autorise l'écriture de texte à l'écran. Après chaque caractère tapé le curseur avance automatiquement de six points, se mettant ainsi à la position correcte pour le caractère suivant. Mode annulé par la touche "DEL".
<**P**>rint	Effectue une copie de l'écran sur imprimante EPSON 8Ø série III (Hardcopy).

ESCAPE Donne accès à quatre utilitaires :

<**C**>ouleur : Modification des couleurs du fond et du tracé.
<**E**>ffacer l'écran.
<**S**>ave : Sauvegarde de l'écran sur cassette.
<**L**>oad : Chargement d'un écran se trouvant sur cassette.
<**Q**>uitter le programme.

De plus, les trois lignes de texte en bas d'écran affichent la commande en cours d'exécution, le mode curseur, le type de trait, la vitesse sélectionnée et les coordonnées du curseur.

Ce que nous avons dit sur les nombreuses REM pour les programmes précédents reste valable, à savoir que vous pouvez les éliminer mais surtout gardez les numéros de ligne multiples de 1ØØ à cause des GOTO et GOSUB qui y font référence.

```
1 ' ********************************
2 ' *       MULTIGRAF V2.15        *
3 ' *                              *
4 ' * Dessin Assiste par Ordinateur *
5 ' * Pour ORIC ATMOS  48K         *
6 ' ********************************
7 '
8 '
9 '
100 CLS:HIMEM38000:POKE775,32:INK0
110 GOSUB8000:PAPER0:INK2:E$=CHR$(27)
130 HIRES
140 GOSUB7000:GOSUB6200
150 CURSETX,Y,1
998 '
1000 '  BOUCLE PRINCIPALE DE SAISIE
1002 '
1100 GET X$:AX=ASC(X$)
1170 IFAX>7ANDAX<12THENGOSUB2000:GOTO1100  ' Mouvements curseur
1175 '
1052 IFAX>47ANDAX<51THEN T=VAL(X$):GOTO1500' Chgmt Mode curseur
1185 '
1190 IFX$="T"THENGOSUB4500' Pattern
1200 IFAX=27THEN1600       ' Utilitaires
1210 IFX$="R"THENGOSUB3000' Rectangle
1220 IFX$="D"THENGOSUB3200' Droite
1230 IFX$="Z"THENGOSUB3400' Zone-Fill
1240 IFAX=127ANDZ=-1THENCURSETX,Y,2:X=XM:Y=YM:GOSUB6400 ' Annulation
1250 IFX$="E"THENGOSUB4000' Texte
1260 IFX$="C"THENGOSUB3600' Cercle
1270 IFX$="B"THENGOSUB3800' Boule
1280 IFX$="A"THENGOSUB6000' Aide
1281 '
1285 '  Vitesse deplacements
1287 '
1290 IFX$=","THENV$="<Lente ":V=1:POKE775,35
```

```
1300 IFX$="."THENV$=">Rapide":V=10:POKE775,32
1310 IFX$="P"THENGOSUB5000
1500 GOSUB6200:GOTO1100
1598 '
1600 '  UTILITAIRES
1602 '
1610 CLS
1620 PRINTE$"S"E$"E<E>ffacer l'ecran <Q>uitter le prog."
1625 PRINTE$"S"E$"E<S>ave ou <L>oad un ecran, <C>ouleur"
1630 PRINTE$"S"E$"Eou bien n'importe quelle touche.";
1640 GETX$
1650 IFX$="E"THENHIRES:CURSETX,Y,T:PRINTCHR$(17);:GOTO1500
1148 IFX$="Q"THEN TEXT:END
1655 IFX$="C"THEN 1900
1660 IFX$="Q"THEN TEXT:END
1670 IFX$="S"THEN1700
1680 IFX$="L"THEN1800
1690 GOTO1500
1698 '
1700 '  Sauvegarde ecran
1702 '
1710 CLS:POKE618,7
1720 INPUT"NOM DE L'ECRAN ";NN$
1730 PRINT"Lancez le K7 et Pressez une touche"
1740 GETX$
1750 CSAVENN$,A#A000,E#BF3F
1760 GOTO1500
1798 '
1800 '  Chargememt ecran
1802 '
1810 CLS:POKE618,7
1820 INPUT"NOM DE L'ECRAN ";NN$
1850 CLOADNN$
1860 GOTO1500
1898 '
1900 '  Changement couleur ecran/aff
1902 '
1905 CLS:POKE618,3:INPUT"COULEUR DU FOND (0-7) ";CR:PAPERCR
1910 INPUT"COULEUR AFFICHAGE (0-7) ";CR:INKCR
1920 GOTO1500
1998 '
2000 '  Mouvements curseur
2002 '
2003 '  Haut/Bas/Droite/Gauche
2004 '
2010 CURSETX,Y,T
2015 IF PEEK(521)<>56THEN2100
2020 IFAX<>8THEN2030
2022 IFX-V>=0THENX=(X-V)ELSEX=MX
2024 GOTO2060
2030 IFAX<>9THEN2040
2032 IFX+V<MXTHENX=(X+V)ELSEX=0
2034 GOTO2060
2040 IFAX<>10THEN2050
2042 IFY+V<MYTHENY=(Y+V)ELSEY=0
2044 GOTO2060
2050 IFY-V>=0THENY=(Y-V)ELSEY=MY
2060 CURSETX,Y,2
2070 GOSUB6200:RETURN
2098 '
2100 '  Diagonales
2102 '
```

```
2110 IFAX<>8THEN2120
2112 IFX-V>=0THENX=(X-V)ELSEX=MX
2114 IFY-V>=0THENY=(Y-V)ELSEY=MY
2116 GOTO2060
2120 IFAX<>9THEN2130
2122 IFX+V<MXTHENX=(X+V)ELSEX=0
2124 IFY+V<MYTHENY=(Y+V)ELSEY=0
2126 GOTO2060
2130 IFAX<>10THEN2140
2132 IFX-V>=0THENX=(X-V)ELSEX=MX
2134 IFY+V<MYTHENY=(Y+V)ELSEY=0
2136 GOTO2060
2140 IFX+V<MXTHENX=(X+V)ELSEX=0
2142 IFY-V>=0THENY=(Y-V)ELSEY=MY
2144 GOTO2060
2998 '
3000 '  Rectangle
3002 '
3010 GOSUB 6400
3020 IFZ=-1THENXM=X:YM=Y:M$="LRectgle":RETURN
3030 IFT=2THENCURSETXM,YM,2:CURSETX,Y,2
3040 DRAWXM-X,0,TM:DRAW0,YM-Y,TM:DRAWX-XM,0,TM:DRAW0,Y-YM,TM
3050 IFT=1THENRETURN
3060 X=X+SGN(XM-X):Y=Y+SGN(YM-Y):CURSETX,Y,1
3070 RETURN
3198 '
3200 '  Droite
3202 '
3210 GOSUB 6400
3220 IFZ=-1THENXM=X:YM=Y:M$="LDroite":RETURN
3240 DRAWXM-X,YM-Y,TM
3250 RETURN
3398 '
3400 '  Zones
3402 '
3410 GOSUB6400
3420 IFZ=-1THENXM=X:YM=Y:M$="LZone":RETURN
3430 CURSETX,Y,2:CURSETXM,YM,2
3440 RX=INT(ABS((XM-X)/6)):IFXM<XTHENX=XM
3445 IFRX<1THENRX=1
3450 RY=ABS(YM-Y):IFYM<YTHENY=YM
3455 IFRY=0THENRY=1
3460 CURSETX,Y,2
3500 CLS:POKE618,7
3510 PRINT"0-7:Coul.Aff 12:Flash 16-23:Coul.Fond"
3520 PRINT"32-127:Dessin de la Zone"
3530 INPUT"VOTRE CHOIX ";FI
3540 IF FI>7ANDFI<12ORFI>12ANDFI<16ORFI>23ANDFI<32ORFI>127THEN3500
3550 FILLRY,RX,FI
3560 RETURN
3598 '
3600 '  Cercle
3602 '
3610 GOSUB6400
3620 IFZ=-1THENXM=X:YM=Y:M$="LCercle":RETURN
3640 RD=SQR((XM-X)^2+(YM-Y)^2)
3650 X=XM:Y=YM:CURSETX,Y,1
3660 CIRCLERD,TM
3670 RETURN
3798 '
3800 '  Boule
3802 '
```

```
3810 GOSUB6400
3820 IFZ=-1THENXM=X:YM=Y:M$="LBoule":RETURN
3840 RD=SQR((XM-X)^2+(YM-Y)^2)
3850 X=XM:Y=YM:CURSETX,Y,1
3860 FORN=2TORD
3870 CIRCLEN,TM
3880 NEXTN
3890 RETURN
3998 '
4000 '  Ecrire a l'ecran
4002 '
4010 CLS:PRINT:M$="LTexte"
4020 T=2:V=6:MX=230:MY=190:V$="Rapide"
4030 GOSUB6200
4040 X$=KEY$:IFX$=""THEN4040ELSEAX=ASC(X$)
4050 IFAX=127THENRESTORE:GOSUB7000:POKE#20C,255:RETURN
4060 IFAX=20THENPRINTCHR$(20);:GOTO4040
4070 IFAX<32THENGOSUB2000:GOTO4040
4090 CHARAX,0,T
4100 AX=9:GOSUB2000
4110 GOSUB6200
4200 GOTO4040
4498 '
4500 '  Choix du trait
4502 '
4510 CLS:POKE618,7
4520 PRINT"CHOIX DU TYPE DE TRAIT-"
4530 PRINT"VALEURS AUTORISEES: 0-255"
4540 INPUT"Votre choix:";P
4550 IFP<0ORP>255THEN4510
4560 PATTERNP:RETURN
4999 '
5000 '  HARDCOPY
5002 '
5010 CLS
5020 PRINT"ALLUMEZ L'IMPRIMANTE"
5030 PRINT"<R>etour au pgm, ou pressez une touche";
5040 GETX$:IF X$="R"THEN RETURN
5100 LPRINT E$"A"CHR$(6);
5110 LPRINT E$"D"CHR$(23)CHR$(0);
5120 FOR I=#A000 TO #A027
5130 LPRINT CHR$(9)E$"K"CHR$(200)CHR$(0);
5140 FOR J=7960 TO 0 STEP-40
5150 DA=PEEK(I+J)-64:IFDA<0THENDA=0
5160 PRINT ,
5170 LPRINT CHR$(DA);
5180 NEXTJ
5190 LPRINT CHR$(10);
5200 NEXTI
5210
5997 '
5998 '  MESSAGES EN BAS D'ECRAN
5999 '
6000 '  Aide
6002 '
6005 CLS
6007 PRINTCHR$(29)
6010 PRINTE$"R"E$"@<R>ectangle <D>roite <C>ercle <B>oule"
6020 PRINTE$"R"E$"@<Z>one <T>rait <A>ide <E>crit <P>rint"
6030 PRINTE$"R"E$"@<DEL>:Anul <ESC>:Utilit 0,1,2:Curseur";
6050 GETX$:PRINTCHR$(29)
6060 RETURN
6198 '
```

```
6200 '  Statut
6202 '
6205 IF PEEK(526)<10 THEN RETURN
6210 CLS:POKE618,6
6220 PRINT"MODE:"E$M$E$"HCURS:"T$(T)" TRAIT:"P
6230 PRINT"VITESSE:"E$"E"V$"  "E$"AX="X"Y="Y;
6240 RETURN
6398 '
6400 '  Memorisation coordonnees
6402 '
6410 Z=NOTZ
6420 IFZ=-1THENXM=X:YM=Y:TM=T:T=2:CURSETXM,YM,2:RETURN
6430 T=TM:M$="BDeplct"
6440 RETURN
6998 '
7000 '  INITIALISATIONS
7002 '
7010 READ X,Y,T,P,V,MX,MY,A(1)
7020 READ M$
7030 READT$(0),T$(1),T$(2)
7040 READV$
7200 RETURN
7498 '
7500 '  Donnees de depart
7502 '
7510 DATA 100,100,2,255,10,239,199,1
7520 DATA BDeplct
7530 DATA Efface,Ecrit,Inverse
7540 DATA >Rapide
7997 '
7998 '---------FIN DU PROG----------
7999 '
8000 '  **  INTRODUCTION  **
8002 '
8010 E1$=CHR$(27):E$=" "+E1$:PAPER4
8100 PRINTCHR$(17)CHR$(4):DOKE#BBD0,2578:DOKE#BBF8,2578
8110 PRINTE$"A  MULTIGRAF Programme de D.A.O"
8120 PRINTCHR$(4):PRINT
8130 PRINT" Ce programme vous permettra d'exploi-";
8140 PRINT"ter les possibilites graphiques de"
8150 PRINT"l'ATMOS avec le maximum de confort.":PRINT
8160 PRINT"DEPLACEMENTS CURSEUR:"
8170 PRINT"4 fleches: Horizontal/Vertical"
8180 PRINT"4 fleches+SHIFT:Diagonales":PRINT
8190 PRINT"VITESSE DEPLCT:":PRINT"< Lente, point par point"
8200 PRINT"> Rapide, 10 points par 10 points":PRINT
8210 PRINT"Les autres commandes s'obtiennent par"
8220 PRINT"leur initiale:<D>roite,<C>ercle,etc."
8230 PRINT"<A>ide donne la liste des commandes"
8240 PRINT"acceptees.":PRINT
8250 PRINT"L'ecran affiche la commande en cours"
8260 PRINT"d'execution, le mode curseur (ecrit,"
8270 PRINT"efface,inverse) et sa position."
8280 PRINT"Annulation d'une commande: DEL":PRINT
8300 DOKE49080,275:PRINTE1$"L   <D>ETAILS OU <P>ROGRAMME";
8310 GETX$:IFX$="P"THENPRINTCHR$(17):CLS:RETURN
8320 IFX$<>"D"THEN8310
8400 CLS:POKE48080,22
8410 PRINT:PRINTE$"A     EXPLICATIONS DETAILLEES":PRINT
8420 PRINT" Commandes necessitant 2 parametres:":PRINT
8430 PRINT"-Un premier appui sur la lettre-cle"
8440 PRINT"marque la 1ere position,materialisee"
```

```
8450 PRINT"par un point.Ensuite, on deplace le"
8460 PRINT"curseur a la 2eme position,et on tape"
8470 PRINT"a nouveau la lettre-cle.":PRINT
8480 PRINT"EX: Les 2 extremites d'une droite,les"
8490 PRINT"diagonales d'un rectangle,le centre et";
8500 PRINT"un point du perimetre pour les cercles";
8510 PRINT"et les boules.":PRINT
8520 PRINT" Commandes n'exigeant qu'un parametre:":PRINT
8530 PRINT"-Celles-ci sont executees immediate-"
8540 PRINT"ment, l'ecran vous renseigne sur les"
8550 PRINT"valeurs autorisees.":PRINT
8560 PRINT"La commande <P>rint recopie l'ecran"
8570 PRINT"sur une imprimante EPSON serie 80.":PRINT
8580 DOKE49040,278:PRINTE1$"L POUR COMMENCER PRESSER UNE TOUCHE'
8590 GETX$:PRINTCHR$(17);:RETURN
9000 CLS
9010 PRINTPEEK(526);
9020 WAIT 10:GOTO9010
49908 '
```

