ORIC INTERNATIONAL
PRESENTE

A LA DECOUVERTE DU TELESTRAT

écrit par Patrice GUERRIN

SOMMAIRE

CHAPITRE 5
LA GESTION DU CLAVIER

CHAPITRE 6
LES STRUCTURES DE CONTROLE ET LES OPERATEURS LOGIQUES ET ARITHMETIQUES

CHAPITRE 7
LES FONCTIONS DE TRAVAIL SUR LES CHAINES ET LES FONCTIONS DE CONVERSION

CHAPITRE 8
LES COMMANDES ET FONCTIONS GRAPHIQUES BASSE RESOLUTION

CHAPITRE 9

LES COMMANDES GRAPHIQUES HAUTE RESOLUTION

CHAPITRE 10

LES COMMANDES ET FONCTIONS SYSTEME DU BASIC

CHAPITRE 11

LES CONSTANTES ET FONCTIONS DE TRAVAIL SUR LES NOMBRES

CHAPITRE 12
LES COMMANDES DE GESTION DU SON

CHAPITRE 13
LES COMMANDES DE GESTION DU DISQUE

CHAPITRE 14
LES COMMANDES ET FONCTIONS DE GESTION DES FICHIERS

CHAPITRE 15
LES MODES DE CLAVIER DU TELESTRAT

CHAPITRE 16
LES COMMANDES ET FONCTIONS DE GESTION DE L'HORLOGE

CHAPITRE 17
LES COMMANDES ET FONCTIONS DE GESTION DES ERREURS

CHAPITRE 18
LES COMMANDES SYSTEME DU TELESTRAT

CHAPITRE 19
LES COMMANDES DE GESTION DES CANAUX D'ENTREES/SORTIES

CHAPITRE 20
LES COMMANDES ET FONCTIONS DE TRAVAIL SUR LA MEMOIRE

CHAPITRE 21
LES COMMANDES DE GESTION DE L'IMPRIMANTE

CHAPITRE 22
LA REDEFITION DES CARACTERES

CHAPITRE 23
LES COMMANDES DE GESTION DE LA RS232

CHAPITRE 24
LES COMMANDES DE GESTION DU MINITEL

INTRODUCTION

Ce livre se propose de vous enseigner le langage HYPERBASIC de l'ordinateur ORIC TELESTRAT. La lecture de "A LA DECOUVERTE DU TELESTRAT" ne nécessite aucune connaissances préalables, que ce soit dans le domaine informatique, ou mathématique. Etre sérieux et appliqué, voilà qui devrait grandement suffire à vous ouvrir les portes du langage BASIC. Toutes les règles, les astuces de la programmation vous sont présentées d'une manière simple et progressive, accompagnées par de nombreux exemples commentés.

En complément aux instructions présentées dans ce livre, il est recommandé de faire également usage du manuel ' Index de l'HYPERBASIC ' fourni avec le **TELESTRAT**. A chaque instruction rencontrée , regardez les explications complémentaires données dans ce manuel.

Que vous soyez débutant, ou programmeur avancé, ce livre vous concerne. Il vous aidera à tirer le meilleur parti de cette ' bête à communiquer ' qu'est le **TELESTRAT**.

Et maintenant, commençons ...

CHAPITRE 1

INSTALLATION DE L'ORIC TELESTRAT

A/ MATERIEL

1. MATERIEL FOURNI AVEC LE TELESTRAT

Avant de pouvoir découvrir les mille et un talents de votre ORIC TELESTRAT, il va tout d'abord falloir le mettre en service. Voici comment procéder. Après déballage, vous devez normalement avoir les éléments suivants :

- une unité centrale ORIC TELESTRAT 64 Ko,
- un lecteur de disquette nommé MICRODISC,
- une alimentation,
- une disquette MASTER STRATSED contenant de nombreux programmes utilitaires et de nombreux exemples,
- un index de l'HYPERBASIC,
- un manuel des applications Télématiques,
- une planche d'autocollants pour transformer votre clavier en AZERTY ou FRENCH,
- un cable plat en nappe de couleur grise, permettant de relier le TELESTRAT au MICRODISC,
- un cable PERITEL,
- un cable MINITEL.

Vous devez également avoir :

- un téléviseur ou un moniteur muni d'une prise PERITEL.

2. MATERIEL SOUHAITABLE

- un MINITEL, (modèle quelconque, retournable si vous voulez réaliser un serveur)
- un téléphone,
- 1, 2 ou 3 lecteurs de disquette supplémentaires,
- une imprimante,
- 1 ou 2 Joysticks,
- une souris.

B / CONNECTION

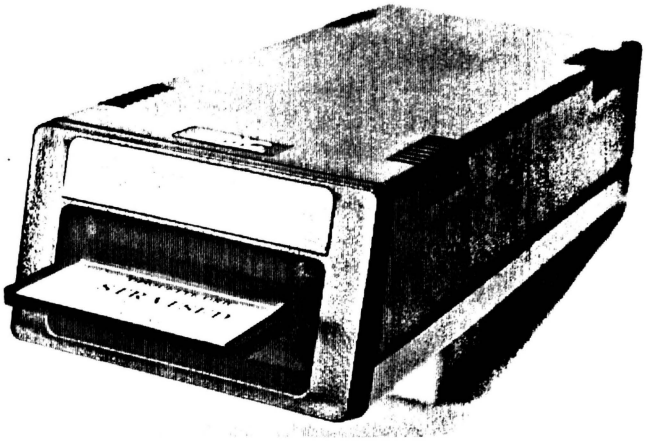
1/ Le TELESTRAT

Voici l'unité centrale ORIC TELESTRAT :

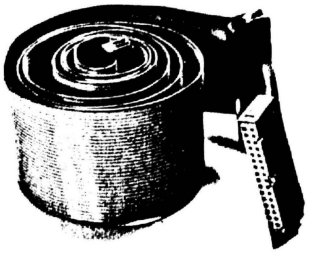
Sur le haut du TELESTRAT, une fenêtre qui bascule vous laisse entre-apercevoir deux cartouches :
A gauche, la cartouche HYPERBASIC.
A droite, la cartouche TELEMATIC.
Vérifiez que ces deux cartouches sont bien placées, et bien enfoncées.

2/ Le MICRODISC

Branchez maintenant le MICRODISC (nom du lecteur de disquette) au TELESTRAT.

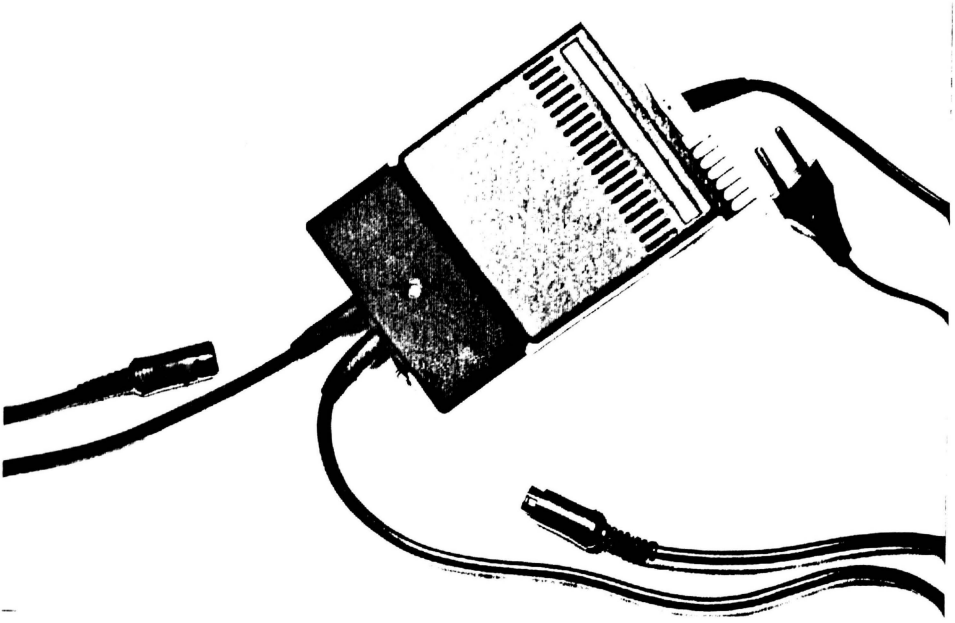


Vous disposez pour cela d'un cable plat de couleur grise :



Toutes les prises de branchement des cables des périphériques (lecteur de disquette, imprimante, minitel ...) se situent à l'arrière de l'ORIC. Reliez le TELESTRAT au MICRODISC avec le cable gris.

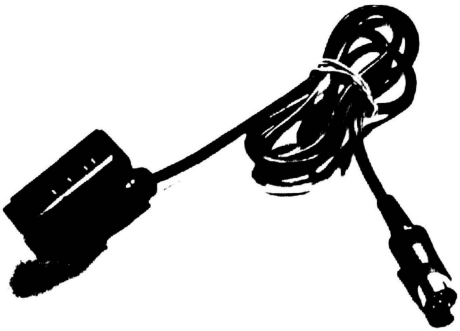
3/ L'alimentation



Cette alimentation comprend 3 cordons :

- Le premier, terminé par une prise 220 V, se branche simplement au secteur.
- Le second relie l'alimentation au TELESTRAT. Il s'agit du fil dont l'origine est la plus près du bouton rouge de l'alimentation. Le bout de ce fil est plus petit que celui servant à alimenter le lecteur de disquette. Branchez ce bout à la prise 'POWER' ou encore prise d'alimentation que vous trouverez derrière le TELESTRAT, entre la prise 'MICRODISC', et la prise 'MINITEL'.
- Le troisième cordon de l'alimentation se branche en bas et à gauche de l'arrière du lecteur de disquette.

4/ La prise PERITEL



Le plus gros côté de la prise péritel va sur votre téléviseur, ou sur votre moniteur. Le plus petit s'enfiche dans la prise 'RGB' du TELESTRAT.

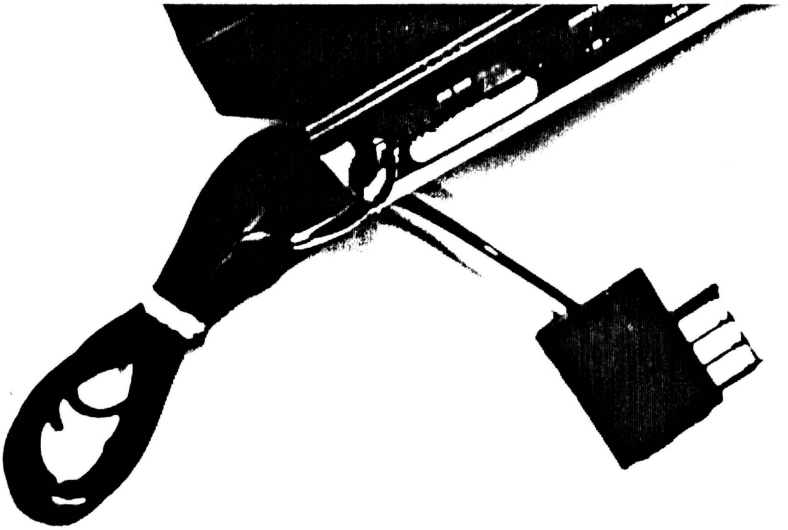
5/ Le cable MINITEL

Pour brancher le MINITEL, utilisez le cordon suivant :

Le premier bout du cordon, celui qui possède 8 broches, se branche dans la prise 'MINITEL' située derrière votre TELESTRAT. Le deuxième bout du cordon noir se connecte à la prise péri-informatique du MINITEL.

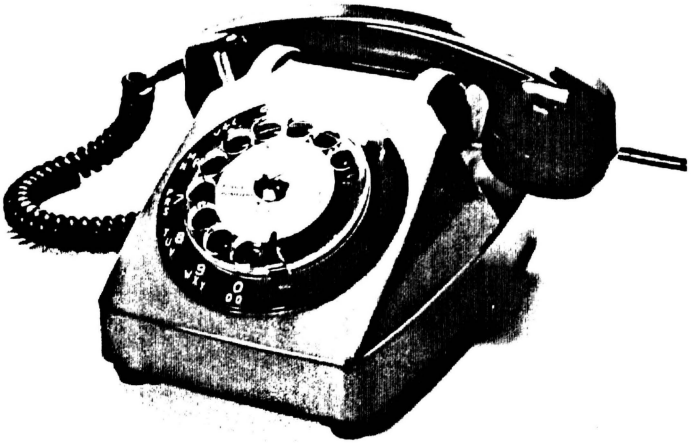
6/ Le cable gigogne

Voici le cable gigogne partant du TELESTRAT :

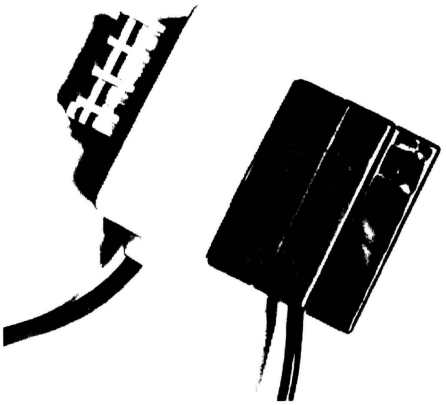


Reliez le à la prise prévue spécialement sur le minitel . Et surtout, n'essayez pas de débrancher le cable gigogne partant du TELESTRAT, puisqu'il y est soudé !

7/ Le téléphone



Reliez le fil partant de votre appareil téléphonique à la prise gigogne du TELESTRAT, en l'enfonçant tout simplement à l'intérieur de la première.

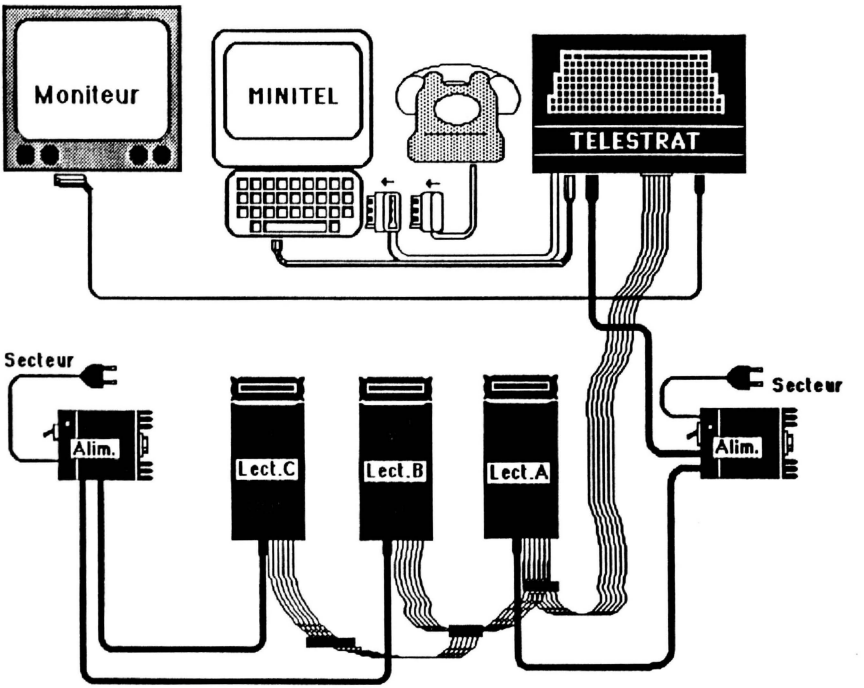


8/ Lecteur(s) de disquette suplémentaire(s)

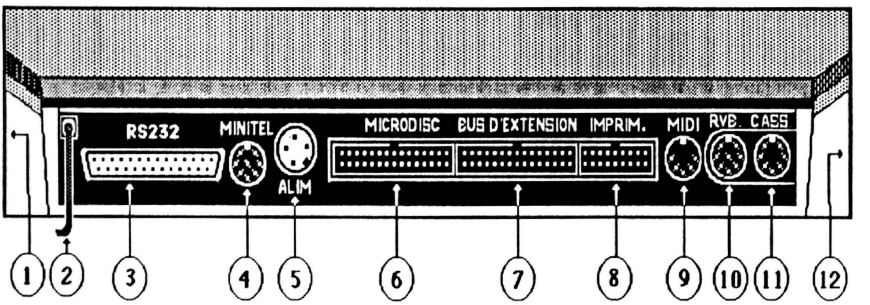
En plus de celui fourni avec le TELESTRAT, il est possible de brancher 1, 2 ou 3 autres lecteurs de disquette. Cela permet de monter à 4 le nombre de lecteurs possibles. Pour d'autres lecteurs de disquette, il conviendra de relier le cable plat en nappe de couleur grise partant de ceux-ci à la prise en nappe grise du précédent lecteur, après avoir préalablement enlevé le morceau de plastique noir protégeant l'entrée de cette prise. L'alimentation fournie lors de votre achat alimente un TELESTRAT, ainsi qu'un lecteur, celui donné initialement. Au cas où vous auriez plusieurs lecteurs, il faut savoir que :

- Pour 1 ou 2 lecteurs de disquettes suplémentaires, une seconde alimentation est nécessaire.
- Pour 3 lecteurs de disquette suplémentaires, il vous faudra 2 alimentations en plus.

9/ Schéma général des branchements



Vu le nombre important d'extensions du TELESTRAT, il convient d'en faire ici le résumé :



De gauche à droite, nous trouvons :

1 - Prise souris
2 - Cable téléphonique
3 - Interface série RS232
4 - Prise MINITEL
5 - Alimentation
6 - Branchement du MICRODISC
7 - Bus d'extension
8 - Sortie imprimante au standart CENTRONICS
9 - Extension MIDI (musique)
10 - Prise RVB-PERITEL
11 - Prise magnétocassette avec télécommande possible
12 - Prise Joystick

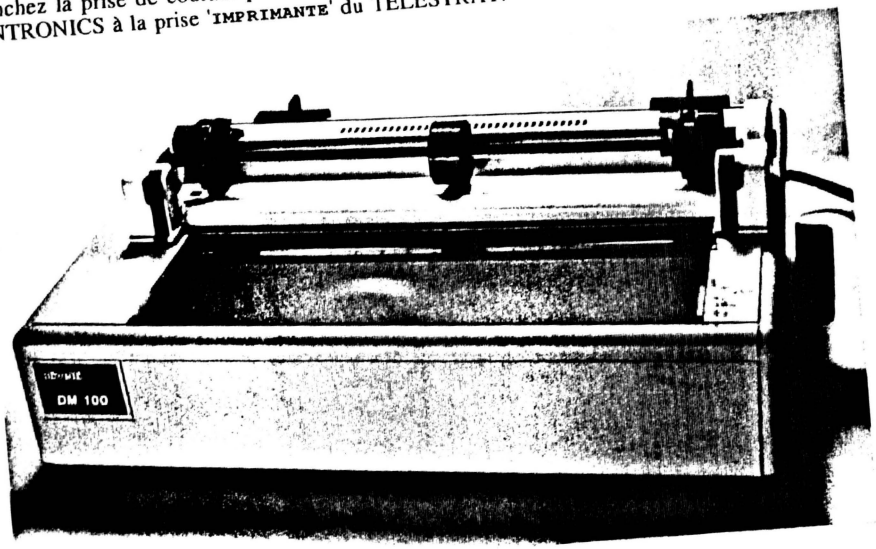
Comme vous le remarquez, le TELESTRAT est doté de très nombreuses prises d'extension. Presque tout ce qui est du domaine de l'informatique peut lui être relié. Ce n'est donc pas pour rien qu'on l'appelle ' LA BETE A COMMUNIQUER ! '.

Enlevez le cache plastique obstruant l'entrée des disquettes dans le MICRODISC. (S'il y en a un , ce n'est pas le cas de tous les lecteurs de disquettes). Pour cela, enfoncez le bouton juste en dessous de l'entrée des disquettes, puis, retirez le morceau de plastique noir se trouvant à l'intérieur. Conservez-le. Son rôle est de protéger la tête magnétique du lecteur de disquette contre les vibrations durant un transport.

Prenez la disquette noire 'STRATSED' et insérez-la entièrement dans le lecteur de disquette, la face A de la disquette obligatoirement en haut. Un 'clac' se fait alors entendre. La disquette est insérée...

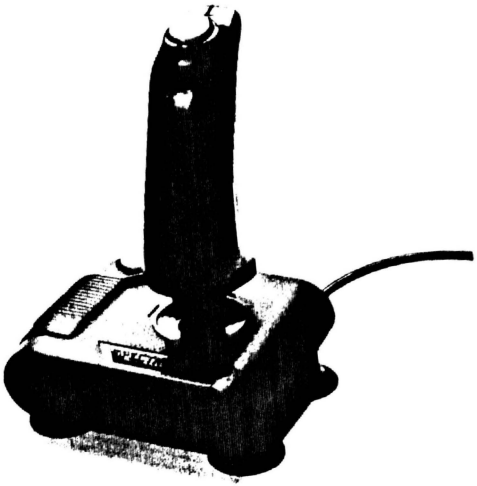
10/ Si vous possédez une imprimante au standard CENTRONICS :

Branchez la prise de courant partant de votre imprimante au secteur. Reliez ensuite le cable CENTRONICS à la prise '**IMPRIMANTE**' du TELESTRAT.



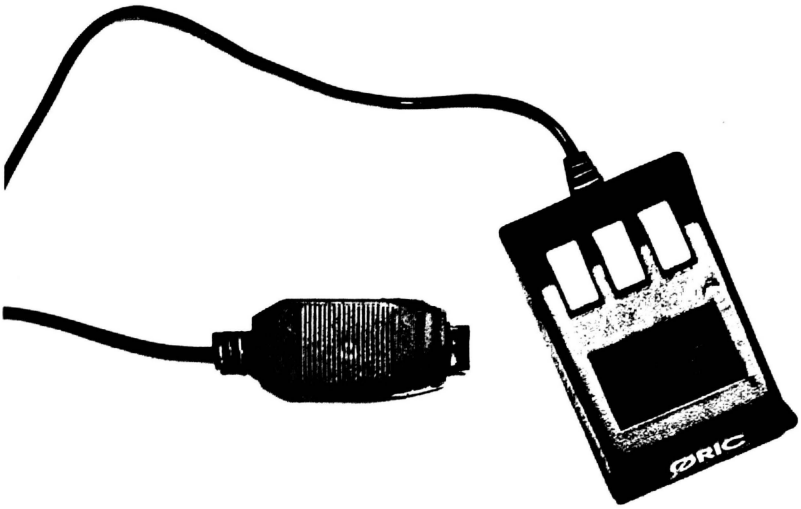
11/ Si vous possédez un ou deux Joysticks :

Pour un Joystick, branchez le sur la prise Joystick de gauche, car celle de droite est plus particulièrement prévue pour une souris. Pour deux Joysticks, branchez en un à chaque prise.



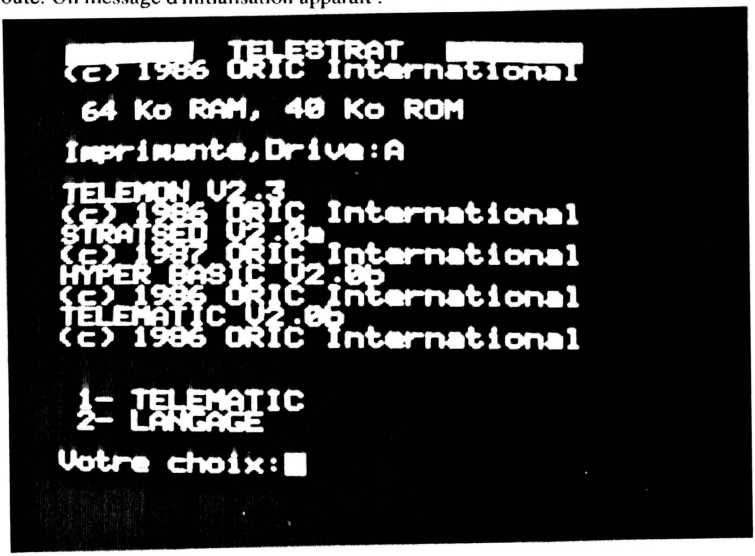
12/ Si vous possédez une souris :

La souris se relie obligatoirement à la prise Joystick de droite.



C/ MISE EN MARCHE

Tout est branché ? C'est bien. Allumez le téléviseur, puis l'alimentation, grâce au petit bouton rouge situé sur la face avant de celle-ci. Une fois ce bouton poussé, un voyant rouge s'allume, signalant le bon fonctionnement de l'alimentation. Vous entendez alors un léger bruit produit par le lecteur de disquette qui se met en marche. Une lumière rouge s'allume également sur celui-ci, prévenant de sa mise en route. Un message d'initialisation apparaît :



On vous demande maintenant de choisir si vous désirez accéder à la partie TELEMATIC du TELESTRAT, ou au langage BASIC. Tapez sur la touche 2, car c'est l'apprentissage du BASIC qui nous intéresse ici.

```
1- TELEMATIC
2- LANGAGE

Votre choix :
```

Le message '44 Ko libres' apparaît dès votre appui sur la touche 2. En dessous de ce message, une flèche et un curseur clignotant s'affichent.

CHAPITRE 2

INITIATION A LA NOTION DE LANGAGE BASIC
PRESENTATION DU CLAVIER

A/ INITIATION A LA NOTION DE LANGAGE BASIC

Tout étant installé, nous allons enfin pouvoir commencer à travailler. Mettez le son de votre téléviseur à environ la moitié de sa possibilité maximale. Les sons produits par le TELESTRAT sortent en effet par le haut parleur du téléviseur.

" Le BASIC, mais qu'est ce que c'est ?"

Voici sans doute une des premières questions que vous vous êtes posé. Un peu d'histoire : le BASIC est un langage informatique inventé en 1965 à Darmouth College, aux Etats-Unis. C'est un langage de programmation assez simple, que l'on peut en fait considérer comme un langage intermédiaire entre l'homme et la machine. Parmi tant d'autres, comme le FORTH, le FORTRAN, le COBOL, le langage C, le PASCAL, l'ADA, ... , le BASIC est le langage le plus pratiqué au monde. Il a grandement collaboré au développement de la micro-informatique familliale. Le BASIC, tout d'abord standard, est aujourd'hui un langage assez diversifié : il en existe de nombreuses versions. En fait, on pourrait même dire que chaque ordinateur possède sa propre version de ce langage, version plus ou moins performante selon les capacités spécifiques de chacun d'eux. Sur le point de la performance, nous verrons peu-après que le TELESTRAT rassemble un des plus complets, des plus puissants et des plus rapides BASIC actuels dans la catégorie des ordinateurs 8 bits. Il est également le seul à être doté d'un serveur MINITEL intégré. Le BASIC est donc un langage simple et puissant qui va vous permettre de converser, ou plutôt, de donner des ordres à votre ordinateur. Ces ordres seront entrés sous forme de mots en Anglais, car un programme BASIC est en fait un ensemble de mots, qui, combinés ensembles, définissent le travail que devra exécuter la machine .

BASIC signifie :

(B) Beginners (A) All purpose (S) Symbolic (I) Instruction (C) Code

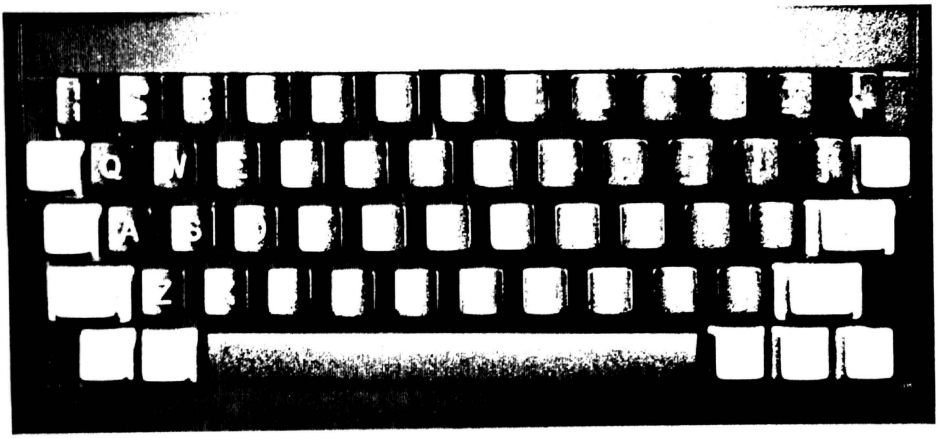
Ou plus clairement : langage pour débutants pour tout faire au moyen d'instructions symboliques codées. Parlons maintenant du langage BASIC spécifique au TELESTRAT. Celui-ci dispose de plus de 220 instructions, qu'elles soient arithmétiques, graphiques, ou sonores ...
De plus, et contrairement à la majorité des ordinateurs présents sur le marché à ce jour, le BASIC du TELESTRAT est compilé, et non pas interprété. Interprété veut dire que les instructions d'un programme en BASIC sont prises une par une, examinées, comparées à celles présentes en mémoire, puis exécutées si elles existent. On devine tout de suite que cela demande du temps. La faiblesse d'un BASIC interprété se situe donc au niveau de sa vitesse d'exécution, qui est généralement peu élevée.

Une nouvelle forme de BASIC a fait son apparition il y a de cela quelques années : le BASIC compilé. La compilation consiste à transcrire le programme BASIC de départ en un programme directement exécutable par la machine : le LANGAGE MACHINE. Il en résulte une très grande rapidité d'exécution : de 4 à 100 fois plus rapide que certains autres ordinateurs dotés d'un BASIC interprété. Un exemple ? Exécuter une boucle vide 10000 fois prend 1/10 ème de seconde alors que cela prend environ 100 fois plus de temps sur d'autres ordinateurs. C'est ça, le TELESTRAT !

Il n'est pas du tout nécessaire de connaître la langue anglaise pour pratiquer le BASIC. De même, on ne vous demande pas non plus de sortir de la SILICON VALLEY ! Etre attentionné, sérieux et appliqué, voila qui devrait grandement suffire à vous ouvrir les portes du monde de la programmation en langage BASIC.

B/ PRESENTATION DU CLAVIER

Regardez le clavier : il y a deux types de touches, les noires et les rouges. En noir, vous devez reconnaître les mêmes touches que celles d'une machine à écrire. Le clavier du TELESTRAT est un QWERTY, ou clavier anglo-saxon.
Que ceux qui ont l'habitude d'un clavier français AZERTY ne s'inquiètent pas, car il est également disponible, comme nous le verrons prochainement. Pour le moment, contentez vous du clavier QWERTY, qui n'en reste pas moins le plus répandu dans le monde.



De chaque côté du clavier, on trouve 11 touches rouges. Ce sont les touches de fonctions. Nous les détaillerons au fur et à mesure de notre progression dans l'apprentissage du langage BASIC.
Mais revenons en à notre TELESTRAT. Il est toujours allumé devant nous, attendant sagement qu'on lui donne des instructions à exécuter. Tout à gauche de l'écran, vous apercevez une flèche suivie d'un curseur qui clignote. Un curseur est un petit carré blanc clignotant permettant de savoir où sera affiché le prochain caractère entré au clavier. Ce curseur se déplace d'un cran à chaque fois qu'une touche est enfoncée, et va à la ligne lorsqu'il arrive à la fin de l'écran.

Allons, n'hésitez pas, tapez sur une touche ... Il faut savoir que le TELESTRAT ne peut en aucun cas être endommagé par ce que vous allez taper, quoi que ce soit que vous tapiez. Donc, pas de danger !

Ecrivez par exemple : BONJOUR

Rien ne se passe , seul le mot est présent à l'écran ... Maintenant que vous avez entré BONJOUR, tapez sur la touche rouge marquée 'RETURN'. Que se passe-t'il ? Le même bruit que celui produit à l'allumage du TELESTRAT réapparaît, et la lumière du lecteur de disquette s'éclaire. Puis, environ une seconde plus tard, le message '`fichier absent`' s'affiche. Pourquoi ? Parce que le TELESTRAT a comparé le mot BONJOUR à tous les mots BASIC de sa mémoire, et ne l'a pas trouvé.

Il a donc considéré cette chaîne de caractères comme le nom d'un programme à charger à partir du lecteur de disquette. Le programme de nom 'BONJOUR' n'existant pas, il a tout simplement affiché le message d'erreur '`fichier absent`'.
Tout mot tapé au clavier et de 9 caractères maximum est donc considéré comme le nom d'un fichier à charger à partir du lecteur de disquette et entrainera la mise en marche de celui-ci.

Tapez : BONJOUR TELESTRAT (Puis RETURN)

Le message '`nom de fichier incorrect`' s'inscrit alors, car BONJOUR TELESTRAT n'est évidemment pas un mot du BASIC. Ce ne peut pas non plus être un nom de programme à charger, car ce message fait plus de 9 caractères. Le TELESTRAT nous prévient alors que si c'est un nom de fichier que l'on voulait charger, ce nom est incorrect, car sa longueur est supérieure à 9 caractères.

Au dessus de la touche marquée RETURN se trouve la touche DEL. Comme RETURN, DEL est une touche de fonction. Elle permet d'effacer le dernier caractère entré, et donc le caractère juste derrière le curseur.

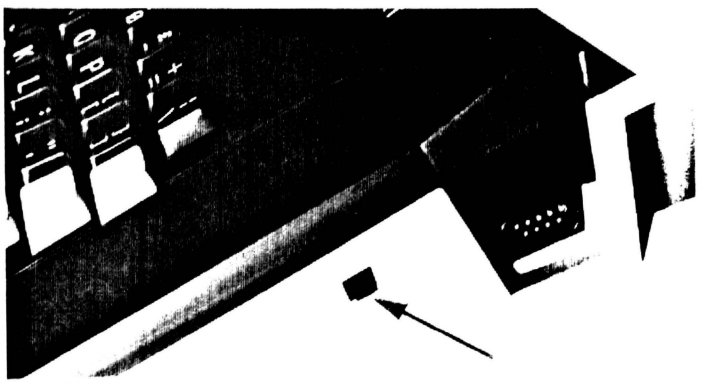
Exemple : appuyez sur la lettre G puis sur la touche DEL. Le G s'efface et le curseur recule d'une colonne.
Autre exemple : tapez HELLO puis 5 fois la touche DEL pour effacer ce mot.

DEL est une abréviation de DELETE qui se traduit par 'effacer'.

Pour obtenir l'affichage de la même lettre plusieurs fois de suite, il n'est pas nécessaire d'appuyer successivement sur la lettre recherchée. Il suffit de laisser enfoncé la touche concernée. La répétition automatique aura lieu immédiatement et conduira à l'apparition de la lettre à l'écran de façon répétitive tant que la touche enfoncée n'aura pas été relachée.

Les deux touches SHIFT en rouge permettent d'accéder aux autres lettres du clavier. Par exemple, si vous tapez sur la touche 1, c'est 1 qui va s'afficher à l'écran. Mais pour le point d'exclamation situé au dessus du 1, il faudra appuyer sur une des deux touches SHIFT en la laissant enfoncée, puis appuyer sur 1.

Lors de l'exécution d'un programme, il peut quelquefois arriver que celui-ci fasse n'importe quoi, et ne vous rende pas la main (Qu'il ne vous redonne pas le contrôle du clavier).
Dans ce cas là, appuyez sur le bouton 'RESET'. C'est un petit bouton noir situé sur le côté droit de votre TELESTRAT, près de la prise Joystick. Dès l'appui sur celui-ci, l'HYPERBASIC vous redonnera la main en ne détruisant pas les données déjà mémorisées, comme un programme BASIC par exemple. Enfoncez ce bouton pour essayer. L'écran se vide alors et le curseur réapparaît en haut et à gauche de l'écran.



Vous avez sans doute remarqué, si le son de votre téléviseur est bien réglé, que l'appui sur une touche émettait systématiquement un bruit : 'clic'.

Ce 'clic' vous confirme que la touche a bien été pressée. Il s'arrête en faisant un CTRL-F. Enfoncez d'abord la touche de fonction CTRL, puis, cette touche étant toujours appuyée, pressez la touche F. Le bruit des touches a désormais disparu, et cela, jusqu'à une nouvelle pression sur CTRL-F.

CTRL-T est la bascule majuscules / minuscules. Un premier appui sur CTRL-T (CTRL enfoncé et appui sur T) provoquera le passage en mode minuscules. Un second appui sur CTRL-T repassera le TELESTRAT en mode majuscules.

Les 4 touches flèchées près de la barre d'espace servent bien sûr à déplacer le curseur. A noter qu'il est obligatoire d'entrer une instruction BASIC sur une ligne débutant par une flèche. (⇒)

CHAPITRE 3

NOTIONS PRELIMINAIRES

A/ NOTION DE CALCULS SIMPLES

La première instruction BASIC que vous allez apprendre est l'instruction PRINT. PRINT veut dire imprimer ou afficher en Français.

Entrez : **PRINT 2** (Puis **RETURN**)

Le TELESTRAT affiche un **2**.

Essayez : **PRINT 2+3** et appuyez sur **RETURN**.

Le chiffre 5 apparaît alors à la ligne suivante. Vous avez en effet demandé au TELESTRAT de vous donner le résultat de l'opération 2+3, qui est 5. + est le signe d'addition.

Essayez aussi : **PRINT 500000+150000**, puis **RETURN**.

Le résultat est 650000. Cela vous montre que le TELESTRAT peut exécuter des calculs très élevés. Et bien plus que les centaines de milliers, puisque le nombre maximum positif possible est de l'ordre de 1 E + 38, et le nombre maximum négatif est - 1 E + 38. Ces nombres sont exprimés en puissances de 10 qui se notent E. Croyez moi, de nombreuses machines à calculer envient la puissance de calcul de votre TELESTRAT. Il est non seulement puissant, mais rapide. Pas question d'attendre 1 à 2 secondes pour connaître le COSinus ou le SINus d'un nombre. Tout s'affiche instantanément.

Le signe de division est /.

PRINT 4/2 (Puis **RETURN**)

Donne le résultat de la division 4/2, soit 2.

PRINT 5/2 (Puis **RETURN**)

Le nombre affiché est 2.5. Remarquez que lorsque le résultat n'est pas entier, l'ORIC TELESTRAT affiche tous les nombres après la virgule, du moins après le point, car la virgule équivaut à un point dans l'informatique (Notation anglaise).

A partir de maintenant, nous ne vous dirons plus d'appuyer sur **RETURN** après chaque ordre entré. Il faudra le faire automatiquement. Ce que vous marquez n'est en effet examiné que lors de l'appui sur **RETURN** (validation).