Explications du programme

1ØØ : POKE 775,32 choisit une vitesse d'autorepeat plus rapide que la normale, afin d'accélérer les mouvements curseur.

11Ø : Si vous voulez éviter de taper les lignes 8ØØØ-fin, qui ne contiennent que des explications, il faudra aussi supprimer le GOSUB 8ØØØ de cette ligne.

1ØØØ-128Ø: Boucle saisissant la touche frappée (X$), et effectuant le branchement vers les divers sous-programmes concernés. AX est le code ASCII de X$ et T sa valeur s'il s'agit d'un nombre ; XM et YM mémorisent les coordonnées du curseur, avant son déplacement. Z est un pointeur valant 1 en temps normal et −1 après pointage des premières coordonnées d'une figure, stockées dans XM et YM.

129Ø : Modification de la vitesse de déplacement (variable V, valant 1 ou 1Ø).

16ØØ-186Ø: Utilitaires; Effacement de l'écran, sauvegarde et relecture cassette. NN$ est le nom de l'écran à sauvegarder ou recharger.

2ØØØ-2Ø7Ø: Gestion des mouvements simples du curseur, avec vérification de sa position: s'il dépasse les limites de l'écran, il repasse de l'autre côté de celui-ci. La variable T détermine si le curseur doit être effacé ou non de son ancienne position.

21ØØ-2144 : Gestion des mouvements curseur complexes (diagonales). Identique à ci-dessus, avec la différence que chaque fois X et Y sont testées et incrémentées ou décrémentées ensemble.

3ØØØ-3Ø7Ø : Sous-programme dessinant un rectangle, dont les angles opposés ont les coordonnées XM, YM et X,Y. Le drapeau Z détermine si on pointe le premier ou le deuxième lot de coordonnées. TM stocke l'ancienne valeur de T.

32ØØ-325Ø : Trace une droite entre XM, YM et X,Y.

34ØØ-356Ø : Définition d'une zone. Comme la coordonnée horizontale de celle-ci est en nombre de cellules et non pas de points, la ligne 344Ø opère la conversion et RX contient la valeur trouvée. Au deuxième passage dans le sous-programme, la valeur de l'attribut ou motif à y placer est demandée et un petit mémo des principales valeurs affiché en bas d'écran.

36ØØ-367Ø : Trace un cercle ; son rayon (RD) est calculé à la ligne 364Ø à l'aide du célèbre théorème de pythagore (Le carré de l'hypoténuse, ici le rayon, est égal à la somme des carrés des deux côtés opposés XM-X et YM-Y).

38ØØ-389Ø : Dessine une boule, en traçant tous les cercles dont le rayon est compris entre 1 et le rayon demandé.

NOTE : Si une cercle ou une boule déborde de l'écran, une erreur sera provoquée et le programme s'arrêtera. Il suffira alors de le relancer par GOTO 1ØØØ.

4ØØØ-42ØØ : Gestion de l'écriture de texte. Le mode curseur inverse est imposé, car la seule manière d'effacer un caractère est de réécrire par-dessus dans ce mode. Il en va de même pour le nombre de points sautés en déplacement, qui est de six afin de respecter l'espacement entre caractères. A la ligne 4Ø5Ø, l'instruction POKE≠2ØC,255 force les majuscules si on quitte le mode afin de permettre la saisie des caractères de commande.

45ØØ-452Ø : Sélection du trait utilisé pour les tracés (variable P, argument d'une instruction PATTERN).

5ØØØ-521Ø : Copie l'écran sur une imprimante EPSON. Ce programme est présenté avec des explications détaillées au § 4.4.

6ØØØ-6Ø6Ø : Affichage du petit texte d'aide, résumant les commandes acceptées.

62ØØ-624Ø : Affichage des paramètres en vigueur, dans les trois lignes en bas d'écran. M$ est la commande en cours d'exécution, ou alors "déplacement". Le tableau T$( ) est "Ecrit", "Efface" ou "Inverse" en fonction de la valeur de T. V$ est la vitesse de déplacement, puis viennent X et Y. Ce sous-programme est très fréquemment appelé, en fait à chaque passage dans la boucle principale. L'accès en est interdit uniquement en autorepeat, pour ne pas ralentir le curseur.

64ØØ-644Ø : Autre routine appelée par chaque commande, qui charge XM et YM au pointage de la première paire de coordonnées et gère le drapeau Z.

7ØØØ-754Ø : Attribue leur valeur initiale à toutes les variables du programme. Au départ, le curseur est positionné à X=1ØØ, Y=1ØØ en mode inverse et vitesse rapide. MX et MY sont les valeurs maximales permises pour X et Y, respectivement 239 et 199.

8ØØØ-Fin : Introduction fournissant toutes les explications nécessaires à l'utilisation immédiate de MULTIGRAF, mais que vous pouvez très bien omettre si les travaux de frappe vous rebutent. Dans ce cas, ne manquez pas de corriger également la ligne 11Ø qui appelle ce sous-programme.

Précisons pour terminer que le programme occupe environ 6,3 K. en mémoire tel qu'il est présenté, valeur qui passe à 5,5 K. sans les REM et à 3,9 K. (!) si le texte d'introduction est également supprimé.

5

Le son Atmos

Muni d'une "puce" spécialisée chargée de la gestion du son, l'Atmos présente de ce fait des dispositions très au-dessus de la moyenne en ce domaine. Outre un certain nombre de sons préprogrammés tout indiqués pour les jeux, des notes ou du bruit blanc peuvent être générés sur trois canaux différents à l'aide d'instructions prévues à cet effet.

Ces trois canaux peuvent produire du son de fréquence déterminée, ou alors des notes de la gamme dont la hauteur et l'enveloppe sont programmables. Le bruit blanc, s'il est utilisé, sera mélangé à une des trois sorties.

5.1. Principes de fonctionnement

Pour bien comprendre comment exploiter le son de l'Atmos, il faut savoir qu'en fait les instructions spécialisées MUSIC et SOUND produisent un son conditionné, dans une certaine mesure, par la nature du son prépro-

grammé les ayant précédé : Ainsi, il sera impossible d'obtenir une note pure après un EXPLODE (qui ne contient quasiment que du bruit blanc). D'une manière générale :

— Si le bruit blanc est utilisé sur un canal il faudra ensuite l'"éliminer" par un son prédéfini qui n'en contient pas, comme par exemple la répétition sonore du clavier ou CTRL G.

— Il faudra tenir compte du fait qu'on n'aura pas nécessairement le même résultat en exécutant deux fois de suite la même instruction à des endroits différents d'un programme, certaines modifications apportées au son de départ restant actives après changement des paramètres qui les ont créées.

— Les sons continus durent indéfiniment sauf avis contraire. Ils seront interrompus, ou bien par l'émission d'un autre son quel qu'il soit, ou bien par l'instruction PLAY Ø,Ø,Ø,Ø.

— Enfin, vous pourrez jouir à loisir de vos créations sonores en connectant l'Atmos à votre chaîne Hi-Fi, par l'intermédiaire de la prise DIN 7 broches du magnétocassette. Le son est disponibles entre la broche 5 et la masse (2), ou la broche 4 et la masse, avec un niveau de 4Ø millivolts environ au volume maximum.

5.2. Commandes disponibles

Les Sons pré-programmés

PING	Son cristallin, identique à la sonnerie clavier (CTRL G ou ou CHR$(7)).
ZAP	Sifflement du type "canon laser".
SHOOT	Bruit simulant un coup de feu — contient du bruit blanc.
EXPLODE	Bruit d'explosion — contient du bruit blanc.

Et bien sûr, le "clic" produit par le clavier pour lequel il n'existe pas d'instruction, et qu'on obtient par un CALL (il se simule également très bien avec l'instruction PLAY, voir plus loin).

SOUND

Syntaxe: SOUND <canal>,<N>,<volume>

Application: Génère une fréquence continue avec les paramètres suivants:

Canal : 1, 2 ou 3 produira un son pur seul sur le canal spécifié;
4, 5 ou 6 mélangera un bruit avec le son d'un des 3 canaux (4 avec le canal 1, 5 avec le canal 2, etc...).

N : Détermine la hauteur du son, qui peut être trouvée par la formule suivante:

N=62ØØØ/<fréquence>

Cette équation est valable pour des valeurs de N allant de 1 (62 Khz) à 4ØØØ environ (15 Hz).

Volume : Modifiable par pas fixes, de 1 à 15. La valeur Ø donne le contrôle à l'instruction PLAY, définissant l'enveloppe du son (voir ci-après).

Exemple: SOUND 1,62,9 produira un son continu à 1ØØØ Hertz.

MUSIC

Syntaxe: MUSIC <Canal>,<Octave>,<Note>,<Volume>

Application: Génère une note répondant aux paramètres fournis qui sont:

Canal : Ne peut prendre que les valeurs 1,2 ou 3.

Octave : Entier compris entre Ø (octave la plus basse) et 6 (octave la plus haute).

Note : Entier compris entre 1 et 12, correspondant aux notes suivantes:

1 — DO	7 — FA#
2 — DO#	8 — SOL
3 — RE	9 — SOL#
4 — RE#	1Ø — LA
5 — MI	11 — LA#
6 — FA	12 — SI

Volume: Choix du niveau, comme pour SOUND.

PLAY

Syntaxe: PLAY <Son>,<Bruit>,<Enveloppe>,<Durée>

Application: PLAY *ne produit pas* de son, mais modifie un son prédéfini ou généré par SOUND et MUSIC, lorsque le volume est mis à Ø dans ces instructions.

Son : Sélectionne des canaux qui émettront un son pur, de Ø à 7:

Valeur	*Canaux activés*
Ø	Aucun
1	1
2	2
3	1 + 2
4	3
5	1 + 3
6	2 + 3
7	1 + 2 + 3

Bruit : Sélectionne les canaux produisant du bruit blanc, avec le même code que pour <son>.

Enveloppe : De 1 à 7, choix du "contour" d'une note ou d'un son.

Numéro	*Caractéristiques*
1	Son unique, avec attaque nette et decay lent.
2	Son unique, attaque lente et arrêt net.
3	Comme 1, mais répétitif.
4	Répétitif, avec attaques et decays lents.
5	Une attaque nette, puis son continu.
6	Comme 2, répétitif.
7	Une attaque lente, puis son continu.