PRINT 8.242617/1.3394881

De même que + permet d'additionner, et que / de diviser, - sert à soustraire.

Exemple : **PRINT 10-5**

Ou encore : **PRINT 2-5**

Le résultat de 2-5 est -3. Les nombres négatifs existent donc aussi.

Ainsi :

```
PRINT -4-5
```

donnera -9. Le signe de multiplication s'obtient en faisant SHIFT 8, ce qui donne *. Ce signe s'emploie dans le domaine de l'informatique, et n'est donc pas le même que x utilisé couramment (qui ne peut être distingué du x minuscule).

```
PRINT 2*2*2
```
donne 8.

```
PRINT 2*5*2*4
```

Pour élever un nombre à la puissance, il convient d'utiliser le signe ↑ qui s'obtient par SHIFT 6.

Exemple :

```
PRINT 2↑3
```
ou 2 puissance 3 est égal à 3 fois le nombre 2 multiplié par lui même, soit 2*2*2=8.

```
PRINT 3↑5
```
donne 3*3*3*3*3= 243

En bref, nous venons de voir les 5 signes opératoires les plus élémentaires qui sont :

+ Addition.
- Soustraction.
* Multiplication.
/ Division.
↑ Elévation à la puissance.

Nous avons également appris que le TELESTRAT acceptait des nombres non entiers, tel que 2.35456, ou des nombres très élevés tel que 13948584755677478, ou bien des nombres négatifs entiers ou non, tel que -451.523.

Pour assurer la priorité dans un calcul, on utilise les parenthèses ( ). Imaginons que nous voulions effectuer 2+3, puis, multiplier le tout par 4. Cela donne normalement :

```
PRINT 2+3*4
```

Mais, PRINT 2+3*4 va-t-il donner le résultat attendu ? Non, car 2+3 font 5, et 5 multiplié par 4 donne 20. Or, PRINT 2+3*4 renvoie 14. Pourquoi ? Tout simplement parce que la multiplication est prioritaire, et que le TELESTRAT effectue d'abord 3*4, ce qui fait 12, et 12 + 2 = 14.

Vous comprenez maintenant l'importance des parenthèses :

```
PRINT (2+3)*4
```

Cette fois-ci, on obtient bel et bien le résultat escompté, 20. En effet, tout calcul entre parenthèses est prioritaire, même sur une multiplication. 2+3 sera d'abord effectué, puis le résultat de cette addition sera multiplié par 4. D'où (2+3)*4 = 20 à ne pas confondre avec 2+3*4 qui est en fait 2+(3*4) si l 'on met des parenthèses, et qui donne 14.

Voici la règle des priorités que respecte le TELESTRAT :

- Les calculs entre parenthèses.
- Les élévations à la puissance.
- La multiplication et la division (même priorité).
- L'addition et la soustraction (même priorité).

Ce tableau vous explique que pour n'importe quel calcul, les opérations entre parenthèses seront d'abord exécutées, puis les élévations, les multiplications, ... s'il y en a.

L'HYPERBASIC accepte des calculs très compliqués, pourvu que le résultat final soit compris entre - 1 E + 38 et 1 E + 38.

Exemple: PRINT 10*2+(2*4)-3/6+5*2-10*10+6*7/(3+2)-53

Vu la puissance et la rapidité de calcul du TELESTRAT, on en conclu rapidement qu'une machine à calculer ne fait pas le poids face à ce génie-ordinateur.

B/ NOTION DE CHAINES DE CARACTERES

L'instruction PRINT sert aussi à l'affichage de chaînes de caractères à l'écran. Cependant, il est indispensable de mettre au début et à la fin d'une chaîne de caractères des guillemets ("), et ceci, pour que le TELESTRAT sache qu'il s'agit d'une chaîne de caractères.

Exemple : PRINT "ORIC TELESTRAT"

Va afficher la chaîne de caractères ORIC TELESTRAT sur la prochaine ligne. Le message est obligatoirement entouré de guillemets.

PRINT "COMMENT ALLEZ VOUS AUJOURD'HUI ?"

Il ne faut pas confondre PRINT "4+2" et PRINT 4+2.

Le premier PRINT affichera la chaîne de caractères 4+2, tandis que le deuxième donnera 6.

L'instruction PRINT peut être remplacée par un point d'interrogation. Cette possibilité s'explique par le fait que PRINT est une des instructions les plus utilisées dans un programme BASIC, car elle permet de visualiser n'importe quelle valeur, chaîne, variable, ... à l'écran. On a donc cherché à simplifier celle-ci.

? "ORIC"

donne le même résultat que :

PRINT "ORIC".

C/ NOTION DE VARIABLES

Si vous faites PRINT DODO au lieu de PRINT "DODO", on obtient le chiffre 0. Le TELESTRAT a en effet considéré DODO comme une variable numérique. Cette variable n'ayant pas été déclarée (une valeur ne lui étant pas assignée), elle vaut 0 (par défaut).
Donnons tout de suite une valeur à la variable DODO : DODO=5 demande que la variable DODO soit égale à 5. Le TELESTRAT a alors mis le nom et la valeur de la variable DODO dans une partie de sa mémoire.

Et à chaque fois que vous voudrez connaître la valeur de DODO, il faudra entrer :

PRINT DODO

Et la valeur 5 sera donnée.

On comprend tout de suite pourquoi une variable s'appelle 'variable'. Si vous faites DODO=12, la valeur de DODO a désormais changée, et vaut 12. Un PRINT DODO vous le confirmera.

On a donc la possibilité de changer la valeur d'une variable à n'importe quel moment.

LIVRE=341.12994 donne la valeur 341.2994 à la variable LIVRE.

La valeur d'une variable peut se définir de diverses façons. En voici quelques unes :

VAR = 10	: donne la valeur 10 à VAR.
VAR = 2+7	: donne le résultat de 2+7 à la variable VAR, soit VAR=9.
VAR = 6/2	: donne le résultat de 6/2 à VAR. VAR=3
VAR = 1.32	: VAR vaut 1.32
VAR = 4*A	: donne à VAR la valeur de la variable A multipliée par 4. Si A=2, alors VAR sera égal à 4*2 =8.
VAR = VAR*2	: Multiplie la valeur de VAR par 2. Si VAR=2, alors VAR vaudra 4 après un VAR=VAR*2.
VAR = A*B/43	: VAR est égal au résultat des variables A et B divisées par 43.

Dans le cas d'un VAR=VAR+4, la variable VAR sera augmentée de 4. Si VAR=2 au départ, VAR vaudra 6 après un VAR=VAR+4, bien que l'équation mathématique 2=2+4 ne soit pas vraie.

Chaque nom de variable que vous choisirez peut être composé d'un ou de plusieurs caractères, avec un maximum d'environ 110 caractères.

Exemple de longue variable :

BONJOURCOMMENTALLEZVOUSCETAPRESMIDIILFAITTRESBEAU=10

PRINT BONJOURCOMMENTALLEZVOUSCETAPRESMIDIILFAITTRESBEAU

Réponse du TELESTRAT : 10

Il y a une limite au choix du nom d'une variable : il ne faut pas que ce nom commence par un mot BASIC. Le seul mot BASIC que vous connaissez pour le moment est PRINT. Et bien utilisons le.

PRINT=5

Le TELESTRAT n'admet pas PRINT=5 car PRINT est un mot du BASIC.

ORIC=2

ORIC=2 n'est pas accepté non plus, car OR est un mot du BASIC et le mot ORIC est composé de OR.

Un nom de variable peut être composé d'une instruction BASIC, à la seule condition que cette instruction ne soit pas au début de la variable. Exception faite pour les instructions : OR, AND, XOR, TO, THEN, ELSE, STEP.

APRINT=1 est une variable possible, car l'instruction PRINT ne commence pas par le nom de la variable. Par contre, PRINTING=4 n'est pas valide, car PRINT est au début du nom de la variable PRINTING.

Une autre restriction concernant les noms de variables: ils ne peuvent pas être composés de signes, mais seulement de chiffres et de lettres.

Exemple : A1=10 est accepté.

W: =10 n'est pas accepté car : est un signe.

Les noms de variables en minuscules sont automatiquement transformés en majuscules.

Le nom d'une variable doit toujours commencer par une lettre, et non pas par un chiffre. La variable A1 est valide, mais 1A ne l'est pas.

D/ NOTION DE VARIABLES ALPHANUMERIQUES

Parallèlement aux variables numériques qui concernent les nombres, les variables alphanumériques s'utilisent avec des chaînes de caractères. Les restrictions pour le choix du nom d'une variable alphanumérique sont les mêmes que celles des variables numériques. Le dernier caractère du nom d'une variable alphanumérique est '$'. Ceci est obligatoire. Ce signe, le dollar, permet au TELESTRAT, ainsi qu'à l'utilisateur, de différencier une variable numérique d'une variable alphanumérique.

Exemple : A$="L'HYPERBASIC, C'EST CHIC !"

Comme pour un PRINT "CHAINE", la chaîne de caractères suivant le '=' est entre guillemets ou doubles quotes. (").
Après avoir donné au TELESTRAT la chaîne de caractères correspondant à A$, un simple PRINT A$ réaffiche le message que contient A$.

Autre exemple :

CHAINE$="TELESTRAT, L'ORDINATEUR QUI EPATE. (EPATE, EPATE, OUI, MAIS DES ...)"

PRINT CHAINE$

Le nombre maximum de caractères que peut admettre une variable alphanumérique est 255. Si ce nombre est dépassé, le message d'erreur '`chaine trop longue`' est affiché.

Il y a plusieurs façons de définir le contenu d'une variable alphanumérique. En voici quelques exemples :

DL$="CHAINE"	:	le message que contient DL$ est "CHAINE".
DL$=AL$+"SALUT"	:	la chaîne de caractères que contient DL$ est l'addition de la variable alphanumérique AL$ et de "SALUT"
DL$=R$	:	le contenu de DL$ est égal au contenu de R$.
DL$=HELLO$+BOY$:		DL$ est la somme de la chaîne que contient HELLO$ et de la chaîne que contient BOY$.

Le signe + additionne deux chaînes de caractères.

A$="ORIC "

```
B$="TELESTRAT"

PRINT A$+B$
```

Donnera ORIC TELESTRAT. PRINT A$+B$ demande que les chaînes A$ et B$ soient affichées sur la même ligne. L'espace laissé après le "ORIC " dans la chaîne A$ compte pour 1 caractère.

Au lieu de faire PRINT A$+B$, pourquoi ne pas essayer :

```
A$="ORIC"

B$="TELESTRAT"

C$=A$+B$

PRINT C$
```

On réunit dans la variable C$ les variables A$ et B$ grâce au signe +. Il ne reste plus qu'un PRINT C$ à exécuter pour voir apparaître le message 'ORIC TELESTRAT'.

E/ NOMBRES ET CHAINES REUNIS

Pour afficher un nombre, 10 par exemple, on tape :

```
PRINT 10
```

Pour écrire un message, 10 aussi, cela donne :

```
PRINT "10"
```

Pour fusionner les deux écritures, on utilise le signe ';'.

```
PRINT "10";10
```

Un autre exemple :

```
PRINT "MON AGE=";15
```

Va afficher MON AGE= 15.

Le ';' écrit la chaîne "MON AGE" et le nombre 15 à la suite l'un de l'autre, et sur la même ligne.

```
PRINT "J'AI";20;" ANS."
```

Les ';' ne sont pas indispensables. Le dernier PRINT s'écrit donc aussi :

```
PRINT "J'AI"20" ANS."
```

Comme l'exemple ci-dessus vous le montre, l'inexistance de ';' ne facilite pas la lisibilité d'un programme.

```
A$="JE SUIS AGE DE "

AGE=25
```

```
B$=" ANS"

PRINT A$;AGE;B$
```

Voici d'autres exemples insistant sur la différence entre une chaîne de caractères et une opération mathématique, ou un nombre.

```
PRINT "5 =";5
PRINT "2X4 =";8
PRINT "4+7 =";4+7
PRINT "4:2 =";4/2
PRINT "2 A LA PUISSANCE 5 DONNE";2*2*2*2*2
```

L'utilisation des signes X et : pour multiplié et divisé est permise, car ceux-ci sont entre guillemets, et font parti d'une chaîne de caractères: ils sont là en tant que message et non en tant qu'opérateur.

F/ NOTION DE PROGRAMME

Toutes les commandes données depuis le début de ce chapitre ont été tapées en mode direct. C'est ce que l'on appelle aussi des commandes entrées en mode immédiat. Aussitôt entrées au clavier, aussitôt exécutées, et aussitôt oubliées. Pour exécuter de nouveau la même instruction, il faut la retaper. Il existe un moyen permettant de garder les instructions en mémoire, et de les exécuter autant de fois que l'on le désire : un programme composé de lignes BASIC.

Un programme est un ensemble de lignes. Sur chaque ligne, on met une ou plusieurs instructions. On choisit le numéro des lignes entre 0 et 65535, puis on y entre des instructions BASIC. Les lignes sont exécutées par ordre croissant : la ligne 10 passera en toute logique avant la ligne 15, elle même passant avant la ligne 1000 ...

Un exemple de programme BASIC :

```
10    PRINT "COMMENT ";
20    PRINT "TROUVEZ VOUS LE ";
30    PRINT "BASIC ?"
```

Pour entrer une ligne, on tape son numéro puis la ou les instructions qu'elle contiendra. On appuie ensuite comme d'habitude sur la touche **RETURN**. Les lignes ci-dessus sont numérotées de 10 en 10 pour permettre l'insertion d'une ou de plusieurs lignes suplémentaires en cas d'erreur, ou d'oubli.

Une instruction spéciale permet de démarrer un programme mis en lignes : RUN. RUN provoque l'éxécution du programme BASIC présent en mémoire. Vous pouvez faire RUN autant de fois que vous le souhaitez.

Après chaque programme copié, il faudra désormais entrer l'instruction RUN pour le mettre en route. Tapez ce programme :

```
10   PRIX=200
20   TVA=PRIX*0.186
30   PRIXFINAL=PRIX+TVA
40   PRINT "PRIX TOTAL :";PRIXFINAL
```

Faites RUN. Le prix TTC d'un article valant 200 francs auquel est ajouté 18,6 % de taxe vous est donné.

CHAPITRE 4

EDITEUR PLEINE PAGE

Avant de continuer notre progression dans l'apprentissage du langage BASIC, il est préférable, voire nécessaire d'en savoir un peu plus sur l'éditeur pleine page du TELESTRAT.
On définit par 'éditeur pleine page' tout ce qui permet d'afficher, d'effacer, de déplacer, ... , des caractères. En bref, l'éditeur pleine page s'occupe de la gestion de l'affichage.
Commençons par quelque chose de simple.

Copiez :

```
10  PRINT "EDITEUR PLEINE PAGE"
20  PRINT "SUR ORIC TELESTRAT."
```

Faites RUN pour voir apparaître les messages que contiennent les deux PRINT. Si vous voulez maintenant revoir le listing de ce programme, c'est à dire les 2 lignes que vous venez d'entrer, tapez l'instruction **LIST (RETURN)**. LIST va afficher à l'écran le programme BASIC que vous avez précédemment copié.

Les espaces entre chaque instruction ne sont pas obligatoires. Lorsque vous devez copier la ligne :

```
10  PRINT "HELLO"
```

Vous pouvez aussi entrer :

```
10PRINT"HELLO "
```

Ou :

```
10      PRINT           "HELLO"
```

Pour une raison de lisibilité d'un programme, le TELESTRAT génère automatiquement les espaces où il le faut lorsque vous tapez la fonction LIST. Et là n'est pas son moindre avantage. En voici un exemple:

```
10PRINT "GENERATION D'ESPACE AUTOMATIQUE"
```

Après un RETURN, la ligne est entrée en mémoire. après un LIST, la ligne 10 est maintenant :

```
10      PRINT "GENERATION D'ESPACE AUTOMATIQUE"
```

Le TELESTRAT a mis des espaces à l'endroit où il le jugeait indispensable pour nous faciliter la lecture.

Au fait, vous auriez aussi pu faire :

```
10 print      "ESPACE"
```

Et après un LIST, cela nous donne :

```
10      PRINT "ESPACE"
```

Le TELESTRAT comprend en effet non seulement les mots BASIC écrits en majuscules, mais aussi en minuscules. Lors d'un LIST, les mots en minuscules seront tous changés en majuscules, sauf s'il s'agit d'un message à afficher (message entre guillemets).

Passons à autre chose . Entrez les deux lignes ci-dessous :

```
10 PRINT "ESSAI DE DESTRUCTION"
20 PRINT "DE LIGNES BASIC."
```

Supposons maintenant que vous vouliez enlever la ligne 20.

Faites :

20 (RETURN)

Puis **LIST**. Vous constatez que la ligne 20 a disparu. Seule la ligne 10 est encore présente.
Conclusion : une ligne se détruit en tapant son numéro, puis RETURN. La ligne est alors enlevée.

Ce livre contient de nombreux petits programmes. A chaque fois que vous en copierez un nouveau, il faudra auparavant détruire le précédent. Les lignes de l'ancien et du nouveau programme pourraient en effet se mélanger. L'instruction NEW détruit entièrement le programme résidant en mémoire.

Exemple : il doit vous rester la ligne 10 du programme de tout à l'heure. Tapez NEW, et LIST. La ligne a alors disparu !

Sur TELESTRAT, la frappe se fait toujours en insertion. Cela veut dire que tout nouveau caractère entré repoussera tous ceux se trouvant à sa droite d'un cran. Ce décalage ne sera pas visible si tous les caractères situés à droite de celui qui vient d'être entré sont des espaces.

Faites :

10 PRNT "TELEVISION"

Et n'appuyez pas sur RETURN. Le curseur est à la fin de la ligne 10, et vous venez de vous apercevoir qu'il manquait un I entre le R et le N de PRNT pour que cela donne PRINT.
Déplacez le curseur jusqu'à ce qu'il soit sur la lettre N (Utilisez les 4 touches flèchées rouges.).

Tapez sur la touche I. Le I s'inscrit alors, et tous les caractères après le I sont repoussés d'un cran à droite. Le mot est maintenant PRINT. Appuyez sur **RETURN** pour valider la ligne. Vous voyez donc que l'on peut positionner le curseur sur une ligne quelconque, et insérer des caractères n'importe où.

Vous pouvez aussi effacer n'importe quel caractère d'une ligne.
Effaçons donc TELE de TELEVISION pour qu'il ne reste plus que VISION. Remontez le curseur sur la ligne 10, et placez le curseur sur le V de TELEVISION, et appuyez 4 fois sur DEL. TELE s'efface et il ne reste plus que VISION. Appuyez sur **RETURN** pour valider cette nouvelle ligne.

La ligne est maintenant :

```
10   PRINT "VISION"
```

Une autre façon d'effacer des caractères s'obtient avec **SHIFT + DEL**. Au contraire de DEL qui efface le caractère situé derrière le curseur et déplace d'un cran à gauche tous les caractères après le curseur, **SHIFT + DEL** efface le caractère situé sous le curseur, ce qui provoque un décalage à gauche des caractères restant après le curseur.

Replacez le curseur sur la ligne 10 et positionnez-vous juste après le PRINT. Appuyez sur SHIFT + DEL en laissant environ une seconde les deux touches enfoncées pour qu'il y ait une répétition automatique de l'effacement. Vous voyez alors le message "VISION" disparaître. Tous les caractères après le curseur ont été effacés par SHIFT + DEL.

Voici maintenant d'autres fonctions de l'éditeur pleine page :

SHIFT + ⇐ ET SHIFT + ⇒ :

La touche SHIFT alliée aux touches fléchées droite et gauche permet un déplacement rapide le long de la ligne courante.

Ecrivez en mode direct (sans no de ligne) :

MATHEMATIQUES (n'appuyez pas sur RETURN).

Si vous appuyez sur SHIFT+ ⇒, le curseur se place directement à la fin du mot MATHEMATIQUES .

Et si vous faites maintenant SHIFT + ⇐, le curseur revient automatiquement en début de ligne. Vous en déduisez donc aisément que :

SHIFT + ⇐ : aller au début de la ligne.

SHIFT + ⇒ : aller à la fin de la ligne.

CTRL-C : permet de sortir d'une ligne (aller à la ligne) sans la valider.

CTRL-N : efface toute la ligne où se trouve le curseur.

CTRL-X : efface toute la ligne à partir de la position courante du curseur.

Exemple :

```
10  PRINT "ESSAI DE CTRL-X"
```

Positionnez-vous après PRINT et faites CTRL-X (CTRL d'abord enfoncé puis appui sur X.). Tous les caractères situés après le curseur sont effacés. Il ne reste que PRINT. Faites RETURN.

CTRL-S : provoque la recopie immédiate de l'écran sur imprimante. (Si vous en possédez une ...)

CTRL-R : écho imprimante. Tout ce que vous marquerez après avoir fait un CTRL-R sera affiché sur imprimante à chaque appui sur RETURN. CTRL-R s'utilisant en mode ON/OFF, un autre appui sur CTRL-R annulera la sortie sur imprimante de tout ce qui est affiché à l'écran.

CTRL-T : mode ON/OFF. Passage en minuscules / majuscules.

Une ligne BASIC accepte un maximum de 110 caractères. Si ce nombre est dépassé, les caractères en trop seront effacés lors d'une insertion ou d'une validation.

Lorsque vous tapez le mot LIST pour voir réapparaître votre programme BASIC à l'écran, il est possible :

• d'appuyer sur n'importe quelle touche du clavier pour stopper momentanément le défilement du programme et de réappuyer à nouveau sur une touche pour que le défilement reprenne.

• d'appuyer 2 fois sur la touche 'ESC' (ESCAPE) pour stopper définitivement le défilement. (1er appui sur ESC = arrêt momentané du défilement et 2ème appui = arrêt définitif.). Pour arrêter le listing (apparition des lignes), on peut également utiliser CTRL-C.

Si le listing de votre programme est plus long que ce que peut contenir l'écran, il s'en suit un scrolling vertical vers le haut (scrolling = défilement), ce qui fait disparaître un certain nombre de lignes vers le haut de l'écran. Et réciproquement, un certain nombre de lignes apparaissent en bas. Quand un listing prend plus de place que l'écran, le TELESTRAT vous offre une nouvelle possibilité : LE DEPLACEMENT LE LONG D'UN LISTING.

Exemple : quand le listing de votre programme défile, appuyez sur CTRL-C pour le stopper. Puis, remontez tout en haut de l'écran et appuyez une fois encore sur la ⇑. Une ligne apparaît alors en haut de l'écran. Cette ligne est celle précédant l'ancienne ligne qui était en haut de l'écran. Vous pouvez encore appuyer sur la flèche du haut. A chaque fois, la ligne précédant celle se trouvant tout en haut apparaît. Descendez maintenant le curseur tout en bas de l'écran grâce à la ⇓. A chaque fois que vous appuierez sur la flèche du bas, le curseur étant tout en bas, la ligne suivant celle présente en bas de l'écran s'affichera. C'est cela qu'on appelle le déplacement le long du listing : on se déplace à volonté en avant/arrière dans le listing grâce aux flèches bas et haut. Vous pouvez bien sûr laisser les touches haut et bas appuyées pour qu'il y ait une répétition automatique, et de même, une plus grande vitesse de déplacement dans le listing.

Le clavier du TELESTRAT est bufferisé. Cela signifie que même lorsqu'un programme est exécuté, les touches sont examinées et prises en compte. Si par exemple vous appuyez sur une touche lors du fonctionnement d'un programme, cette touche ne sera pas affichée car le TELESTRAT est en train de travailler. Quoi qu'il en soit, il met cette touche dans une partie de sa mémoire, et le caractère correspondant à celle-ci sera affiché dès que l'exécution du programme est terminée.

Une des grandes qualités du TELESTRAT réside dans la gestion des erreurs. Il faut en effet savoir qu'il est impossible de faire une erreur. Le TELESTRAT nous prévient dès que nous en faisons une, en affichant le type de l'erreur commise.

Entrez :

PRINT 4+ (Puis RETURN)

Un message apparaît alors en haut de l'écran : '`Attention: expression`'. On vous prévient que vous avez oublié d'entrer un nombre, ou le nom d'une variable, après le +. Entrez un 5, puis **RETURN**. Le TELESTRAT accepte alors la ligne et exécute l'opération 4+5. Voici donc ce qui est fait avant l'exécution ou la validation (appui sur RETURN) d'une ligne :

• On prend les instructions une à une et on regarde s'il y a une erreur.

• S'il n'y a pas d'erreur, alors on exécute les instructions.

• S'il y a une erreur, alors on affiche un message en haut de l'écran prévenant l'utilisateur de la faute qu'il a commise.

• Et de plus, le TELESTRAT se paye le luxe de déplacer le curseur sur le mot qu'il n'a pas compris! Il ne vous reste alors plus qu'à le modifier en conséquence.

Un chapitre entier est consacré à la gestion des erreurs. Vous y trouverez tous les messages d'erreur du TELESTRAT. Nous ne pouvons pas vous en dire plus pour le moment, car vous ne connaissez pas encore les instructions du BASIC ...

CHAPITRE 5

LA GESTION DU CLAVIER

Un ordinateur ne possède pas seulement des instructions capables d'afficher des chaînes de caractères, des nombres ou des résultats d'opérations diverses. Il est également pourvu d'instructions spécialement destinées à la gestion du clavier. Sur TELESTRAT, 3 mots BASIC se chargent de cette gestion. Ils permettent au programmeur (vous) de saisir aussi bien un nombre qu'une chaîne de caractères.

INPUT

La première instruction de gestion du clavier est INPUT. Elle interrompt le déroulement d'un programme et ne reprend ce déroulement que lorsque l'utilisateur a entré une réponse au clavier. (chaîne de caractères ou nombre).

La syntaxe de INPUT est la suivante : INPUT suivi du nom de la variable qui contiendra la réponse entrée.

Exemple :

```
10   PRINT "ENTREZ UN NOMBRE QUELCONQUE : ";:INPUT A
20   PRINT :PRINT "VOUS VENEZ D'ENTRER LE NOMBRE ";A
```

Le TELESTRAT attend en ligne 10 que vous ayez entré un nombre quelconque, puis appuyé sur **RETURN** pour valider votre réponse. (Vous pouvez aussi utiliser la touche DEL pour effacer un ou plusieurs caractères en cas d'erreur).

Explication du programme : en ligne 10, le mot BASIC PRINT affiche ce que l'on attend que l'utilisateur fasse. Dans ce cas, il s'agit de demander l'entrée d'un nombre. Le point-virgule (;) après la chaîne de caractères dit au TELESTRAT de mettre la prochaine chaîne à écrire sur la même ligne que celle précédemment inscrite, dans le but d'afficher la réponse sur la même ligne que la question. En clair, le ';' juxtapose les chaînes les unes à côté des autres, ce qui n'est pas fait en temps normal, le TELESTRAT allant automatiquement à la ligne après chaque chaîne affichée.

En ligne 10, INPUT A demande qu'un nombre soit saisit au clavier. A étant une variable numérique, seul un nombre sera accepté. Si vous entrez une chaîne de caractères, la variable A vaudra 0. A prend la valeur donnée lors de votre réponse à : 'ENTREZ UN NOMBRE QUELCONQUE'.

Vous pouvez aussi faire en mode direct : INPUT NOMBRE

Entrez 5 puis RETURN. Un PRINT NOMBRE vous montrera que la variable NOMBRE vaut 5.

```
10   PRINT "ENTREZ UN NOMBRE :";
20   INPUT NOMBRE1
30   PRINT "ENTREZ UN SECOND NOMBRE :";
40   INPUT NOMBRE2
50   PRINT
60   PRINT "LE TOTAL DE VOS DEUX NOMBRES EST :";NOMBRE1+NOMBRE2
```

INPUT ne se limite pas qu'aux nombres. Pour une chaîne de caractères, la variable suivant INPUT doit se déclarer comme étant une variable alphanumérique. Faites en mode direct : **INPUT CHAINE$**

Entrez une chaîne de caractères suivie de l'habituel **RETURN**. Puis : **PRINT CHAINE$**

Et la chaîne que vous avez entré lors du INPUT sera affichée. La seule différence entre la syntaxe de INPUT pour la saisie d'un nombre et d'une chaîne de caractéres est donc la variable suivant INPUT. Cette variable est numérique si l'on veut saisir un nombre, et est alphanumérique (terminée par $) si l'on désire une chaîne de caractères.

• Si l'on rentre une chaîne de caractères lorsque l'on demande un nombre, la valeur de la variable devant contenir ce nombre sera égale à 0.

• Par contre, un nombre est accepté comme une chaîne de caractères, car les chiffres ne sont ni plus ni moins que des caractères.

Exemples : **INPUT A**

Votre réponse : **IMPRIMANTE (RETURN)** puis : **PRINT A**

Réponse : **0**. Vous avez entré une chaîne de caractères alors que l'on vous demandait un nombre.

Essayez maitenant : **INPUT A$**

Votre réponse : **125 (RETURN)** puis : **PRINT A$**

Réponse du TELESTRAT : **125** car une chaîne de caractères peut comporter des chiffres.

La réponse à une question formulée avec un INPUT ne doit pas excéder 110 caractères.

```
 10   PRINT "QUEL EST VOTRE NOM ? ";
 20   INPUT NOM$
 30   PRINT "ET VOTRE PRENOM ? ";'
 40   INPUT PRE$
 50   PRINT
 60   IDENTITE$=NOM$+" "+PRE$
 80   PRINT "DONC, SI JE COMPRENDS BIEN, VOUS VOUS APPELEZ :
";IDENTITE$;" !!!"
```

Le fonctionnement de ce programme est simple à comprendre : votre nom que contiendra la variable NOM$ vous est demandé en ligne 20, puis votre prénom en ligne 30 que contiendra PRE$. NOM$ et PRE$ sont réunis dans la variable IDENTITE$ en mettant un espace entre NOM$ et PRE$ pour que les 2 chaînes ne soient pas collées. Il ne reste plus qu'à afficher IDENTITE$ pour connaître votre identité complète.

GET

Une autre instruction gérant le clavier est GET + variable alphanumérique. GET nécessite la même syntaxe que INPUT mais n'attend pas que l'on entre une chaîne de caractères à l'écran et que l'on appuie sur RETURN.
Cette instruction ne saisit en effet qu'une seule touche, et donc, la première frappée. La valeur de la touche frappée sera affectée à la variable alphanumérique suivant GET.

Exemple :

```
10    GET VA$
```

RUN. Appuyez sur la touche 3. Pour vérifier que votre touche a bien été affectée à la variable VA$, un simple PRINT VA$ suffira.

GET n'accepte que les variables alphanumériques : GET A$ est toléré, mais GET A est impossible.

Autre exemple :

```
10    PRINT "APPUYEZ SUR UNE TOUCHE SVP"
20    GET REPONSE$
30    PRINT
40    PRINT "MERCI D'AVOIR APPUYE SUR LA TOUCHE ";REPONSE$
```

La variable REPONSE$ peut également contenir un caractère numérique. (caractères de 1 à 9).

KEY$

La dernière fonction de gestion du clavier est KEY$.

KEY$ permet de saisir une touche au clavier, mais contrairement à GET ou à INPUT, le TELESTRAT n'interrompt pas l'exécution du programme jusqu'à ce qu'une touche soit pressée. Il donne à une variable la valeur de la touche pressée à l'instant où il exécute la fonction KEY$. Si aucune touche n'est pressée lors d'un KEY$, la chaîne renvoyée est une chaîne vide.

La syntaxe de KEY$ diffère un peu de GET et INPUT. Pour demander par exemple au TELESTRAT de saisir immédiatement une touche et d'affecter la valeur de cette touche à la variable T$, on fait : T$=KEY$

Exemple :

```
10    A$=KEY$
20    PRINT "LA TOUCHE APPUYEE ACTUELLEMENT EST : ";A$
```

En ligne 10, on affecte à la variable A$ la touche appuyée au moment ou A$=KEY$ est exécuté. Puis, le contenu de A$ est sorti à l'écran en ligne 20. Rien ne sera affiché si aucune touche n'a été enfoncée lors du KEY$, le TELESTRAT mettant une chaîne vide dans la variable A$.

CHAPITRE 6

LES STRUCTURES DE CONTROLE ET LES OPERATEURS LOGIQUES ET ARITHMETIQUES

Nous entamons maintenant une partie importante de ce livre : les structures de contrôle qui incluent les prises de décisions, les déplacements dans un programme, les répétitions d'une action, etc ...

IF ... THEN ... ELSE

Le TELESTRAT a la possibilité de tester une ou plusieurs conditions, et selon que cette ou ces conditions sont réalisées ou non, il peut exécuter telle ou telle tâche. Cela se traduit par IF ... THEN ... (ELSE ...) qui est l'instruction type d'une structure de décision.

IF signifie SI
THEN signifie ALORS
ELSE signifie AUTREMENT

Ce qui nous amène à traduire l'instruction IF .. . THEN ... (ELSE ...) par : SI ... ALORS ... (AUTREMENT ...) ou plus précisément : SI expression ALORS instruction(s) (AUTREMENT instruction(s)). ELSE est entre parenthèses car il est facultatif et s'écrit d'ailleurs sans les parenthèses.

Pour ceux qui n'auraient toujours pas compris, sachez que IF ... THEN ... (ELSE ...) signifie : SI condition vraie (ou fausse) ALORS faire cela (AUTREMENT faire ceci) ou SI CA (variable numérique ou alphanumérique, nombre,calcul divers, ...) = CA (nombre, variable, chaîne ...) ALORS FAIRE CECI OU FAIRE CELA.

```
10    IF A=1 THEN PRINT A ELSE PRINT "A EST DIFFERENT DE 1"
```

Cette structure de décision dit que SI (IF) la variable A est égale à 1 (A=1) ALORS (THEN) on affiche la valeur de A (PRINT A) AUTREMENT on signale que A est différent de 1 (PRINT "A EST DIFFERENT DE 1").

Tapez :

```
5    A=0    (puis RUN)
```

et :

```
5    A=15 (puis RUN)
```

Vous obtenez les deux possibilités de la structure de décision située en ligne 10.

Le ELSE n'est pas obligatoire, en voici un exemple :

```
10   PRINT "ETES VOUS INTELLIGENT ? ( OUI / NON )";
20   INPUT REPONSE$
30   IF REPONSE$="OUI" THEN PRINT "EH BIEN, LA MODESTIE NE VOUS
ETOUFFE PAS !":END
40   PRINT "VOUS N'ETES PAS INTELLIGENT ? PAS DE CHANCE !!!"
```

Après vous avoir posé une question concernant votre quotient intellectuel, on analyse votre réponse en ligne 30. Si REPONSE$="OUI", c'est à dire si vous avez répondu OUI, alors les instructions après le THEN sont exécutées. (PRINT "EH BIEN , ...).

Il n'y a pas de ELSE en ligne 30. Donc, si vous n'avez pas répondu OUI, c'est que votre réponse est NON ou autre chose. (simple logique). La condition testée étant fausse (REPONSE$="OUI"), et ne rencontrant pas de ELSE après THEN, le TELESTRAT passe à la ligne suivante et affiche le message s'y trouvant (40 PRINT "VOUS N'ETES ...")

END

En ligne 30, une nouvelle instruction ne vous a pas échappé : END. Elle ordonne au TELESTRAT d'arrêter immédiatement l'exécution du programme. Si cette instruction n'était pas présente en ligne 30 et que le réponse soit OUI (condition vraie), alors le message après THEN serait normalement affiché, mais le message de la ligne 40 aussi, car l'ordre d'arrêter le programme en ligne 30 n'aurait pas été donné.

LES OPERATEURS LOGIQUES ET ARITHMETIQUES

Les tests d'instructions nécessitent une connaissance préalable des opérateurs arithmétiques que voici :

Vous connaisez déjà : + - * / ⇑

Il existe aussi :

< INFERIEUR
> SUPERIEUR
<= INFERIEUR OU EGAL
>= SUPERIEUR OU EGAL
<> DIFFERENT

Voici quelques exemples de ce qui peut être testé grâce à IF ... THEN ... (ELSE ...) :

```
IF A=1 THEN PRINT "A=1"
```

Teste si A est égal à 1. (=)

```
IF A>10 THEN PRINT "A EST PLUS GRAND QUE 10"
```

Teste si A est strictement supérieur à 10. (>)

```
IF A<5 THEN PRINT "A EST PLUS PETIT QUE 5"
```

Teste si A est strictement inférieur à 5. (<)

```
IF A>=0 THEN PRINT "A EST POSITIF"
```

Teste si A est plus grand ou égal à 0. (>=)

```
IF A<=0 THEN PRINT "A EST NEGATIF"
```

Teste si A est inférieur ou égal à 0. (<=)

```
IF A<>5 THEN PRINT "A N'EST PAS EGAL A 5"
```

Teste si A est différent (<>) de 5.

```
IF 4/2=2 THEN PRINT "4/2 = 2"
```

Teste si le résultat de 4/2 est égal à 2.

```
IF 4*3=2*6  THEN PRINT "C'EST BON !!!"
```

Teste si 4*3 = 2*6.

Ces divers exemples de IF ... THEN . .. (ELSE ...) ne sont pas exhaustifs. IF ... THEN ... ELSE s'utilise aussi pour les variables alphanumériques (chaînes de caractères).

Exemples :

```
IF A$="BONJOUR" THEN ...
```

Teste si la chaîne A$ contient la chaîne BONJOUR qui est obligatoirement mise entre guillemets, sinon, BONJOUR serait considéré comme une variable numérique. Et en aucun cas, une variable numérique ne peut être égale à une variable alphanumérique.

```
IF CHAINE$<>"SALUT" THEN ...
```

S'utilise pour tester si une chaîne est différente d'une autre.

Les signes > et < permettent de comparer 2 chaînes de caractères et de savoir laquelle des 2 est la plus grande.

```
IF "A"<"B" THEN PRINT "OK ..."
```

La lettre A est bien sûr inférieure à la lettre B dans l'alphabet normal.

```
IF "MARCHONS">"MARCHEZ" THEN ...
```

Pour comparer 2 chaînes de caractères, le TELESTRAT compare d'abord le premier caractère de chaque chaîne, puis le deuxième si les deux premiers caractères sont égaux, ...

Dans le cas de "MARCHONS" et "MARCHEZ", aucune décision ne sera prise sur la supériorité ou l'infériorité d'une chaîne par rapport à l'autre en examinant les 5 premiers caractères de chaque chaîne, ceux-ci étant égaux. ("MARCH"="MARCH"). Par contre, il n'est est pas de même pour le 6ème caractère de chaque chaîne. Pour la première chaîne, il s'agit d'un O, et pour la deuxième d'un E. Le E étant plus petit que le O dans l'alphabet, la chaîne "MARCHEZ" sera plus petite que "MARCHONS". La condition IF "MARCHONS">"MARCHEZ" sera donc vrai car O>E. De plus, la chaîne "MARCH" est inférieure à la chaîne "MARCHE", car le nombre de caractères de "MARCH" est inférieur à celui de "MARCHE". Si une chaîne débute par une autre "MARCHEZ" et "MARCHE" la plus petite est la plus courte des deux.

Exemples de programmes contenant des structures de condition :

```
 10    PRINT "TAPEZ SUR LA TOUCHE SE SITUANT A COTE DE LA LETTRE
'W'."
 20    GET A$
 30    IF A$="Q" THEN PRINT :PRINT "OK ... MERCI" !!!":END
 40    PRINT :PRINT "CE N'EST PAS LA BONNE LETTRE  !"
```

```
 10   PRINT "VOTRE PRENOM SVP : ";
 20   INPUT PRE$
 30   IF PRE$="ARTICHAUD" THEN PRINT "BONJOUR ARTICHAUD, COMMENT
ALLEZ VOUS ?":END
 40   PRINT "COMMENT  ? VOUS NE VOUS APPELEZ PAS ARTICHAUD ?"
 50   PRINT "JE REFUSE DE M'ENTRETENIR AVEC VOUS !"
```

AND

L'instruction AND fait également partie des opérateurs logiques et arithmétiques. Utilisée avec la strucure IF ... THEN ... (ELSE ...), AND permet de vérifier que deux conditions sont remplies. AND se traduit par ET. Les instructions après le THEN ne seront exécutées que si les deux conditions sont vérifiées.

IF (1ère condition) AND (2ème condition) THEN (instruction(s)) (ELSE (instructions)).

Ou : SI 1ère condition ET 2ème condition (sous entendu : réalisées) ALORS instruction(s) AUTREMENT instruction(s)

AND ne renvoyant la valeur "vrai" que si deux conditions sont remplies, cela nous donne la table de vérité suivante :

```
VRAI et VRAI = VRAI
VRAI et FAUX = FAUX
FAUX et VRAI = FAUX
FAUX et FAUX = FAUX
```

```
 10   PRINT "ENTREZ UN NOMBRE ENTRE 0 ET 5";
 20   INPUT NOMBRE
 30   IF NOMBRE>=0 AND NOMBRE<=5 THEN PRINT "MERCI" ELSE PRINT
"TRICHEUR !"
```

La ligne 30 nous montre très bien l'utilisation de l'instruction AND. Les lignes 10 et 20, demandent l'entrée d'un nombre compris entre 0 et 5. Ce nombre est stocké dans la variable NOMBRE.
La ligne 30 quant à elle se charge de détecter si le nombre entré est compris dans l'intervalle (0,5): IF NOMBRE >=0 AND NOMBRE <=5 THEN .... Les instructions après le THEN ne seront exécutées que si :

NOMBRE >=0 **AND** (et) NOMBRE <=5

Si le nombre entré n'est pas compris entre 0 et 5, les instructions après le ELSE sont exécutées.

```
 10   PRINT  "ETES VOUS JEUNE (O/N) ?";'
 20   INPUT A$
 30   PRINT "ETES VOUS VIEUX (O/N) ?";
 40   INPUT  B$
 50   IF A$="O" AND B$="O" THEN PRINT "VOUS ETES ILLOGIQUE : NE
RACONTEZ PAS N'IMPORTE QUOI !!!":END
 60   PRINT "EH BIEN TANT MIEUX ( TANT PIS ! ) POUR  VOUS !"
```

Après avoir saisi et placé dans les variables A$ et B$ les réponses aux questions posées, la ligne 50 s'assure que vous n'êtes pas jeune et vieux en même temps. On vérifie donc que les réponses aux deux questions ne sont pas O et O (OUI et OUI) :

IF A$="O" AND B$="O" THEN ... teste si on a répondu aux deux questions par l'affirmative. Si c'est le cas, il en résulte une exécution des instructions suivant le THEN.

OR

Une autre instruction classable parmi les opérateurs logiques, et qui s'utilise également avec IF ... THEN ... (ELSE ...) est OR. Cet opérateur détermine un résultat en fonction de deux conditions.

OR se traduit par OU. Le résultat final sera VRAI si l'une OU l'autre des deux conditions est vérifiée. Cela nous amène à la table de vérité suivante :

```
VRAI ou VRAI = VRAI
VRAI ou FAUX = VRAI
FAUX ou VRAI = VRAI
FAUX ou FAUX = FAUX
```

Si une seule des deux conditions suivant le IF est vraie, alors le total sera considéré comme vrai.

```
 10   PRINT "JOUEZ AU LOTO !"
 20   PRINT "ENTREZ UNE LETTRE ENTRE A ET Z";:GET C$
 30   IF C$<"A" OR C$>"Z" THEN PRINT "N'IMPORTE QUOI !":END
 40   IF C$="E" OR C$="T" THEN PRINT "GAGNE ! PAR ICI LA MONNAIE !"
 50.  PRINT "PAS DE CHANCE, CE N'EST PAS LA BONNE LETTRE. ESSAYEZ
DE NOUVEAU ..."
```

On trouve dans ce programme deux structures de décision alliées à OR :

```
 30   IF C$<"A" OR C$>"Z" THEN PRINT "N'IMPORTE QUOI"
```

Cette ligne s'assure que ce que vous venez d'entrer est bien une lettre de l'alphabet. Il faut pour cela que la chaîne C$ contienne une lettre entre A et Z. Si le contenu de C$ est plus petit que la lettre A ou si le contenu de C$ est plus grand que la lettre Z, alors la lettre entrée n'est pas celle que l'on attendait.

Deux conditions sont à tester :

- C$<"A" Si une seule de ces conditions est vraie,
- C$>"Z" alors la solution est vraie.

```
 40     IF C$="E" OR C$="T" THEN PRINT "GAGNE ! PAR ICI LA MONNAIE !"
```

L'utilisateur venant d'entrer une lettre entre A et Z (on vient de le tester en ligne 30), on compare cette lettre avec les deux bonnes lettres du tirage : E et T.

IF C$="E" OR C$="T" THEN ... demande à ce que les instructions suivant THEN soient exécutées si C$ contient E ou T.

Il n'est pas indispensable d'utiliser le signe '=' pour savoir si une variable est nulle ou non.

```
 IF A THEN PRINT "A<>0"
```

IF A teste si A est différent de 0. Si oui, alors les instructions suivant THEN sont exécutées. Pour une chaîne, cela nous donne :

```
 IF A$ THEN PRINT "chaine NON VIDE"
```

Si la chaîne A$ n'est pas vide, alors les instructions suivant THEN sont prises en compte car IF A$ correspond à IF A$<>"" (chaîne vide).

Depuis que nous avons appris à mettre des instructions en lignes de programme, ces lignes s'exécutent dans l'ordre croissant : la ligne 10 avant la ligne 20, la 20 avant la 30 ... Ceci est plus que normal. Le TELESTRAT n'en possède pas moins deux instructions permettant de se déplacer dans un programme.

GOTO

La première a pour syntaxe GOTO + No de ligne. GOTO signifie GO TO ou ALLER A (numéro de ligne spécifié).
Cette instruction demande au TELESTRAT d'aller directement à la ligne donnée après GOTO et d'exécuter les instructions s'y trouvant.

```
10   GOTO 60
20   PRINT "L'UTILISATION ";
30   GOTO 80
40   PRINT "VOUS MONTRE ";
50   GOTO 20
60   PRINT "CE PROGRAMME ";
70   GOTO 40
80   PRINT "DE GOTO !"
```

Regardons ce qui se passe après un RUN :

- En ligne 10, GOTO 60 nous envoie à la ligne 60 qui affiche "CE PROGRAMME".
- En ligne 70, GOTO 40 nous envoie à la ligne 40 qui affiche "VOUS MONTRE".
- En ligne 50, GOTO 20 nous envoie à la ligne 20 qui affiche "L'UTILISATION".
- En ligne 30, GOTO 80 nous envoie à la ligne 80 qui affiche "DE GOTO".
- Et c'est fini.

GOTO est une instruction très puissante. Elle rompt l'uniformité de direction d'exécution d'un programme (de la ligne la plus petite à la ligne la plus grande) et dirige le programme vers une ligne quelconque (mais existante).

```
10   I=10
20   PRINT I
30   I=I-1
40   IF I=0 THEN END
50   GOTO 20
```

Ces 5 lignes vont afficher consécutivement les nombres de 10 à 1.
On initialise la variable I avec la valeur 10, puis la valeur de I est affichée (ligne 20). On décrémente de 1 la valeur de I en ligne 30. (I=I-1 est en fait un non sens mathématique car si I vaut 10, 10 <> 10-1. Il faut comprendre par I=I-1 que la valeur de la variable I est décrémentée de 1.).

La ligne 40 teste si la variable I vaut 0, et si c'est le cas, arrête le programme. Sinon, GOTO 20 en ligne 60 nous renvoie à la ligne 20 pour continuer l'affichage décroissant des nombres de 10 à 1.

GOSUB

La deuxième instruction détournant l'exécution d'un programme est GOSUB + No de ligne. GOSUB = GO SUBROUTINE ou encore aller au sous-programme commençant à la ligne spécifiée.

GOSUB + No de ligne permet d'appeler un sous programme qui sera obligatoirement terminé par l'instruction RETURN que l'on traduit par RETOUR DE SOUS-PROGRAMME.

Lorsque le TELESTRAT rencontre GOSUB, le numéro de ligne où a eu lieu l'instruction GOSUB est conservé dans une partie de la mémoire appelée la pile. Puis, il se rend à la ligne donnée après GOSUB et il exécute toutes les instructions qu'il trouve jusqu'à ce qu'il rencontre RETURN. Dès ce moment, le TELESTRAT reprend dans la pile le numéro de la ligne où a eu lieu le GOSUB, et y retourne.

```
10   PRINT "MAIS QUEL EST DONC VOTRE AGE  ?";
20   INPUT AGE
30   GOSUB 40:END
40   PRINT
50   PRINT "MAIS C'EST BIEN SUR, VOUS AVEZ" ;AGE;"ANS !!!"
60 RETURN
```

Votre âge entré, la ligne 30 envoie l'interpréteur de lignes à la ligne 40 grâce à GOSUB 40. Les lignes 40 et 50 affichent votre âge. En ligne 50, RETURN fait retourner le programme à la ligne où l'on a fait GOSUB, soit à la ligne 30. Il continue l'exécution de la ligne 40 et trouve END. Le programme s'arrête.

LES LABELS

Une grande nouveauté facilitant la création de programme a été ajoutée à votre TELESTRAT : les labels. On peut donner un nom à une partie de son programme, et exécuter cette partie à tout moment en incluant simplement le nom de cette partie dans un programme. Pour définir un label, il suffit de mettre au début d'un ligne le caractère] suivit du nom du label.

Exemple :

```
10 ]AFFICHEBONJOUR
20   PRINT "BONJOUR"
30 RETURN
```

Vous venez de définir un label nommé AFFICHEBONJOUR qui affiche le message BONJOUR. Le nombre de caractères d'un label ne se limite qu'au nombre de caractères qu'accepte une ligne (environ 110). Tous les caractères d'un label sont significatifs.
Après avoir copié le programme ci-dessus, tapez en mode direct AFFICHEBONJOUR. Le mot BONJOUR s'inscrit alors. Vous venez en effet d'appeler le sous-programme ayant pour nom AFFICHEBONJOUR, et ce, rien qu'en entrant son nom. AFFICHEBONJOUR est un diminutif de GOSUB AFFICHEBONJOUR et c'est d'ailleurs à cause de ce GOSUB que nous trouvons un RETURN en ligne 30. Quoi qu'il en soit, les programmes commençant par des labels peuvent également être appelés par GOTO + LABEL.

Récapitulatif : les labels permettent de poser des "pancartes" au début de sous-programmes. Pour cela, on donne un nom à ces sous-programmes grâce à] (bracket droit) suivi du nom du sous-programme. Le nom de ce sous programme peut comporter autant de caractères significatifs que le permet une ligne. L'intérêt des labels est très grand : D'abord, ils facilitent grandement la lecture d'un programme, et de plus, ils simplifient la vie du programmeur. Il n'est en effet plus nécessaire de savoir que la ligne 5200 correspond au sous-programme posant une question par exemple.

```
10 ]AFFICHE
20    PRINT "HYPERBASIC"
30 RETURN
```

Pour appeler ce sous-programme nommé AFFICHE, vous avez plusieurs solutions :

- l'appeler par GOSUB AFFICHE.
- l'appeler tout simplement par AFFICHE car AFFICHE est un diminutif de GOSUB AFFICHE (le GOSUB peut être omis).
- l'appeler par son numéro de ligne GOSUB 20. Dans cette optique, le label ne servira à rien.

On peut également utiliser GOTO + label. A ce moment là, il n'y aura pas de RETURN à la fin du sous-programme à appeler car GOTO ne conserve pas d'adresse de retour, à l'inverse de GOSUB.

```
10 ]DEBUT
20   PRINT "IL EST ";
30   HEURE
40   IF H=25 THEN END
50   GOTO DEBUT
60 ]HEURE
70   PRINT H;" HEURE."
80   H=H+1
90 RETURN
```

Ce programme affiche l'heure de 0 à 24. On commence par mettre le label DEBUT à la ligne 10 pour spécifier le début du programme. Puis, la ligne 20 affiche IL EST. En ligne 30, HEURE commande au TELESTRAT d'aller au sous-programme HEURE qui commence à partir de la ligne 60. HEURE correspondant à GOSUB HEURE, l'adresse de retour est conservée. Le sous-programme HEURE affiche alors l'heure qui est contenue dans la variable H. On augmente le contenu de H de 1 en ligne 80, puis RETURN fait revenir l'exécution du programme en ligne 40. Cette ligne teste si l'heure (H) est égale à 25, et stoppe le programme si c'est le cas. Sinon, la ligne 50 GOTO DEBUT nous fait retourner au début

Nous avons déjà vu que l'instruction RUN démarrait un programme. Mais elle permet aussi de commencer l'exécution d'un programme à une ligne spécifiée :

```
10   PRINT A
20   PRINT B
```

Si vous entrez RUN après avoir copié ces deux lignes, l'ordinateur affiche deux 0, car les variables A et B valent initialement 0. Toutes les variables sont en effet mises à 0 lors d'un RUN. Au cas où l'on désire commencer le programme en ligne 20, il suffit de faire RUN 20. RUN + numéro de ligne démarre un programme à un numéro de ligne donné. RUN LABEL démarre le programme à partir du numéro de label donné.

GOTO s'utilise également en mode direct et possède d'ailleurs un avantage sur l'instruction RUN : Elle conserve les valeurs des variables.

Exemple : faites en mode direct B=1 puis GOTO 20. La valeur affichée est bien 1, ce qui vous montre que les variables ne sont pas mises à 0 lors d'un GOTO.

N'oubliez d'ailleurs pas que vous avez aussi la possibilité de démarrer un programme en mode direct par GOTO + LABEL ou GOSUB + LABEL ou GOSUB + numéro de ligne ou LABEL.

FOR ... TO ... NEXT ... STEP

Une des tâches les plus fréquentes qu'accomplit l'ordinateur est la répétition d'une action. C'est pour cela qu'existent les boucles FOR ... TO ... (STEP) ... NEXT

FOR V= N1 TO N1+n ... NEXT

Cette boucle va obliger l'ordinateur à répéter n fois l'action correspondant à toutes les instructions comprises entre FOR et NEXT. V définit la variable numérique qui controlera la boucle.

Exemple :

```
10   FOR I=1 TO 3
20    PRINT I
30   NEXT
```

La boucle ci-dessus affichera les nombres 1 2 3. Le TELESTRAT donne d'abord la valeur 1 à la variable I. Puis il exécute la ligne 20 où la valeur de I (pour le moment 1) est affichée. En ligne 30, NEXT vérifie la valeur de I et retourne en ligne 10 où la valeur de I est augmentée de 1. La ligne 20 affiche 2, ... Au 3ème passage de cette boucle, la ligne 30 vérifie si la valeur de I est égale à la valeur spécifiée après TO (3). Comme c'est le cas, le programme s'arrête.

FOR I=1 TO 3 ... NEXT I se traduit par : pour I qui vaut 1 au départ jusqu'à ce que I = 3 (valeur finale) : exécuter tout ce qui se trouve entre la boucle FOR ... TO ... jusqu'à NEXT I.

En clair, FOR ... TO ... NEXT exécute les instructions se trouvant entre FOR ... TO et NEXT depuis la valeur de départ située après le = jusqu'à la valeur finale indiquée après TO. FOR I=1 TO 3 demande à ce que toutes les instructions entre FOR ... TO et NEXT soient exécutées 3 fois. La 1ère fois, I vaut 1, la deuxième, I vaut 2 et la troisième, I vaut 3. 3 correspondant à la valeur donnée après le TO, on sort de la boucle.

Le nom de la variable d'un FOR ... TO ... NEXT n'est pas obligatoire à la fin d'un NEXT, bien qu'il facilite la lecture d'un programme. Ainsi :

```
10   FOR A=1 TO 2
20   NEXT A
```

Correspond à :

```
10   FOR A=1 TO 2
20   NEXT
```

IMPORTANT : une boucle FOR ... TO doit toujours commencer en début de ligne. Sinon, votre programme peut se bloquer et devenir irrécupérable ...

```
10   FOR I=1 TO 10
```

Est possible, tandis que :

```
10   PRINT :FOR I=1 TO 10
```

N'est pas possible. Souvenez vous en !

Il en est de même pour NEXT qui doit impérativement être placé en début de ligne. Seul un NEXT en début de ligne décrémentera la boucle FOR/NEXT

On a quelquefois besoin de faire des boucles à l'intérieur d'autres boucles. Cela s'appelle "l'emboîtement de boucles". Il faut savoir que l'on peut emboîter jusqu'à 10 boucles les unes dans les autres. Ce maximum est à ne pas dépasser.

Voici par exemple comment l'on peut emboîter deux boucles l'une dans l'autre :

```
10  FOR W=1 TO 2 ------------------------------|
20   FOR X=4 TO 6 ------------|                 |
    ........        BOUCLE    |      BOUCLE     |
    ........      INTERIEURE  |    EXTERIEURE   |
70   NEXT X        ------------|                 |
80  NEXT W        ------------------------------|
```

Lorsque vous créez une boucle à l'intérieur d'une autre boucle, il est obligatoire que le NEXT de la boucle emboîtée (boucle intérieure) se trouve avant le NEXT de la boucle extérieure.

```
10   FOR X=0 TO 10
20    FOR Y=1 TO 5
30     PRINT X,Y
40    NEXT Y
50   NEXT X
```

Dans une boucle conventionnelle FOR ... TO ... NEXT le pas de comptage est mis à 1. Cela signifie qu'à chaque fois que la boucle fait un tour, la valeur de la variable de comptage est augmentée de 1.

On peut obtenir un comptage différent de 1 grâce à l'instruction STEP (non indispensable) qui définit un pas.

```
10   FOR I=0 TO 20 STEP 2
20    PRINT I
30   NEXT
```

Va afficher les nombres 0,2,4,6,8,10 ... 20. Car l'on a demandé de marquer les nombres de 0 à 20 avec un pas de 2, c'est à dire de 2 en 2.

```
10   FOR S=1 TO 20 STEP 5
20    PRINT S
30   NEXT
```

Affiche les nombres de 1 à 20 avec un pas de 5, donc, de 5 en 5. Un pas peut avoir des valeurs négatives qui permet de décompter :

```
10   FOR I= 5 TO 1 STEP -1
20    PRINT I
30   NEXT I
```

Il est absolument nécessaire de mettre un STEP si l'on veut que le comptage soit fait à l'envers car FOR I=5 TO 1 n'est pas égal à FOR I=5 TO 1 STEP -1.

Une boucle FOR ... TO ... NEXT est 4 fois plus rapide à l'exécution qu'une boucle avec STEP. (FOR ... TO ... (STEP ...) ... NEXT)

Dans un listing, les boucles sont indentées : toutes les instructions contenues dans une boucle sont décalées d'un espace vers la droite, ce qui donne une meilleur lisibilité.

Si deux boucles sont imbriquées, toutes les instructions présentes dans la 2ème boucle sont décalées de 2 espaces, 3 espaces pour 3 boucles imbriquées, et ainsi de suite. (Avec un maximum de 10 boucles emboitées les unes dans les autres.)

Comme pour la plupart des instructions, les valeurs à spécifier lors d'un FOR ... TO ... (STEP...)... NEXT peuvent être des variables, des valeurs, des expressions calculées, ...

Exemples :

```
10  FOR I=1 TO 5 = valeur entière.
10  FOR I=1.5 TO 4.5 = valeur non entière.
10  FOR I=1 TO I+5 = expression calculée qui correspond à FOR I=1 TO 6 car I
vaut au début 1 et 1+5 donne 6. (Dangereux !!!).
10  FOR I=2*4 TO 5*2 = expression calculée.
10  FOR I=DEBUT TO FIN STEP PAS = variables.
```

Attention: il faut toujours sortir d'une boucle par NEXT et jamais par GOTO.

COUNT

Il arrive parfois que l'on veuille créer une boucle sans pour cela avoir obligatoirement besoin de créer une variable, comme c'est le cas pour FOR ... TO ... (STEP ...) ... NEXT.

COUNT n ... UNCOUNT est la structure de contrôle qui permet de répéter n fois une action sans avoir besoin de variable. COUNT veut dire compter et UNCOUNT décompter.

COUNT n .... (instructions) ... UNCOUNT fait : compter n fois (... exécuter n fois tout ce qui se trouve à l'intérieur de la boucle) ... Décompter. Toutes les instructions entre COUNT et UNCOUNT seront répétées n fois.

```
10  COUNT 100
20   PRINT "BONJOUR ! ";
30  UNCOUNT
```

Va afficher 100 fois le mot BONJOUR.

De même que FOR ... TO ... NEXT, COUNT doit être impérativement en début de ligne. Lors d'un LIST, tout ce qui se trouve entre COUNT et UNCOUNT est décalé d'un espace vers la droite pour 1 boucle, de 2 espaces pour deux boucles imbriquées, ..., le nombre suivant COUNT ne peut excéder 65535.

La structure COUNT ... UNCOUNT est la plus rapide de toutes les structures de répétition d'une action. Ainsi, COUNT 10000 : UNCOUNT ne prend qu' 1/10 ème de seconde !

```
10  COUNT 65535
20   PRINT "APPUYEZ SUR CTRL-C QUAND VOUS EN AUREZ ASSEZ !"
30  UNCOUNT
```

Une boucle COUNT/UNCOUT à vide ne nécessite qu'environ 14 µs (1 µs équivaut à 1 micro seconde, soit 1 millionième de seconde).

REPEAT, UNTIL

Toujours dans la série répétition d'une action, voici maintenant la structure de contrôle REPEAT ... UNTIL. REPEAT veut dire répéter et UNTIL se traduit par "jusqu'à ce que". Toutes les instructions comprises entre REPEAT et UNTIL (condition) seront exécutées jusqu'à ce que la condition logique donnée après UNTIL soit vraie.

```
10  REPEAT
20   PRINT "APPUYEZ SUR LA TOUCHE A"
30   GET A$
40  UNTIL A$="A"
```

Cette boucle se répétera tant que la condition logique A$="A" ne sera pas vraie. Il vous faut donc appuyer sur la touche A pour pouvoir sortir de la boucle. Sinon, le programme retournera au début de la boucle REPEAT ... UNTIL jusqu'à ce que la bonne touche soit entrée.

```
10   ARRET = 1000
20    REPEAT
30    PRINT X
40    X=X+10
50   UNTIL X>ARRET
```

Cette boucle affichera la valeur de X tant que la condition X>ARRET ne sera pas vraie. Comme ARRET = 1000, il faudra attendre que X soit égal à 1010 pour pour la condition X > ARRET soit vérifiée.

ATTENTION : mettre REPEAT en début de ligne est impératif. UNTIL doit aussi être en début de ligne pour ajuster l'indentation des lignes

WHILE, WEND

La dernière structure de contrôle disponible est WHILE (condition) ... WEND. C'est en quelque sorte l'inverse de REPEAT ... UNTIL (condition). L'avantage d'une boucle WHILE (condition) ... WEND est que la condition étant testée au début de la boucle, cela nous permet de 'sauter' la boucle si la condition n'est pas vérifiée dès le début.

```
10   WHILE KEY$=""
20    PRINT " VOUS N'APPUYEZ SUR AUCUNE TOUCHE POUR LE MOMENT ..."
30   WEND
40   PRINT " VOUS VENEZ D'APPUYER SUR UNE TOUCHE ..."
50   PRINT "C'EST FINI"
```

La boucle entre WHILE et WEND est exécutée tant que la condition après WHILE est vérifiée. Dans le programme ci-dessus, tant qu'une touche n'aura pas été appuyée (KEY$=""), il y aura affichage du message en ligne 20. Et dès le moment où une touche sera détectée, la boucle WHILE... WEND sera 'sautée' et on se dirigera à la ligne suivant le WEND, dans le cas présent, à la ligne 40.

```
10   WHILE X > -100
20    PRINT "X =";X
30    X=X-1
40   WEND
```

Tant que X sera supérieur à -100, la valeur de X sera affichée. Rappelez vous que X vaut 0 au début du programme car RUN initialise toutes les variables et les met à 0. Nous avons vu que END permettait d'arrêter un programme à n'importe quel moment, rien que par sa présence. Une deuxième instruction permet également cela : STOP .

Il y a une différence entre END et STOP : lorsque l'ordinateur rencontre un STOP, il y a affichage du message "`execution arretee (ligne 1)`", l étant le numéro de la ligne où figure l'instruction STOP.

```
10 STOP
```

Faites RUN. "`execution arretee (ligne 10)`" apparaît.

POP

Cette instruction permet de sortir d'un sous-programme appelé par RETURN. Lors de l'appel d'un sous-programme par GOSUB, le seul moyen de sortir de ce sous-programme est RETURN. POP annule l'instruction GOSUB, et permet à l'utilisateur d'aller n'importe où ailleurs dans le programme.

Exemple :

```
10   PRINT A
20   GOSUB AUGMENTE
30   GOTO 10
40 ]AUGMENTE
50   A=A+1
60   IF A=100 THEN POP:GOTO FIN
70 RETURN
80 ]FIN
90 END
```

CHAPITRE 7

LES FONCTIONS DE TRAVAIL SUR LES CHAINES ET LES FONCTIONS DE CONVERSION

Travailler sur des chaînes, cela consiste à savoir en extraire un ou plusieurs caractères, à les transformer en nombres, en minuscules, en majuscules, ...

LEN

Une instruction très utile et très simple à comprendre est LEN (nom de la chaîne). Elle renvoie le nombre de caractères que comporte une chaîne.

Exemple : `PRINT LEN("TELEMATIC")`

Renverra 9 puisque le mot TELEMATIC est composé de 9 caractères.

```
10   PRINT "ENTREZ UNE CHAINE DE CARACTERES QUELCONQUE : ";
20   INPUT CHAINE$
30   NOMBREDECARACTERE=LEN(CHAINE$)
40   PRINT "VOTRE CHAINE CONTIENT";NOMBREDECARACTERE;"
CARACTERES."
50   PRINT "ENCORE UN ESSAI ? (O/N) ";
60   GET A$
70   IF A$="O" THEN 10 ELSE END
```

En ligne 20, on donne à la variable NOMBREDECARACTERE le nombre de caractères que contient la chaîne entrée, puis on l'affiche en ligne 40.

Vous remarquerez qu'en ligne 70, IF A$="O" THEN 10 ELSE END est l'équivalent de IF A$= "O" THEN GOTO 10 ELSE END.

Conclusion : GOTO peut être omis, mais seulement après un THEN ou un ELSE.

Attention : THEN LABEL équivaut à THEN GOTO LABEL (différents de GOSUB LABEL)

```
10   PRINT "ENTREZ UNE CHAINE DE 10 CARACTERES MAXIMUM : ";
20   INPUT A$
30   IF LEN(A$)>10 THEN 10 ELSE PRINT "MERCI !"
```

LEN (A$) est considéré comme une valeur et peut donc être testé (ligne 30) ou être affecté à une variable. Dans une chaîne, les espaces comptent pour un caractère.

LEFT$

Une fonction extrayant un certain nombre de caractères d'une chaîne est LEFT$. LEFT$ (nom de la chaîne, nombre de caractères à extraire) prend un nombre donné de caractères à une chaîne à partir de la gauche de celle-ci.

```
10   A$="COMMENT ALLEZ VOUS ?"
20   B$=LEFT$(A$,7)
30   PRINT B$
```

B$=LEFT$(A$,7) va extraire les 7 caractères de A$ à partir de la gauche (LEFT = GAUCHE en français) de cette chaîne et va les mettre dans B$. B$ contiendra alors COMMENT car les 7 premiers caractères de la chaîne "COMMENT ALLEZ VOUS ?" sont COMMENT.

```
10   A$="ORIC TELESTRAT, LE MICRO-SERVEUR PERSONNEL"
20   PRINT "VOICI TOUTES LES POSSIBILITES D'EXTRACTION DE
CARACTERES DE LA CHAINE : ";A$
30   PRINT :PRINT "APPUYEZ SUR UNE TOUCHE":GET A$
40   FOR I=1 TO LEN(A$)
50    PRINT LEFT$(A$,I)
60   NEXT
```

Ce petit programme va afficher une par une toutes les parties que l'on peut extraire de la chaîne A$ en partant de la gauche.

RIGHT$

L'instruction RIGHT$ fait exactement l'inverse de LEFT$. Elle permet d'extraire un nombre donné de caractères d'une chaîne en partant de la droite. RIGHT signifie DROITE en français. La syntaxe de RIGHT$ est la même que celle de LEFT$, à savoir RIGHT$ (nom de la chaîne, nombre de caractères à extraire).

```
10   A$="ESSAI DE LA FONCTION RIGHT$"
20   PRINT RIGHT$(A$,15)
```

RIGHT$ (A$,15) va extraire les 15 caractères de droite de la chaîne A$. Ces 15 caractères donnent:
`FONCTION  RIGHT$`

Voici comme pour LEFT$ le programme donnant toutes les parties que l'on peut extraire d'une chaîne de caractères à partir de la droite.

```
10   A$="ORIC TELESTRAT, LE MICRO-SERVEUR PERSONNEL"
20   PRINT "VOICI TOUTES LES POSSIBILITES D'EXTRACTION DE
CARACTERES DE LA CHAINE : ";A$
30   PRINT :PRINT "APPUYEZ SUR UNE TOUCHE":GET A$
40   FOR I=1 TO LEN(A$)
50    PRINT RIGHT$(A$,I)
60   NEXT
```

Si le nombre de caractères à extraire d'une chaîne avec RIGHT$ ou LEFT$ dépasse celui que contient une chaîne, tous les caractères de la chaîne seront restitués.

MID$

LEFT$ et RIGHT$ extraient des caractères à gauche ou à droite d'une chaîne. MID$ quant à lui va nous permettre de les extraire au milieu d'une chaîne, ou plutôt à partir d'un caractère donné. La syntaxe est MID$ (nom de la chaîne, nombre de caractères à extraire, numéro du caractère à partir duquel commence l'extraction).

```
10   A$="BONJOUR ORIC TELESTRAT"
20   PRINT MID$(A$,9,4)
```

MID$(A$,9,4) demande à ce que soit affichés les 4 caractères situés à partir du 9ème caractère de la chaîne A$. Ces 4 caractères forment le mot ORIC.

Pour vous montrer que MID$ permet d'extraire le nombre de caractères que l'on veut à partir du caractère choisi, faites :

```
A$="ESSAI  FONCTION  MID$"
```

Puis `PRINT  MID$(A$,1,5)`

Puis `PRINT  MID$(A$,7,8)`

Puis `PRINT  MID$(A$,16,5)`

Si le nombre de caractères à extraire est supérieur a celui que contient une chaîne, tous les caractères à partir du caractère de départ seront pris en compte. Si la position du caractère de départ est supérieure au nombre total de caractères d'une chaîne, la chaîne renvoyée sera vide.

MIDDLE$

MIDDLE$ se traduit par 'milieu' et permet de centrer une chaîne à l'écran dans un espace donné.

Syntaxe : MIDDLE$ (nom de la chaîne, longueur donnée).

Exemple:

```
10   A$="MICRO-INFORMATIQUE"
20   PRINT
30   PRINT MIDDLE$(A$,40)
```

L'écran étant divisé en lignes de 40 colonnes, PRINT MIDDLE$ (A$,40) va centrer la chaîne A$ au milieu de l'écran. La chaîne A$ fait 18 caractères, et l'écran 40. Le TELESTRAT laissera 11 espaces de chaque côté du mot MICRO-INFORMATIQUE, car 18+(11*2)= 40 colonnes.

La longueur de l'espace où l'on veut afficher une chaîne doit être supérieure au nombre de caractères que contient cette chaîne. Si ce n'est pas le cas, aucun espace ne sera affiché. Un nombre d'espace impair sera réparti en deux espaces inégaux d'un côté et de l'autre d'une chaîne.

```
10   PRINT "ENTREZ UNE CHAINE DE MOINS DE 40 CARACTERES  : ";
20   INPUT A$
30   IF LEN(A$)>39 THEN 10
40   PRINT
50   PRINT MIDDLE$(A$,40)
```

Ce programme centre une chaîne quelconque sur une ligne de l'écran.

SPC$

La fonction SPC$(n) génère une chaîne "vide" de n espaces.

```
10      PRINT  SPC$(10);"ORIC"SPC$(10)
```

Laisse d'abord 10 espaces puis affiche ORIC.

SPC$ (n) demande à l'ordinateur d'afficher n espaces. Cela permet de formater un affichage, c'est à dire de laisser le nombre d'espaces voulu avant ou après l'inscription à l'écran d'une chaîne.

```
10   FOR X=0 TO 30
20    PRINT SPC$(X);"SALUT"
30   NEXT
```

LOB$, LO$, UP$

LO$ (nom de la chaîne) transforme tous les caractères d'une chaîne en minuscules.

LOB$ (nom de la chaîne) transforme tous les caractères d'une chaîne en minuscules sauf la première lettre qui n'est pas touchée.

UP$ (nom de la chaîne) transforme tous les caractères d'une chaîne en majuscules.

Exemples :

```
A$="HELLO"
PRINT LO$(A$)
```

Résultat : `hello` car LO$ convertit tous les caractères d'une chaîne en minuscules.

Faites:

```
10   A$="HELLO"
20   PRINT LOB$(A$)
```

Résultat : `Hello` car LOB$ convertit une chaîne en minuscules sauf le premier caractère qui reste en majuscule.

```
10   A$="hello"
20   PRINT UP$(A$)
```

Résultat : `HELLO` car UP$ transforme tous les caractères d'une chaîne en majuscules.

Le programme ci-dessous centre et affiche avec LO$, LOB$ et UP$ la chaîne que vous avez défini.

```
10   PRINT " ENTREZ UNE CHAINE DE MOINS DE 40 CARACTERES : ";
20   INPUT A$
30   IF LEN(A$)>39 THEN 10
40   PRINT:PRINT MIDDLE$(LO$(A$),40)
50   PRINT:PRINT MIDDLE$(LOB$(A$),40)
60   PRINT:PRINT MIDDLE$(UP$(A$),40)
```

ASC, CHR$

Dans un ordinateur, chaque caractère est stocké sous forme de code. Ce code a pour nom code ASCII (American Standard Code For Information Interchange) et peut varier de 0 à 255.

Deux instructions se chargent d'effectuer les conversions CODE ASCII -> CARACTERE ou CARACTERE --> CODE ASCII : ASC et CHR$. ASC (caractère) affiche le code ASCII équivalent au caractère entre parenthèses.

```
10   A$="A"
20   PRINT ASC(A$)
```

Réponse : `65`

PRINT ASC(A$) correspond à PRINT ASC ("A") puisque A$="A". Le code ASCII de A est donc 65. Et vous pouvez le vérifier grâce à l'instruction CHR$(code ASCII) qui permet de connaître le caractère correspondant à un code ASCII.

Exemple : `PRINT ASC("A")`

Réponse : `65`

```
PRINT CHR$(65)
```

Réponse : `A`

Il y a une façon encore plus simple de connaître le code ASCII de la lettre B par exemple :

```
PRINT'B'
```

Réponse : `66`

Tout caractère entre apostrophes (') est considéré comme le code ASCII de ce caractère. Ainsi, A='C' correspond à A=67 car 67 est le code ASCII de C.

Ce programme va visualiser tous les codes ASCII entre 32 et 127. Les codes ASCII de 0 à 32 et de 128 à 255 ne sont pas donnés car ce sont des codes spéciaux qui peuvent perturber l'affichage à l'écran. Vous devez appuyer sur une touche après avoir vu à quel caractère correspondait un code ASCII.

```
10   FOR I=32 TO 127
20    PRINT "CODE ASCII ";I;" : CARACTERE ";CHR$(I)
30    GET A$
40   NEXT
```

La liste complète des codes ASCII vous est donnée en annexe 4.

STR$, VAL

Deux instructions faisant partie des fonctions de conversion transforment une valeur numérique en une chaîne alphanumérique et vice-versa. Il s'agit tout d'abord de STR$ qui change un nombre en une chaîne de caractères.

```
10   X=10
20   A$=STR$(X)
30   PRINT A$
```

Réponse `10`.

A$=STR$(X) a demandé à ce que la variable alphanumérique A$ contienne la valeur de la variable X. X étant initialisé à 10, A$="10".

VAL est l'instruction inverse de STR$ et donne la VALeur d'une chaîne numérique.

Exemple :

```
10   A$="1500"
20   NOMBRE=VAL(A$)
30   PRINT NOMBRE
```

Le résultat est tout simplement 1500 car VAL a transformé le nombre que contenait A$ en valeur numérique. VAL renvoie 0 si la chaîne à convertir contient une chaîne non numérique.

Les fonctions VAL et STR$ marchent également avec les nombres en notation exponentielle du genre 1,434E+10, avec les nombres négatifs, et avec les nombres hexadécimaux et binaires que nous allons voir ci-dessous.