Durée : Choix de la longueur de son conformé par l'enveloppe, peut avoir une valeur entière comprise entre 1 et 37ØØØ. Agit de trois manières différentes:

1) S'il s'agit d'une enveloppe spécifiant un son unique (N° 1 et 2), détermine la durée de celui-ci.

2) Pour une enveloppe produisant un son répétitif (N° 3, 4 et 6), définit l'intervalle entre deux répétitions. Donc si cet intervalle est suffisamment court, il devient lui-même une fréquence.

3) S'il s'agit d'un son continu (enveloppes 5 et 7), définit simplement la longueur de l'attaque.

   N étant la longueur souhaitée en millisecondes, on trouvera <durée> avec la formule suivante:

   <durée>=N(ms)*2

Exemples:

Pour voir comment on peut produire un son avec PLAY sans utiliser au préalable MUSIC ou SOUND, taper:

```
PLAY 0,7,3,120
```

Puis:

```
SHOOT:PLAY 0,7,3,120
```

On obtient deux bruits distincts (rappelant un hélicoptère) avec la même instruction, simplement en "travaillant" des sons de départ différents: La répétition sonore du clavier pour la première ligne, et SHOOT pour la deuxième.

Simulation du "clic" du clavier par programme: 2 méthodes.

1) PLAY 7, Ø, 1, 2Ø

On remarque que cette instruction reproduit fidèlement le dernier "clic" entendu, "RETURN" et les flèches ne produisant pas exactement le même que le restant du clavier.

2) CALL #FB14 produit le son généré par une touche normale,

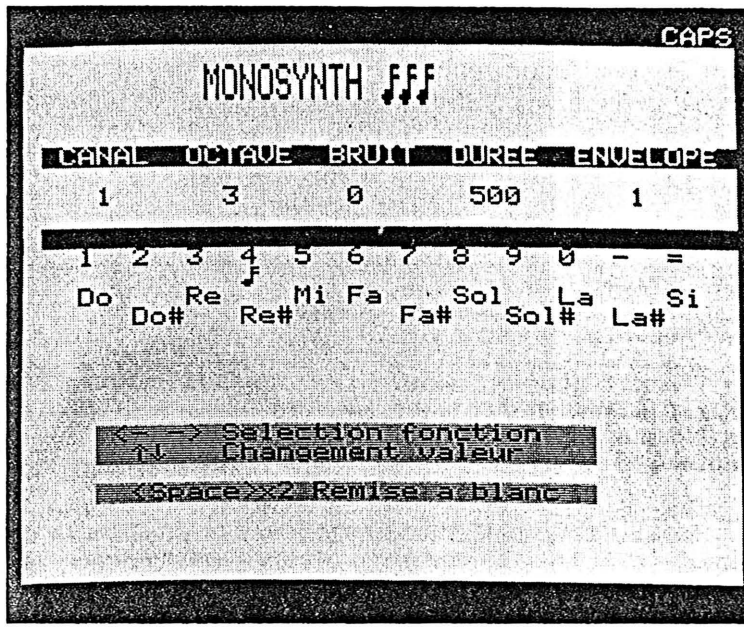
   CALL #FB2A produit le son généré par RETURN, DEL, etc...

5.3. *Une application: le programme MONOSYNTH*

Il est extrêmement difficile de véhiculer l'impression de sons ou des instructions qui les gèrent par des mots, nous avons donc conçu le petit

programme ci-dessous qui vous donnera la possibilité de vous servir du clavier comme d'un instrument de musique. Mais aussi et surtout, celle d'agir sur tous les paramètres définissant un son.

Ces paramètres sont affichés à l'écran, avec les noms et les valeurs qu'ils ont dans les instructions MUSIC et PLAY. Ainsi, vous pourrez exactement savoir "qu'est-ce qui fait quoi", et éventuellement retenir les combinaisons les plus intéressantes pour d'autres applications.



L'écran de "Monosynth"

On peut voir sur l'écran, de haut en bas:

— Une ligne contenant les noms des cinq paramètres principaux: Canal et Octave, arguments de MUSIC, puis Bruit, Durée et Enveloppe, arguments de PLAY.

— Juste en dessous, les valeurs de chacun des paramètres. L'une d'entre elles clignote, cela veut dire qu'on peut la modifier: Elle augmente

d'une unité à chaque appui sur "flèche vers le haut", et diminue avec la flèche vers le bas. On passe d'un paramètre à l'autre avec les flèches droite et gauche.

— Deux lignes plus bas, on trouve la liste des touches du clavier produisant une note: Il s'agit de la rangée supérieure contenant les touches numériques, de "1" à "=", et encore en dessous leur valeur dans la gamme. Chaque fois qu'une note est jouée, un petit caractère spécial en forme de note s'affiche en dessous.

— Un petit texte d'aide en bas d'écran termine la présentation.

Une dernière remarque: Vous pourrez parfaitement, si vous le désirez, adapter le programme pour mémoriser des mélodies et les rejouer ensuite. Il vous faudra dans ce cas créer un tableau contenant les divers paramètres, que vous pourrez sauvegarder et relire sur cassette en utilisant STORE et RECALL.

```
1 '       ** MONOSYNTH **
2 '
10 CLS:GOSUB 1000
18 '
20 '      PRESENTATION ECRAN
22 '
100 PRINT:PRINT" "CHR$(4)CHR$(27)"J        MONOSYNTH @@@"CHR$(4)
110 PRINT:PRINT:PRINT:PLOT 1,5,17:PLOT 1,9,17
120 PRINT"CANAL  OCTAVE  BRUIT  DUREE  ENVELOPE"
130 POKE#26F,3
134 '
135 '       AIDE
136 '
140 PRINT@2,19;E$"S<- -> Selection fonction "E$"W"
150 PRINT E$"S ↑&   Changement valeur  "E$"W"
160 PRINT:PRINT E$"S <Space>x2 Remise à blanc"E$"W"
164 '
165 '       IMPRESSION PARAMETRES
166 '
170 POKE 618,10:F=1:NM=1
180 FOR N=1 TO 5
190 PLOT M(N,3),7,STR$(M(N,0))+"  ":PLOT M(N,3)+6,7,8
200 NEXT:PLOT M(4,3)-1,7,8
220 PLOT M(F,3)-1,7,12:PRINT@2,10;
230 PRINT " 1  2  3  4  5  6  7  8  9  0  -  =":PRINT
240 PRINT " Do    Re    Mi Fa    Sol   La    Si"
250 PRINT "    Do#   Re#      Fa#   Sol#  La# //
296 '
297 '       BOUCLE PRINCIPALE
298 '       SAISIE NOTES
299 '
```

```
300 GET X$:AS=ASC(X$):NO=VAL(X$)
310 IF X$="0" THEN NO=10
320 IF X$="-" THEN NO=11
330 IF X$="=" THEN NO=12
340 IF NO=0 THEN 440
350 PLOT NM*3,11,32:NM=NO:PLOT NO*3,11,64
354 '
355 '       JEU NOTES
356 '
360 ON M(1,0) GOTO 370,380,380
370 MUSIC M(1,0),M(2,0),NO,0:PLAY 1,M(3,0),M(5,0),M(4,0):GOTO 300
380 MUSIC M(1,0),M(2,0),NO,0:PLAY 7,M(3,0),M(5,0),M(4,0):GOTO 300
398 '
400 '       SAISIE NOUVEAUX PARAMETRES
402 '
440 IF AS=32 THEN PRINT CHR$(6);:GOTO 300
450 IF AS=9 AND F<5 THEN F1=1:GOSUB 900
460 IF AS=8 AND F>1 THEN F1=-1:GOSUB 900
470 IF AS=11 AND M(F,0)<M(F,2) THEN M(F,0)=M(F,0)+1:IF F=4 THEN
M(F,0)=M(F,0)+9
480 IF AS=10 AND M(F,0)>M(F,1) THEN M(F,0)=M(F,0)-1:IF F=4 THEN
M(F,0)=M(F,0)-9
484 '
485 '       IMPRESSION NOUVEAUX PARAM.
486 '
490 PLOT M(F,3),7,STR$(M(F,0))+"   "
500 GOTO 300
900 PLOT M(F,3)-1,7,8:F=F+1:PLOT M(F,3)-1,7,12:RETURN
999 '
1000 '      INITIALISATIONS
1002 '
1010 FOR N=1 TO 5
1020 READ M(N,0),M(N,1),M(N,2),M(N,3)
1030 NEXT
1040 DOKE 46592,1031:DOKE 46594,1031:DOKE 46596,7172:DOKE 46598,6204
1050 DOKE 46384,2056:DOKE 46386,2056:DOKE 46388,5162:DOKE 46390,8
1070 E$="   "+CHR$(27):RETURN
1200 DATA 1,1,3,3,3,0,6,10,0,0,6,17,500,0,32767,24,1,1,7,33
```