HEX$

Le systéme Hexadécimal est le système utilisé entre autre par un ordinateur. Contrairement au système décimal auquel nous sommes habitués et qui est en base 10, le système hexadécimal agit en base 16.

En système décimal (base 10), les 10 premiers chiffres sont :

0,1,2,3,4,5,6,7,8,9

En système hexadécimal (base 16), les 16 premiers chiffres sont :

0,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F

Les nombres de A à F correspondent aux nombres de 10 à 15 en décimal. Pour convertir un nombre en hexadécimal, on utilise la fonction HEX$(nombre) qui convertit un nombre en HEXadécimal.

```
PRINT  HEX$(10)
```

Donne **#A**.

Le signe # signifie hexadécimal. Un nombre hexadécimal ne peut excéder la valeur 65535 (#FFFF).

De même, il ne peut être que positif. Si vous dépassez l'intervalle (0,65535) ou (#0,#FFFF) en hexadécimal, il y aura affichage du message "**valeur incorrecte**." .

Pour afficher un nombre hexadécimal en décimal, il suffit de faire PRINT + nombre en hexa (abrégé de hexadécimal.)

Ainsi, PRINT HEX$(16) donne #10 en hexadécimal et PRINT #10 donne 16 en décimal.

L'hexadécimal, de même que le binaire, seront des systèmes numériques que vous emploierez peu, voire jamais lors de la réalisation d'un programme. Toutefois, lorsque l'on désire en savoir plus sur son ordinateur, il est indispensable de maîtriser ces deux systèmes numériques.

```
10   FOR I=0 TO 255
20    PRINT I;" EN DECIMAL VAUT ";HEX$(I);" EN HEXA"
30    IF KEY$ THEN GET A$
40   NEXT
```

Va afficher les nombres décimaux et hexadécimaux de 0 à 255. Tout appui sur une touche interrompra momentanément le listing et tout nouvel appui le fera reprendre.

BIN$

Un ordinateur ne travaille qu'avec des 0 et des 1. Le système qui comprend les nombres 0 et 1 est le système numérique binaire ou encore base 2.

C'est le système préféré de votre TELESTRAT. Encore une fois, le système binaire vous sera très peu utile au cours de votre apprentissage du langage BASIC. Mais il pourra s'avérer intéressant par la suite.
L'instruction BIN$(nombre) donne l'équivalent binaire d'un nombre.

Exemple : `PRINT BIN$(10)` va afficher : `%00001010`

De même que # est le signe définissant un nombre hexadécimal, % est le signe définissant un nombre décimal.

Le contraire de PRINT BIN$(10) est PRINT %00001010 qui donnera comme résultat la valeur 10.

```
10   PRINT "ENTREZ UN NOMBRE DECIMAL :  ";
20   INPUT A
30   PRINT A;" CORRESPOND A ";BIN$(A);" EN BINAIRE."
```

Le système binaire n'accepte que les nombres positifs compris entre 0 et 65535.

Ce programme va afficher les équivalences décimales, hexadécimales et binaires des nombres de 0 à 255. Un appui sur une touche interrompra momentanément le listing tandis qu'un autre appui le fera continuer.

```
10   FOR I=0 TO 255
20    PRINT I;"   ";HEX$(I);"   ";BIN$(I)
30    IF KEY$ THEN GET A$
40   NEXT
```

CHAPITRE 8

LES COMMANDES ET FONCTIONS GRAPHIQUES BASSE RESOLUTON

Votre ordinateur met à votre disposition 3 écrans graphiques basse résolution que nous allons détailler ci-dessous.

A/ LE MODE TEXT

PRINT

Le mode TEXT est celui dans lequel vous êtes lorsque vous allumez votre ordinateur. Le mode TEXT est tout spécialement prévu pour l'affichage de texte, d'où son nom. Vous connaissez déjà l'instruction PRINT pour afficher une chaîne de caractères.

```
10   PRINT "MODE TEXT"
```

PRINT tout seul sautera une ligne

```
10   PRINT
```

2 chaînes de caractères liées par un point-virgule (;) seront affichées sur la même ligne.

```
10   PRINT "ORIC ";
20   PRINT "TELESTRAT"
```

PRINT est une des instructions les plus utilisées en BASIC, puisqu'elle permet à tout moment d'afficher une chaîne, un nombre, une variable, ... l'écran TEXT est composé de 28 lignes (de 0 à 27) de 40 colonnes (de 0 à 39) chacune.

L'écran se divise en 8 zones de tabulations de 8 colonnes chacune. Si l'on met une virgule à la fin d'une chaîne précédée de PRINT, cette chaîne est affichée au début d'une tabulation.

Ce programme va vous montrer toutes les tabulations de l'écran :

```
10   COUNT 100
20    PRINT "SALUT",
30   UNCOUNT
```

Lorsque le TELESTRAT affiche une variable numérique, il met automatiquement un espace avant si elle est positive, et un - si elle est négative.

```
10      PRINT  "J'AI";FR;"FRANCS  EN  POCHE."
```

PRINT @Y,X;... affiche une chaîne ou une variable aux coordonnées X (entre 0 et 39) et Y (entre 0 et 26).

```
10   PRINT @15,10;"ICI ON EST EN X=10 ET Y=15 !"
```

PRINT @15,10; ... affiche à l'écran une chaîne ou une variable au coordonnées X=10 et Y=15.

```
10   PRINT "ENTREZ VOTRE NOM :";
20   INPUT A$
30   PRINT :PRINT "COORDONNEE X (ENTRE 0 ET 39) :";
40   INPUT X
50   PRINT :PRINT "COORDONNEE Y (ENTRE 0 ET 26) :";
60   INPUT Y
80   CLS
90   PRINT @Y,X;A$
```

La ligne 80 contient une instruction inconnue. Mais que fait-elle ? Eh bien tapez CLS en mode direct. L'écran s'efface. Le rôle de CLS est de nettoyer l'écran en y effaçant tout. Le curseur se retrouve en haut et à gauche. CLS s'obtient en mode direct en tapant CTRL L.

SPC$(n) lié avec PRINT laisse n espaces avant l'affichage d'une chaîne.

```
10   PRINT SPC$(10);"ICI SPC$ LAISSE 10 ESPACES."
```

PLOT

Une instruction similaire à PRINT @Y,X;... est PLOT X,Y,"chaîne" qui affiche aux coordonnées X et Y la chaîne de caractères donnée.

```
10  PLOT 20,10,"COUCOU !"
```

Place la chaîne "COUCOU !" aux coordonnées X=20 et Y=10.

```
10   CLS
20   FOR Y=0 TO 26
30    PLOT Y,Y,"PLOT ! PLOT ! PLOT !"
40   NEXT
```

Différence avec PRINT : très rapide et ne déplace pas le curseur .

WIDTH

La largeur normale de l'écran est de 40 colonnes. Dès que 40 caractères ont été entrés, il s'en suit un retour chariot (retour au début de la ligne) et un saut de ligne, ce qui nous fait passer au début de la ligne suivante. Le nombre de caractères au bout duquel doit avoir lieu un aller à la ligne est initialement de 40, mais peut être changé par : WIDTH largeur.

Exemple : `WIDTH 10` provoquera l'aller à la ligne à chaque fois que 10 caractères auront été entrés. Laissez une touche appuyée quelques temps pour voir ...quelque chose de plus amusant :

Faites `WIDTH 2`.

L'affichage ne se fait plus que sur deux colonnes ! . Vous avez 2 possibilités pour revenir au mode d'affichage normal : faire WIDTH 40 ou appuyer sur le bouton RESET.

CURSOR OFF, CURSOR SET

Lorsque l'on réalise un programme, la présence du curseur peut devenir gênante lors de son exécution. On aimerait bien pouvoir enlever ce curseur clignotant.
C'est le rôle de CURSOR OFF qui enlève le curseur et de CURSOR SET qui le fait réapparaître. CURSOR OFF et CURSOR SET sont des instructions qui ne marchent qu'en mode programme.

Si vous faites un CURSOR OFF en mode direct, le curseur sera toujours présent : on ne peut pas le faire disparaître, un éditeur pleine page sans curseur est un non sens.

```
10   CURSOR OFF
20   PRINT MIDDLE$("LE CURSEUR A DISPARU !",40)
30   PRINT
40   PRINT MIDDLE$("APPUYEZ SUR UNE TOUCHE",40)
50   GET A$
60   PRINT
70   PRINT MIDDLE$("LE VOILA DE NOUVEAU !",40)
80   PRINT
90   CURSOR SET
```

PAPER, INK

Lorsque vous allumez votre TELESTRAT, les couleurs de départ sont encre blanche sur fond noir.

8 couleurs sont pourtant à votre disposition.

Voici comment les obtenir :

0 NOIR
1 ROUGE
2 VERT
3 JAUNE
4 BLEU
5 MAGENTA
6 CYAN
7 BLANC

PAPER veut dire papier et INK signifie encre. Pour obtenir un écran d'une couleur voulue, il faut taper PAPER + numéro de la couleur choisie.

Exemple :

```
PAPER 1
```

Et l'écran devient rouge. La couleur des caractères est toujours blanche. Pour changer la couleur de l'encre, faire INK + numéro correspondant à la couleur choisie.

INK 0 instaurera par exemple l'encre noire.

Le programme suivant vous révèlera toutes les combinaisons possibles de PAPER et de INK.

```
10   CURSOR OFF
20   FOR I=1 TO 7
30    INK I
40    FOR J=0 TO 7
50     PAPER J
60     FOR X=1 TO 3000
65     NEXT
70    NEXT J
80   NEXT I
90   PAPER 0:INK 7:CURSOR SET
```

Si la couleur de fond est la même que la couleur de l'encre, plus rien n'est visible.

Exemple : `PAPER 2:INK 2`

On appelle "attribut" tous les caractères spéciaux qui contrôlent les couleurs. Lorsque vous faites un PAPER et un INK, les 2 colonnes les plus à gauche de l'écran sont occupées : elles sont réservées pour l'affichage des couleurs.

D'autres attributs spéciaux peuvent être obtenus par la touche ESCAPE ou par la touche FUNCT.

ESCAPE	0	ENCRE NOIRE
ESCAPE A	1	ENCRE ROUGE
ESCAPE B	2	ENCRE VERTE
ESCAPE C	3	ENCRE JAUNE
ESCAPE D	4	ENCRE BLEUE
ESCAPE E	5	ENCRE MAGENTA
ESCAPE F	6	ENCRE CYAN
ESCAPE G	7	ENCRE BLANCHE
ESCAPE H	8	TEXTE STANDARD
ESCAPE I	9	TEXTE ALTERNE
ESCAPE J	10	STANDARD DOUBLE HAUTEUR
ESCAPE K	11	ALTERNE DOUBLE HAUTEUR
ESCAPE L	12	STANDARD CLIGNOTANT
ESCAPE M	13	ALTERNE CLIGNOTANT
ESCAPE N	14	STANDARD DOUBLE HAUTEUR CLIGNOTANT
ESCAPE O	15	ALTERNE DOUBLE HAUTEUR CLIGNOTANT
ESCAPE P	16	PAPIER NOIR
ESCAPE Q	17	PAPIER ROUGE
ESCAPE R	18	PAPIER VERTE
ESCAPE S	19	PAPIER JAUNE
ESCAPE T	20	PAPIER BLEU
ESCAPE U	21	PAPIER MAGENTA
ESCAPE V	22	PAPIER CYAN
ESCAPE W	23	PAPIER BLANC

Si l'on désire que tout l'écran clignote, on regarde dans le tableau ci-dessus et on s'aperçoit que 12 est le code correspondant au clignotement. Un INK 12, et le tour est joué.

Les couleurs peuvent s'obtenir au clavier. Par exemple, ESCAPE + B ou FUNCT + B inscrira toute la ligne où se trouve le curseur en vert, ESCAPE + L ou FUNCT + L la fera clignoter, ...

Une grande particularité du TELESTRAT est qu'il accepte les couleurs dans les listings. Et cela, peu d'ordinateurs se le permettent ! Pour afficher UN GRAND BONJOUR en rouge, faites :

10 PRINT ". Une fois les guillemets ouverts, faites un FUNCT + A puis tapez UN GRAND BONJOUR !" qui s'inscrit en rouge. Si vous faites LIST, vous devez voir que UN GRAND BONJOUR est toujours affiché en rouge. Lors d'un RUN, UN GRAND BONJOUR sera aussi affiché en rouge. Conclusion : les couleurs sont acceptées dans les chaînes de caractères et donc, dans les listings de programmes.

Pour afficher des caractères en double hauteur, il faut tout d'abord savoir que PRINT CHR$(4) demande à ce que tous les caractères soient affichés sur 2 lignes. CHR$(27) correspond au code ASCII de ESCAPE et CHR$(27)+"J" instaure la double hauteur.

Ce programme affiche une chaîne en double hauteur.

```
10   CLS:PRINT
20   PRINT CHR$(4);
30   PRINT CHR$(27);"J";
40   PRINT "COMMENT ALLEZ VOUS ?"
50   PRINT CHR$(4);:PRINT
```

La ligne 10 efface l'écran. La ligne 20 demande à ce que les caractères soient écrits sur 2 lignes en même temps. La ligne 30 nous met en mode double-hauteur standard. La ligne 50 enlève le mode de double écriture des caractères.

IMPORTANT : pour écrire un message en double hauteur, il est obligatoire d'être sur une ligne paire, sinon, le message s'écrira à l'envers.

La vidéo-inverse consiste à afficher les caractères de la couleur inverse de celle présente à l'écran. Par exemple, si vous êtes en fond noir sur encre blanche, les caractères seront affichés en fond blanc sur encre noire en vidéo inverse. Ce mode 'inverse' s'obtient au clavier en tapant CTRL-V qui fonctionne en mode ON/OFF. En mode programme, on utilisera PRINT CHR$(22) pour passer en vidéo inverse et à nouveau PRINT CHR$(22) pour revenir en mode normal.

```
10   PRINT CHR$(22);"QUE PENSEZ VOUS DE LA VIDEO INVERSE?";CHR$(22)
```

B/ LE MODE LORES 0

Le mode LORES 0 s'obtient simplement en tapant **LORES 0**. Comme le mode TEXT, la couleur de fond est noire et l'encre est blanche. Lorsque vous tapez LORES 0, la valeur 8 est mise dans chaque colonne 0 de chaque ligne. La valeur 8 définit un affichage standard. PRINT @Y,X;. .. et PLOT X,Y,... sont les deux instructions qui permettent un affichage de chaînes, variables, ... en mode LORES 0.

Exemple :

```
10   CURSOR OFF
20   LORES 0
30   PRINT @10,10;"LE MODE LORES 0"
40   PLOT 20,20,"C'EST SYMPA !"
50   CURSOR SET
60   TEXT
```

La ligne 60 portant l'instruction TEXT fait passer le TELESTRAT en mode TEXT. Si vous vous mettez en mode LORES 0 et que vous dépassez la ligne 26, l'écran reviendra en mode TEXT, bien que cela ne soit pas visible, les 2 modes ayant les mêmes couleurs de fond et d'encre. En mode LORES 0, CLS ne sert pas à effacer l'écran. L'instruction effaçant l'écran est LORES 0.

C/ LE MODE LORES 1

Voici un mode bien plus utile que le mode LORES 0 et que l'on obtient en entrant LORES 1.

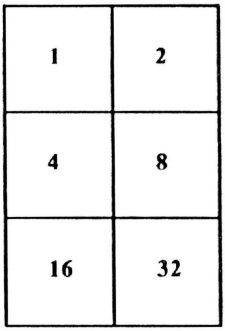
```
10   PAPER 1
20   LORES 1
```

Faites RUN. L'écran devient noir. En effet, l'écran LORES 1 est également en fond noir sur encre blanche. L'écran est divisé en 27 lignes de 40 colonnes. 2 sortes de caractères sont à votre disposition : les caractères standards qui s'utilisent avec le mode TEXT et le mode LORES 0, et les caractères alternés que l'on obtient en mode LORES 1.

Ainsi, si vous avez mis en route le programme donné plus haut et que vous êtes en mode LORES 1, vous remarquez que l'appui sur des touches affiche des caractères bizarres. Ce sont en fait des caractères qui sont divisés en 6 carrés. Affichage de tous les caractères altèrnés disponibles en mode LORES 1 :

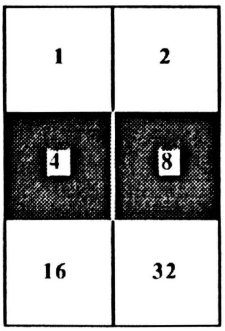
```
10   CLS:PAPER 1
20   LORES 1
30   FOR I=32 TO 110
40    PRINT CHR$(I);
50   NEXT
```

Voici comment trouver le code ASCII et la lettre correspondant à un motif recherché.



Comme dit précédemment, chaque caractère est décomposé en 6 blocs. Chaque bloc peut être vide ou rempli. Pour connaître le code ASCII correspondant à un caractère alterné, il faut le dessiner dans les 6 blocs ci-dessus. Prenons un exemple: nous recherchons le code ASCII du caractère alterné "-".

On le dessine :



On additionne les nombres se trouvant dans les cases pleines. Ce qui nous donne 4+8=12. On ajoute 32 (plus petit code ASCII des caractères normaux) ce qui fait 12+32=44.

Le code ASCII du caractère alterné "-" est donc 44. Essayez :

```
10   CLS:PAPER 1
20   LORES 1
30   FOR I=1 TO 40
40    PLOT I,10,CHR$(44)
50   NEXT
```

Une ligne droite s'affiche. Cela vient du fait que nous avons mis 30 caractères alternés - côte à côte.

```
10   CLS
20   PAPER 1
30   LORES 1
40   FOR I=1 TO 1000
50    PRINT CHR$(50);
60   NEXT
```

Vous trouverez en annexe 4, les caractères alternés correspondants aux codes ASCII allant de 32 à 94.

CHAPITRE 9

LES COMMANDES GRAPHIQUES HAUTE RESOLUTION

HIRES

L'instruction HIRES (abrégé de HIgh RESolution) fait passer votre ORIC en mode haute résolution graphique. Dans de ce mode, l'écran se compose d'une grille de 200x240 points.

Seules 3 lignes en bas de l'écran restent en mode basse résolution (mode TEXT) et vous permettent d'afficher des messages en même temps que des graphismes. C'est dans le mode HIRES (Haute Résolution) que votre TELESTRAT vous donnera ses meilleures capacités graphiques.

En mode HIRES, la largeur de l'écran est appelée coordonnée X et la hauteur coordonnée Y. X peut varier entre 0 et 239, et Y entre 0 et 199. Cette surface de 240x220 points que comprend l'écran HIRES est bien sûr destinée au dessin.

CURSET

CURSET est une instruction qui définie la position du 'curseur'. En fait, il n'y pas de curseur en mode HIRES, mais CURSET sert à définir la position de départ d'une droite, d'un cercle, d'un rectangle ...

CURSET X,Y,FB demande trois paramètres :

X : il s'agit de la coordonnée horizontale du 'curseur' graphique (entre 0 et 239).
Y : il s'agit de la coordonnée verticale du curseur graphique (entre 0 et 199).
FB : le code FB (FORE / BACK) est utilisé par la majorité des instructions de dessin.

Voici comment est affiché un point à l'écran selon la valeur de FB :

Si FB vaut 0 : le point à afficher aura la couleur de l'arrière plan (papier) et ne sera pas visible à l'écran.
Si FB vaut 1 : le point prendra la couleur du premier plan (encre) et sera visible à l'écran.
Si FB vaut 2 : la couleur du point sera inversée. Si par exemple le point était de la couleur du premier plan, alors il deviendra de la couleur de l'arrière plan et vice-versa.
Si FB vaut 3 : la couleur du point n'est pas modifiée. La valeur prise par le paramètre FB doit être entre 0 et 3.

```
10   HIRES
20   CURSET 120,100,1
```

La ligne 10 nous fait passer en mode HIRES. La ligne 20 affiche un point aux coordonnées X=120 et Y=100, soit au milieu de l'écran. Le point est visible car le paramètre FB vaut 1, ce qui correspond à l'affichage du point de la couleur de l'encre. (L'écran HIRES est initialisé avec du papier noir et de l'encre blanche).

```
10    HIRES
20    FOR I=1 TO 199
30      CURSET I,I,1
40    NEXT
50    FOR I=199 TO 1 STEP -1
60      CURSET I,I,2
70    NEXT
80    GOTO 10
```

Les lignes de 20 à 40 dessinent une droite de 200 points qui coupe l'écran. Puis, les lignes 50 à 70 effacent cette droite. Cette disparition des points vient du paramètre FB qui a pris la valeur 2 : tous les points sont inversés.

CURMOV

Il est quelquefois utile de déplacer le curseur d'un certain nombre de points dans les directions horizontales ou verticales. CURMOV est l'instruction qui se charge d'effectuer ce déplacement.

CURMOV X,Y,FB a la même syntaxe que CURSET, à savoir que :

X représente le déplacement de la coordonnée X (entre -239 et 239),
Y le déplacement de la coordonnée Y (entre -199 et 199),
FB le paramètre d'avant arrière plan (entre 0 et 3).

```
10   HIRES
20   CURSET 50,50,1
30   CURMOV 10,0,1
```

Cet exemple affiche un point aux coordonnées X=50 et Y=50 et et déplace le curseur de 10 points vers la droite. Le curseur se trouve désormais en X=60 (50+10) et Y=50 (50+0).

X et Y peuvent être négatifs. Si le point de départ est en 100,100 par exemple, et que l'on veuille faire reculer le curseur de 10 horizontalement, le CURMOV sera CURMOV -10,0,1, ce qui donne :

```
10   HIRES
20   CURSET 100,100,1
30   CURMOV -10,0,1
```

Et le curseur se trouve en 90 (100-10) ,100.

DRAW

Plus intéressantes que les instructions destinées à tracer des points, voici celles qui vont dessiner des droites :

DRAW X,Y,FB trace une droite de X et de Y points à partir de la position de curseur. FB est toujours le paramètre d'avant arrière plan.

Les cordonnées X et Y de DRAW sont relatives, c'est à dire qu'elles sont ajoutées aux coordonnées courantes du curseur graphique pour obtenir le point jusqu'auquel on tracera une droite, le point de départ étant donné par la position du curseur.

Exemple :

```
10   HIRES
20   CURSET 80,50,1
30   DRAW 50,10,1
```

La ligne 20 précise que le curseur graphique est aux coordonnées X=80 et Y=50. La ligne 30 demande à ce que l'on trace à partir de ce curseur une droite de 50 points en horizontal et de 10 points en vertical.

Une ligne droite est donc tracée entre le point de départ en 80,50 et le point d'arrivée en 80+50,50+10, soit en 130,60.

```
10   HIRES
20   FOR Y=0 TO 199
30   CURSET 0,Y,1
40    DRAW 239,0,1
50   NEXT
```

Les coordonnées X et Y suivant DRAW peuvent également être négatives si l'on désire tracer des droites à gauche ou en haut de la position actuelle du curseur.

Lorsque que vous utiliserez des instructions graphiques, tout dépassement des intervalles donnés pour chaque paramètre d'une instruction verra le message '`valeur incorrecte`' s'inscrire.

L'écran HIRES s'efface en tapant HIRES, et les trois lignes de texte en bas de l'écran s'effacent avec CLS. Pour revenir en mode texte, utilisez TEXT.

ADRAW

Bien plus utile et facile à utiliser dans le tracé de droites, **ADRAW X1,Y1,X2,Y2,FB** trace directement une droite entre les points de coordonnées X1,Y1 et X2,Y2.

Exemple : pour tracer une droite entre le point 10,30 et le point 150,160, un simple :

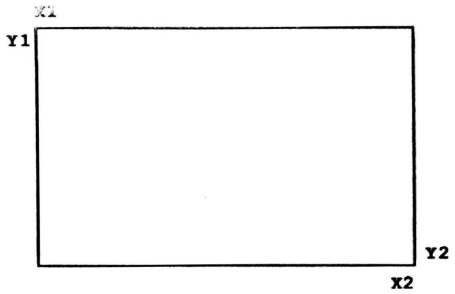
```
10   HIRES
20   ADRAW 10,30,150,160,1
```

fera l'affaire ...

Autre exemple :

```
10   HIRES
20   ADRAW 0,0,239,199,1
30   ADRAW 239,0,0,199,1
```

Va dessiner une croix à l'écran. On ne fait d'ailleurs que tracer une droite entre les 2 extrémités opposées de l'écran. Pour bien se repérer en mode HIRES, il est préférable de bien visualiser les coordonnées des extrémités de l'écran :



CIRCLE

Passons à l'affichage des cercles. **CIRCLE R,FB** trace un cercle de rayon R avec le paramètre FB.

Essayez :

```
10   HIRES
20   CURSET 100,100,1
30   CIRCLE 50,1
```

CIRCLE 50,1 affiche un cercle de rayon 50 et de la couleur de l'encre. (FB=1). Le centre de ce cercle est le point défini par CURSET.

Pour que le point au milieu du cercle soit éteint, le paramètre FB de CURSET 100,100,1 doit être à 0, ce qui donne : CURSET 100,100,0.

On arrive avec un peu de pratique à réaliser d'assez belles figures géométriques avec l'instruction CIRCLE. C'est ce que va vous montrer ce programme :

```
 10   HIRES
 20   INK 3
 30   CURSET 120,100,1
 40   FOR I=1 TO 100
 50    CIRCLE I,1
 60   NEXT
 70   HIRES
 80   CURSET 120,100,1
 90   FOR I=1 TO 100 STEP 2
100    CIRCLE I,1
110   NEXT
120   HIRES
130   CURSET 120,100,1
140   FOR I=1 TO 100 STEP 5
150    CIRCLE I,1
160   NEXT
```

BOX

Après les cercles, voici venus les rectangles. 2 instructions ont pour tâche de tracer des rectangles : La première est **BOX H,V,FB**: H est la largeur du rectangle, V la hauteur, et FB le paramètre avant arrière plan.

La position du curseur détermine l'endroit où sera affiché le coin supérieur gauche du rectangle.

```
10   HIRES
20   CURSET 10,10,1
30   BOX 50,40,1
```

La ligne 20 dit que le coin supérieur gauche du rectangle affiché à la ligne 30 se trouvera en 10,10. BOX 50,40,1 trace un rectangle de 50 points de longueur et de 40 points de hauteur.

Autre exemple de BOX :

```
10   HIRES
20   INK 3
30   FOR I=1 TO 180 STEP 5
40    CURSET 20,20,1
50    BOX I,I,1
60   NEXT
```

Une fois un rectangle dessiné avec BOX, les coordonnées du curseur graphique sont les mêmes. BOX se traduit par 'boîte'.

ABOX

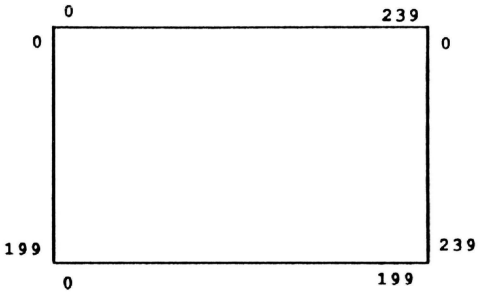
La deuxième instruction dessinant des rectangles se nomme ABOX. C'est un peu le même genre d'amélioration que ADRAW comparé à DRAW : il n'y a pas besoin de déterminer la position de départ du rectangle, celle-ci étant spécifiée dans les paramètres de ABOX.

ABOX X1,Y1,X2,Y2,FB trace un rectangle dont le coin supérieur gauche est en X1,Y1 et le coin inférieur droit est en X2,Y2. FB a toujours la même fonction.

```
10   HIRES
20   ABOX 20,20,100,100,1
```

Trace un rectangle dont le côté supérieur gauche est en 20,20 et dont le côté inférieur droit est en 100,100. On en déduit donc que la longueur du rectangle est de 80 points (X2-X1 = 100-20 = 80) et la hauteur est également de 80 points (Y2-Y1 = 100-20 = 80).

La position courante du curseur n'est pas affectée par un ABOX, et ne varie donc pas.



CHAR

Le mode TEXT est le mode destiné par excellence à afficher des caractères. Le mode HIRES, bien que plus spécialement prévu pour le dessin, accepte aussi l'affichage de caractères. Mais pas directement. Il convient de passer par 2 instructions, à savoir CHAR ou SCHAR.

CHAR N,S,FB affiche sur un écran HIRES des caractères standards ou alternés.

N : c'est le code ASCII du caractère à afficher.

S : S=0 si le caractère à afficher est standard, et S=1 si le caractère à afficher est alterné.

FB : comme toujours, FB (entre 0 et 3) est le code d'avant/arrière plan.

Le coin supérieur gauche du caractère demandé sera affiché à la position courante du curseur.

Exemple :

```
10   HIRES
20   CURSET 100,100,0
30   CHAR 65,0,1
```

CHAR 65,0,1 affiche un A au milieu de l'écran. 65 est le code ASCII du caractère A. 0 demande à ce que le A soit pris dans l'ensemble des caractères standards. (Essayez une deuxième fois le programme ci-dessus avec S=1 pour obtenir un caractère alterné).

Le paramètre FB de CURSET 100,100,0 est mis à 0 pour que seul le caractère affiché par CHAR soit visible.

N'oubliez pas que si vous ne connaissez pas le code ASCII d'un caractère, vous avez la possibilité d'utiliser le caractère entre quote ('). Ainsi : CHAR 65,0,1 correspond à CHAR 'A', 0,1

```
10   HIRES
20   X=10:Y=10
30   A$="BONJOUR, COMMENT CA VA ?"
40   FOR I=1 TO LEN(A$)
50    CURSET X,Y,0
60    CHAR ASC(MID$(A$,I,1)),0,1
70    X=X+8:Y=Y+8
80   NEXT
```

SCHAR

Bien que CHAR permette d'afficher soit un caractère standard, soit un caractère alterné, cette instruction ne permet pas d'afficher des chaînes de caractères. Du moins, pas très facilement, comme vous le montre le dernier programme : on est obligé d'extraire et d'afficher un par un les caractères d'une chaîne.

Aussi existe-t-il l'instruction SCHAR "chaîne de caractères" qui affiche une chaîne entière en HIRES.

```
10    HIRES
20     CURSET 10,100,1
30     SCHAR "ORIC TELESTRAT, THE BEST COMPUTER IN THE WORLD !"
```

Le curseur graphique se déplace de 6 points à droite à chaque fois qu'un caractère est affiché. A noter que la chaîne inscrite à l'écran sera obligatoirement de la couleur du premier plan (encre), le paramètre FB étant forcé à 1.

```
10   HIRES
20   CURSET 0,100,0
30   FOR I=32 TO 127
40    SCHAR CHR$(I)+" "
50   NEXT
```

Affiche tous les caractères standard sur l'écran HIRES.

```
10   HIRES
20   CURSET 0,100,0
30   FOR I=32 TO 127
40    CHAR I,1,1 ' OU SCHAR CHR$(I+128)+" "
50   NEXT
```

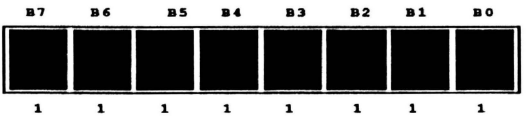
PATTERN

Une instruction que vous allez tout de suite trouver très intéressante est PATTERN. Normalement, lorsque vous tracez une droite, un cercle, un rectangle, ..., les dessins sont tracés en lignes continues.

PATTERN *(valeur) va vous laisser déterminer* si vous ne voulez tracer qu'un point sur deux d'une droite, qu'un point sur trois ...

Le registre de PATTERN utilise 8 cases. Les cases pleines sont considérées comme des points qui seront tracés, et les cases vides comme des points non tracés. A l'initilisation du TELESTRAT, toutes les cases de PATTERN sont remplies, ce qui nous donne :

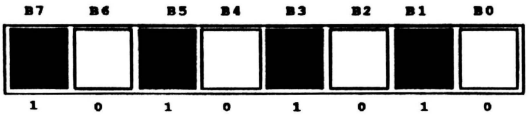
VALEUR DE PATTERN A L'INITIALISATION



%11111111 = 255 = #FF

Une fois que l'on a rempli certaines des 8 cases du registre PATTERN pour déterminer si les points seront affichés ou non, on remplace chaque case remplie par un 1 et chaque case vide par un 0.

Et l'on se retrouve en mode binaire. Toutes les cases étant remplies, %11111111 donne 255 en décimal. C'est la valeur la plus élevée qu'accepte PATTERN. Lorsque PATTERN vaut 255 (PATTERN 255), tous les points sont tracés. Essayons d'obtenir un affichage d'un point sur deux seulement. Cela nous donnera des pointillés. On fait comme tout à l'heure, à savoir que l'on commence par dessiner une grille de 8 cases. Nous n'allons remplir qu'une case sur 2.



%10101010 = 170 = #AA

On remplace chaque case remplie par 1 et chaque case vide par 0, ce qui nous donne le motif binaire %10101010 qui correspond à 170 en décimal. Faites un PATTERN 170, et un point sur deux seulement sera tracé. Essayez :

```
10   HIRES
20   PATTERN 170
30   CURSET 0,0,1
40   BOX 150,150,1
```

Et vous obtiendrez un rectangle dont les côtés sont affichés en pointillés. Pareil pour les cercles :

```
10   HIRES
20   PATTERN 170
30   CURSET 120,100,1
40   CIRCLE 90,1
```

Il n'est pas nécessaire de faire un dessin du registre, puis d'y remplir certaines cases, puis de les transformer en 1 ou 0, puis de transformer ces 1 ou 0 en décimal ...

En sachant qu'un 1 correspond à une case remplie et un 0 à une case vide, il suffit de mettre la valeur après PATTERN en binaire.

Exemple : PATTERN %11001100 va afficher 2 points allumés, puis deux points éteints. Chaque point allumé est remplacé par un 1, et chaque point éteint par un 0.
Les couleurs existent également en mode HIRES. PAPER et INK ont le même rôle qu'en mode TEXT. Ils donnent une couleur au papier ou à l'encre suivant la valeur donnée après PAPER ou INK.

Voici le tableau des couleurs en mode HIRES qui est du reste le même que pour le mode TEXT :

0	NOIR
1	ROUGE
2	VERT
3	JAUNE
4	BLEU
5	MAGENTA
6	CYAN
7	BLAN
8	TEXTE STANDARD
10	DOUBLE HAUTEUR
12	CLIGNOTEMENT

PAPER 3 mettra l'écran HIRES en fond jaune, INK 2 en encre verte ... Lorsque vous changez la couleur de l'écran HIRES, les points de 0 à 11 de chaque ligne sont utilisés. On ne pourra donc plus y dessiner des cercles, des rectangles ...

FILL

Le remplissage d'une partie de l'écran avec une couleur donnée se fait avec FILL.

FILL L,H,A remplit L lignes de H segments de large avec la couleur A. L'écran graphique est divisé en 200 lignes, chaque ligne étant elle même divisée en 40 segments de 6 points chacun.

FILL servira à colorier un rectangle de L lignes de haut (entre 0 et 200) et de H segments (entre 0 et 40) avec la couleur A (entre 0 et 255). Le coin supérieur gauche du rectangle à colorier est défini par CURSET.

```
10   HIRES
20   CURSET 120,100,1
30   FILL 100,20,17
```

Va remplir à partir du milieu de l'écran (120,100) 100 lignes de 20 segments chacune de couleur rouge. 17 est le code couleur rouge.

Vous trouverez les codes allant avec FILL au chapitre précédent (voir tableau avec ESC et FUNCT).

Chaque ligne étant composée de 40 segments de 6 points, vous pouvez mettre 40 couleurs différentes par ligne. Comme il y a 200 lignes, l'écran HIRES peut contenir jusqu'à 8000 couleurs différentes !

```
10   HIRES
20   PAPER 3
30   CURSET 120,100,3
40   FILL 200,20,192
50   CURSET 120,0,3
60   FOR I=1 TO 90 STEP 5
70    CIRCLE I,1
80   NEXT
```

Et encore :

```
10 HIRES
20 CURSET 0,0,1
30 FILL 200,40,193
```

Et hop, vous êtes en prison !

CHAPITRE 10

LES COMMANDES ET FONCTIONS SYSTEME DU BASIC

REM ET '

Un long programme BASIC devient vite illisible et surtout, incompréhensible. C'est pour l'éviter qu'existe l'instruction REM qui permet d'insérer des REMarques tout au long d'un listing. Très utile pour documenter un programme (expliquer sa structure), REM se place généralement en début de ligne. (Mais peut aussi suivre des instructions).

```
10   REM CALCUL D'UN NOMBRE A LA PUISSANCE 2
20   PRINT "ENTREZ VOTRE NOMBRE : ":
30   INPUT A
40   B=A↑2
50   PRINT  A;" A LA PUISSANCE 2 DONNE";B
```

La ligne 10 nous informe sur ce qu'est ce programme : il calcule un nombre à la puissance 2. Pendant l'exécution, la REM est ignorée. Elle n'a aucune influence sur le déroulement d'un programme. Tout ce qui suit une REM sur la ligne où il se trouve est ignoré. Ainsi:

```
10    REM CA VA ? :PRINT
```

Ici, le PRINT ne sera pas exécuté car il suit la REM.

Par contre :

```
10    PRINT : REM CA VA  ?
```

Le PRINT sera exécuté car il précède la REM.

```
10   REM CE PROGRAMME UNIQUEMENT
20   REM COMPOSEE DE REM EST TRES
30   REM UTILE, PUISQU'IL NE FAIT
40   REM RIEN !!!
```

L'apostrophe est un abrégé de REM et a la même fonction dans un programme.

```
10 ' L'APOSTROPHE, EST COMME
20 ' REM, ET SERT A COMMENTER
30 ' UN PROGRAMME.
```

Il y a pourtant deux différences entre REM et ' :

- Si l'on utilise l'apostrophe en début de ligne, laisser obligatoirement un espace entre le numéro de la ligne et le '.
- L'apostrophe n'a pas besoin d'être précédée de 2 points en milieu de ligne, ce qui n'est pas le cas de REM.

```
10    GOTO 100: REM ALLER AU SOUS PROGRAMME LIGNE 100
20    GOTO 100' ALLER AU SOUS PROGRAMME LIGNE 100
```

En ligne 10, les 2 points sont obligatoires entre GOTO 100 et REM. En ligne 20, ils peuvent être omis avec '.