Explications du programme:

La grande majorité des variables utilisées par le programme sont regroupées dans le tableau à deux dimensions M(X,Y), organisé comme suit:

Y=Paramètre → / *X=Fonction*		Ø *Valeur en cours*	1 *Val. Min. autorisée*	2 *Val. Max. autorisée*	3 *Position horiz. à l'écran*
1	CANAL	M(1,Ø)	M(1,1)	M(1,2)	M(1,3)
2	OCTAVE	2,Ø	2,1	2,2	2,3
3	BRUIT	3,Ø	3,1	3,2	3,3
4	DUREE	4,Ø	4,1	4,2	4,3
5	ENVELOPE	5,Ø	5,1	5,2	5,3

— Ce tableau regroupe donc toutes les valeurs affichées à l'écran, et une variable (F) déterminant quelle fonction est en cours de modification permet de tout traiter dans une seule boucle.

— Deux caractères spéciaux sont utilisés, "@ " est transformé en note de musique et "&" en flèche vers le bas.

1ØØ-16Ø : Affichage du logo en double hauteur, et de tous les textes à l'écran.

17Ø-25Ø : Impression des valeurs initiales des divers paramètres. PLOT place un code "caractères normaux" (8) après chaque valeur, de telle sorte que le code "clignotant" (12) servant à repérer la fonction active n'agit que sur une d'entre elles à la fois.

3ØØ-35Ø : Boucle principale. NO est la valeur de la note jouée, et les lignes 31Ø-33Ø y placent une valeur correcte pour les touches non numériques. La ligne 34Ø détecte l'utilisation des flèches, qui sont traitées plus loin. NM mémorise la précédente note jouée, pour effacer le caractère "note" avant de le replacer.

36Ø-38Ø : La note saisie est jouée par une instruction MUSIC et conformée par PLAY, leurs divers paramètres utilisant les valeurs sélectionnées à l'écran.

4ØØ-48Ø : Gestion des déplacements latéraux (changement de fonction) et haut/bas (changement de la valeur d'une fonction). F1 est la valeur à ajouter à la valeur en cours du paramètre modifié (1 si on l'augmente et —1 si on le diminue).

49Ø-9ØØ : Impression de la nouvelle valeur du paramètre modifié à la position contenue dans M(F,3).

1ØØØ-1Ø3Ø: Initialisations: Les valeurs de départ, position d'impression, ainsi que les maxima et minima de chaque paramètre sont chargés à partir de DATA dans le tableau M( ).

1Ø4Ø-1Ø5Ø: Création des deux paramètres spéciaux utilisés par le programme.

5.4. Bombardier

Jeu intéractif du type "jeu d'arcades", Bombardier n'a pas de relation directe avec le son en général. Nous avons toutefois pensé qu'il serait agréable de terminer cet ouvrage par un programme à vocation résolument ludique.

Comme pour tous les logiciels de cette catégorie, la définition en est simple mais la réalisation relativement complexe. Le lecteur patient pourra extraire nombre de "trucs" intéressants en épluchant ce programme qui est, précisons-le, écrit entièrement en Basic.

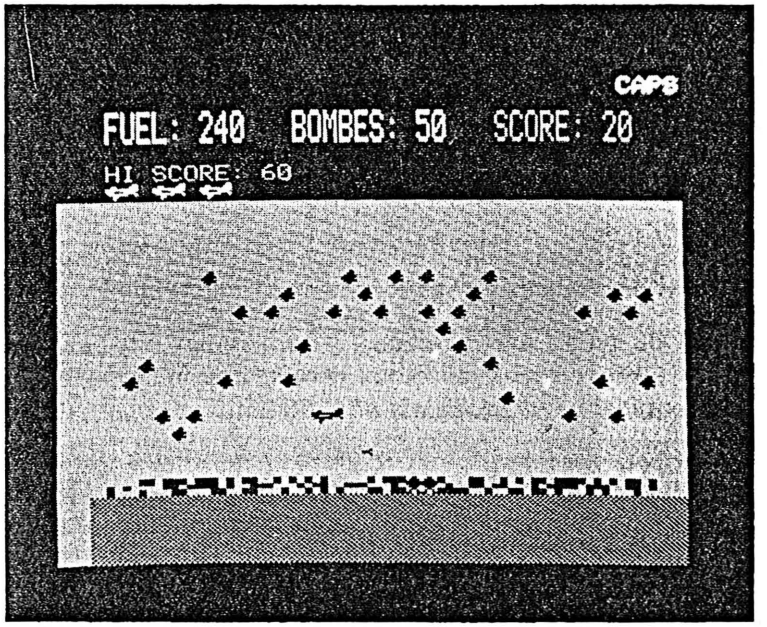
Le principe du jeu est le suivant: Vous pilotez un bombardier de nuit dans le ciel ennemi, et à vos pieds se déroule un décor composé de caractères semi-graphiques aléatoires; bombarder ces derniers ne rapporte rien, mais de temps à autre une fusée ou un réservoir apparaissent, qui vous rapportent des points si vous les touchez.

Pour corser la difficulté, la D.C.A. vous canarde sans relâche (et bien souvent vous touche!) et vous consommez trente litres de fuel à chaque tour d'écran sur les 3ØØ que vous avez au départ. Seule, la destruction des réservoirs au sol vous évitera la panne sèche au bout d'un certain temps.

On a le choix entre deux niveaux de difficulté, et tous les 1ØØØ points un avion supplémentaire (3 au départ) sera accordé, ainsi que huit bombes.

Comme d'habitude, le programme est constitué de sous-programmes chargés de tâches précises, et qui ne se suivent pas forcément dans l'ordre où ils sont appelés. D'une manière générale, les routines servant le plus souvent sont au début, et celles utilisées une ou deux fois à la fin du programme.

Le score, la quantité de munitions, de fuel et d'avions restants sont affichés en haut de l'écran, ainsi que le High score qui ne s'effacera d'ailleurs pas tant que l'ordinateur n'est pas éteint (ni RUN, ni NEW n'y feront rien).



En mission de nuit...

L'avion change d'altitude avec les flèches haut et bas, et une bor larguée en pressant la barre d'espacement. La D.C.A. "suit" l'avi ses déplacements verticaux, et tire aléatoirement dans une fourch trois cases autour de sa position. Vous pouvez monter au-des plafond de la D.C.A., mais il devient très malaisé de toucher les

Enfin et pour mémoire, "Bombardier" n'occupera que 3,5 Koctets de RAM tel quel et moins de 2,5 K si le thème musical du début et les REM sont supprimés.

```
1 '     **********************
2 '     *                    *
100 '   *  BOMBARDIER  V2.0  *
101 '   *  JEU DE REFLEXES   *
102 '   *  & D'ADRESSE       *
103 '   *                    *
104 '   **********************
105 '
106 '
110 PAPER4:INK0:E$=CHR$(27)
120 POKE618,14
180 GOSUB6000
190 EXPLODE
198 '
200 '       DESSIN DECOR
202 '
210 GOSUB9000:GOSUB8400:CLS
220 PAPER4:INK0:K=2
225 FORN=48040TO48240STEP40:DOKEN,784:NEXT:POKE48241,7
230 D=0:X=37:Y=7:A=2.2:AM=2:NO=30:AX=0:CA=300:A1=2:B1$=" "
260 GOSUB2000
270 GOSUB5000
280 FORN=23TO26
290 PLOT2,N,"{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{"
300 NEXT
998 '
1000 '      BOUCLE PRINCIPALE
1002 '
1100 X$=KEY$:IFX$<>""THENAX=ASC(X$)
1105 IFCA<0THENAX=10
1110 IFAX>9ANDAX<12THENGOSUB2000
1120 IFX$<>" "THEN1130
1125 IFB1$=" "THENB1$=B$:Y1=Y:X1=X+1:AS=SCRN(X1,22):MU=MU-1
1127 '
1130 ONDGOTO3000:'  Avion detruit
1140 GOSUB4000  :'  Dessin decor
1145 '
1150 PLOTX,Y,R$
1160 IFX<=2THENX=36:CA=CA-30:CO=CO+10:GOSUB5000ELSEX=X-1
1165 D=(X=A1)*(Y=AL)
1170 PLOTX,Y,V$
1180 PLOTAM-1,AL,0:PLOTAM,AL,"¦"
1185 ONKGOTO1200,1190
1190 PLAY7,7,6,NO:K=1
1197 '
1198 '      Explosion bombe
1199 '
1200 IFB1$=" "ORMU=0THEN1100
1210 Y1=Y1+1
1220 IFY1<23THEN1280
1225 PLOT1,22,7:EXPLODE:WAIT5:PLOT1,22,0
1230 PLOTX1,22," ":K=2
1231 '
1232 '      Cible touchee
1233 '
1235 IFAS=37THENCO=CO-50:AS=0:GOTO1250
1237 IFAS=65THENCO=CO-100:AS=0:CA=CA+30:GOTO1250
1240 IFX1<>A1THEN1270
1250 PAPER7:INK0:EXPLODE:WAIT5:PAPER0:INK7:CO=CO+200:PAPER4:INK0:K=2
1255 FORN=48040TO48240STEP40:POKEN,16:NEXT:POKE48241,7
1260 GOSUB5000
1270 B1$=" ":GOTO1100
```

```
1280 PLOTX1,Y1-1," "
1290 PLOTX1,Y1,B1$
1300 GOTO1100
1998 '
2000 '      Deplacements Avion
2002 '
2030 IFAX=11ANDY>7THENPLOTX,Y,R$:Y=Y-1:NO=NO+1:K=2:GOTO2100
2040 IFAX<>10THEN2100
2050 IFY<21THENPLOTX,Y,R$:Y=Y+1:NO=NO-1:K=2ELSED=1
2100 PLOTX,Y,V$:RETURN
2998 '
3000 '      DESTRUCTION AVION
3002 '
3010 FORN=0TO5
3015 IFCO>DEEK(#400)THENDOKE#400,CO
3020 PAPERRND(1)*7:INKRND(1)*7:EXPLODE:WAITRND(1)*10
3030 WAIT2:NEXTN
3035 IFVA=1THEN3040
3036 VA=VA-1:GOSUB8000:WAIT250:CLS:GOTO220
3040 PAPER7:INK0:GOSUB5000
3060 PRINT@10,7;CHR$(4)E$"NGAME  OVER":PRINTCHR$(4);
3070 RUN
3998 '
4000 '      DESSIN DECOR
4002 '
4010 PLOT2,22,9
4020 A$=CHR$(RND(1)*60+32)
4030 PLOTA,22,A$:PLOTA,AL," "
4031 YL=12:IFY>12THENYL=Y
4032 AL=INT(RND(1)*R1+YL-(R1-1))
4033 AM=A1
4035 PLOTA-1,AL,7:PLOTA,AL,"*"
4040 A=A+.8:A1=INT(A):IFA1>37THENA=3.2:A1=3
4050 RETURN
4998 '
5000 '      COMPTEURS
5002 '
5010 PRINT@2,0;CHR$(4)
5015 IFCO/1000>=MITHENVA=VA+1:MU=MU+8:MI=MI+1
5020 PRINT@1,1;E$"E"E$"JFUEL:"CA;E$"E BOMBES:"MU;E$"A SCORE:"CO:PRINTCHR$(4)
5022 PRINT@2,4;E$"BHI SCORE:"DEEK(#400):PRINTE$"F";
5023 FORN=1TOVA:PRINTV$" ";:NEXT
5030 RETURN
5998 '
6000 '      MESSAGE DE DEBUT
6002 '
6005 IFDEEK(#400)<>21845THENFORN=1TO11:READZ,Q:NEXT:GOTO6150
6010 CLS
6020 PRINT@8,3;CHR$(4)E$"E"E$"JBVRP       PRESENTE:"
6030 PRINT@13,11;E$"NBOMBARDIER"
6040 PRINT@8,21;CHR$(4)E$"EA  GAME  OF  SKILL !"
6055 O=3
6060 FORN=1TO11
6070 READA,B
6075 MUSIC2,O,A,12
6080 PLAY3,0,1,2000
6090 WAITB
6095 PLAY0,0,0,0
6100 NEXTN
6110 IFO=2THEN6120ELSEO=2:RESTORE:GOTO6060
6120 DOKE#400,0:GOSUB8200
6150 RETURN
6200 DATA5,30,5,30,3,15,8,75,5,30
```

```
6210 DATA8,40,10,30,7,20,5,30,3,30,5,90
7998 '
8000 '      CRASH
8002 '
8020 PRINT@14,13;CHR$(4)E$"G"E$"NCRASH!"E$"H"E$"@"CHR$(4)
8030 RETURN
8198 '
8200 '      TEXTE DEBUT
8202 '
8210 CLS
8220 PRINT:PRINT
8230 PRINTE$"GVOTRE BOMBARDIER EST EN MISSION DE":PRINT
8240 PRINTE$"GNUIT AU DESSUS DU TERRITOIRE ENNEMI":PRINT
8250 PRINT:PRINTE$"G      ATTENTION A LA D.C.A!"
8260 RETURN
8398 '
8400 '      EXPLICATIONS
8402 '
8405 PLOT0,10,22
8410 PRINT@1,10;"          REGLES  DU  JEU"
8420 PRINTSPC(38);:PRINT"  ^ @ # : HAUT/BAS/TOUT DROIT  "
8430 PRINT"  ESPACE: LARGUE UNE BOMBE  ":PRINTSPC(38);
8440 PRINTE$"I%"E$"H  :"E$"F150 POINTS"
8450 PRINTE$"IA"E$"H  :"E$"F100 PTS & 30 LITRES DE FUEL"
8460 PRINT"CURS.:"E$"F200 POINTS":PRINTSPC(38);
8500 PRINT:PLOT0,19,22:PLOT0,20,22
8510 PRINT"CHAQUE PASSAGE CONSOMME 30 LITRES      ";
8520 PRINT"TOUS LES 1000 PTS: 8 BOMBES & 1 AVION ";
8530 PRINT:PRINT:PRINT:PRINTE$"L"E$"FTAPER <1> OU <2> SELON NIVEAU      ";
8540 X$=KEY$:IFX$="1"ORX$="2"THENR1=4-VAL(X$):RETURN
8550 GOTO8540
8999 '
9000 '      REDEFINITION CARACTERES
9002 '
9100 READAD:IFAD=0THEN9700
9110 FORN=0TO7
9120 READDA:POKEAD+N,DA
9130 NEXT
9140 GOTO9100
9500 DATA 47400,0,8,8,8,8,8,28,54
9510 DATA 46904,0,0,0,2,28,2,0,0
9520 DATA 47928,4,37,21,14,63,14,21,37
9530 DATA 47072,6,14,31,62,31,12,0,0
9540 DATA 47624,12,30,63,63,30,12,18,51
9550 DATA 46840,0,12,63,56,31,6,4,0
9560 DATA 46816,3,7,63,30,49,0,0,0
9570 DATA 47064,42,21,42,21,42,21,42,21
9580 DATA 46592,8,8,8,8,42,28,8,0
9590 DATA 46360,0,8,16,63,16,8,0,0
9690 DATA0
9699 '
9700 '      INITIALISATIONS
9702 '
9710 READ X,Y,A,AL,NO,AM,A1,VA,MU,CA,MI
9720 READV$,B$,F$,R$,B1$
9799 RETURN
9800 DATA 37,7,3.2,20,30,3,3,3,50,300,1
9810 DATA _\,g,%,"  "," "
```

Explications du programme:

Plusieurs caractères spéciaux sont créés par le jeu:

"%", jeu graphique : Fusée
"A", jeu graphique : Réservoir
"g", jeu normal : Bombe
"g", jeu graphique : Explosion au sol
"£", jeu normal : Avant de l'avion
"\ ", jeu normal : Arrière de l'avion
":", jeu normal : Nuage de fumée (explosion de D.C.A.)

On remarque que plusieurs caractères sont placés dans le jeu graphique, car toute la ligne ou défile le décor composé de mosaïques graphiques est précédée d'un attribut "deuxième jeu". Ainsi, le "g" qui représente une bombe dans le jeu normal (tant que la bombe est en l'air) se transformera instantanément en explosion une fois arrivé au sol. De plus, l'effet d'explosion est renforcé par la modification de l'attribut de couleur de la première ligne pendant un court instant, qui passe du noir au blanc quand aucune cible n'est touchée. En cas de coup réussi, c'est tout l'écran qui change de couleur accompagné par une succession de "EXPLODE" du plus bel effet...