LIST

Vous connaissez déjà LIST qui affiche un programme à l'écran. Une partie seule d'un programme se 'liste' en indiquant après LIST les deux numéros de ligne (départ et fin) séparés par une virgule. Copiez le programme suivant :

```
10 '
20 '
30 '
40 '
50 '
```

LIST affiche tout le programme.

LIST 20,40 devrait afficher les lignes 20 à 40 y compris. Mais un bug (erreur) s'est glissé dans le programme qui effectue l'instruction LIST. Et si vous faites un LIST 20,40 la ligne 50 est aussi affichée. Il faudra faire avec. LIST 30 affiche toutes les lignes à partir de la ligne 30.

DELETE

Effacer une ligne de programme se fait en entrant directement le numéro de celle-ci et en tapant RETURN. Effacer un programme en entier se fait en tapant NEW. Et pour détruire un ensemble de lignes ?

Si nous avons par exemple 200 lignes à effacer dans un programme, nous pourrions les détruire une par une. Mais ce serait long, trop long. Aussi existe-t-il la fonction DELETE (ligne de début, ligne de fin) qui efface toutes les lignes comprises entre les 2 numéros de lignes spécifiés après DELETE. Si un seul numéro de ligne est donné après l'instruction DELETE, les lignes seront effacées depuis la ligne donnée jusqu'à la fin.

Exemples fictifs: DELETE 100,160 détruit les lignes de 100 à 160.
DELETE 0,150 détruit les lignes depuis le début jusqu'à la ligne 150.
DELETE 150 détruit les lignes depuis la ligne 150 jusqu'à la fin.

ATTENTION : DELETE tout seul correspond à un NEW et efface tout le programme. De plus, l'instruction DELETE ne s'utilise qu'en mode direct.

NEW

NEW est l'instruction vue au chapitre 4 qui détruit entièrement un programme ainsi que les variables stockées en mémoire qui sont toutes remises à 0. Il est recommandé de faire un NEW à chaque fois que l'on copie un nouveau programme. Ceci afin d'éviter que des lignes non désirables de l'ancien programme cohabitent avec celles du nouveau.

CONT

L'exécution d'un programme peut être reprise après son arrêt par un CTRL-C ou par un STOP en entrant en mode direct le mot CONT. Cette instruction fait CONTinuer l'exécution à la ligne où a eu lieu l'arrêt du programme.

Contrairement à RUN, les variables ne sont pas effacées, et c'est là où réside l'intérêt de CONT.

Tapez:

```
10   FOR I=1 TO 100000
20    PRINT I
30   NEXT
```

Puis RUN. Des nombres apparaisent. Interrompez ce programme en tapant CTRL-C. Faites ensuite CONT en mode direct. L'exécution du programme reprend, et la variable I n'est pas remise à 0.

DIM

L'utilisation des tableaux est très utile pour la conservation de données numériques ou alphanumériques. La première chose à faire est de DIMensionner un tableau en indiquant le nombre d'éléments qu'il comprendra. Un tableau se dimensionne par DIM constante (nombre d'éléments du tableau).

Par exemple, DIM A(5) réservera en mémoire de la place pour 6 éléments de A(0) jusqu'à A(5).

A(0)=200 donne la valeur 200 au 1er élément du tableau A(5). Un tableau de 11 éléments peut être utilisé sans que l'on ait besoin de le dimensionner.

Si l'on déclare par exemple D(3)=13 sans avoir précédemment utilisé un DIM, un tableau de 10 éléments est automatiquement créé. L'ordinateur fera tout simplement un DIM D(10) à votre place.

Particularité de l'instruction DIM : **elle doit obligatoirement être placée en début de ligne et ne peut être suivie d'aucune instruction**.

```
10   DIM ELEMENT(50) ' RESERVE DE LA PLACE EN MEMOIRE POUR UN
TABLEAU DE 51 ELEMENTS
20   FOR I=0 TO 50
30    ELEMENT (I)=2*I
40   NEXT
```

La ligne 10 crée un tableau de 51 éléments (de 0 à 50 = 51 élements). La boucle FOR ... NEXT donne à chaque élément le double de la valeur de l'indice qu'a chaque élément. Vous pouvez maintenant faire en mode direct :

```
FOR  I=0  TO  50:PRINT  ELEMENT(I):NEXT
```

Pour qu'une variable chaîne soit conservée dans un tableau, on ajoute un $ au nom de la variable qui définit ce tableau. DIM A$(10) réserve de la place en mémoire pour 11 chaînes de caractères.

```
 5   DIM AMIS$(50)
10   PRINT "COMBIEN AVEZ VOUS D'AMIS ? ";
30   INPUT A
50   PRINT :PRINT "ENTREZ LE NOM DE CHACUN DE VOS";A;"AMIS"
60   PRINT
70   FOR I=0 TO A
80    PRINT "NOM DE L'AMI NO";I;" :";
90    INPUT AMI$(I)
100  NEXT
110  PRINT "MERCI"
120  PRINT :PRINT
130  PRINT "VOUS AVEZ DONC";A;" AMIS QUI SE NOMMENT :":PRINT
140  FOR I=0 TO A
150   PRINT AMI$(I);" - ";
160  NEXT
```

L'intérêt des tableaux réside dans le fait qu'ils permettent le stockage d'éléments d'information similaires de données sans la définition d'une variable individuelle pour chaque élément séparément, mais aussi dans le fait qu'ils permettent l'indexation de leurs indices.

CLEAR

CLEAR est une instruction qui met toutes les variables à 0, y compris les tableaux. Les variables numériques sont mises à 0 et les chaînes de caractères sont vidées.

Exemple : A=1

Puis CLEAR.

Et PRINT A

Réponse : 0 car CLEAR a effacé toutes les variables.

Autre exemple :

```
 10   DIM A(10)
 20   FOR I =0 TO 10
 30    A(I)=I
 40   NEXT
 50 ' AFFICHAGE DES VALEURS DU TABLEAU A(10) AVANT UN CLEAR
 60   FOR I=0 TO 10
 70    PRINT A(I)
 80   NEXT
 90   CLEAR:PRINT :PRINT
100 ' AFFICHAGE DES VALEURS DU TABLEAU A(10) APRES UN CLEAR
110   FOR I=0 TO 10
120    PRINT A(I)
130   NEXT
```

Lors d'un RUN, un CLEAR est automatiquement effectué.

TALK SET, TALK OFF

L'entrée en mode direct de l'instruction TALK SET provoque l'affichage des messages de compilation (fait de transformer un programme BASIC en langage machine) d'un programme BASIC.

La compilation se fait en deux passes, et la première passe est divisée en 3 parties (a,b,c).

Vous voyez donc s'afficher :

```
Passe  Ia-Ib-Ic-II
```

Il y a également affichage du numéro de la ligne en cours de compilation.

L'intérêt de l'instruction TALK SET est qu'elle donne le numéro de la ligne où un label n'a pas été trouvé :

Tapez :

```
 10     GOTO EDIT
```

Puis entrez LIST

Le message `Passe Ia-Ib-Ic-II` apparaît et `10 ligne/label EDIT absent` également.

Mais imaginez que vous ayez un programme de plusieurs centaines de lignes. Comment se rappeller où se trouve la ligne GOTO EDIT ?

Entrez TALK SET puis LIST

Le message "`Passe Ia-Ib-Ic-II`" apparaît, et en dessous : `10 Label EDIT absent`.

Cette fois- ci, vous savez que le label absent est en ligne 10. Plus besoin de perdre des heures à le rechercher !

TALK OFF stoppe le mode TALK SET. Il n'y a plus affichage des messages de compilation.

CHAPITRE 11

LES CONSTANTES ET LES FONCTIONS DE TRAVAIL SUR LES NOMBRES

Nombreux sont ceux qui croient que tout ce qui a trait aux mathématiques est forcément difficile. Il ne faut pas se baser sur des idées préconçues. Nous allons vous montrer dans ce chapitre qu'avec un peu de concentration et de bonne volonté, les fonctions mathématiques s'acquièrent toutes seules. Nous n'allons d'ailleurs pas étudier que des fonctions mathématiques, mais aussi des fonctions de travail sur les nombres par lesquelles nous allons d'ailleurs commencer.

RAND

Les jeux seraient bien monotones sans hasard. La fonction RAND (limite) tire un nombre entier au hasard entre 0 et le nombre limite indiqué entre parenthèses.

```
10   A=RAND(40)
20   PRINT A
```

A=RAND (40) affecte à la variable A un nombre tiré au hasard entre 0 et 39. La limite précisée pour tirer un nombre au hasard est exclue. RAND (10) tirera un nombre entre 0 et 9, mais le nombre 10 sera exclu. Pour que ce nombre soit inclu, on peut faire un RAND (11).

RND

La fonction RND tire également des nombres au hasard. Mais contrairement à RAND qui tire des nombres entiers, RND(i) tire un nombre compris entre 0 (compris) et 1 (exclus). Le paramètre i suivant RND vaut généralement 1. Le nombre tiré au hasard sera donc compris entre 0 et 0.999999.

`PRINT RND(1)` vous donnera un nombre compris dans l'intervalle (0,0.999999).

Un nombre compris entre 0 et 1 n'a pas grand intérêt me direz vous. Oui, mais pourquoi ne pas le multiplier par un autre nombre pour qu'il soit compris dans un intervalle désiré ?

RND(1)*5 tire un nombre compris entre 0 et 4.999999. On a tout simplement multiplié RND (1) par le nombre, limite exclue, que l'on voulait.

INT

Tant que vous utiliserez RND, le nombre tiré au hasard ne sera pas forcément entier. Cependant, INT (nombre) sert à convertir un nombre décimal en nombre entier. Le nombre généré avec un INT est toujours inférieur au nombre original.

`PRINT INT (10.1)` donnera `10`

`PRINT INT (8.9)` donnera `8`

Et `PRINT INT (-15.3)` renverra `-16` car -16 est le nombre inférieur à -15.3. Mais revenons en à RND. La fonction INT va nous aider à convertir en entier un nombre décimal obtenu avec RND.

```
10   PRINT "ENTREZ UNE LIMITE POUR RND :";
20   INPUT A
30   NB=INT(RND(1)*A)
40   PRINT "LE NOMBRE ENTIER TIRE AU HASARD EST";NB
50   PRINT "CE NOMBRE EST COMPRIS ENTRE :";0;" <";NB;" <";A-1
```

RND(0) tire un nombre aléatoire, mais celui-ci sera toujours le même si l'on refait à nouveau un RND (0).

```
10     COUNT 10
20       PRINT RND(0)
30     UNCOUNT
```

RANDOM

Lorsque l'on utilise les fonctions RND et RAND, l'ordinateur ne tire pas les nombres "au hasard". Ces nombres sont calculés grâce à une racine qui produit des tables de nombres aléatoires. Ainsi, plusieurs nombres tirés par RND ou RANDOM proviendront d'une même racine. RANDOM va servir à changer cette racine et à la réiniatiliser avec un nombre tiré au hasard.

Exemple :

```
10   FOR I=1 TO 10
20    COUNT 10
30     RANDOM
40     PRINT INT(RND(10)*10)+1
50    UNCOUNT
60    PRINT:PRINT
70   NEXT
```

Voici un petit jeu qui illustre la fonction RAND :

```
  10   CLS:PRINT
  20   PRINT "LE BUT DU JEU CONSISTE A TROUVER LE"
  30   PRINT "NOMBRE TIRE AU HASARD ENTRE 0 ET 100 ..."
  40   PRINT :PRINT :RANDOM:X=RAND(101)
  50   PRINT "VOTRE NOMBRE (ENTRE 1 ET 100) : ";
  60   INPUT NB
  70   IF X>NB THEN PRINT :PRINT "LE NOMBRE A TROUVER EST SUPERIEUR
A";NB:PRINT :GOTO 50
  80   IF X<NB THEN PRINT :PRINT "LE NOMBRE A TROUVER EST INFERIEUR
A ";NB:PRINT :GOTO 50
  90   PRINT "BRAVO !!! C'ETAIT BIEN LE NOMBRE";X
 100   PRINT "VOULEZ VOUS REJOUER ? (O/N) ";
 110   INPUT REP$
 120   IF REP$="O" THEN 10 ELSE END
```

TRUE, FALSE

2 constantes de travail sur les nombres qui s'utilisent généralement avec structure de condition IF ... THEN ... ELSE sont TRUE et FALSE.

TRUE se traduit par VRAI et renvoie la valeur 1 pour signifier que le résultat d'une expression est vraie.

FALSE (FAUX) renvoie la valeur 0 pour signifier que le résultat d'une expression est fausse.

Exemple avec TRUE :

```
10   A=30
20   REPEAT
30    PRINT A;
40    A=A-1
50   UNTIL A=TRUE
```

UNTIL A=TRUE n'arrête la boucle REPEAT ... UNTIL que quand A=1 (TRUE=1).

Exemple avec FALSE :

```
10   PRINT "ENTREZ UN  NOMBRE >5"
20   INPUT NB
30   IF NB>5 THEN PRINT TRUE ELSE PRINT FALSE
```

Renvoie la valeur 1 si le nombre entré est supérieur à 5, et 0 si le nombre est inférieur ou égal à 5.

Les variables TRUE et FALSE ont surtout comme rôle de simplifier la lecture d'un programme.

NOT

NOT est un opérateur logique qui veut dire NON. Normalement, la valeur 0 est affectée à une condition fausse, et la valeur 1 à une condition vraie. NOT inverse cette valeur.

On l'utilise surtout avec IF ... THEN ... ELSE pour orienter un programme si une condition n'est pas réalisée. Cela donne IF NOT (condition) THEN ... (ELSE ...).

La condition suivant NOT doit être obligatoirement entre parenthèses.

```
10   PRINT "ENTREZ LE NOMBRE 9 SVP :";
20   INPUT A
30   IF NOT(A=9) THEN 10 ELSE PRINT "MERCI D'AVOIR ENTRE UN OEUF!"
```

Dans un programme, un IF NOT (W) correspond à IF W=0. De même que FALSE et TRUE, NOT sert en particulier à simplifier la lecture d'un programme, car :

```
10   INPUT A$
20   IF NOT(A$="OK") THEN PING
```

équivaut à :

```
10   INPUT A$
20   IF A$<>"OK" THEN PING
```

Passons maintenant aux instructions mathématiques de votre TELESTRAT.

ABS

ABS (n) renvoie la valeur absolue du nombre n. Cette fonction est très utile lorsque l'on ne désire travailler qu'avec des nombres positifs.

```
10    PRINT ABS(-10)
```

Donnera 10 car la valeur absolue de -10 est 10.

ABS(1) donne 1,
ABS(-54) donne 54,
ABS(0) donne 0, ...

SGN

SGN (n) permet de connaître le signe d'un nombre n.

Si le nombre est positif, SGN vaudra +1.
S'il est négatif, SGN vaudra -1,
S'il est nul, SGN vaudra 0.

Exemple :

```
PRINT  SGN(10)
```

Résultat : 1 car 10 est un nombre positif.

```
10   PRINT "ENTREZ UN NOMBRE :";
20   INPUT A:PRINT
30   PRINT "C'EST UN NOMBRE ";
40   IF SGN(A)=-1 THEN PRINT "NEGATIF"
50   IF SGN(A)=0 THEN PRINT "NUL"
60   IF SGN(A)=1 THEN PRINT "POSITIF"
```

SGN est un abrégé de (S)i(GN)e

DEGRE, RADIAN

Pour tous les calculs mathématiques, deux modes sont mis à votre disposition : le mode DEGRE, et le mode RADIAN.

A l'allumage du TELESTRAT, c'est le mode RADIAN qui est présent par défaut. Pour passer en mode degré, on tape tout simplement DEGRE.

Pour convertir un nombre en degrés x en radians, il faut effectuer le calcul :

(x*PI/180)

Pour convertir un nombre en radians x en degrés, il faut effectuer le calcul :

(x/PI*180)

PI

PI est une constante et vaut normalement 3.14159265.

```
PRINT  PI
```

Si l'on est en mode RADIAN, PRINT PI va donner 3.14159265. En mode DEGRE, PI=180 degrés.

```
10    RADIAN ' mise en mode RADIAN
20    PRINT PI ' Affiche valeur de PI en radians
30    PRINT ' Saute une ligne
40    DEGRE ' Mise en mode DEGRE
50    PRINT PI ' Affiche la valeur de PI en degrés
```

La constante PI est très utile pour le calcul des aires et des circonférences de cercles.

L'aire d'un cercle est PI*rayon*rayon.

La circonférence d'un cercle est 2*PI*rayon.

```
 1 ' CALCUL DE L'AIRE ET DE LA CIRCONFERENCE D'UN CERCLE
 2 '
10   HIRES:RADIAN
20   PRINT "ENTREZ UN RAYON (1-99) :  ";
30   INPUT RAYON
40   IF RAYON<1 OR RAYON>99 THEN PRINT "J'AI DIT ENTRE 1 ET 99 !"
:GET A$:GOTO 10
50   CURSET 120,100,0:CIRCLE RAYON,1
60   CLS:PRINT "L'AIRE DE CE CERCLE EST";PI*RAYON*RAYON
70   PRINT "LA CIRCONFERENCE EST : ";2*PI*RAYON
```

DEG. RAD

Une fois que le mode DEGRE ou le mode RADIAN est entré, tous les calculs sont faits dans un de ces deux modes.

Cependant, deux instructions permettent de passer 'momentanément' dans le mode opposé.

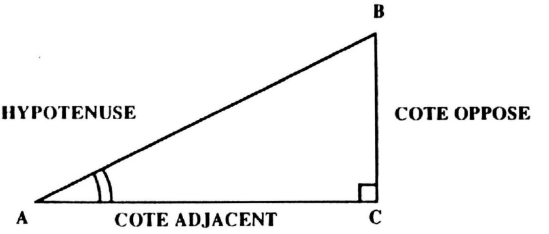
DEG (angle) renvoie la valeur en degrés d'un angle exprimée en radians.
RAD (angle) renvoie la valeur en radians d'un angle exprimée en degrés.

Ces deux instructions ne sont à employer que lors d'un calcul à un endroit donné d'un programme d'une valeur en degrés ou en radians. Si tous les calculs sont faits dans le même mode, autant instaurer directement le mode voulu avec DEGRE et RADIAN.

```
10   CLS
20   PRINT
30   PRINT "ENTREZ UN ANGLE EN RADIANS : ";
40   INPUT ANGLE
50   PRINT "CELA DONNE";DEG(ANGLE);" DEGRES"
60   PRINT
70   PRINT "ENTREZ UN ANGLE EN DEGRES  : ";
80   INPUT ANGLE
90   PRINT "CELA DONNE";RAD(ANGLE);" EN RADIANS"
```

COS. SIN. TAN

Les fonctions COS (n), SIN (n) et TAN (n) calculent le COSinus,le SINus, et la TANgente de l'angle n donné.



Dans un triangle rectangle ABC, le SINUS est toujours le COTE OPPOSE à l'angle en A.

Ici:

$$\text{Le côté adjacent (COS) est égal à } \frac{AC}{AB}$$

$$\text{Le côté opposé (SIN) est égal à } \frac{BC}{AB}$$

$$\text{La tangente (TAN) est égal à } \frac{BC}{AC}$$

$$TAN(x) = SIN(x) / COS(x)$$

ATN

La fonction inverse de TAN (x) est ATN(x) (Arctangente). Elle renvoie la valeur d'un angle dont ont connaît la tangente.

Remarque : n'oubliez jamais que la valeur donnée après une fonction trigonométrique sera exprimée en radians ou en degrés selon le mode dans lequel se trouve le TELESTRAT.

```
 10   RADIAN
 20   PRINT SIN(90)
 30   DEGRE
 40   PRINT SIN(90)
```

Traçons la courbe représentative de la fonction COS :

```
 10 ' COURBE DE COS
 20   HIRES
 30   DEGRE
 40   CURSET 0,100,1
 50   DRAW 239,0,1
 60   CURSET 0,0,1:DRAW 0,199,1
 70   FOR I=199 TO 678
 80    CURSET X,100+COS(I)*90, 1
 90    X=X+0.5
100   NEXT
```

Dans le calcul 100+COS(I)*90 en ligne 80, 90 exprime l'amplitude de la courbe. Vous pouvez la faire varier entre 1 et 98.

Plus l'amplitude sera faible, et plus la courbe sera aplatie.
Plus l'amplitude sera importante, et plus la courbe sera haute.

Si vous voulez que la courbe de COS ne s'arrête pas à la fin de l'écran, tapez ces deux lignes :

```
 70     FOR I=199 TO 5000
 95     IF X=240 THEN X=0
```

Il arrivera un moment où aucune courbe ne se tracera plus car les points de la nouvelle courbe seront les mêmes que ceux de l'ancienne.

Courbe représentative de SIN :

```
 10   HIRES
 20   DEGRE
 30   CURSET 0,100,1:DRAW 239,0,1
 40   CURSET 0,0,1:DRAW 0,199,1
 50   FOR I=179 TO 658
 80    CURSET X,100+SIN(I)*90,1
 90    X=X+0.5
100   NEXT
```

Vous savez déjà que CIRCLE s'occupe de tracer un cercle. Mais il serait intéressant de savoir comment les points formant ce cercle sont déterminés. Chaque point de coordonnées X,Y est calculé avec les fonctions SIN et COS.

X=COS(I)*R
Y=SIN(I)*R

I est un nombre variant de 0 à 2*PI. Car 2PI est le nombre qu'il faut parcourir pour faire le tour d'un cercle de rayon 1. R est le rayon du cercle.

Ce programme trace un cercle de rayon donné en mode HIRES :

```
 10   HIRES
 11   PRINT "RAYON (1-90) : ";
 12   INPUT RAYON
 20   DEGRE
 30   FOR I=0 TO 2*PI STEP 0.01
 40    CURSET 100+COS(I)*RAYON,100+SIN(I)*RAYON,1
 50   NEXT
```

Et en mode TEXT :

```
 10   TEXT:CLS
 20   CURSOR OFF:DEGRE
 30   FOR I=0 TO 2*PI STEP 10
 40    PLOT 20+COS(I)*5,15+SIN(I)*5,"*"
 50   NEXT
```

Courbe SIN en mode TEXT :

```
  5   CLS:CURSOR OFF
 10   FOR I=0 TO 3600 STEP 10
 20    PRINT SPC$(20+18*SIN(I));"*"
 30   NEXT
```

Changez le SIN en ligne 20 par un COS pour découvrir la courbe COS en mode TEXT.

Et pour en terminer avec COS et SIN, suivons le parcours d'un petit circuit auto simulé avec COS :

```
 10 ' CIRCUIT AUTO
 20   TEXT
 30   CLS
 40   CURSOR OFF
 50   DEGRE
 60   FOR T=0 TO 3600 STEP 10
 70    PRINT SPC$(15+10*COS(T));"*"
 80    PRINT SPC$(25+10*COS(T));"*"
 90   NEXT
100   RADIAN
```

LOG

LOG (n) calcule le logarithme décimal d'un nombre n. Le nombre n doit être positif, et N doit être différent de 0 car LOG (0) n'existe pas.

Courbe représentative de LOG (+LOG et -LOG) :

```
10 ' LOG
20   HIRES
30   DEGRE
40   CURSET 0,100,1:DRAW 239,0,1
50   CURSET 0,0,1:DRAW 0,199,1
60   FOR I=1 TO 240
70    CURSET X,100-LOG(I)*40,1
80    CURSET X,100+LOG(I)*40,1
90    X=X+1
100  NEXT
```

EXP

EXP (n) est la fonction exponentielle et renvoie le nombre 2.7183 à la puissance n.

```
10   DEGRE
20   PRINT EXP (10)
```

LN

LN (n) donne le Logarithme Népérien d'un nombre n. Il s'agit de la fonction inverse de EXP.

```
10   DEGRE
20   PRINT LN (10)
```

SQR

SQR (n) donne la racine carrée du nombre n. La racine carrée d'un nombre correspond à son élévation à la puissance 1/2.

SQR(10) = 10 ⇑ 0,5

Une racine carrée est obligatoirement positive.

On a pour SQR un exemple de l'utilité de ABS qui renvoit la valeur absolue d'un nombre. Quel que soit le nombre n que l'on donnera à SQR, ABS(n) nous permettra de le rendre positif.

```
10   FOR I=0 TO 10
20    PRINT RACINE DE";I;"=";SQR(I);
30   NEXT
```

Et

```
10   FOR I=-10 TO 0 STEP -1
20    PRINT "GRACE A ABS : ":PRINT
30    PRINT " RACINE DE";I;"=";SQR(ABS(I))
40   NEXT
```

Résumé des (nombreuses) fonctions mathématiques :

`ABS(n)`	Valeur absolue de n
`ATN(n)`	Arc tangente de n
`COS(n)`	Cosinus de n
`EXP(n)`	Exponentielle de n
`INT(n)`	Partie entière de n
`LN (n)`	Logarithme népérien de n
`LOG(n)`	Logarithme décimal de n
`PI`	Constante PI : 3.14159265
`SGN(n)`	Renvoi 1 si n>0, -1 si n<0 et 0 si n=0
`SIN(n)`	Sinus de n
`SQR(n)`	Racine carrée de n
`TAN(n)`	Tangente de n

Ceux qui désireraient en savoir plus sur les fonctions mathématiques peuvent directement consulter des ouvrages s'y référant. Et que ceux qui n'ont rien compris ne s'inquiètent pas. Ces instructions sont en géneral très très peu utilisées. On ne s'en sert que dans des programmes du type traceur de courbes, etc...

CHAPITRE 12

LES COMMANDES DE GESTION DU SON

Le son produit par le TELESTRAT sort du haut-parleur de votre téléviseur ou moniteur.

ZAP, PING, SHOOT, EXPLODE, OUPS.

Sans que l'on n'ait besoin d'aucune théorie musicale, on peut déjà utiliser 5 sons prédéfinis, à savoir :

ZAP	(bruit de rayon laser)
PING	(bruit de clochette)
SHOOT	(bruit de coup de feu)
EXPLODE	(bruit d'explosion)
OUPS	

En tapant simplement un des 5 mots BASIC (ZAP, PING, SHOOT, EXPLODE, OUPS) appartenant aux sons prédéfinis, un bruit se fait entendre.

Exemple :

```
ZAP
```

Ou :

```
EXPLODE
```

Ou :

```
OUPS
```

WAIT

Pour qu'un son soit répété, on inclut l'instruction produisant ce son dans une boucle.

```
10   COUNT 500
20    PING
30   UNCOUNT
```

L'instruction PING répétée 500 fois (COUNT 500:PING:UNCOUNT) ne permet pas de distinguer les 500 bruits consécutifs de clochette. Le programme va trop vite.

L'instruction WAIT t provoque un arrêt du programme pendant t dixième de seconde.

WAIT 5 attend 5/10ème de seconde, soit 1/2 seconde.
WAIT 10 attend 1 seconde.

Incluons dans le programme précédent un WAIT 5 :

```
10   COUNT 500
20    PING
30    WAIT 5
40   UNCOUNT
```

Cette fois-ci, chaque PING est séparé du suivant par 1/2 seconde d'intervalle. Chaque coup de clochette se distingue de l'autre. Le bruit donné ressemble d'ailleurs à une alarme.

```
10   COUNT 10
20    PING:WAIT 10
30    ZAP:WAIT 10
40    SHOOT:WAIT 10
50    EXPLODE:WAIT 10
60    OUPS:WAIT 10
70   UNCOUNT
```

PLAY

Disposer de sons prédéfinis est bien, mais pouvoir créer des sons soi-même est encore mieux. L'ORIC a trois canaux de sons. Avoir 3 canaux veut dire que 3 sons différents peuvent être joués en même temps. Chaque canal produit soit de la musique, soit des bruits, ou soit de la musique et des bruits. (au choix).

Avant de jouer un air, il convient de définir les canaux qui seront utilisés. C'est l'instruction PLAY qui se charge de ce travail. PLAY (jouer) ne génère pas un son, mais ouvre simplement les canaux dont on a besoin pour jouer un air.

La syntaxe de PLAY est : **PLAY t,s,e,d**

* Le paramètre t détermine l'ouverture de certains canaux et est compris entre 0 et 7.

Voici la liste des canaux ouverts et fermés selon la valeur de t :

t	CANAL 1	CANAL 2	CANAL 3
0	FERME	FERME	FERME
1	OUVERT	FERME	FERME
2	FERME	OUVERT	FERME
3	OUVERT	OUVERT	FERME
4	FERME	FERME	OUVERT
5	OUVERT	FERME	OUVERT
6	FERME	OUVERT	OUVERT
7	OUVERT	OUVERT	OUVERT

Quand t=5, les canaux 1 et 3 sont ouverts.
Quand t=0, les 3 canaux sont fermés.
Quand t=7, les 3 canaux sont ouverts, ...

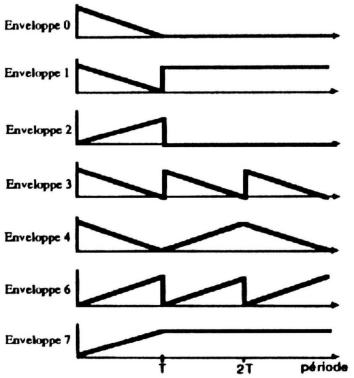
* Le paramètre s détermine les canaux sur lesquels bruit et musique peuvent être mélangés. s (entre 0 et 7) se comporte comme le paramètre t.

Si s=5, le bruit et la musique seront mélangés sur les canaux 1 et 3. Si s=7, le bruit et la musique pourront être mélangés sur les 3 canaux.

* Les paramètres e et d déterminent l'enveloppe et la durée de l'enveloppe d'un son. e varie entre 0 et 7 et d entre 0 et 32767.

Quand un son est à un volume constant, son enveloppe est égale à 0. Dans ce cas, nous n'avons aucun besoin de la notion d'enveloppe. Mais si nous voulons faire varier un son en volume pendant une durée donnée, l'enveloppe prend à ce moment toute son importance.

Le graphe des 7 différentes enveloppes disponibles vous est présenté ci-dessous :



Pour obtenir un son qui monte progressivement en volume, puis qui se stabilise, c'est l'enveloppe 7 que nous allons utiliser. La courbe de l'enveloppe 7 monte en effet progressivement, puis se stabilise définitivement.

Vous remarquez que les enveloppes 0 et 2 stoppent le son à la moitié d'une période, tandis que les enveloppes 1,3,4,6 et 7 sont continues.

Après avoir choisi l'enveloppe du son voulu, il faut choisir la durée de cette enveloppe en millièmes de secondes (entre 0 et 32767). Si s=5000, alors la durée sera de 2500/1000 èmes de seconde, soit de 2,5 secondes.

Si nous choisissons l'enveloppe 7 à nouveau, on remarque que la courbe monte pendant sa période (durée du son), et se stabilise jusqu'à la fin. Pour une période de 5 secondes, le son montera pendant 5 secondes, puis se stabilisera définitivement. (l'enveloppe 7 est continue).

SOUND

L'instruction SOUND produit un son selon les paramètres entrés.

SOUND c,h,v

* c (entre 0 et 6) définit le canal sur lequel sera joué le son.

c=1 : Le canal 1 émet de la musique.
c=2 : Le canal 2 émet de la musique.
c=3 : Le canal 3 émet de la musique.
c=4 : Le canal 1 émet de la musique et du bruit.
c=5 : Le canal 2 émet de la musique et du bruit.
c=6 : Le canal 3 émet de la musique et du bruit.

* h (entre 0 et 2000) définit la hauteur de la note. Plus h est près de 0, et plus la note est aigüe. Plus h est près de 2000, et plus la note est grave.

* v (entre 1 et 16) définit le volume du son joué. 15 est le volume maximum, et 1 le volume minimum. On utilise un volume égal à 0 ou 16 lorsque l'on veut prendre en compte le paramètre d'enveloppe d'un son avec l'instruction PLAY.

Exemples avec SOUND :

```
 10   PRINT
 20   PRINT "ENVELOPPE SONORE NO (1 A 7) ? ";: INPUT ENV
 30   PRINT "DUREE DU SON (0 A 32767) ? ";:INPUT DUREE
 40   PRINT "C'EST PARTI ..."
 50   WAIT 10
 60   SOUND 1,1500,16
 70   PLAY 1,0,ENV,DUREE
 80   PRINT :PRINT"APPUYEZ SUR UNE TOUCHE POUR ARRETER"
 90   GET A$:PLAY0,0,0,0
100   GOTO 10
```

PLAY 0,0,0,0 en ligne 90 désactive tous les canaux et fait s'arrêter le son.

```
 10 ' VAGUES
 20   PLAY 0,1,0,0 ' OUVRE LE CANAL 1 SUR LE BRUIT
 30   A=INT (RND(1)*2)
 40   FOR I=0 TO 31
 50    SOUND 4,A,10
 60    WAIT A
 70   NEXT
 80   GOTO 30
```

****

```
 10   PLAY 1,0,0,0
 20   FOR I=0 TO 99
 30    SOUND 1,80*(I/4),12
 40   NEXT
 50   SOUND 1,0,0
```

****

```
 10   FOR I=1 TO 50
 20    SOUND 1,50,16
 30    PLAY 1,0,1,70
 40    WAIT 10
 50   NEXT
```

****

```
 10   SOUND 1,200,13:WAIT 1
 20   SOUND 1,300,13:WAIT 1
 30   SOUND 1,100,10:WAIT 1
 40   SOUND 1,0,0
```

****

```
 10 ' SON AVION
 20   SOUND 4,10,16
 30   PLAY 1,1,7,0
```

```
10 ' EN VOITURE ! LE  TRAIN DEMARRE ...
20   SOUND 4,100,0
30   PLAY 1,1,3,200
```

Après chaque programme, vous pouvez arrêter le son qui est continu en tapant CTRL F.

Le calcul de la fréquence F d'un son en fonction de sa période P se fait par F=1E6/P/16. Pour obtenir la période P en fonction de sa fréquence F: P=1E6/P/16

MUSIC

MUSIC est avec SOUND une instruction sonore qui s'utilise généralement avec PLAY.

Syntaxe : **MUSIC c,o,n,v**

* c (entre 0 et 3) est le numéro de canal sur lequel sera joué un son.

* o (entre 0 et 6) est le numéro de l'octave sur lequel sera jouée la note. Plus la valeur de l'octave est près de 0, et plus la note jouée est grave. Plus la valeur de l'octave est près de 6, et plus l'octave est aigüe. o=2 correspond au DO du milieu d'un piano.

* n est la hauteur de la note dans la gamme (de 1 à 12 pour les 12 demi-tons de la gamme chromatique).

Chaque valeur de n représente une note, à savoir :

1	2	3	4	5	6	7	8	9	10	11	12
Do	Do#	Ré	Ré#	Mi	Fa	Fa#	Sol	Sol#	La	La#	Si
		Ré b		Mi b			Sol b		La b		Si b

(dièse) désigne une note aigüe, et b (bémol) une note grave.

* v (de 0 à 16) est le volume de la note. 1 = volume nul. 15 = volume maximum. 16 ou 4 définit par PLAY.

Exemple :

La gamme chromatique (ensemble des notes possibles) vous est donnée par ce programme :

```
10   FOR OC=0 TO 6
20    FOR NO=1 TO 12
30     MUSIC 1,OC,NO,12
40     WAIT 3
50    NEXT NO
60   NEXT OC
```

****

Transformer son TELESTRAT en mini-piano n'est pas impossible. Comme en témoigne le programme ci-dessous.

Les touches de 1 à 9 simulent les notes de Do à Sol# (Voir correspondance numéro-note plus haut.). Les touches 0, - et = simulent les notes de 10 à 12. La touche O augmente de 1 la valeur de l'octave. Arrivée à 6, l'octave repasse à 0.

L'arrêt du programme se fait par un appui sur Q. Il est préférable d'enlever le bruit des touches avant de démarrer le programme. (CTRL F)

```
10   GET A$
11   IF A$<>"O" AND OC=6 THEN 20
12   IF OC<6 THEN OC=OC+1 ELSE OC=0
20   A=VAL(A$)
30   IF A$="-" THEN A=11
40   IF A$="=" THEN A=12
50   IF A$="Q" THEN PLAY 0,0,0,0:END
60   IF A=0 THEN A=10
70   MUSIC 1,OC ,A,12
80   WAIT 2:PLAY 0,0,0,0
90   GOTO 10
```

****

```
10   FOR I=12 TO 1 STEP -1
20    MUSIC  3,1,I,0
30    PLAY 0,7,1,15000
40   NEXT
```

****

CHAPITRE 13

LES COMMANDES ET FONCTIONS DE GESTION DE DISQUE

Dès que vous éteignez le TELESTRAT, le programme résidant en mémoire est détruit. Il est en effet stocké en mémoire "volatile" (appelée RAM). Tout ce qui est en RAM disparaît quand vous coupez l'alimentation.

Or, vous n'allez pas le retaper à chaque fois que vous en aurez besoin. Cela prendrait des heures. Il faut pouvoir stocker ce programme quelque part où il ne sera pas détruit lors de l'arrêt du TELESTRAT. Et ce quelque part, c'est le lecteur de disquette.

Un lecteur de disquette est un support de stockage. Le nom de votre lecteur est MICRODISC. Le programme gérant le lecteur de disquette est le STRATSED. (STRAT est un abrégé de TELESTRAT et SED signifie Système d'Exploitation du Disque). Lorsque vous allumez votre ordinateur, un message '`Inserer une disquette`' apparaît si aucune disquette n'est présente dans le MICRODISC. En effet, à chaque allumage du TELESTRAT, le STRATSED présent sur la disquette doit être chargé en mémoire.

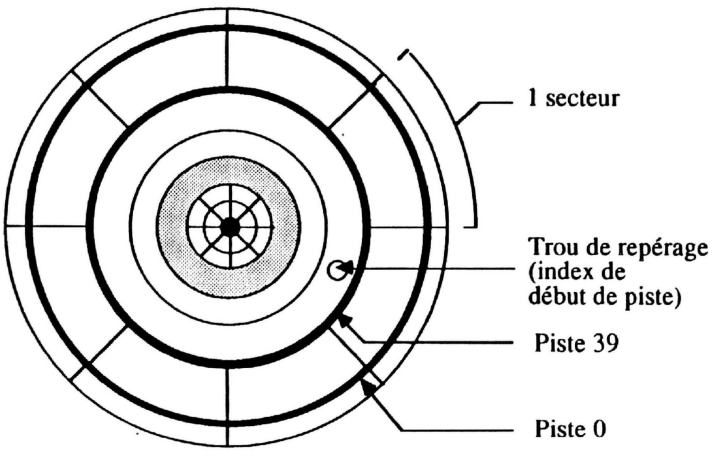
La disquette est un des supports de stockage les plus modernes et les plus rapides.