Étant donné leur profusion, nous avons jugé utile d'ajouter une liste détaillée des variables employées:

1) Chaînes

B$: Bombe; caractère "g".
B1$: Contient un espace en temps normal, et B$ lorsque la bombe est en chute.
V$: Avion; 2 caractères.
R$: Contient 2 espaces, pour effacer l'avion lorsqu'il se déplace.
E$: Caractère ESCAPE — CHR$(27).

2) Coordonnées

A : Abscisse du curseur.
AM : Mise en mémoire de l'abscisse curseur.
A1 : Abscisse de la D.C.A.
AL : Altitude de la D.C.A. (ordonnée).
X,Y : Abscisse et ordonnée de l'avion.
X1,Y1 : Abscisse et ordonnée de la bombe.

3) Compteurs

CO : Nombre de points (score).
CA : Compteur carburant.
MU : Compteur munitions.
VA : Nombre d'avions restants.

4) Drapeaux

D : Égal à Ø si tout est normal, à 1 si l'avion est touché.
MI : Nombre de milliers de points gagnés, détermine l'acquisition d'avions supplémentaires.

Divers

R1 : Niveau du jeu (1 ou 2).
NO : Paramètre dans une instruction PLAY, détermine la hauteur du son en fonction de l'altitude de l'avion.
AS : Code du caractère se trouvant au point d'impact de la bombe; si celui-ci correspond à une fusée ou un réservoir, les compteurs sont incrémentés en conséquence.
AX : Code ASCII de la touche pressée.

18Ø : Appel d'un sous-programme jouant un petit thème musical, au premier démarrage du programme (6ØØØ).

21Ø : Appel des sous-programmes d'initialisation et création de caractères spéciaux (9ØØØ), et d'affichage des explications (8ØØØ).

22Ø-23Ø : Initialisation de certaines variables. La routine située en 9ØØØ n'étant appelée qu'à la première exécution du programme, certaines variables doivent être initialisées localement quand on veut rejouer.

28Ø-3ØØ : Remplit le bas de l'écran avec une zone grisée (le "sol").

1ØØØ-119Ø : Boucle principale, gérant les déplacements de l'avion, le son qu'il fait et les mouvements de la bombe si elle a été lâchée.

12ØØ-123Ø : Génère les sons et image correspondant à l'explosion de la bombe au sol.

1235-13ØØ : Détermine si une cible a été touchée, en testant le code du caractère situé à l'abscisse de la bombe. Si c'est le cas, génère les son et image appropriés et incrémente les compteurs.

2ØØØ-21ØØ : Efface l'avion de son ancienne position, et le redessine à la nouvelle.

3ØØØ-3Ø7Ø : Le branchement s'effectue vers cette routine si le drapeau de destruction (D) est à 1, résultat de la concordance entre l'altitude de la D.C.A. (AL) et l'ordonnée de l'avion (Y). Dans ce cas, l'écran prend une succession de couleurs aléatoires accompagnées d'explosions. S'il ne reste plus d'avions, le message "GAME OVER" s'affiche et le programme est relancé.

4ØØØ-4Ø5Ø : Dessine des mosaïques graphiques aléatoires au niveau du sol, en parsemant de ci de là quelques cibles tout aussi aléatoires...

5ØØØ-5Ø3Ø : Réactualise le compteur à chaque tour d'écran de l'avion ; se charge aussi de l'attribution d'avions supplémentaires tous les 1ØØØ points. si vous voulez changer cette valeur, il suffit de modifier la ligne 5Ø15.

6ØØØ-621Ø : Joue la musique "de générique" à la première exécution du programme. Cette partie peut être omise, en veillant à supprimer le "GOSUB 6ØØØ" de la ligne 18Ø.

8ØØØ-8Ø3Ø : Affiche le message "CRASH" si l'avion est détruit et qu'il en reste encore.

84ØØ-855Ø : Affichage des explications (points donnés par chaque cible, commandes, etc.) en début de programme.

9ØØØ-981Ø : Création des caractères spéciaux dont les données, contenues dans des DATA, sont précédées de l'adresse de début du caractère ; initialisation des principales variables.

Appendice 1
Le jeu de caractères ASCII

CODE		CARACTÈRE	CODE		CARACTÈRE
Décimal	*Hexa*		*Décimal*	*Hexa*	
32	2Ø	Espace	8Ø	5Ø	P
33	21	!	81	51	Q
34	22	"	82	52	R
35	23	#	83	53	S
36	24	$	84	54	T
37	25	%	85	55	U
38	26	&	86	56	V
39	27	'	87	57	W
4Ø	28	(	88	58	X
41	29	)	89	59	Y
42	2A	*	9Ø	5A	Z
43	2B	+	91	5B	[

CODE *Décimal*	CODE *Hexa*	CARACTÈRE	CODE *Décimal*	CODE *Hexa*	CARACTÈRE
44	2C	,	92	5C	
45	2D	—	93	5D	]
46	2E	.	94	5E	^
47	2F	/	95	5F	£
48	3Ø	Ø	96	6Ø	c
49	31	1	97	61	a
5Ø	32	2	98	62	b
51	33	3	99	63	©
52	34	4	1ØØ	64	d
53	35	5	1Ø1	65	e
54	36	6	1Ø2	66	f
55	37	7	1Ø3	67	g
56	38	8	1Ø4	68	h
57	39	9	1Ø5	69	i
58	3A	:	1Ø6	6A	j
59	3B	;	1Ø7	6B	k
6Ø	3C	<	1Ø8	6C	l
61	3D	=	1Ø9	6D	m
62	3E	>	11Ø	6E	n
63	3F	?	111	6F	o
64	4Ø	@	112	7Ø	p
65	41	A	113	71	q
66	42	B	114	72	r
67	43	C	115	73	s
68	44	D	116	74	t
69	45	E	117	75	u
7Ø	46	F	118	76	v
71	47	G	119	77	w
72	48	H	12Ø	78	x
73	49	I	121	79	y
74	4A	J	122	7A	z
75	4B	K	123	7B	{
76	4C	L	124	7C	¦
77	4D	M	125	7D	}
78	4E	N			
79	4F	O			

Appendice 2
Index des adresses mémoire

Adresse	*Fonction*	*Page(s)*
Pointeurs		
#9A,#9B (154,155)	Début zone programme	46, 53
#9C,#9D (156,157)	Fin prog./début var. simples	46, 48, 53
#9E,#9F (158,159)	Fin var. simples/début tableaux	48, 51
#AØ,#A1 (16Ø,161)	Fin tableaux/début chaînes	5Ø, 51
#A2,#A3 (162,163)	Adresse la plus basse dans la zone chaînes	51
#A4,#A5 (164,165)	Début de la dernière chaîne entrée	47
#A6,#A7 (166,167)	HIMEM	47
#BØ,#B1 (176,177)	LOMEM	47
#B8,#B9 (184,185)	Adresse dernière variable entrée	47

Adresse	*Fonction*	*Page(s)*
L'Ecran		
#12,#13 (18,19)	Adresse début de la ligne du curseur ..	58
#268 (616)	Numéro de la ligne du curseur	58,94
#269 (617)	Numéro de la colonne du curseur	59,94
#26A (618)	Modes curseur et clavier	59,85
#26B (619)	Couleur du fond, pour tout l'écran	60,95
#26C (620)	Couleur affichage, pour tout l'écran ...	60,95
#278,279 (632,633)	Adresse début de la deuxième ligne de l'écran	
#27A,27B (634,635)	Adresse début de la première ligne de l'écran accessible au curseur	61,92
#27C,27D (636,637)	Nombres de lignes prises par le scrolling (L*40)	61,93
#27E (638)	Numéro de la dernière ligne de l'écran accessible au curseur	61,93
Le clavier		
#208 (520)	Code de la dernière touche frappée ...	55
#209 (521)	Pointeur Normal/CTRL/SHIFT/FUNCT .	56
#20C (524)	Pointeur Majuscules/Minuscules	56
#20E (526)	Décompte de l'autorepeat	56
#24E (590)	Vitesse de démarrage de l'autorepeat ..	57
#26F (623)	Vitesse de l'autorepeat 1	57
#306,307 (774,775)	Vitesse de l'autorepeat 2	58
Imprimante et K7		
#24D (589)	Vitesse transfert K7. 0=Rapide, 1=Lente	
#256 (598)	Largeur Imprimante (normal=80)	64
Le graphique		
#213 (531)	Type de tracé en vigueur (PATTERN) ..	123
#219 (537)	Abscisse du curseur HIRES	123
#21A (538)	Ordonnée du curseur HIRES	123
#21F (543)	Pointeur TEXT/HIRES	124
#2E0 (736)	PARAMS — Début du bloc de transfert de données vers les routines graphique et son	124

Appendice 3
Messages d'erreur

Si une commande, donnée, ou syntaxe inacceptable est validée, Atmos répondra par un des messages ci-dessous suivis du mot "ERROR IN <N° de ligne>, ce dernier indiquant à quel endroit du programme s'est produite l'erreur.