A/ LES DISQUETTES

Le format des disquettes ORIC est au standard 3" (3 pouces). Une disquette 3" est une sorte de boitier rigide de 8x10 cm. A l'intérieur de ce boitier se trouve une feuille plastique circulaire (71 mm de diamètre) qui conserve des informations.

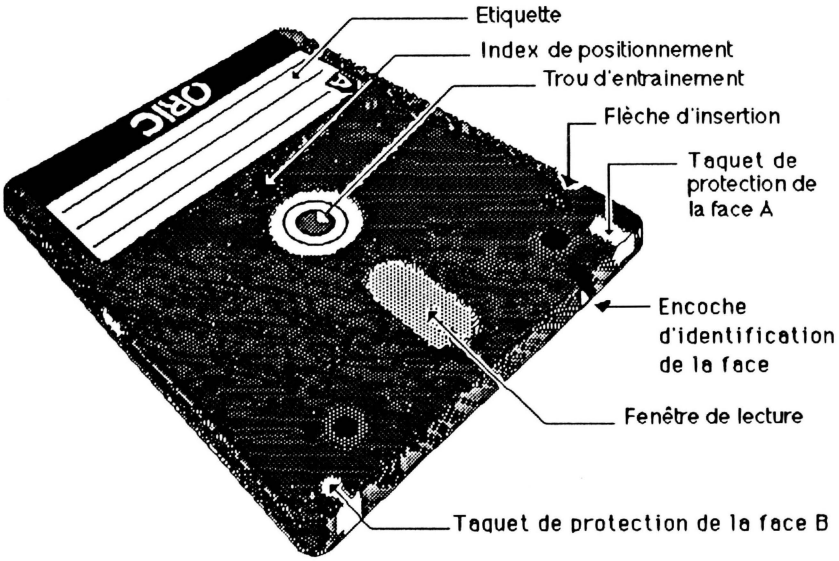
Pour pouvoir retrouver rapidement des programmes sur une disquette, le TELESTRAT procède à un découpage "imaginaire" de la disquette. Chaque face est divisée en pistes (jusqu'à 42 par face), et chaque piste est divisée en secteurs (jusqu'à 19 secteurs par piste). Chaque secteur est composé de 256 octets.

Voici la structure d'une disquette :



Une disquette 3" dispose, de même qu'une cassette, de deux faces. La face A et la face B (ou la face 1 et la face 2). Le MICRODISC est un lecteur double face, c'est à dire qu'il lit les deux faces d'une disquette sans que l'on ait besoin pour cela de la retourner. Donc, lorsque vous introduisez une disquette dans le lecteur, la face A doit être obligatoirement en haut . Une disquette avec la face B en haut ne rentrera pas.

Voici l'anatomie d'une disquette 3 pouces :



Sur le haut de la disquette, vous trouvez deux taquets blancs. Lorsque ceux-ci sont enfoncés, la disquette est protégée en écriture . (On ne peut plus ni sauver des programmes, ni en effacer). Votre lecteur étant double face, il suffira de pousser le taquet de la face A pour que toute la disquette soit protégée contre l'écriture et l'effacement de programmes.

Le côté de la disquette portant une étiquette est le côté que l'on tient lors de l'insertion de celle-ci dans le lecteur.

Au milieu de la disquette se trouve la fenêtre de lecture. Elle est protégée par un volet qui ne disparaît que lors de l'entrée de la disquette dans le lecteur. Bien que la disquette 3" soit réputée pour sa solidité et sa résistance, il convient de respecter certaines règles lors de son usage :

- Après utilisation d'une disquette, la remettre dans sa boîte.

- Ne jamais ouvrir le volet de la disquette et toucher la surface magnétique. La survie de vos programmes en dépend !

- Ne pas poser d'objets lourds dessus.

- Eloigner les disquettes des sources de rayonnement électromagnétiques. (moniteurs, alimentation, aimants ...)

- Une disquette ne doit pas être exposée à une chaleur ou à une humidité excessive. En moyenne, la température doit être entre 5 à 50 degrés.

Le TELESTRAT accepte jusqu'à 4 lecteurs. Chaque lecteur est appelé par une lettre de l'alphabet allant de A à D. Ainsi, le lecteur livré avec le TELESTRAT est appelé lecteur A. Le deuxième sera appelé lecteur B, ... Pour brancher un second lecteur au premier, il faut relier le cable en nappe grise partant du lecteur B à la prise présente sur le cable en nappe grise du lecteur A. Pour 1 ou 2 lecteurs supplémentaires, une autre alimentation sera nécessaire. Pour 3 lecteurs, deux alimentations.

Pour une disquette de 42 pistes par face, chaque piste comprenant 17 secteurs et chaque secteur 256 octets, la capacité de stockage est de 42(pistes)x17(secteurs)x256(octets)x2(faces)= 365568 octets ou encore 365568/1024 = 357 Ko par disquette (1 Kilo-octets=1024 octets)

B/ SYNTAXE DES NOMS DE FICHIERS

Un nom de fichier est généralement composé de 3 parties :

- Le nom du lecteur

Le nom du lecteur sur lequel sera chargé, sauvé, détruit , ... le programme désiré est une lettre de A à D suivi d'un tiret (-). Si le nom du lecteur n'est pas précisé, la valeur par défaut du n° du lecteur sera prise en compte. (Le lecteur par défaut à l'allumage est le lecteur A).

- Le nom du fichier

Un programme est reconnu grâce à son nom. Ce nom comprend 9 caractères au maximum. Ceux-ci doivent être alphanumériques ou numériques. Aucun signe, symbole n'est accepté. Un message '`nom de fichier incorrect`' est généré si une erreur a été commise dans la frappe d'un nom de fichier.

- L'extension.

Il s'agit d'un groupe de 3 caractères au plus séparés du nom de fichier par un point. L'extension permet de distinguer un type de fichier.

Par exemple, pour un fichier appelé BAS : on saura que ce fichier est un programme BASic.
Pour .COM, ce sera du COde Machine (langage machine) ...

Exemples de noms de fichiers :

ORIC.BAS charge sur le lecteur A (lecteur par défaut) le programme de nom ORIC ayant l'extension .BAS. Il s'agit donc d'un programme BASIC. Le nom du lecteur n'étant pas précisé, le programme est chargé à partir du lecteur par défaut (Lecteur A à l'initialisation du TELESTRAT). ORIC.BAS correspond ici à A-ORIC.BAS

B-PRG.SAS charge sur le lecteur B le programme de nom PRG et d'extension .SAS

DD.H charge le programme de nom DD et d'extension .H à partir du lecteur par défaut A.

VOITURE charge le programme de nom VOITURE à partir du lecteur A. Le nom de l'extension n'étant pas donné, l'extension par défaut (.COM) est prise en compte. VOITURE équivaut donc à VOITURE.COM ou encore à A-VOITURE.COM

C-DODO charge le programme DODO.COM à partir du lecteur C.

Exemples de noms invalides:

E-RIDEAU Le lecteur E n'existe pas. Il n'y a que 4 lecteurs de A à D.

XXXXXXXXXXX.BAS Nom de fichier trop long.

TOTO.BASIC Extension trop longue (3 caractères maximum).

S+.VDT Le caractère + n' est pas alphanumérique ou numérique ==> Erreur.

Tout caractère minuscule présent dans un nom de fichier ou d'extension sera transformé en majuscule. (essai.bas devient ESSAI.BAS)

DIR

La disquette fournie avec le TELESTRAT est sans doute actuellement dans votre lecteur. Et vous aimeriez bien savoir ce qu'elle contient. DIR affiche le catalogue d'une disquette, c'est à dire que tous les noms des fichiers présents sur la disquette sont affichés.

Exemple : `DIR`

Prenons une disquette imaginaire. Voici ce qui s'affiche :

```
A- (DF/42/17) Nom: ESSAI CATALOGUE

JEU       .BAS   10      DESSIN    .COM 33
IMPRIME   .CAT    3      ECRAN     .SCR  6
TOTO      .BIN   24P

1250 secteurs libres,  5 fichiers
```

Pour expliquer ce qu'affiche le catalogue d'une disquette, commençons par le haut :

`A- (DF/42/17) Nom : ESSAI CATALOGUE`

A- : Veut dire que c'est le catalogue de la disquette présente dans le lecteur A qui est effectué.

(DF/ : Pour dire que le lecteur A est un lecteur (D)ouble (F)ace. (SF pour un (S)imple (F)ace)

42/17) : annonce que la disquette est composée de 42 pistes par face et de 17 secteurs par piste.

Nom : ESSAI CATALOGUE : ici, c'est le nom donné à la disquette qui apparaît.

Ensuite, tous les noms des fichiers sont affichés.

Par exemple :

`JEU       .BAS  9`

Le nom du fichier présent sur la disquette est JEU. Son extension est .BAS. Et ce fichier occupe 9 secteurs. Enfin, à la fin du catalogue, on trouve :

1250 secteurs libres, 5 fichiers. Qui donne le nombre de secteurs encore disponibles, ainsi que le nombre total de fichiers que comprend la disquette.

L'affichage du catalogue d'une disquette peut être interrompu par une pression sur n'importe quelle touche. Pour continuer, tapez la barre d'ESPACE. Les touches ESC et CTRL C provoquent un arrêt définitif de l'apparition du catalogue.

Un nom de fichier après DIR provoquera l'affichage seul de ce nom, à condition qu'il existe.

Exemple : sur la disquette fournie avec le TELESTRAT se trouve un fichier appelé ANIBAL.BAS

Si vous faites :

```
DIR "ANIBAL.BAS"
```

Seul le nom du fichier ANIBAL.BAS sera donné dans le catalogue. Le nom d'un lecteur peut aussi faire partie du nom de fichier situé après DIR. Par exemple :

```
DIR "C-ANIBAL.BAS"
```

Ou tout simplement : `DIR "C"` pour obtenir le catalogue de la disquette présente dans le lecteur C.

Il est possible d'inclure des JOKERS dans un nom de fichier. Un JOKER (*) s'utilise lorsque l'on veut voir apparaître plusieurs noms de fichiers ayant un rapport entre eux. Le joker (?) remplace une lettre

`DIR "*.COM"` Liste seulement tous les fichiers d'extension .COM.

`DIR "B-*.COM"` Pareil, mais sur le lecteur B.

`DIR "TOT?.COM"` Liste tous les fichiers de 4 lettres commençant par TOT et d'extension .COM

`DIR "TOT?.*"` Liste tous les fichiers de 4 lettres commençant par TOT quelleque soit l'extension.

lecteur-.DRV$

Le lecteur par défaut est le lecteur sur lequel un fichier est chargé, sauvé, détruit, ..., si le nom du lecteur n'est pas précisé avec le nom d'un fichier.

Lorsque l'on allume le TELESTRAT, le lecteur A est le lecteur par défaut. Pour changer le nom du lecteur par défaut, il faut taper le nom du nouveau lecteur suivi d'un tiret.

```
B-
```

Et désormais, tous les fichiers dont le nom du lecteur n'est pas spécifié seront chargés à partir du lecteur B.

PRINT DRV$ affiche le nom du lecteur par défaut :

```
PRINT DRV$
```

On peut aussi se servir de DRV$ comme variable alphanumérique dans un programme lorsque l'on ne connaît pas le nom du lecteur courant.

```
10 LOAD DRV$+"-JEU.BAS"
```

EXT, EXT$

De même que le lecteur par défaut est le lecteur A, l'extension par défaut est .COM.
PRINT EXT$ donne le nom de l'extension par défaut en renvoyant une chaîne de 3 caractères maximum.

EXT "chaîne" change le nom de l'extension par défaut.

EXT "SCR" et l'extension par défaut est désormais .SCR.

Avant un EXT "SCR", TOTO correspondait à TOTO.COM. Désormais, TOTO = TOTO.SCR.

INIT

Lorsque vous achetez une disquette vierge 3" dans le commerce, le TELESTRAT ne peut pas l'utiliser directement. Il faut d'abord qu'il la formate, c'est à dire qu'il "découpe" la disquette en pistes et en secteurs, et qu'il y mette le STRATSED qui est le système d'exploitation du MICRODISC.

INIT est l'instruction qui va demander au TELESTRAT de formater une disquette.

ATTENTION : vous ne pouvez pas utiliser la disquette fournie avec le TELESTRAT pour sauver des programmes. Il faut que vous alliez en acheter une ou plusieurs.

INIT efface l'écran. On vous demande d'insérer la disquette à formater dans le lecteur A, puis d'appuyer sur une touche. Les données courantes sont alors présentées :

A-(DF/42/17) La disquette sera initialisée dans le lecteur A avec 42 pistes par face de 17 secteurs chacune. Le nom de la disquette vous est demandé. (RETURN après avoir donné un nom).

Formater (O/N) : vous demande si la disquette doit être formatée. S'il s'agit d'une disquette que vous venez d'acheter, alors répondez oui. Sinon, ce n'est pas nécessaire si vous voulez simplement détruire tous les fichiers d'une disquette.

Voulez-vous recommencer (O/N) : au cas où vous auriez d'autres disquettes à formater.

La touche ESC permet de sortir prématurement de la procédure d'initialisation d'une disquette.

LOAD

LOAD "nom de fichier" charge en mémoire un fichier situé sur une disquette. LOAD signifie charger. Vous connaissez déjà le mode direct, c'est à dire que seul le nom d'un fichier suffit pour que celui-ci soit chargé.

JEU.BAS <==> LOAD "JEU.BAS"

Pareil dans un programme :

```
10   JEU.BAS
```

ou :

```
10   LOAD "JEU.BAS"
```

Si le nom d'un fichier n'est pas trouvé, le message '**fichier absent**' apparaît.

L'instruction LOAD n'est pas indispensable, mais elle a quand même deux avantages par rapport au chargement d'un fichier en n'indiquant que son nom :

- *LOAD* permet d'utiliser les variables alphanumériques. A$ en mode direct ne chargera pas le fichier dont le nom est dans A$. Tandis que LOAD A$ le fera.

```
10   PRINT "ENTREZ UN NOM DE FICHIER"
20   INPUT A$
30   LOAD A$
```

- Le nom entre guillemets suivant LOAD peut contenir un mot BASIC. C'est impossible en mode direct.

FORMI.DDD n'est pas accepté comme nom de fichier car FOR est une instruction BASIC.

LOAD "FORMI.DDD" est quant à lui accepté.

SAVE

SAVE "nom de fichier" sauve un fichier sur une disquette.

```
10   PRINT "PROGRAME D'ESSAI DE SAUVEGARDE D'UN"
20   PRINT "FICHIER BASIC AVEC SAVE ..."
30   SAVE "A-PRG.BAS"
40   DIR
```

L'instruction SAVE, comme LOAD, accepte le nom du lecteur de disquette, l'extension, ...Si un fichier existe déjà et que l'on tente à nouveau de le sauver par un SAVE, le message '`fichier existant`' apparaît. Si la disquette est protégée en écriture (taquet blanc de la face A enfoncé), un '`disquette protegee`' vous prévient.

Pour qu'un programme soit exécuté dès la fin de son chargement, on rajoute, "AUTO" après le nom de fichier.

```
10    PRINT "LE PROGRAMME SE LANCE TOUT SEUL !"
```

Copiez cette ligne. Puis faites `SAVE "PRG.BAS",AUTO`. Puis `NEW` pour détruire le programme. PRG.BAS le charge à partir du lecteur de disquette, et il se lance automatiquement.

SAVE accepte plusieurs variantes :

- SAVEO "nom du fichier" : si le programme "nom de fichier" existait déjà sur la disquette, il est effacé. Quoi qu'il en soit, le nouveau fichier est sauvé.

Exemple :

```
10 ' ESSAI 1
20   SAVE "ESSAI"
```

Sauve le programme BASIC ESSAI.

```
10 ' ESSAI 2
20   SAVEO "ESSAI"
```

Détruit le fichier ESSAI présent sur la disquette et le remplace par le nouveau programme ESSAI.

- SAVEM "nom de fichier" : avec un SAVEM, le fichier à sauver est ajouté au fichier existant sur la disquette (et portant le même nom) pour n'en former qu'un. Mais attention : cette fusion de deux fichiers n'est qu'une juxtaposition. Les 2 fichiers sont stockés sous le même nom et seront chargés en même temps.

```
10 ' ESSAI SAVEM
20   SAVE "COUCOU.BAS"
30   SAVEM "COUCOU.BAS"
```

- SAVEU "nom de fichier" : pour SAVEU, si un fichier du même nom existe déjà sur la disquette, alors, on lui donne une extension .BAK. Le fichier devient une ancienne version. Si un fichier du même nom avec un .BAK existait déjà, il est détruit.

SAVEU permet une mise à jour de ses programmes, en prenant la sécurité de sauvegarder l'ancienne version avec l'extension .BAK.

Exemple :

```
10 ' ESSAI 1 DE SAVEU
```

Faites `SAVE "ESSAI.BAS"`

On met à jour le programme précédent en ajoutant une ligne par exemple.

```
10 ' ESSAI 1 DE SAVEU
20 ' PRET !
```

Faites un `SAVEU "ESSAI.BAS"`.

L'ancien fichier ESSAI.BAS sur la disquette devient ESSAI.BAK et le nouveau est ESSAI.BAS.

Un bloc mémoire se sauve en indiquant les adresses de début et de fin de ce bloc :

Syntaxe : `SAVE "nom",A adresse de début,E adresse de fin (,AUTO).`

`SAVE "BLOC",A#5000,E#6000` sauve le bloc de mémoire de #5000 à #6000.

Quand un programme n'est pas en BASIC, l'option ,T adresse peut être utilisée. Elle permet d'exécuter un programme à une adresse différente de celle du début du programme sauvé (Ce que fait ,AUTO).

`SAVE "HELLO.ABC",A#5000,E#6000,T#5200` sauve le programme HELLO.ABC.

Lorsque qu'il sera chargé, son exécution commencera à partir de l'adresse #5200.

AUTO-BOOT

Un programme peut être chargé automatiquement à l'allumage du TELESTRAT. Pour cela, le nom du programme doit être obligatoirement BONJOUR.COM. Si ce nom est trouvé à l'allumage, alors il sera chargé. Un RUN est aussi effectué si le programme a été sauvé en AUTO.

Exemple :

```
10   PRINT :PRINT "CE PROGRAMME SE CHARGE AUTOMATIQUEMENT"
20   PRINT "A L'ALLUMAGE DU TELESTRAT !!!"
```

Tapez `SAVE "BONJOUR.COM",AUTO` pour sauver ces 2 lignes avec exécution automatique au chargement. Eteignez le TELESTRAT et rallumez-le : les deux lignes sont automatiquement chargées et exécutées ...

ESAVE

ESAVE est un abrégé de ECRANSAVE ou ECRANSAUVE et sauve sur disquette l'écran dans le mode où l'on se trouve. (TEXT, LORES ou HIRES.)

En mode TEXT ou LORES, ESAVE "ECRAN.ECR" correspond à `SAVEU "ECRAN.ECR",A#BB80,E#BFDF`.

En mode HIRES, ESAVE "ECRAN.ECR" correspond à `SAVEU "ECRAN.ECR",A#A000,E#BF3F`.

Exemple en mode TEXT :

```
ESAVE "ECRAN"
```

Rechargez votre écran en tapant ECRAN.

En mode HIRES :

```
10  HIRES
20  CURSET 120,100 ,1
30  PAPER 1:INK 0
40  FOR I=1 TO 99 STEP 5
50   CIRCLE I,1
60  NEXT
70  ESAVE "CERCLE"
```

Et à chaque fois que vous voudrez voir apparaître le dessin CERCLE, il faudra taper CERCLE.

Attention cependant à se trouver en mode HIRES lors de l'appel d'un écran haute résolution (HIRES), et en mode TEXT ou LORES lors de l'appel d'un écran basse résolution.

DEL, DELBAK

DEL "NF" s'utilise pour effacer un fichier de nom NF.

Attention : après avoir effacé un programme, celui-ci est irrécupérable. En conséquence, utilisez avec soin la fonction DEL.

```
10   SAVE "ESSAIDEL.BAS":PRINT "JE SAUVE LE PROGRAMME BASIC
ESSAIDEL.BAS"
20   PRINT :PRINT "ET MAINTENANT,JE DETRUIS ESSAIDEL.BAS!":WAIT 30
30   DEL "ESSAIDEL.BAS"
```

DEL accepte aussi les JOKERS : `DEL "*.BAS"` efface tous les fichiers qui ont une extension .BAS.

DEL tout seul efface tous les fichiers de la disquette en demandant une confirmation pour chaque fichier.

Par exemple, si le fichier DESSIN.SCR est trouvé, on vous demandera :

```
DESSIN      .SCR  (O/N)
```

Si vous répondez O, le fichier est détruit. Si vous répondez N, le fichier est conservé. Si vous appuyez sur ESC, la procédure est arrêtée.

Effacer tous les fichiers sur le lecteur B s'obtient en ajoutant après DEL le nom du lecteur entre guillemets :

```
DEL  "B"
```

DELBAK détruit tous les fichiers d'extension .BAK. Cela permet de "nettoyer" la disquette en effaçant les anciennes versions des programmes obtenues avec un SAVEU.

DELBAK est l'équivalent de DEL "* .BAK".

PROT, UNPROT

PROT "NF" protège contre l'effacement un fichier de nom NF. Une fois ce fichier protégé par PROT, il ne sera plus possible de l'effacer, à moins d'utiliser UNPROT "NF" qui le déprotègera.

Lorsque vous protégez un fichier, un P apparaît dans le catalogue (DIR), à côté du nom du fichier protégé.

Exemple : si le fichier SERVEUR.VDT se trouve dans le catalogue :

```
DIR  :

SERVEUR    .VDT  15
```

Après `PROT  "SERVEUR.VDT"` et `DIR` :

```
SERVEUR    .  VDT  15P
```

Le P à côté du nombre de secteurs qu'occupe le programme nous prévient que celui-ci est protégé. Si l'on tente d'effacer un fichier protégé, le message "`(nom de fichier) est protege`" apparaît.

UNPROT (déprotège) est le contraire de PROT et enlève la protection en écriture d'un fichier.

`UNPROT  "SERVEUR.VDT"` enlève la protection de SERVEUR.VDT. Le P dans le catalogue disparaît.

Les Jokers sont admis avec PROT et UNPROT :

`PROT  "*.ECR"` protège tous les fichiers d'extension . ECR et `UNPROT  "*.ECR"` les déprotège.

COPY

Cette commande de syntaxe COPY nom de fichier TO nom de fichier fait la copie d'un fichier. On peut éventuellement changer le nom du nouveau fichier à créer.

COPY "JEU1.COM" TO "JEU.COM" fait une copie de JEU1.COM. La copie de JEU1.COM a pour nom JEU.COM

Lors de la copie d'un fichier, on appelle disquette source la disquette où se trouve le programme à copier et disquette cible le programme où doit être copié le programme source (de départ).

Comme SAVE, COPY a des variantes de syntaxe :

`COPYO "NF1" TO "NF2"` copie le fichier de nom NF1 et lui donne le nom NF2, mais efface le fichier NF2 s'il existait déjà .

`COPYO "STRAT.COM" TO "ESC.COM"` copie le fichier NF1 et donne à la copie le nom NF2 , mais le programme NF2 est effacé s'il existe déjà sur la disquette.

`COPYM "NF1" TO "NF2"` colle bout à bout des fichiers NF1 pour n'en former qu'un : NF2. Comme NF1 représente plusieurs fichiers, on utilise les jokers :

`COPYM "*.BAS" TO "TOTO.BAS"` rassemble dans un fichier de nom TOTO.BAS tous les fichiers d'extension .BAS.

COPY et COPYO acceptent aussi les jokers :

`COPY "*.COM" TO "*.SCR"` copie tous les fichiers d'extension .COM en fichier du même nom, mais d'extension .SCR.

`COPYO "A-*.COM" TO "B-*.BAS"` copie tous les fichiers d'extension .COM présent sur le lecteur A en fichiers d'extension .BAS sur le lecteur B.

BACKUP

BACKUP effectue une copie intégrale d'une disquette (la source) sur une autre disquette (la cible). Si un seul lecteur est connecté, la copie se fera sur le lecteur A. Il y aura alors 2x9 manipulations de disquettes au maximum.

Si le lecteur B est branché, la copie se fera du lecteur A vers le lecteur B.

La disquette source doit avoir le même type de formatage que la disquette cible (même nombre de pistes et de secteurs).

Lors de la copie d'une disquette, on vous demande si vous désirez préalablement formater la disquette cible.

Pour éviter des erreurs de manipulation, il est recommandé d'abaisser le taquet de la disquette source avant de procéder à un BACKUP (copie).

BACKUP demande toute la mémoire disponible. Il faut donc impérativement sauver le programme en mémoire (s'il y en a un) avant de copier une disquette.

REN

REN "NF1" TO "NF2" change le nom d'un fichier NF1 en un nouveau nom NF2.

Par exemple : `REN "IMPR.BAS" TO "VITRE.COM"` rebaptise le fichier IMPR.BAS en VITRE.COM. Il y a affichage de :

```
IMPR        .BAS   -->   VITRE       .COM
                (devient)
```

Les jokers sont utilisables à condition qu'ils soient présents à la même place dans les 2 noms de fichiers.

REN "*.COM" TO "BELFORT.VIL" n'est pas valable.

REN "*.COM " TO "*.VIL" est valable.

DNAME

DNAME "NOM" modifie le nom donné à une disquette lors d'un INIT. Le nouveau nom contient 20 caractères au maximum et est complété par des espaces à droite s'il y en moins. Les codes couleurs sont acceptés.

`DNAME "DISK 5 du 24/06/87"` change le nom de la disquette contenue dans le lecteur courant.

MERGE

MERGE "nom de fichier" mélange le programme BASIC en mémoire avec le programme "nom de fichier" présent sur la disquette. Les lignes du programme BASIC de la disquette sont ajoutées au programme BASIC déjà en mémoire.

Si deux mêmes numéros de lignes sont rencontrés, la priorité sera donnée à la ligne venant du programme 'mergé'. Le TELESTRAT affiche un par un le numéro des lignes rajoutées au programme en mémoire. L'instruction MERGE remet obligatoirement toutes les variables à 0.

Exemple : `MERGE "PR2"` va mélanger les lignes du programme PRG2 au programme déjà en mémoire.

Attention: MERGE ne peut-être utilisé en mode direct.

CHAPITRE 14

LES COMMANDES ET FONCTIONS DE GESTION DE FICHIERS

Le chapitre 13 se consacrait à la gestion de fichiers contenant des programmes, que ceux-ci soient en BASIC, en langage machine, des écrans TEXT ou HIRES, ... Mais il existe aussi un autre type de fichiers : les fichiers de données. Au lieu de stocker des programmes dans des fichiers, nous allons y mettre des valeurs de variables numériques ou alphanumériques.

L'accès à un fichier peut se faire de deux manières : séquentielle ou directe. On en vient donc à appeler ces deux accès possibles par : accès séquentiel ou accès direct.

A/ L'ACCES SEQUENTIEL

C'est la manière la plus simple de conserver des données. Mais c'est aussi la moins performante. En accès séquentiel, les données sont écrites les unes après les autres dans un fichier. On y accède ainsi dans l'ordre où elles ont été écrites. Un pointeur se déplace constamment tout au long du fichier en avançant d'une donnée à l'autre. C'est à la position de ce pointeur que pourront être lues ou écrites des variables.

La sous-performance de la méthode d'accès séquentiel à des données vient du fait qu'à chaque fois qu'une donnée doit être lue ou écrite, le pointeur se place au début du fichier. Et il avance de donnée en donnée, jusqu'à ce qu'il ait trouvé la donnée à lire/écrire. Ce déplacement prend du temps et est proportionnel au nombre de données que contient un fichier et à la place qu'occupe une donnée dans ce même fichier.

FILE

La première chose à faire avant d'utiliser l'accès séquentiel est de déclarer le nombre de fichiers qui vont être simultanément ouvert. FILE NB ouvre NB fichiers en même temps. 16 fichiers au maximum peuvent être ouverts. NB varie de 1 à 16.

Attention : l'instruction FILE effectue un CLEAR et efface donc toutes les variables. Il vaut donc mieux la placer en début de programme.

OPEN

Le nombre de fichiers simultanément ouverts étant déclaré, il convient maintenant d'ouvrir un fichier séquentiel.

OPEN 0,NL,NF ouvre un fichier à accès séquentiel portant un numéro logique NL et ayant comme nom NF. Ce fichier est créé sur la disquette et porte l'extension .DAT qui est l'extension utilisée pour les fichiers de données.

OPEN 0,1,"ESSAI" ouvre un fichier séquentiel de numéro logique 1 et de nom ESSAI. Le fichier ESSAI.DAT est créé sur la disquette s'il n'existe pas. Chaque fichier à accès séquentiel qui vient d'être ouvert occupe 5 secteurs au départ. Le nombre de secteurs croîtra au fur et à mesure que le contenu du fichier augmentera. (OPEN ne marche que si l'on a auparavant donné le nombre de fichiers ouvrables avec FILE).

OPEN réserve un tampon en mémoire dans lequel sont stockés provisoirement les données des fichiers. Une fois que ce tampon est plein, il est sauvé sur disquette.

CLOSE

CLOSE NL est l'inverse de OPEN et ferme un fichier de numéro logique NL (entre 0 et 16). Quand on a fini l'exploitation d'un fichier, il est indispensable de le fermer avec CLOSE, sous peine de ne pas voir ses dernières données stockées sur disquettes. (comme dit précédemment plus haut, un tampon (endroit mémoire) est réservé lors de l'ouverture d'un fichier. C'est dans ce tampon que seront stockées les données. Quand le tampon est plein, il est sauvé sur disquette.)
CLOSE oblige le TELESTRAT à sauver le tampon sur disquette. On est donc sûr que les dernières données seront sauvées sur disquette.

CLOSE 0 ferme tous les fichiers ouverts sans exception.

SPUT

SPUT NL,EA ou EN écrit dans un fichier de numéro logique NL la valeur d'une expression alphanumérique (EA) ou numérique (EN).

```
 10   FILE 1 ' ON DECLARE UN FICHIER A OUVRIR
 20   OPEN 0,1,"TOTO" ' OUVRE UN FICHIER PORTANT LE NUMERO LOGIQUE
1 ET DE NOM TOTO
 30   CHAINE$="chaine ALPHANUMERIQUE"
 40   NOMBRE=10
 50   SPUT 1,CHAINE$ ' ECRIT DANS LE FICHIER LA VARIABLE
ALPHANUMERIQUE CHAINE$
 60   SPUT 1,NOMBRE ' ET LA VARIABLE NUMERIQUE NOMBRE
 70   CLOSE 1 ' FERME LE FICHIER 1, CE QUI SAUVE LES DONNEES
```

Ce programme écrit dans un fichier séquentiel 2 données. Quand le fichier est créé sur disquette (car il n'existait pas), le pointeur est placé au début du fichier.

SPUT 1,CHAINE$ écrit dans le fichier la valeur de CHAINE$ et SPUT 1,NOMBRE écrit la valeur de NOMBRE.

Une chaîne peut avoir jusqu'à 255 caractères, et un nombre une valeur quelconque.

Une règle est à respecter : si l'écriture d'une donnée se fait en fin de fichier, la donnée à écrire peut être soit numérique, soit alphanumérique. Mais si l'écriture ne se fait pas en fin de fichier, il faut que la donnée à écrire soit du même type que celle déjà présente. (On ne peut pas remplacer en plein milieu d'un fichier une chaîne par un nombre ou vice-versa.)

Si une valeur alphanumérique est écrite en milieu de fichier, c'est à dire sur une autre valeur alphanumérique, la nouvelle chaîne sera tronquée ou complétée par des espaces, selon que celle-ci est plus longue ou plus courte que la précédente.

STAKE

STAKE NL,VN ou VA lit une valeur dans un fichier de numéro logique NL et l'affecte à une variable numérique ou alphanumérique selon la valeur lue. Si l'on se trouve en fin de fichier, l'ordinateur renvoit le message '`fin de fichier`'. (Il n'y a plus de données à lire).

Avec l'instruction SPUT de tout à l'heure, nous avons créé un fichier dans lequel nous avons écrit deux données. STAKE va nous permettre de les relire :

```
10   FILE 1 ' DECLARE 1 FICHIER
20   OPEN 0,1,"TOTO" ' REOUVRE LE FICHIER TOTO (NUMERO LOGIQUE=1)
30   STAKE 1,CHAINE$ ' LIT LA PREMIERE DONNEE DU FICHIER
40   STAKE 1,NOMBRE ' AINSI QUE LA DEUXIEME
50   PRINT CHAINE$ ' ET ON AFFICHE
60   PRINT NOMBRE ' LE TOUT ...
70   CLOSE 0 ' FERME TOUS LES FICHIERS.
```

STAKE 1, CHAINE$ lit la chaîne alphanumérique placée en début de fichier ("chaîne ALPHANUMERIQUE") et l'affecte à la variable CHAINE$. Et STAKE 1,NOMBRE prend le nombre situé en deuxième place dans le fichier et l'affecte à la variable A$.

Un STAKE 1,CHAINE à la place de STAKE 1,CHAINE$ n'aurait pas marché car le premier élément du fichier est une chaîne alphanumérique. (message "**mauvais type d'expressions**")

APPEND

APPEND NL place le pointeur de fichier à la fin du fichier de numéro logique NL. APPEND 1 et le pointeur est à la fin du fichier de numéro logique 1.

REWIND

REWIND NL place le pointeur de fichier au début du fichier de numéro logique NL.

Attention : l'instruction REWIND ne marche pas après l'écriture de données : elle détruit les dernières valeurs entrées dans le buffer reservé en mémoire pour un fichier à accès séquentiel. En conséquence, le retour en début de fichier après l'écriture de données ne sera possible que par : CLOSE NL (ferme le fichier et sauve les données du buffer mémoire en même temps) et OPEN 0,NL,"nom de fichier" (ouvre à nouveau le fichier et replace le pointeur en début de fichier par la même occasion.)

```
 10   FILE 1
 20   OPEN 0,1,"XXX"
 30   FOR I=1 TO 50
 40    SPUT 1,RAND(I) ' MET DANS LE FICHIER XXX 50 NOMBRES TIRES AU
HASARD
 50   NEXT
 60   CLOSE 1:OPEN 0,1,"XXX" ' REPLACE LE POINTEUR  EN DEBUT DE
FICHIER AVEC CLOSE/OPEN CAR ON VIENT D'ECRIRE DES DONNEES
 70 ' LIRE DES DONNEES
 80   FOR I=1 TO 50
 90    STAKE 1,W
100    PRINT W
110   NEXT
120   CLOSE 1
```

APPEND et REWIND sont très utiles en séquentiel. Par exemple, lorsque l'on veut lire une donnée, REWIND replace le pointeur au début du fichier. Et pour rajouter une donnée à la fin d'un fichier, APPEND replace le pointeur à la fin de ce fichier.

JUMP

JUMP NL,NE déplace le pointeur du nombre d'enregistrements NE précisé dans le fichier de numéro logique NL. Si ce nombre est trop grand, le pointeur est placé en fin de fichier.

Pour se placer au début d'une donnée de position N dans un fichier de numéro logique NL :

```
REWIND NL:IF N<>1 THEN JUMP NL,N-1
```

Explications : REWIND NL place le pointeur au début du fichier de numéro logique NL (à ne pas utiliser après l'écriture de données = voir instruction REWIND). IF N<>1 THEN JUMP NL,N-1 place le pointeur au début de la donnée numéro N du fichier. Si N=1, ce sera la première donnée du fichier : REWIND suffira.

FST

FST (NL) renvoit la valeur 0 si le pointeur est en fin de fichier, renvoit 1 s'il est sur une variable numérique, et renvoit 2 s'il est sur une variable alphanumérique.

Programme affichant le contenu d'un fichier séquentiel quelconque :

```
 10   FILE 1
 20   OPEN 0,1,"XXX"
 30   WHILE FST (1) ' LIRE LE FICHIER TANT QUE CE N'EST PAS LA FIN
DU FICHIER
 40    IF FST(1)=1 THEN STAKE 1,A:PRINT A
 50    IF FST(1)=2 THEN STAKE 1,A$:PRINT A$
 50   WEND
 60   CLOSE 1
```

Afin de bien vous faire comprendre :

- L'utilité des fichiers de données.

- Le rôle des instructions FILE,OPEN,CLOSE,SPUT,STAKE,APPEND,REWIND, nous vous proposons ci-dessous un programme qui regroupe toutes les instructions de l'accès séquentiel. Il vous permet d'ouvrir un fichier quelconque et d'y lire ou écrire diverses données.

```
 10   CLS:PRINT :FILE 16 ' OUVRIR 16 FICHIERS PAR DEFAUT
 20   PRINT CHR$(129);"       OUVERTURE DE FICHIERS":PRINT
 30   PRINT "NUMERO LOGIQUE DU FICHIER A OUVRIR":PRINT "(ENTRE 1 ET
16) ? ";
 40   INPUT NL
 50   IF NF<1 OR NL>16 THEN 10
 60   PRINT
 70   PRINT CHR$(130);"NOM DU FICHIER : ";
 80   INPUT NOM$
 90   IF LEN(NOM$)>9 THEN PRINT "NOM TROP LONG ...":GOTO 70
100   OPEN 0,NL,NOM$
110   CLS:PRINT :PRINT "VOULEZ VOUS :":PRINT
120   PRINT "1 - ECRIRE DANS LE FICHIER":PRINT
130   PRINT "2 - LIRE LE FICHIER":PRINT
140   PRINT "3 - ARRETER"
150   GET A$
160   IF A$="1" THEN GOTO ECRIRE
170   IF A$="2" THEN GOTO LIRE
```

```
180   IF A$="3" THEN ARRETE
190   PING:GOTO 110
200 ]ECRIRE
210   APPEND 1 ' PLACE LE POINTEUR EN FIN DE FICHIER POUR ECRIRE
220   PRINT :PRINT "VOULEZ VOUS ECRIRE :":PRINT
230   PRINT "1 - UNE DONNEE NUMERIQUE":PRINT
240   PRINT "2 - UNE DONNEE ALPHANUMERIQUE"
250   GET A$
260   IF A$="1" THEN NUMER
270   IF A$="2" THEN ALPHA
280   GOTO 220
290 ]NUMER
300   PRINT "DONNEE NUMERIQUE : ";
310   INPUT NOMBRE
320   SPUT NL,NOMBRE:GOTO 110
330 ]ALPHA
340   PRINT "DONNEE ALPHANUMERIQUE : ";
350   INPUT CHAINE$
360   SPUT NL,CHAINE$:GOTO 110
370 ]LIRE
380   REWIND 1 ' PLACE LE POINTEUR AU DEBUT DU FICHIER POUR
390   ' POUVOIR LIRE LES DONNEES
400   CLS:PRINT "VOULEZ VOUS :":PRINT
410   PRINT "1 - LIRE LES DONNEES NUMERIQUES":PRINT
420   PRINT "2 - LIRE LES DONNEES ALPHANUMERIQUES"
430   PRINT "3 - LIRE TOUTES LES DONNEES"
440   GET A$
450   IF A$="1" THEN GOTO CHOIX1
460   IF A$="2" THEN GOTO CHOIX2
470   IF A$="3" THEN GOTO CHOIX3
480   GOTO 400
490 ]CHOIX1
500   IF FST(NL)=0 THEN 110 ' SI FIN DE FICHIER ON ARRETE
510   IF FST(NL)=2 THEN JUMP NL,1:GOTO CHOIX1 ' SI DONNEE ALPHA
ALORS ON LA SAUTE
520   STAKE NL,DON:PRINT DON:GOTO CHOIX1 '
530 ]CHOIX2
540   IF FST(NL)=0 THEN 110 ' SI FIN FICHIER ON ARRETE
550   IF FST(NL)=1 THEN JUMP NL,1:GOTO CHOIX2 ' SI VARIABLE NUMER
ALORS ON SAUTE
560   STAKE 1,CHAINE$:PRINT CHAINE$
570   GOTO CHOIX2
580 ]CHOIX 3
590   IF FST(NL)=0 THEN 110 ' SI FIN FICHIER ON ARRETE
600   IF FST(NL)=1 THEN STAKE NL,A:PRINT A:GOTO CHOIX3
610   STAKE NL,A$:PRINT A$:GOTO CHOIX3
610 ]ARRETE
620   CLOSE 0 ' FERMER TOUS LES FICHIERS OUVERTS
630   PRINT "AU REVOIR ...":END
```

Sur la disquette fournie avec le TELESTRAT, vous trouverez les programmes :

BUILD.COM Permet de créer des fichiers séquentiels à partir du clavier. Entrez le nom du fichier à créer, puis entrez votre texte. Tapez ESC pour terminer.

TYPEBUILD.COM Ce programme visualise ce qui a été créé par BUILD.

TYPE.COM Permet de visualiser le contenu d'un fichier séquentiel quelconque. Entrez le nom du fichier. L'affichement des données s'interrompt par ESPACE.

Dans un fichier à accès séquentiel, une expression alphanumérique occupe 2 octets + la longueur de la chaîne, et une expression numérique occupe toujours 7 octets.

Globalement, le temps de lecture d'une donnée stockée dans un fichier séquentiel se calcule par : N/1300, N étant le nombre d'octets qu'occupe la donnée (avec un temps d'accès minimum de 0.2 seconde si le résultat de N/1300 est inférieur à 0.2 seconde).

B/ L'ACCES DIRECT

L'accès direct est plus performant que l'accès séquentiel, car comme son nom l'indique, l'accès à une donnée se fait DIRECTement. Le temps d'accès à une fiche est de 0,2 seconde.

En accès direct, les données sont regroupées en fiches. Chaque fiche contient le même type de renseignements. Pour cela, on divise une fiche en CHAMPS (25 champs au maximum).

Avant d'ouvrir une fiche, il faudra définir les champs de chaque fichier, c'est à dire qu'il faudra définir quelles variables seront stockées dans une fiche, et quel nombre de caractères sera réservé dans une fiche pour une variable.

Commençons par voir les instructions agissant en accès direct :

FILE

En accès direct comme en accès séquentiel, FILE N définit le nombre de fichiers simultanément ouverts, (N varie entre 1 et 16)

OPEN

Pour ouvrir un fichier en accès direct, c'est OPEN 1,NL,NOM et non pas OPEN 0,NL, NOM que l'on utilise.

Rappelez vous :

OPEN **0**,..... pour un fichier à accès séquentiel.

OPEN **1**,..... pour un fichier à accès direct.

NL est comme d'habitude le numéro logique et NOM$ est le nom du fichier. (le fichier en accès direct a aussi une extension .DAT).

En accès direct, OPEN ne crée pas un fichier sur disquette, mais il l'ouvre. On utilise un autre utilitaire (que nous verrons plus loin) pour créer un fichier. En conséquence, un fichier direct créé sur disquette par OPEN est inutilisable.

Exemple : `OPEN 1,1,"DIRECT"`

CLOSE

CLOSE NL ferme un fichier de numéro logique NL. CLOSE 0 ferme tous les fichiers ouverts.

En accès direct, il n'est pas nécéssaire de faire un CLOSE pour sauver les derniers enregistrements d'un programme, car les données sont regroupées en fiches, et chaque fiche est sauvée à chaque fois sur la disquette. Un CLOSE est indispensable toutefois pour réactiver le n° logique.

PUT

PUT NL,NF écrit dans le fichier de numéro logique NL la fiche no NF. Si une variable correspond à un nom de champ sa valeur est mise dans la fiche. Sinon, le nom de champ est vidé (0 ou chaîne vide selon le type de champ). Le nombre de fiches peut aller jusqu'à 65535. Chaque fiche peut contenir jusqu'à 512 caractères. Le nombre de fiches se crée automatiquement.

Par exemple : **PUT 1,100** écrit la fiche numéro 100 dans le fichier de numéro logique 1. Si les fiches de 1 à 100 n'existent pas, elles sont automatiquement créées, mais elles contiennent n'importe quoi (attention).

Il est d'ailleurs conseillé de rajouter des fiches par blocs, car écrire 1 ou 100 fiches prend le même temps (2 secondes environ).

TAKE

TAKE NL,NF lit dans le fichier portant le numéro logique NL la fiche numéro NF. Si une variable correspond à un nom de champ, la variable est chargée avec la valeur correspondante. Si le numéro de fiche est plus grand que le nombre de fiches créées, le message "`valeur illegale`" apparaît.

FST

Seul FST(NL) s'utilise en accès direct et renvoit le nombre d'enregistrements (de fiches) contenus dans un fichier.

Pour ajouter une fiche à la fin d'un fichier à accès direct, on utilise l'instruction FST(1)+1, car FST(1) donne le nombre de fiches contenues dans un fichier à accès direct.

```
PUT  1,FST(1)+1
```

CREATE

Si vous n'avez pas compris grand chose à l'accès direct jusqu' à présent, la création de champs devrait vous éclaircir les idées. Le programme qui définit les champs d'une fiche est présent sur la disquette fournie avec le TELESTRAT. Son nom est CREATE25. Tapez CREATE25 pour le charger.(Si le message ' `fichier absent`' apparaît, alors tapez CREATE seulement).

Une fois CREATE25 chargé, un masque apparaît à l'écran. Insérez une disquette vierge dans le lecteur. Le curseur se trouve en haut à gauche de l'écran. On vous demande le nom du fichier à créer. Entrez AMIS par exemple et appuyez sur RETURN. Le fichier n'existant pas, il est automatiquement sauvé sur disquette. Si le fichier avait existé, le TELESTRAT aurait chargé et affiché tous les noms de champs du fichier.

Le curseur est maintenant au début d'un tableau de 25 lignes de haut. Chaque ligne est faite pour définir un champ. On entre tout d'abord le nom de ce champ (8 caractères maximum). Tapez par exemple NOM$ (champ alphanumérique) et appuyez sur RETURN. Le curseur se déplace et vient dans la 3ème colonne. Dans la deuxième colonne, CH s'inscrit. Cela veut dire que le champ défini est alphanumérique (CH pour CHaîne). La 3ème colonne est celle où l'on doit donner le nombre de caractères maximum que pourra contenir la chaîne NOM$ (ou plutôt le nombre d'octets).

Entrez 20 et tapez RETURN. Le curseur va à la ligne. Tapez AGE, puis RETURN. La 2ème colonne se remplit automatiquement avec NU (pour donnée NUmérique). Et la 3ème colonne prend la valeur 5, car un nombre est toujours stocké sur 5 octets, quelle que soit sa valeur .... Nous avons maintenant défini le champ de chaque fiche : dans chaque fiche, il y aura un champ de 20 octets (20 caractères) reservé à la chaîne NOM$, et un champ de 5 octets réservé à la variable AGE. A droite de l'écran s'affiche constamment le nombre d'octets encore libres et le nombre d'octets déjà pris. Une fiche ne peut pas contenir plus de 512 octets. Une chaîne est elle même limitée à 255 caractères. Tout dépassement de cette valeur pendant l'affectation d'un nombre d'octets à une variable alphanumérique sera repoussé (l'ordinateur ne vous fait pas passer à la ligne. Il attend que vous corrigiez votre nombre avec la touche DEL.)

Nos champs sont prêts. Sauvez les sur disquette comme l'indique l'écran en tapant CTRL S. Appuyez sur ESC pour quitter le programme. Si nous vous avons fait entrer les champs NOM$ et AGE, c'est pour une raison bien précise : nous allons ouvrir un fichier qui conserve les noms et les âges de vos amis. Le fichier à accès direct **AMIS.DAT** doit bien sûr être sur disquette, et les champs doivent être créés comme expliqué ci-dessus.
Pour rajouter une fiche à la fin d'un fichier à accès direct, on utilise l'instruction FST(1)+1, car FST(1) donne le nombre de fiches contenues dans un fichier à accès direct .

```
 10   FILE 1 ' DECLARE 1 FICHIER
 20   OPEN 1,1,"AMIS" 'OUVRE LE FICHIER AMIS.DAT
 30   PRINT "COMBIEN AVEZ VOUS D'AMIS ? ";
 40   INPUT A
 50   FOR I=1 TO A
 60    PRINT "NOM DE L'AMI NO";I; " : ";
 70    INPUT NOM$
 90    PRINT "AGE DE ";NOM$;" : ";
100    INPUT AGE
110    PUT 1,I ' SAUVE NOM ET AGE DE L'AMI No X.
120   NEXT
130   PRINT " ET MAINTENANT, VOICI LE NOM ET L'AGE DE VOS AMIS
:":WAIT 30:PRINT
140   FOR I=1 TO A
150    TAKE 1,I
160    PRINT NOM$;"  AGE :";AGE
170   NEXT
180   PRINT "C'EST FINI ..."
190   CLOSE 1
```

Comprenez bien l'action de ce programme : les champs du fichier AMIS ont tout d'abord été définis par CREATE25 (NOM$ et AGE). Ensuite, ce programme demande un par un le nom et l'âge de vos amis. Le nom est mis dans la variable NOM$, et l'âge dans la variable AGE. PUT 1,I sauve chaque nom et chaque âge de chaque ami dans une fiche différente. (Lors du PUT, le TELESTRAT sauve dans une fiche les variables NOM$ et AGE qui ont été définies avec CREATE25.)

Un exemple tout simple :

```
 10   FILE 1 ' DECLARE UN FICHIER
 20   OPEN 1,1,"AMIS"
 30   NOM$="MR CLAVIER ORIC"
 40   AGE=30
 50   PUT 1,1'SAUVE NOM$ ET AGE DANS LE FICHIER DE NUMERO LOGIQUE 1
 60 ' ET DANS LA FICHE 1
 70   CLEAR ' EFFACE LES VARIABLES
 80   TAKE 1,1 ' LIT LA FICHE 1
 90   PRINT NOM$;"  AGE :";AGE;" ANS."
100   CLOSE 1
```

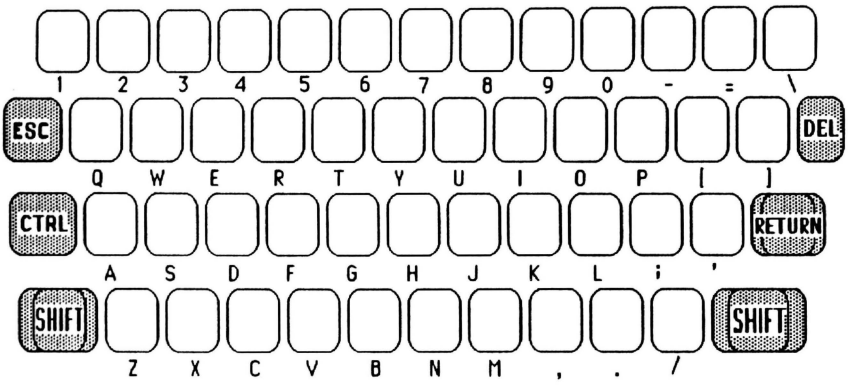
CHAPITRE 15

LES MODES DE CLAVIER DU TELESTRAT

Le TELESTRAT met à votre disposition plusieurs types de claviers différents. Le clavier courant est en QWERTY, au standard anglo-saxon.

QWERTY

Comme vous le devinez, QWERTY fait passer l'ordinateur en mode QWERTY. Si l'on est par exemple en mode AZERTY, il suffit d'entrer QWERTY pour retrouver le clavier d'origine. L'instruction QWERTY redéfinira tous les caractères du clavier.



AZERTY

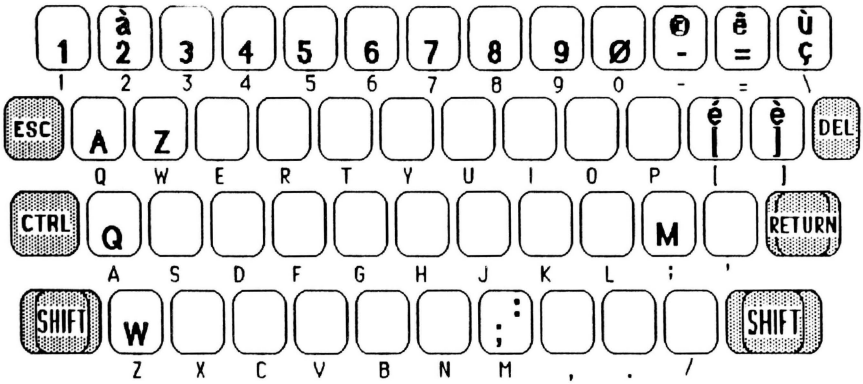
En tapant AZERTY, le clavier se transforme en un clavier AZERTY au standard français. Le TELESTRAT est livré avec une planche de petits autocollants qui vous permet de changer la configuration courante du clavier. Pour obtenir un clavier AZERTY, on tape AZERTY. Puis on colle les autocollants appropriés. (Voir plaquette).

Les caractères accentués français sont inclus au clavier AZERTY : é è ê à ç. Les lettres sont en majuscules, et les chiffres s'obtiennent sans SHIFT, comme pour le clavier QWERTY.

Lors de la conversion du clavier QWERTY en AZERTY,

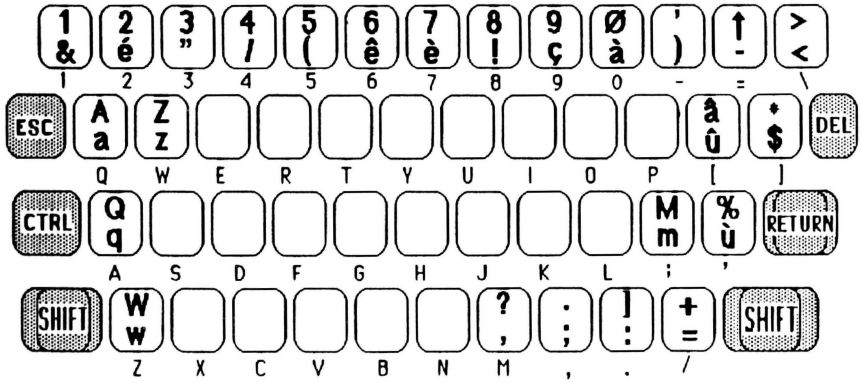
Q	devient	A
W	"	Z
A	"	Q
Z	"	W
M	"	;
;	"	M

Et un ACCENT SET (voir cette commande plus loin) est automatiquement effectué.



FRENCH

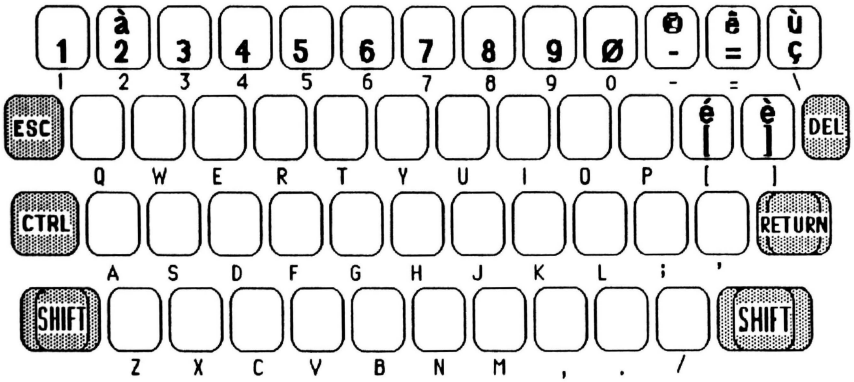
FRENCH convertit le clavier en un clavier au standard des machines à écrire françaises. Le clavier est un AZERTY, mais les chiffres s'obtiennent avec le SHIFT. Les signes sont disposés au clavier comme sur une machine à écrire française. Pour des raisons techniques, le tréma et l'accent circonflexe ne sont pas disponibles seuls.



ACCENT SET, ACCENT OFF

ACCENT SET permet d'obtenir les accents français en mode QWERTY. Voici comment sont redéfinis certains caractères du mode QWERTY :

CARACTERE NORMAL	CARACTERE ACCENTUE
@	à
\	ç
{	é
|	ù
}	è
FUNCT =	ê



L'obtention de l'accent circonflexe est un peu spéciale et se fait en appuyant sur les touches FUNCT et =.

ACCENT OFF est l'inverse de ACCENT SET et rétablit le jeu de caractères standards.

Exemple :

```
10   QWERTY:CLS:PRINT :PRINT
20   ACCENT SET:PRINT "AVEC ACCENT SET : \|@{}"
30   WAIT 30:PRINT
40   ACCENT OFF:PRINT "AVEC ACCENT OFF : \|@{}"
```

CHAPITRE 16

LES COMMANDES ET FONCTIONS DE GESTION DE L'HORLOGE

Vous voulez connaître l'heure ? Rien de plus facile : demandez-la au TELESTRAT ...

CLOCKSET

Dès que vous allumez votre ordinateur, une horloge se met en route. Tous les compteurs de l'horloge (heures, minutes, secondes) sont mis à 0. Bien que l'heure ne soit pas visible à l'écran, elle fonctionne toute seule.

CLOCKSET ADR est l'instruction qui va demander au TELESTRAT d'afficher l'horloge sur l'écran TEXT.
ADR est l'adresse où sera affichée l'horloge. Avant tout, il faut savoir que :

- #BB80 est l'adresse mémoire du début de l'écran TEXT.

- en mode TEXT, l'écran est divisé en 27 lignes de 40 colonnes. Pour définir l'adresse où sera affiché l'horloge, on effectue le calcul #BB80+40*Y+ X, X (entre 0 et 39) et Y (entre 0 et 26) étant les coordonnées où sera inscrite l'horloge.

```
10   CLS
20   CLOCKSET #BB80+40*15+20
```

Affiche l'horloge toutes les secondes aux coordonnées X=20 et Y=15. L'heure apparaît sous la forme HH:MM:SS ou HEURES:MINUTES:SECONDES.

TIME

TIME H,M met à l'heure l'horloge. H est l'heure courante, et M les minutes. Les secondes et les dixièmes de seconde sont remis à 0 à l'exécution de la commande TIME.

```
10   CLS
20   TIME 11,35 ' IL EST 11 H 35 MN
30   CLOCKSET #BB80+40*10+10 ' AFFICHE L'HEURE EN X=10 ET Y=10
```

Le seul endroit où l'affichage de l'horloge n'est pas perturbé par le défilement de l'écran est la ligne 0, appelée ligne de status (car utilisée pour l'affichage de messages).

```
10   TIME 0,0
20   CLOCKSET #BB80+40*0+30
```

Affiche l'horloge en X=30 et Y=0, c'est à dire sur la ligne de status.

TIME$

TIME$ est une variable alphanumérique et renvoie la valeur courante de l'horloge sous la forme : HH:MM:SS:D.

HH : HEURES
MM : MINUTES
SS : SECONDES
D : DIXIEMES

Exemple :

```
 10   TIME 0,0 ' MET L'HORLOGE A 0
 20   WAIT 32 ' ATTEND 32/10 EMES DE SECONDES, SOIT 3 SECONDES ET 2
DIXIEMES
 30   PRINT TIME$
```

Résultat : 00:00:03:2 = 00 heure, 00 minute, 03 secondes et 2 dixièmes de seconde.

Pouvoir connaître à tout moment la valeur de l'horloge à l'aide de TIME$ apporte de nombreux avantages : vous n'avez plus de réveil : demandez à votre TELESTRAT de le remplacer.

```
  10   CLS
  20   TIME 20,30
  30   CLOCKSET #BB80+30
  40   PRINT :PRINT
  50   PRINT MIDDLE$("DORMEZ BIEN ...",40)
  60   IF TIME$<>"09:00:00:0" THEN 60
  70   PRINT :PRINT  "BONJOUR ..."
  80   PRINT :PRINT "IL EST L'HEURE !"
  90   COUNT 20
 100    PING:WAIT 10
 110   UNCOUNT
```

Ce programme imagine qu'il est 20 H 30 lorsque vous mettez l'horloge. Votre intention est de vous réveiller le lendemain matin à 9 H. IF TIME$="09:00:00:0" teste s'il est 9 heures, 00 minute, 00 seconde et 0 dixième de seconde. Si c'est le cas, une batterie de 20 PING imite la sonnerie du réveil. Sinon, on attend qu'il soit 9 heures ...

Ce 'réveil par ordinateur' ne marche qu'à deux conditions :

- Que le TELESTRAT et le moniteur (pour le son) restent branchés.

- Qu'il n'y ait pas de coupure de courant.

```
  10   CLS
  20   TIME 0,0
  30   IF TIME$<>"00:00:05:0" THEN 30 ELSE PRINT "IL VIENT DE
S'ECOULER 5 SECONDES !"
  40   IF TIME$<>"00:00:10:6" THEN 40
  50   PRINT "IL VIENT DE S'ECOULER 10 SECONDES ET 6 DIXIEMES !"

 10   CLS
 20   TIME 0,0
 30   COUNT 1000
 40    PRINT TIME$
 50   UNCOUNT
```

CLOCKOFF

CLOCKOFF stoppe l'affichage de l'horloge à l'écran. L'heure n'est ni effacée, ni stoppée, mais son apparition à l'écran toutes les secondes est interrompue (L'heure est figée.)

```
10   CLS
20   CLOCKSET #BB80+40*10+10
30   TIME 10,30
40   IF TIME$<>"10:30:05:04" THEN 40
50   CLOCKOFF
60   WAIT 30
70   PRINT  @10,10;"          "
```

Ici, CLOCKOFF arrête l'horloge au bout de 5 secondes et 4 dixièmes.

La ligne 70 efface l'horloge de l'écran en mettant 8 espaces vides aux coordonnées X=10 et Y=10. (Là où se trouve l'horloge).

Remarques générales concernant la gestion de l'horloge :

- L'appui sur le bouton RESET interrompt l'affichage de l'horloge.
- L'horloge ne s'utilise pas en mode HIRES (sauf sur les 3 lignes TEXT du bas). Par conséquent, entrez l'instruction CLOCKOFF avant de passer en HIRES, sous peine de voir des traces indésirables à l'écran.
- L'horloge est mise à 0 lors de l'allumage du TELESTRAT. Elle ne fonctionne d'ailleurs que lorsque le TELESTRAT est sous tension.
- Lors d'un accès au lecteur de disquette, l'horloge est momentanément interrompue, le temps qu'un fichier soit chargé.

L'HORLOGE TEMPS REEL

ORIC, le constructeur du Télestrat met à votre disposition une extension horloge temps réel qui se branche sur le bus d'extension de votre TELESTRAT. Elle garde l'heure dans une pile intégrée qui dure plusieurs années. Cette horloge gère automatiquement l'année, le mois, le jour, l'heure, les minutes, les secondes, les dixièmes de seconde. Enfin, tout, ... Il est même permis de choisir le mode 12 ou 24 heures. De plus, une fonction alarme est intégrée à l'horloge.

Pour une prise de rendez-vous par MINITEL, pour avoir toujours l'heure exacte sur son serveur MINITEL, ou pour bien d'autres applications, il suffit de posséder cette horloge.

Nous vous présentons ci-dessous un rappel du fonctionnement de l'horloge, ainsi qu'un petit programme d'application.

Initialisation de l'horloge :

Elle se fait par : POKE #371,12 pour le mode 24 heures.
POKE #371,8 pour le mode 12 heures.

Votre horloge est maintenant en marche, et ne s'arrêtera plus, même lors de l'arrêt du TELESTRAT.

Réglons l'heure, les minutes, et les secondes. (Ne nous occupons pas de l'année, du mois, du jour...)

L'adresse #361 contient l'heure (entre 0 et 23)
L'adresse #362 contient les minutes (entre 0 et 59)
L'adresse #362 contient les secondes (entre 0 et 59)
Pour régler l'heure, on utilise POKE ADRESSE, VALEUR
Pour lire l'heure, on utilise PRINT PEEK(A DRESSE)

Exemple : Il est 14 H 38 MN 10 S (en mode 24 heures).

```
POKE #361,14 pour 14 heures
POKE #362,38 pour 38 minutes
POKE #363,10 pour 38 secondes
```

L'horloge est maintenant à l'heure.

Lecture de l'heure :

Avant de lire l'heure, il faut toujours entrer : CC=PEEK(#360). #360 est l'adresse des dixièmes de secondes. Et lire cette adresse correspond à fixer l'heure.

```
CC=PEEK(#360)

PRINT   PEEK(#361)":"PEEK(#362)":"PEEK(#363)
```

Et l'heure exacte est affichée.

Voici un petit programme d'application de l'horloge temps réel : il affiche automatiquement l'heure à l'initialisation du TELESTRAT, et met également le TELESTRAT à l'heure (instruction TIME). Les secondes sont aussi mises à jour, en agissant directement en #211 qui est l'adresse de décomptage des secondes. Cela vous évite de perdre de 0 à 59 secondes de différence entre l'horloge du TELESTRAT (qui est remise à 0 lors de l'arrêt du TELESTRAT) et l'horloge temps réel, car TIME n'accepte pas les secondes, et les met automatiquement à 0.

```
10   PRINT:CC=PEEK(#360) ' Fixe l'heure
20   PRINT PEEK(#361)" :"PEEK(#362)":"PEEK(#363) ' Affiche l'heure
30   TIME PEEK(#361),PEEK(#362) ' Met l'heure, les minutes
40   POKE #211,PEEK(#363) ' et les secondes
50   NEW ' Destruction du programme et retour au BASIC
```

Sauvez ce programme par : `SAVE"BONJOUR.COM",AUTO`

A chaque allumage du TELESTRAT, l'heure sera affichée. Le programme s'auto-détruira ensuite.

CHAPITRE 17

LES COMMANDES ET FONCTIONS DE GESTION DES ERREURS

L'erreur est humaine. Se tromper en entrant une réponse, une ligne, un mot... arrive couramment. Un certain nombre d'instructions sont prévues pour éviter les erreurs trop fréquentes.

TRACE SET, TRACE OFF

TRACE SET est l'instruction qui va vous permettre de suivre un programme "à la TRACE" pendant son exécution. En effet, les numéros de lignes de programmes s'affichent sur les 2 dernières lignes du bas de l'écran TEXT au fur et à mesure qu'elles sont exécutées.

TRACE OFF arrête le mode TRACE SET, c'est à dire qu'il n'y a plus d'affichage du numéro d'une ligne exécutée après un TRACE OFF.

```
10   CLS:TRACE SET ' MET LE MODE 'TRACE'
20   FOR I=1 TO 100
30    PRINT I:WAIT 10:CLS
40    IF I=10 THEN I=1
50   NEXT
```

Dans cette exemple, notre intention est d'obtenir l'affichage des nombres de 1 à 100, grâce à une boucle FOR-NEXT, et de sortir de la boucle une fois arrivé à 100. Malheureusement, quelque chose ne marche pas, car la boucle ne se termine jamais.

Où est l'erreur ? La commande TRACE vous montre que les lignes 20, 30, 40 et 50 sont constamment exécutées. Il y a donc un problème dans une de ces 4 lignes. A vous de le découvrir. (Pour ceux qui n'auraient pas trouvé, il s'agit de la ligne 40).

La fenêtre de TRACE porte le n°3, on peut la modifier, par exemple: `TRACE SET:WINDOW 3,20,39,5,20` (voir chapitre 19).

HELP, AIDE

Les homologues HELP et AIDE font exactement la même chose, et affichent :

Essayez `WORD,FUNCTION,ERRLIST !`

WORD

WORD provoque l'affichage de la liste de toutes les commandes du TELESTRAT. La barre d'espace interrompt momentanément l'affichage, et la touche ESC définitivement. WORD est très pratique lorsque l'on débute en HYPERBASIC et que l'on ne se souvient plus si HIRES prend un H ou pas.

```
10   CLS
20   COUNT 5
30    WORD
40   UNCOUNT
```

FUNCTION

FUNCTION donne la liste à l'écran de toutes les fonctions, de toutes les constantes, et de tous les opérateurs logiques existant en HYPERBASIC.

Exemple : `FUNCTION`

ERRLIST

ERRLIST est un abrégé de ERREUR LISTE, ou encore, LISTE DES ERREURS, et provoque l'apparition à l'écran des 28 erreurs d'èxécution avec leur numéro. Une touche arrête le défilement du listing, une autre le fait rependre, et ESC l'arrête définitivement.

Exemple : `ERRLIST`

ERRGOTO

ERRGOTO LIGNE ou LABEL envoit le déroulement d'un programme à un numéro de ligne ou à un label donné si une erreur a été détectée.

```
 10   ERRGOTO 100
 20   PRINT "NOM DU FICHIER A CHARGER : ";
 30   INPUT A$
 40   LOAD A$
 50 END
100   PRINT "CE FICHIER N'EXISTE PAS ! RECOMMENCEZ !"
110   PRINT :GOTO 20
```

Ici, ERRGOTO 100 demande à ce que le programme aille à la ligne 100 lorsqu'une erreur aura lieu. L'erreur que l'utilisateur peut faire est d'entrer un nom de fichier inexistant. A ce moment, le programme est détourné vers la ligne 100. ERRGOTO donne la possibilité d'afficher nous même nos messages d'erreur. Le nombre d'ERRGOTO est illimité, ce qui permet de définir différents messages d'erreur à plusieurs endroits dans un même programme.

ERROR

ERROR EN déclenche volontairement une erreur de numéro EN. La liste des erreurs et de leur numéro apparaît avec un ERRLIST. Ainsi, pour voir apparaître le message d'erreur correspondant au numéro 1, il suffit de faire : `ERROR 1`

Le numéro des messages d'erreur de l'HYPERBASIC sont compris entre 0 et 27. Les numéros de 28 à 255 sont réservés à l'utilisateur.

```
10   CLS
20   PRINT
30   PRINT "ENTREZ UN CHIFFRE ENTRE 1 ET 10 : ";
40   INPUT A
50   IF A<1 OR A>10 THEN ERROR 18
```

ERRNB

ERRNB est une variable qui renvoie le numéro d'erreur quand une erreur s'est produite. En connaissant le numéro de l'erreur avec ERRNB, on a la possibilité d'afficher des messages d'erreur plus compréhensibles.

Reprenons l'exemple de ERRGOTO :

```
 10   ERRGOTO GESTIONERREUR
 20   PRINT "NOM DU FICHIER A CHARGER : ";
 30   INPUT A$
 40   LOAD A$
 50 END
100 ]GESTIONERREUR
110   IF ERRNB=4 THEN PRINT "LE FICHIER ";A$;" N'EXISTE PAS"
120   IF ERRNB=12 THEN PRINT "LE NOM ";A$;"EST INCORRECT."
```

Dans l'exemple de chargement de fichier précédent, ERRGOTO détournait le programme en cas d'erreur. Malheureusement, on ne pouvait traiter qu'une seule erreur (fichier absent), car le numéro de l'erreur n'était pas connu.

Avec ERRNB, on sait si le fichier à charger n'existe pas (ERRNB=4), ou si le nom du fichier est incorrect . (ERRNB=12).

ERRNL

La variable ERRNL donne le numéro de ligne où s'est produite une erreur. Combinée avec ERRNB, ERRNL permet de donner à l'utilisateur le maximum d'informations sur l'erreur qu'il a commise :

```
 10   ERRGOTO 100
 20   PLOT 100,20,"A"
 30 END
100   PRINT "ERREUR NO";ERRNB;" EN LIGNE";ERRNL
```

En combinant ERRGOTO, ERRNB et ERROR, il devient possible de faire des erreurs 'maison'. L'HYPERBASIC utilise les numéros d'erreur de 0 à 27. Il reste à l'utilisateur les numéros d'erreur de 28 à 255.

En générant une erreur dont le numéro est compris entre 28 et 255, on définit soit même les messages d'erreur qui vont s'afficher selon la faute commise.

Exemple :

```
 10   ERRGOTO 100
 20   PRINT "AGE : ";
 30   INPUT AGE
 40   IF AGE<0 OR AGE>120 THEN ERROR 30
 50   PRINT :PRINT "NOM ( MAXIMUM 7 CARACTERES ) : ";
 60   INPUT NOM$
 70   IF LEN(NOM$)>7 THEN ERROR 31
 80   PRINT "OK ...";NOM$;" AGE ;AGE;"ANS ."
 90 END
100   IF ERRNB=30 THEN PRINT "NE RENTREZ PAS N'IMPORTE QUOI !"
110   IF ERRNB=31 THEN PRINT "VOTRE NOM EST TROP LONG : 7
CARACTERES AU MAXIMUM ..."
```

Dans ce programme, nous générons 2 erreurs. Si l'âge entré n'est pas compris entre 0 et 120, ERROR 30 génère une erreur de numéro 30. Une erreur étant détectée, le programme se dirige vers la ligne 100 (à cause de ERRGOTO 100).

La ligne 100 affiche le message conséquent à l'erreur de numéro 30. Pareil si le nom de la personne est trop grand. (ERRNB=31)

RESUME

Lorsque qu'une erreur a lieu, ERRGOTO détourne le déroulement normal d'un programme à un numéro de ligne donné. RESUME permet de revenir à la ligne du programme où s'est produite l'erreur, ce qui permet de continuer le déroulement du programme. (Après avoir bien sûr corrigé l'erreur survenue).

```
10   CLS:ERRGOTO ERREUR
20   PLOT I,10,"A":WAIT 2
30   I=I+1
40   GOTO 20
50 ]ERREUR
60   CLS:I=1
70   RESUME
```

En ligne 20, on augmente inconsidérement la variable I. Or, la coordonnée X de l'écran ne peut dépasser 39, sous peine d'erreur. Quand I=40, une erreur se produit en ligne 20, et le programme est détourné au label ERREUR. On en profite pour remettre la variable fautive à 1 (I=1), et RESUME fait retourner le programme à la ligne où l'erreur s'est produite.

ERROFF

ERROFF interdit l'arrêt d'un programme par une erreur. Après un ERROFF, les messages ne sont donc plus affichés.

Un CTRL C (arrêt du programme) est considéré comme une erreur (no 19 = exécution arrêtée). En conséquence, il ne sera plus possible de stopper un programme si celui-ci contient un ERRGOTO, à moins que ERROFF ait été placé en début de programme.

L'ANALYSEUR SYNTAXIQUE DE L'EDITEUR PLEINE PAGE

Le TELESTRAT innove en mettant à votre disposition un analyseur syntaxique très puissant. Les débutants l'apprécieront. En effet, il est impossible de valider une ligne que le TELESTRAT ne comprend pas.

Sur la plupart des ordinateurs existant actuellement, le message 'Syntax error' (erreur de syntaxe) s'affiche lorsque quelque chose dans une ligne n'a pas été compris. Que faire alors ? Trouver tout seul sa faute (ce qui n'est pas évident), et retaper entièrement la ligne.

Ce problème ne se posera désormais plus pour tous les possesseurs de TELESTRAT, car l'ordinateur met non seulement automatiquement le curseur sur le caractère qui n'a pas été compris, mais en plus ..... il vous dit ce qu'il attendait.

Voici des exemples :

Si vous tapez A= (Return), le curseur se positionne après le '=', et le message '`Attention : expression numerique`' apparaît en haut de l'écran (sur la ligne de status).

Le TELESTRAT a en effet vu que vous n'aviez pas donné de valeur à la variable A, et le message en conséquence a été affiché. La ligne ne sera pas validée tant que vous n'aurez pas donné une valeur à A (A=10 ou A=B ..).

ACCENT,TRACE,CURSOR tous seuls verront le curseur se placer après ACCENT, TRACE, CURSOR, et le message '**Attention : SET ou OFF**' s'afficher. ACCENT,TRACE et CURSOR ne peuvent être suivis que par SET ou OFF.

PRINT ABS (10 et '**Attention : ")" ?**' apparaîtra et le curseur se positionnera après le 10 . Vous avez oublié de fermer la parenthèse.

PINGPRINT"HELLO" collé donnera '**Attention : plus de parametre**'. Le curseur est juste après le PING. Cela signifie que la commande PING n'a plus besoin de paramètres, et dans ce cas là, il manque deux points entre PING et PRINT "HELLO".

PRINT 'A donne '**Attention : 2eme "'" ? '** : Vous avez oublié la deuxième quote. (')PLOT tout seul et '**Attention : expression numerique**' apparaît.

PLOT 10 affiche '**Attention : "," ?**'car vous n'avez pas mis de virgule après le 10.

PLOT 10,10, donne '**Attention : expression alphanumerique**" : l'entrée d'une chaîne est attendue.