BAD SUBSCRIPT

Un élément de tableau est appelé en utilisant un indice hors des limites déclarées dans DIM, lorsque le tableau a été dimensionné.

BAD UNTIL

Un UNTIL est rencontré, sans que la boucle ait commencé par REPEAT.

CAN'T CONTINUE

Réponse à l'instruction CONT, si le programme a été modifié pendant l'interruption.

DISP TYPE MISMATCH

Affiché si des instructions graphiques sont utilisées en mode texte, ou si PLOT est utilisé en HIRES.

DIVISION BY ZERO

Refus de diviser un nombre par zéro.

FORMULA TOO COMPLEX

Trop de IF — THEN imbriqués sur une même ligne, ou une expression mathématique trop longue est rencontrée..

ILLEGAL DIRECT

Tentative d'utiliser en mode direct des commandes ne fonctionnant que dans un programme (GET, INPUT, etc.).

ILLEGAL QUANTITY

Une valeur hors limites a été donnée comme argument à une instruction ou fonction (par ex. PRINT PEEK(3ØØ).

NEXT WITHOUT FOR

Une commande NEXT est rencontrée sans que la boucle ait débuté par FOR.

OUT OF DATA

Tentative d'utiliser READ alors que les DATA ont déjà été lus en entier.

OUT OF MEMORY

Plus de mémoire, ou bien plus de 16 boucles du type FOR-NEXT, REPEAT-UNTIL imbriquées.

OVERFLOW

Les calculs ont fourni un nombre dépassant $1.7*1Ø^{38}$, ou inférieur à $2.93*1Ø^{-39}$.

REDIM'D ARRAY

Réponse à une tentative de redimensionner un tableau déjà déclaré dans un DIM.

REDO FROM START

Des données incorrectes ont été fournies à un INPUT. Seule erreur ne repassant pas en mode direct, l'INPUT est réexécuté.

RETURN WITHOUT GOSUB

Un RETURN est rencontré sans être préalablement passé par un GOSUB. Peut aussi être provoqué par un POP mal utilisé.

STRING TOO LONG

Tentative d'attribuer plus de 255 caractères à une chaîne.

SYNTAX ERROR

Une erreur de syntaxe a été rencontrée dans une commande directe ou une ligne de programme.

TYPE MISMATCH

Discordance entre un type de variable et les données qu'on essaie de lui attribuer (par ex. A="ABCD"), ou entre une commande et l'argument qu'on lui donne (ex. SIN(A$)).

UNDEF'D STATEMENT

Un numéro de ligne inexistant a été fourni comme argument à GOTO, THEN ou GOSUB.

UNDEF'D FUNCTION

Tentative d'utiliser une fonction avec FN, sans l'avoir préalablement définie par DEF FN.

Appendice 4

Différences entre l'Atmos et l'Oric-1

L'Atmos n'est pas à proprement parler un nouvel ordinateur, il s'agit en fait d'une amélioration (substantielle) de l'Oric-1, apparu en 1982. Les modifications apportées sont les suivantes:

1) Au niveau du matériel:

* Un nouveau clavier a été installé, du type "machine à écrire" entièrement mécanique, qui remplace l'ancien clavier constitué de petites touches peu mobiles.

  La disposition générale des touches est restée la même, et une nouvelle touche (FUNCT) a été ajoutée. Celle-ci n'est pas reconnue par le Basic, et n'est exploitable que par l'intermédiaire d'une variable-système (voir § 2.3.3).

* Deux résistances ajustables sont accessibles sur la partie inférieure du boîtier, l'une par un trou de forme rectangulaire et l'autre par un trou circulaire. Elles permettent de régler l'émission des couleurs dans certaines limites, la première agissant sur la balance couleurs et la seconde sur le contraste.

Les accès aux périphériques (port d'extension, port parallèle imprimante, prise cassettes et vidéo) demeurent inchangés sur l'Atmos.

2) Au niveau du logiciel:

C'est dans ce domaine que se trouve le plus grand nombre de modifications et d'améliorations, même si elles sont moins spectaculaires.

* La ROM a été changée, et contient une nouvelle version du Basic, la 1.1. Celle-ci possède toutes les instructions du Basic 1.Ø de l'Oric-1, avec en plus:

— PRINT @, autorisant l'adressage direct du curseur (déplacement) par l'instruction PRINT.

— STORE et RECALL, permettant la sauvegarde et la lecture de tous types de tableaux de variables sur cassette.

* D'autre part, un certain nombre d'instructions et de fonctions ont été améliorés ou déboguées:

— CLOAD possède maintenant deux options supplémentaires, "V" qui autorise la vérification des sauvegardes (VERIFY), et "J" qui permet de charger un programme en mémoire à la suite du programme résident, sans effacer celui-ci (MERGE).

  En outre, CLOAD détecte maintenant automatiquement le type de fichier qu'il charge, une syntaxe particulière n'est plus nécessaire pour charger une zone-mémoire. Dans ce dernier cas également, les pointeurs ne sont plus déplacés après le chargement et on ne retourne plus automatiquement en mode direct.

  Enfin, si on utilise CLOAD"", le nom du programme trouvé est affiché en cours de chargement. Et si le programme spécifié dans CLOAD<Nom de fichier> n'est pas trouvé du premier coup, la mention "Found" suivie des noms des programmes rencontrés est affichée.

— Le caractère Escape (CHR$(27)) ne peut plus s'afficher sur les colonnes protégées, comme c'était le cas auparavant.

— L'instruction REM peut être remplacée par "'", même en début de ligne.

— Les bogues de la fonction TAB (positionnement erroné, impossibilité de l'utiliser plusieurs fois) ont été éliminés.

— La fonction STR$ a été également corrigée (le premier caractère de la chaîne générée était CHR$(2) au lieu d'un espace).

— Le bogue de ELSE a été corrigé (une variable en virgule flottante juste avant ELSE provoquait une erreur de syntaxe). Par contre, un autre est apparu : Il est désormais impossible de spécifier plusieurs instructions après ELSE.

— L'interférence qui existait entre le scanning-clavier et la sortie imprimante, et qui se manifestait par l'émission aléatoire du caractère CHR$(127) vers cette dernière lors de l'utilisation de LPRINT et LLIST, a été supprimée.

* En ce qui concerne l'organisation-mémoire, les différences sont minimes mais nombreuses :

— Les variables-système ont, pour leur grande majorité, changé d'emplacement. C'est le cas pour toutes celles concernant le clavier, et la moitié de celles concernant l'écran.

— Les routines de la ROM ne sont évidemment plus du tout situées aux mêmes emplacements.

— Le mauvais positionnement de HIMEM à la mise sous tension, qui se manifestait par un empiètement de l'espace-chaîne sur l'écran, a été corrigé.

Il est à noter toutefois que les grandes zones de la mémoire sont demeurées inchangées, ainsi que les emplacements des pointeurs s'y référant.

Index alphabétique

L'Oric Atmos est une nouvelle petite bombe dans le monde de l'informatique individuelle, qui reprend toutes les qualités de son prédécesseur l'Oric-1 en en éliminant les défauts.

Malgré la similitude apparente entre les deux machines, leurs différences sont nombreuses, et pour cette raison cet ouvrage a été conçu selon le même principe que « Tout savoir sur l'Oric-1 » en exploitant à fond les nouvelles possibilités créées par l'Atmos. Il s'agit également d'un ouvrage d'approfondissement, et non d'initiation.

Outre l'examen détaillé, illustré d'exemples, des possibilités les plus sophistiquées du Basic, de l'exploitation des variables systèmes et des « Trucs et astuces » qui en découlent, une très large part est faite aux programmes d'applications.

Vous trouverez, entre autres, une gestion de fichier carnet d'adresses, un générateur de caractères très performant, un mini-traitement de textes et un programme complet de D.A.O... Et les amateurs de jeux n'ont pas non plus été oubliés, qui découvriront plusieurs exemples de sous-programmes spécialisés, ainsi qu'un grand jeu en temps réel du type « jeu d'arcades ».

EYROLLES

8221