GOTO affiche '**Attention : No ligne ou label**' : la syntaxe de GOTO est GOTO + numéro de ligne ou GOTO + LABEL

Pour un FOR ... NEXT : **FOR** donne '**Attention : variable numerique**', **FOR A** donne '**Attention : "=" ?**', **FOR A=** donne '**Attention : expression numerique**', **FOR A=1** donne '**Attention : "TO" ?**', ...

En bref, une vingtaine de messages viendront vous prévenir en cas d'erreur en entrant une ligne de programme. Sans compter le curseur qui se place directement là où un caractère n'a pas été compris... En ce qui concerne GOTO et GOSUB, le TELESTRAT affiche constamment les messages :

- '**Ligne/Label(nom du label ou numéro de la ligne) non défini**' si un numéro de ligne ou un nom de label n'existe pas.

Exemple : tapez en mode direct GOTO DEBUT

- '**Label (nom du label) defini plusieurs fois**' si un label a été déclaré plus d'une fois dans un programme.

Exemple :

```
10   GOTO XX
20 ]XX
30   PRINT
40 ]XX
50   PRINT
```

Le label XX est présent à deux reprises : en ligne 20 et en ligne 40. Le message '**Label XX defini plusieurs fois**' s'affiche.

Les instructions NEXT, UNTIL, UNCOUNT ou WEND tapées seules sans FOR, REPEAT, COUNT, WHILE verront les messages '**NEXT sans FOR**', '**UNTIL sans REPEAT**', '**UNCOUNT sans COUNT**', '**WEND sans WHILE**' apparaître.

CHAPITRE 18

LES COMMANDES SYSTEME DU TELESTRAT

ASCII

La commande ASCII affiche à l'écran le tableau complet des codes ASCII compris entre #20 et #7F (En Hexadécimal).

```
    0 1 2 3 4 5 6 7 8 9 A B C D E F

20    ! " # $ % & ' ( ) * + , - . /
30  0 1 2 3 4 5 6 7 8 9 : ; < = > ?
40  @ A B C D E F G H I J K L M N O
50  P Q R S T U V W X Y Z [ \ ] ^ £
60  © a b c d e f g h i j k l m n o
70  p q r s t u v w x y z { | }
```

Pour obtenir la valeur hexadécimale correspondant à un caractère, on utilise les nombres donnés en haut du tableau, et aussi ceux situés à gauche.

Exemple : la lettre E est en 3ème rangée. On lit 40 sur la gauche. E est en 5ème colonne. Le code ASCII de E est #45 (#40 + #5).

Pour le signe ? : #20 + #F = #2F

PRINT ASC ("caractère") donne le code ASCII du caractère entre parenthèses.

PRINT 'caractère' affiche le code ASCII décimal du caractère entre quotes. Mais pour rechercher plusieurs codes ASCII (en hexadécimal), la commande ASCII en mode direct s'avère plus pratique.

SEI, CLI

Un ordinateur est constamment en contact avec ses périphériques. Plusieurs dizaines de fois par seconde, les périphériques envoient au TELESTRAT une demande d'interruption (appelée IRQ). A ce moment, le TELESTRAT arrête de travailler, et va traiter les périphériques. Cet arrêt n'est bien sûr pas visible, car il ne dure que quelques millièmes de seconde.

Pendant les interruptions, le TELESTRAT effectue beaucoup de choses : il scrute le clavier, il met à l'heure l'horloge, il interroge les entrées/sorties (RS-232C, CENTRONICS, SOURIS, JOYSTICK, CLAVIER...).

Ce traitement répété des périphériques prend du temps, et ralentit l'exécution d'un programme BASIC. C'est pour cela qu'existe l'instruction SEI qui interdit les interruptions, et CLI qui les autorise à nouveau.

Attention : SEI ne s'utilise pas n'importe comment. Il faut savoir que si l'on interdit les interruptions, tout ce qui était fait pendant celles-ci n'est plus exécuté :

- La gestion du clavier n'est plus effectuée. Résultat : le clavier est bloqué. (Le TELESTRAT ne s'occupe plus si vous tapez sur une touche ou non).

- L'horloge s'arrête.

-Toutes les entrées/sorties ne sont plus scrutées. (RS232C, IMPRIMANTE, MINITEL, JOYSTICK, SOURIS ...)

Exemple : tapez SEI en mode direct. Le clavier est bloqué. La seule manière de 'reprendre la main' est d'appuyer sur le bouton RESET.
Un programme ou une partie d'un programme utilisant les instructions SEI-CLI va 15 à 20 % plus vite qu'un programme où les interruptions sont inhibées.

Ainsi :

```
10   SEI
20   FOR I=1 TO 200
30    PRINT I
40   NEXT
50   CLI
```

Ira beaucoup plus vite que :

```
10   FOR I=1 TO 200
20    PRINT I
30   NEXT
```

Car les interruptions ne sont pas autorisées. N'oubliez pas de mettre CLI à la fin de votre programme, sous peine de voir le clavier bloqué après exécution.

La scrutation du clavier n'ayant plus lieu après un SEI, les instructions INPUT,GET, KEY$ n'ont aucun effet.

```
10   SEI
20   PRINT "APPUYEZ SUR UNE TOUCHE"
30   PRINT
40   PRINT "CE N'EST PAS LA PEINE, LA GESTION DU"
50   PRINT "CLAVIER N'EST PLUS EFFECTIVE !"
60   PRINT
70   PING
80   PRINT "UN CONSEIL : APPUYEZ SUR LE BOUTON RESET !"
```

Toutes les commandes de gestion de l'horloge (TIME, CLOCK, WAIT), des entrées/sorties sont aussi ineffectives. En conclusion, n'utilisez SEI et CLI que pour encadrer des portions de programmes comprenant des calculs, des affichages, des boucles ...

Exemple :

```
10   PRINT "LONGUEUR (ENTRE 1 ET 150) : ";
20   INPUT L
30   SEI
40   HIRES
50   FOR  I=1 TO L
60    CURSET 0,0,1
70    BOX L,L,170
80   NEXT
90   CLI
```

NMI

Cette commande débloque l'ordinateur sans perdre le contenu de la mémoire. Elle est comparable à l'action sur le bouton RESET.

NMI s'utilise généralement en mode direct, à condition que le clavier ne soit pas bloqué.

```
10   SEI ' BLOQUE LE CLAVIER, ...
20   NMI ' FAIT UN 'RESET'
```

RESET

Beaucoup plus méchante que NMI, l'instruction RESET réinitialise le TELESTRAT : le programme résidant en mémoire est détruit, ainsi que les variables, les données ... Tout est détruit. Le message de présentation du TELESTRAT réapparaît.

```
10   FOR I=5 TO 1 STEP -1
20    PRINT I;"SECONDES AVANT LA DESTRUCTION TOTALE  !": PING
30    WAIT 10
40   NEXT
50   RESET
```

RESET s'obtient au clavier en appuyant sur la touche DEL et sur le bouton RESET en même temps.

CHAPITRE 19

LES COMMANDES DE GESTION DES CANAUX D'ENTREES/SORTIES

Comme vous l'avez remarqué au chapitre 1, le TELESTRAT est muni de nombreuses entrées/sorties (prise imprimante, RS232C, prise MINITEL, écran, ...).

On regroupe sous le nom d'entrées/sorties (abrégé en E/S) l'ensemble des opérations permettant la liaison d'un ordinateur à ses périphériques.

On désigne par périphérique tout ce qui n'est pas le microprocesseur et les mémoires d'un ordinateur. Par exemple, le clavier est un périphérique d'entrée, car il donne constamment des informations au microprocesseur (sorte de 'cerveau' d'un ordinateur). L'écran est une sortie. Il peut recevoir des données.

Une entrée/sortie est soit une entrée, soit une sortie, mais jamais les deux en même temps. Par définition, une entrée a toujours un numéro inférieur ou égal à 7, et une sortie un numéro supérieur ou égal à 8.

LISTE DES ENTREES/SORTIES :

0 à 7 = ENTREES :

0 CLAVIER, JOYSTICK, SOURIS
2 MINITEL EN ENTREE
3 RS232C EN ENTREE

8 à 20 = SORTIES :

8 ECRAN FENETRE No 0
9 ECRAN FENETRE No 1
10 ECRAN FENETRE No 2
11 ECRAN FENETRE No 3
14 SORTIE PARALLELE CENTRONICS
15 MINITEL EN SORTIE
16 RS232C EN SORTIE
17 ECRAN EN MODE EMULATION VIDEOTEX

Pour regrouper les E/S (Entrées/Sorties), on utilise 4 canaux. Un canal est un intermédiaire entre nous et les E/S. Les canaux du TELESTRAT prennent un numéro de 0 à 3. Chaque canal peut contenir de 0 à 4 E/S.

Lorsque vous ouvrez une ou plusieurs entrées dans un canal par exemple, toutes les informations reçues sur cette ou ces entrées seront envoyées sur la ou les sorties ouvertes sur le même canal.

OPCH

OPCH C, N ouvre l'E/S numéro N sur le canal C.

Exemple : le canal utilisé par défaut est le canal 0. Lors de son initialisation, le TELESTRAT ouvre deux E/S sur le canal 0. L'E/S numéro 0, et la numéro 8 : **OPCH 0,0 et 0,8.**

Regardez les numéros correspondant aux E/S ci-dessus. L'entrée N° 0 est le clavier, et la sortie N°8 est l'écran. OPCH 0,0 et OPCH 0,8 demandent à ce que tout caractère provenant de l'entrée N°0 (clavier) soit répercuté sur la sortie n° 8 (Ecran). C'est ce qui est fait : à chaque fois que vous tapez une touche au clavier, elle s'affiche à l'écran.

Supposons que vous vouliez que tout ce qui est affiché sur l'écran apparaisse aussi sur l'imprimante : un simple OPCH 0,14 et le tour est joué !

Examinons le canal 0 après un OPCH 0,14 :

- Le canal 0 contient 1 entrée : le clavier (0).

- Il contient 2 sorties : l'écran (8), et l'imprimante (14).

A chaque fois qu'un caractère est envoyé sur le canal 0, il est automatiquement et immédiatemment répercuté sur la ou les sorties ouvertes sur le canal 0. Dans notre exemple, tout caractère tapé au clavier (entrée no 0) est immédiatemment envoyé à l'écran (Sortie No 8), ainsi qu'à l'imprimante (Sortie No 14).

OPCH est un abrégé de OPen CHannel. (ouvrir canal N).

CLCH

CLCH C,N ferme l'E/S numéro N sur le canal C. N'oubliez pas qu'un canal accepte 4 E/S au maximum. Si vous tentez d'ouvrir une E/S alors qu'un canal en contient déjà 4, celle-ci sera ignorée. CLCH signifie CLose CHannel = fermer le canal N.

Après avoir copié chaque exemple de ce livre concernant les canaux, il est obligatoire d'appuyer sur le bouton RESET pour les réinitialiser à leurs valeurs de départ. Vous voulez que tous les caractères entrés s'inscrivent directement sur l'imprimante, mais qu'ils ne s'affichent plus à l'écran:

```
10   CLCH 0,8 ' FERME L'ENTREE NO 8 (ECRAN) SUR LE CANAL 0
20   OPCH 0,14 ' OUVRE LA SORTIE NO 14 (IMPRIMANTE) SUR LE CANAL 0
```

Résultat : tout caractère provenant de l'E/S No 0 (clavier) est renvoyé (en mode direct seulement et pas en mode programme) sur l'E/S No 14 (imprimante).

(Faites un RESET avant de copier le programme ci-dessous)

```
10   CLCH 0,8 ' FERME LA SORTIE ECRAN
20   OPCH 0,15 ' OUVRE LA SORTIE MINITEL
```

Après avoir exécuté ces deux lignes, tout caractère entré au clavier est immédiatement affiché sur le minitel.

Autre exemple :

```
10 ' APPUYEZ SUR LE BOUTON RESET
20   CLCH 0,0 ' FERME L'ENTREE CLAVIER
30   OPCH 0,2 ' OUVRE L'ENTREE MINITEL
```

Ici, tout caractère venant du MINITEL (2) sera affiché à l'écran (8).

PRINT]

PRINT]C;EA ou EN envoie sur le canal de numéro C (entre 0 et 3) la chaîne de caractères EA ou l'expression numérique EN. PRINT] a la même syntaxe que PRINT, à savoir PRINT]C;@10,10,... Les données spécifiées après le PRINT] seront envoyées sur toutes les sorties ouvertes sur le canal C. Le canal utilisé par défaut pour l'affichage étant le canal 0, PRINT "BONJOUR " correspond à PRINT]0;"BONJOUR".

Mais pour envoyer une chaîne sur un autre canal que le canal 0, il convient d'utiliser PRINT]C;...

Exemple :

```
10   OPCH 2,14 ' OUVRE LA SORTIE 14 SUR LE CANAL 2
20   PRINT ]2;"AFFICHAGE SUR IMPRIMANTE"
```

L'imprimante ayant été ouverte sur le canal 2, PRINT]2; ... envoie toute séquence de caractères sur imprimante.

```
10   OPCH 3,8 ' OUVRE LA SORTIE ECRAN
20   OPCH 3,14 ' ET L'IMPRIMANTE
30   OPCH 3,15 ' ET LE MINITEL
40   PRINT "ENTREZ UNE CHAINE :";
50   INPUT C$
60   PRINT ]3;C$
```

Avec ce programme, toute chaîne envoyée sur le canal 3 s'affiche à l'écran, sur l'imprimante, sur le minitel. Que vous n'ayez pas d'imprimante, de minitel ... n'est pas important. Le TELESTRAT sait si une imprimante est branchée ou pas, et par conséquent, il envoie ou non une chaîne de caractères sur celle-ci.

GET]

GET]C;VA saisit un caractère sur le canal numéro C, et l'affecte à la Variable Alphanumérique VA. Comme pour PRINT, GET A$ correspond à GET]0;A$.

```
10   OPCH 1,3 ' MET LE MINITEL EN ENTREE SUR LE CANAL 1
20   GET ]1;A $ ' SAISIT UN CARACTERE
30   PRINT "LE CARACTERE SAISI EST : ";A$
40   GOTO 20
```

WINDOW

L'instruction WINDOW est une application intéressante des E/S. Avec WINDOW on peut définir 4 fenêtres d'écran différentes. Les fenêtres ont des numéros allant de 0 à 3. L'écran que vous utilisez actuellement est la fenêtre d'écran 0 qui est la fenêtre par défaut.

L'écran actuel (fenêtre 0) se compose de 28 lignes de 40 colonnes chacune. Ce sont les coordonnées maximum qu'admet l'écran TEXT.

WINDOW N,Dx,Fx,Dy,Fy définit une fenêtre d'écran de numéro N, et de coordonnées de départ Dx,Fx (position en haut et à gauche de la fenêtre) et d'arrivée Dy,Fy (position en bas et à droite de la fenêtre).

Condition obligatoire régissant l'ouverture d'une fenêtre :

Coordonnées : Dx<Fx Dy<Fy

```
10    CLS
20    WINDOW 0,5,35,5,15
```

Avec WINDOW 0,5,35,5,15 la fenêtre d 'écran par défaut (0) est maintenant comprise entre les coordonnées Dx=5, Fx=35, et Dy=5, Fy=15.

Un appui sur le bouton RESET réinstaure l'écran normal en 27x40. Nous allons maintenant définir deux fenêtres à l'écran, et y écrire quelque chose.

```
10    CLS
20    WINDOW 0,0,39,0,10 ' DEFINIT LA FENETRE 0
25    WINDOW 1,10,30,15,20 ' ET  LA FENETRE 1
30    OPCH 0,9 ' OUVRE LA FENETRE 1 SUR LE CANAL 0
40    WORD ' AFFICHE LA LISTE DES COMMANDES SUR LES 2 FENETRES
ECRAN
```

Pour écrire une chaîne sur les fenêtres de 1 à 3, on met sur un canal le numéro de sortie de ces fenêtres, on y envoie les informations désirées.

Tout ce qui est entré au clavier est normalement affiché sur la fenêtre 0. Et pourquoi ne pas le faire apparaitre seulement sur une fenêtre no 1 ?

```
10    CLS
20    WINDOW 0,0,39,0,10 ' OUVRE FENETRE 0
30    WINDOW 1,15,25,15,20 ' OUVRE FENETRE 1
40    CLCH 0,8 ' FERME LA SORTIE 8 (FENETRE 0) SUR LE CANAL 0
50    OPCH 0,9 ' OUVRE A LA PLACE LA SORTIE 9 (FENETRE 1)
```

POS

POS, de même que WINDOW, ne sont pas des instructions de gestion de canaux, mais leur approche nécessite une connaissance des canaux d'E/S, ce qui explique leur présence à cet endroit du livre. POS (EN) est une variable qui donne des informations sur la position horizontale du curseur :

POS (0) donne la position horizontale du curseur dans la fenêtre 0.
POS (1) dans la fenêtre 1.
POS (2) dans la fenêtre 2.
POS (3) dans la fenêtre 3.

```
10    PRINT "POS(ITION) : ";
20    PRINT "X=";POS(0)
```

CROSS

CROSS E,S transfère des informations entre l'entrée numéro E, et la sortie numéro S.

CROSS 2,14 transfère tous les caractères provenant du MINITEL sur l'imprimante.

L'instruction CROSS est un mode de communication direct. On ne s'en sort qu'en tapant CTRL C.

CROSS 0,14 envoie toute donnée venant du clavier (0) vers l'imprimante (14).

CROSS utilise le canal 1 pour envoyer ses données. Cette information vous est utile, si par exemple, vous désirez envoyer les informations reçues par CROSS dans une seconde sortie qui aura été préalablement ouverte sur le canal 1.

CROSSX

CROSSX E1,S1,E2,S2 établit une double communication entre l'entrée E1, la sortie S1, et l'entrée E2, et la sortie S2.

CROSSX 0,15,2,17 prend les caractères tapés au clavier (0), les répercute au minitel (15), et les caractères provenant du minitel (2) sont affichés à l'écran (Ecran en mode émulation VIDEOTEXT = Sortie No 17).

La liaison entre le clavier et le minitel est multiplexée (elle se fait dans les deux sens : minitel --> Ecran mode émulation VIDEOTEXT et Clavier --> Minitel)

La sortie du mode de communication 'CROSSX' s'opère par un CTRL-C.

CROSSX utilise 2 canaux : les canaux 1 et 2.

CHAPITRE 20

LES COMMANDES ET FONCTIONS DE TRAVAIL SUR LA MEMOIRE

Une mémoire est en fait un ensemble de cases dans lesquelles sont stockées différentes valeurs. Un ordinateur n'est pas capable de calculer en décimal, du moins pas directement. Son système de numérotation est le système binaire (base 2). Celui-ci n'utilise que des suites de 0 et de 1 pour représenter des nombres. Ces chiffres 0 ou 1 sont appelés des bits.

Un ensemble de 8 bits forme un octet qui est le contenu d'une case mémoire. En d'autres termes, chaque case mémoire de votre ordinateur est un ensemble de 8 bits appelé octet, chaque bit pouvant prendre la valeur 0 ou 1.

Dans un octet, les bits portent un numéro de 7 à 0 en allant de la gauche vers la droite. Quand un bit est actif (1), il représente deux fois la valeur du bit situé à sa droite. Quand il est inactif (0), il vaut 0. La plus petite valeur que peut contenir un octet est 0, et la plus grande est 255. (Ce nombre s'obtient en sachant que le binaire est un systéme en base 2, et qu'il y a 8 bits dans un octet. 2^8 donne 256, et de 0 à 255, cela fait bien 256).

```
BIT      :  7   6   5   4   3   2   1   0
BINAIRE  :  1   1   1   1   1   1   1   1
VALEUR   : 128 64  32  16  8   4   2   1

Décimal  :  255  (128+64+32+16+8+4+2+1)
```

Pour connaître la valeur décimale du nombre binaire 01001011 :

```
BIT      :  7   6   5   4   3   2   1   0
BINAIRE  :  0   1   0   0   1   0   1   1
VALEUR   :  0  64   0   0   8   0   2   1

Décimal  :  75 (0+64+0+0+8+0+2+1)
```

La fonction de conversion BIN$ (nombre) vous facilite grandement la tâche, puisqu'elle convertit directement un nombre en binaire.

```
PRINT  BIN$(255)  ==>  %11111111
```

Le signe % est utilisé pour spécifier qu'il s'agit d'un nombre binaire. L'autre base numérique qu'utilise votre ordinateur est la base 16, qui est le système de numérotation hexadécimale.

DECIMAL	HEXADECIMAL
1	1
2	2
3	3
4	4
5	5
6	6
7	7
8	8
9	9
10	A

11	B
12	C
13	D
14	E
15	F
16	10
17	11

En hexadécimal, les nombres décimaux de 10 à 15 sont représentés par les lettres de A à F.

HEX$(nombre) convertit un nombre en hexadécimal.

PRINT HEX$(255) ==> #FF

PRINT HEX$(%11111111) ==> #FF

est le signe utilisé par les nombres hexadécimaux.

ORGANISATION DE LA MEMOIRE DU TELESTRAT

L'organisation complète de la mémoire vous est donnée en annexe. Votre ordinateur a une mémoire de 64 Ko (Kilo-octet). Un Ko vaut 1024 octets, et non pas 1000 comme le laisserait présager le système international de poids (1 kg = 1000 g).

Avoir une mémoire de 64 Ko peut se traduire par 'disposer de 65536 cases'. (64*1024=65536).

La mémoire de l'ORIC se compose de deux parties :

- La RAM (Random Acces Memory ou encore 'mémoire vive') :

C'est dans la RAM que sont stockées toutes les données que nous entrons. Des informations peuvent être lues ou écrites en RAM.

Composition de la RAM :

La mémoire de l'ORIC va de l'adresse #0000 à l'adresse #FFFF, ou encore de 0 à 65535. La RAM occupe la zone de #0000 à #BFFF. La mémoire RAM est décomposée en pages de 256 octets (256=#100).

La page 0 est l'ensemble des cases entre l'adresse #0 et l'adresse #FF, la page 1 va de #100 à #2FG...

La zone de #0000 à #07FF, est occupée par le TELESTRAT. On y trouve tous les renseignements concernant le travail du TELESTRAT. Tous les pointeurs, toutes les données y sont conservées.

(Exemple : il y a constamment la coordonnée X du curseur en mode TEXT à l'adresse #220, et en #224 la coordonnée Y).

Le programme BASIC que vous entrez est conservé à partir de l'adresse #800 en mémoire (en montant). Lorsque vous mettez en route votre programme BASIC, il est compilé (traduit en langage machine) à partir de l'adresse #9800 en descendant.

En mode HIRES, les caractères sont stockés dans la zone #9800-#9FFF. L'écran HIRES commence en #A000, et fini en #BF3F.

En mode TEXT, les caractères sont dans la zone #B400-#BB7F, et l'écran TEXT en #BB80-#BFDF.

La ROM (Read Only Memory ou encore 'mémoire morte') :

Le contenu de la mémoire ROM ne peut pas être changé. La lecture de la ROM est possible, mais pas son écriture. Cette zone mémoire contient en effet le programme en langage machine qui est en quelque sorte 'l'âme' de votre ordinateur. C'est en ROM qu'est stocké le programme qui effectue les calculs, les affichages, les dessins, qui comprend le langage BASIC...

Composition de la ROM :

La ROM occupe la zone #C000-#FFFF. Cela représente 16 Ko de mémoire. La 'mémoire morte' est divisée en 8 banques (zones) mémoire de 16 Ko chacune. Selon la tâche qu'il doit effectuer le TELESTRAT utilise une banque ou l'autre.

POKE

POKE AD,N écrit dans une case mémoire d'adresse AD (entre #0000 et #BFFF) le nombre N (entre 0 et 255).

Exemple : l'écran TEXT commence à l'adresse #BB80.

POKE #BB80,65 affiche la lettre A en haut et à gauche de l'écran. Pourquoi ? A a pour code ASCII 65, et entrer un code ASCII à l'écran provoque l'affichage du caractère y correspondant.

```
10   POKE #BB80,17
20   POKE #BB81,65
30   POKE #BB82,66
40   POKE #BB83,67
```

POKE AD,"chaîne" place directement une chaîne de caractères en mémoire. Par exemple, POKE #BB80,"A" est pareil que POKE #BB80,65

```
10   CLS
20   POKE #BB80,"ET ALORS ?"
```

Ou :

```
10   CLS
20   FOR I=#BB80 TO #BFDF
30    POKE I,CHR$(96)
40   NEXT
50   WAIT 40
60   NMI
```

PEEK

PEEK (AD) renvoie le contenu de l'octet situé à l'adresse AD.

```
10   POKE #BB80,80
20   PRINT PEEK (#BB80)
```

```
10   FOR I=0 TO #100
20    PRINT PEEK(I);"  ";HEX$(PEEK(I));"  ";BIN$(PEEK(I))
30   NEXT
```

DOKE

L'instruction DOKE AD,N stocke en mémoire une valeur sur deux octets. Un DOKE équivaut à un Double POKE. N varie entre 0 et 65535.

Exemple : `DOKE #5000,800`

Le nombre 800 est stocké en #5000 et en #5001 sous la forme : 32 en #5000 et 03 en #5001, puisque 3*256+32 = 800

DEEK

DEEK(AD) renvoie la valeur conservée sur deux octets à l'adresse AD.

PRINT DEEK (#500)

Une fois un fichier chargé, vous pouvez connaître :

Son adresse de début avec : `PRINT HEX$(DEEK(#52D))`
Son adresse de fin avec: `PRINT HEX$(DEEK(#52F))`
Son adresse d'exécution automatique (s'il s'agit d'un programme en langage machine) avec :
`PRINT HEX$(DEEK(#531))`

CALL

CALL AD démarre un programme en langage machine commençant à l'adresse AD. L'emploi de CALL nécessite des connaissances en langage machine. Ne l'utilisez qu'avec précaution, sous peine de 'planter' la machine.

```
CALL #C000
```

HIMEM

HIMEM AD fixe l'adresse de l'emplacement mémoire le plus élevé (AD) disponible sous BASIC. Cette commande qui s'utilise exclusivement en mode direct permet de réserver un espace mémoire pour le langage machine, ou pour diverses données. Ainsi, le programme BASIC et les données à conserver n'empièteront pas l'une sur l'autre.

La valeur de HIMEM est sauvée en même temps qu'un programme BASIC. Il faut refaire la commande HIMEM avant chaque exécution d'un programme, sauf si celui-ci est chargé à partir du lecteur de disquette.

Exemple : avant d'écrire un programme de serveur MINITEL en BASIC, il est obligatoire de faire un HIMEM #4000, car la zone mémoire de #4000 à la fin est utilisée par le TELESTRAT pour y conserver des données (arborescence, ...) par la suite, il sera inutile de faire HIMEM à chaque utilisation.

MOVE

MOVE AD1,AD2,AD3 déplace un bloc mémoire situé entre les adresses AD1 (début) et AD2 (fin) à partir de l'adresse AD3.

Déplaçons l'écran HIRES (#A000-#BF3F) à partir de l'adresse #5000 :

```
10   HIRES
20   CURSET 120,100,0
30   FOR I=1 TO 99 STEP 5
40    CIRCLE I,1
50   NEXT
60   PAPER 2
70   MOVE #A000,#BF3F,#5000 ' DEPLACE L'ECRAN HIRE EN MEMOIRE
```

Pour réobtenir le dessin :

```
10   HIRES
20   MOVE #5000,#6F40,#A000 ' RETRANSFERE LE DESSIN HIRE A SA
PLACE
```

FRE

La fonction système FRE (qui n'est autre qu'une variable) donne le nombre d'octets encore libres en mémoire pour un programme BASIC. Ce nombre dépend du mode d'écran dans lequel on se trouve, puisque l'écran HIRES occupe 8 Ko de mémoire à lui tout seul.

```
10   CLS
20   PRINT "ACTUELLEMENT, IL Y A";FRE;"OCTETS DISPONIBLES..."
```

GRAB

Une grande partie de la mémoire du TELESTRAT est réservée à l'écran HIRES. Aussi, si vous ne l'utilisez pas et que vous envisagez d'écrire un programme BASIC assez long, il est possible de libérer la place occupée par l'écran HIRES et par la zone des caractères (#9800-#B400) en tapant GRAB. Cette commande s'utilise uniquement en mode direct (comme pour HIMEM).

Après l'exécution de GRAB, vous ne pourrez plus passer en mode HIRES tant que vous n'aurez pas entré : RELEASE.

RELEASE

S'utilisant en mode direct, la commande RELEASE récupère la place en mémoire libérée après un GRAB. L'écran HIRES est de nouveau utilisable.

Faites :

```
GRAB
PRINT FRE "OCTETS LIBRES"
RELEASE
PRINT FRE "OCTETS LIBRES"
```

CHAPITRE 21

LES COMMANDES DE GESTION DE L'IMPRIMANTE

Le TELESTRAT accepte toutes sortes d'imprimantes. Mais chacune ayant ses codes spéciaux, il était impossible d'adapter chaque instruction à chaque imprimante existante sur le marché. La différence apparaissait surtout sur la façon de recopier un écran HIRES. Aussi a-t-on choisi le format d'impression correspondant à celui le plus généralement utilisé pour des imprimantes : le mode "bit image" de type EPSON ou "compatible PC". La plupart des imprimantes du marché sont compatibles EPSON. Que vous ne possédiez pas cette compatibilité n'est pas vital. Vous pourrez tout de même sortir sur papier des caractères en mode TEXT. Pour les dessins en mode HIRES, il vous faudra faire un programme spécial les reproduisant.

Une imprimante a généralement deux types de standard de connexion:

- le standard parallèle CENTRONICS : le branchement s'effectue à la prise 'CENTRONICS' de votre ordinateur. Ce standard est le plus répandu en micro-informatique. L'envoi d'informations vers l'imprimante se fait 'en parallèle'. Cela signifie que les bits sont envoyés 8 par 8 (donc octet par octet).

- le standard RS232 : le branchement s'opère à la prise RS232 de votre ordinateur. L'envoi de données est réalisé bit par bit. Si vous utilisez une imprimante au standard RS232, ce chapitre ne vous concerne pas. La façon de brancher et d'utiliser une imprimante RS232 vous est en efffet donnée au chapitre concernant les commandes de gestion de la RS232.

LPR SET, LPR OFF

Dès sa mise en marche, le TELESTRAT affiche le nom de tous les périphériques branchés. (Drive : A-B, imprimante ...). Si l'on branche une imprimante après l'allumage du TELESTRAT, sa présence ne sera pas reconnue, et toute tentative d'impression sera ignorée.

LPR SET reteste la présence de l'imprimante à l'ordinateur. L'impression de données devient alors possible si elle est connectée.

LPR OFF signale à l'ordinateur que l'imprimante n'est plus en service. Elle évite le bloquage d'un programme qui cherche à imprimer quelque chose alors que l'imprimante est éteinte.

LFEED SET, LFEED OFF

Normalement, une imprimante effectue un saut de ligne automatiquement quand elle reçoit le signal RETURN. D'autres ont besoin en plus d'un signal de saut de ligne. (appelé LINE FEED).

Si votre imprimante est dans ce cas, LFEED SET enverra directement un signal de saut de ligne à l'imprimante après chaque RETURN.

LFEED OFF n'enverra plus de saut de ligne après RETURN.

```
10   PRINT "VOTRE IMPRIMANTE A-T'ELLE UN SAUT DE"
20   PRINT "LIGNE AUTOMATIQUE APRES CHAQUE RETURN ?"
30   PRINT (O/N) ?";
40   INPUT A$
50   IF A$="N" THEN LFEED SET ELSE LFEED OFF
```

LWIDTH

LWIDTH EN détermine le nombre de caractères qui seront imprimés sur une même ligne, avant qu'un RETURN soit envoyé, et que l'imprimante passe à la ligne suivante.

La valeur par défaut de LWIDTH est de 80 colonnes.

```
10   PRINT "LARGEUR DE L'IMPRESSION : ";
20   INPUT A
30   LWIDTH A
```

LPRINT

LPRINT "chaîne" ou variable a le même rôle que l'instruction PRINT, sauf que l'affichage se produit sur imprimante et non pas à l'écran.

```
10   LPRINT "O R I C *****************"
20   LPRINT "T E L E S T R A T ******"
30   LPRINT "6 4 K o *****************"
```

Ou :

```
10   PRINT "QUELLE PHRASE VOULEZ-VOUS IMPRIMER ? ";
20   INPUT E$
30   LPRINT E$
```

Et :

```
10   LPRINT "CODES ASCII DE 32 A 127"
20   FOR I=32 TO 127
30    LPRINT CHR$(I);" ";
40   NEXT
```

LOUT

LOUT EN envoie à l'imprimante le code ASCII ayant pour numéro EN. Cette commande est beaucoup plus rapide et pratique que LPRINT CHR$(...), en particulier pour les codes de contrôle.

LPRINT CHR$(65); équivaut simplement à LOUT 65, ou encore LOUT 'A'.

```
10   LOUT 96
20   LOUT 'O'
30   LOUT 'R'
40   LOUT 'I'
50   LOUT 'C'
```

LLIST

LLIST provoque l'impression sur imprimante d'un programme. La syntaxe de LLIST est la même que celle de LIST : LLIST ligne ou LLIST ligne, ligne permet de lister une partie de programme seulement.

```
10 ' LLIST
20   LLIST 10
30   LPRINT :LLIST 10,40
40 ' C'EST FINI !
```

LDIR

LDIR donne l'affichage du catalogue d'une disquette sur imprimante. LDIR accepte les mêmes paramètres que DIR, à savoir les noms, les extensions, les jokers. Attention toutefois à ne pas mettre de codes couleurs dans le nom de votre disquette. Cela risquerait de donner n'importe quoi sur une imprimante qui ne traite que le noir et le blanc.

Exemples :

```
LDIR
LDIR "TOTO.BAS"
LDIR "B"
LDIR "*.COM"
```

TCOPY

T(ext)COPY effectue la recopie complète de l'écran TEXT ou LORES sur imprimante. En mode HIRES, seules les 3 lignes du bas de l'écran sont recopiées.

```
10   CURSOR OFF
20   CLS
30   FOR I=1 TO 26
40    PRINT "ABCDEFGHIJKLMNOPQRSTUVWXYZ"
50   NEXT
60   TCOPY
```

HCOPY

H(ires)COPY recopie sur imprimante un écran haute résolution. L'écran est imprimé point par point. Les 3 lignes TEXT du bas de l'écran ne sont pas prises en compte. Le format d'impression de HCOPY est le mode 'bit image' des imprimantes EPSON ou compatibles PC.

```
10   HIRES
20   CURSET 0,0,1
30   BOX 239,199,1
40   CURSET 120,100,1
50   CIRCLE 90,1:CIRCLE 50,1:CIRCLE 10,1
60   HCOPY
```

VCOPY

V(idéotext)COPY recopie une page VIDEOTEX sur imprimante. La page doit bien sûr être en mémoire. Seul les codes ASCII sont imprimés. Les graphismes disparaissent.

```
10   CROSS 2,17
20 ' APPUYEZ SUR CTRL-C QUAND L'ECRAN EST TRANSMIS
30   CURSOR OFF
40   VCOPY
```

LBUF

Une imprimante est un périphérique relativement lent. A cause de la très grande différence entre la vitesse d'impression et la vitesse d'envoi des données, vous êtes parfois obligé d'attendre un temps assez long pour reprendre le contrôle du clavier. C'est pour diminuer cette attente que le TELESTRAT est doté d'une 'mémoire tampon' (appelée aussi buffer).

A chaque fois qu'une instruction d'impression est rencontrée, tout ce qui est à imprimer est conservé dans une partie de la mémoire, et le TELESTRAT vous rend immédiatement la main. Il continuera à imprimer les données reçues pendant que vous pourrez faire autre chose.

La taille mémoire de ce buffer (tampon) est à l'origine de 2048 octets (2Ko). Pour l'impression d'un écran HIRES de 8 Ko par exemple, le buffer ne sera pas assez grand. Il faudra attendre que l'imprimante ait reçu 6 Ko (soit les 3/4 du dessin) pour que le TELESTRAT vous rende la main. Il ne restera en effet plus que 2Ko à imprimer qui seront immédiatement transférés dans le buffer.

L'instruction LBUF permet de redéfinir la taille du buffer imprimante, et ainsi, de diminuer le temps d'attente pendant l'impression.

LBUF ADRDEB,ADRFIN demande à ce que la zone comprise entre ADRDEB et ADRFIN soit occupée par le buffer imprimante.

Exemple : LBUF #5000,#7000 définit un nouveau buffer commençant en #5000, et finissant en #7000 (exclu).

La taille de ce nouveau buffer est de :

```
PRINT  #7000-#5000
```

8192 octets, ou encore 8192/1024 = 8 ko.

Les valeurs de LBUF par défaut sont : de #CA00 à #D200 (adresse de fin exclue)

CHAPITRE 22

LA REDEFINITION DES CARACTERES

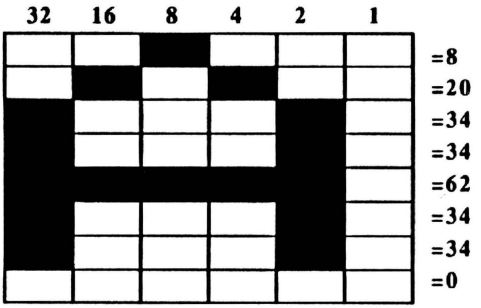
Lorsque vous appuyez sur la touche A, un A s'affiche à l'écran. Le TELESTRAT va en fait chercher dans sa mémoire le dessin correspondant à la lettre A, et l'affiche.

Les caractères de l'ORIC sont dits redéfinissables, car rien ne vous empêche de changer le motif de la lettre A en une petite balle par exemple.

Chaque caractère est stocké dans un tableau situé en mémoire. Dans ce tableau, 8 données sont réservées pour chaque caractère. Chaque donnée correspond à une ligne horizontale du caractère, et regroupe 6 points.

Un caractère se définit dans une grille de 6x8 blocs, ou encore, 8 lignes verticales de 6 points chacune.

Voici par exemple comment est stockée la lettre A :



Sur ce dessin, toutes les cases remplies forment la lettre A. Les nombres au dessus de chaque bloc représentent la valeur en bits des cases hachurées. Les chiffres sur le côté de chaque rangée représentent la somme des bits de la rangée. En effet, chaque rangée est composée de 6 bits. La rangée n° 2, par exemple, vaut 20 : les cases correspondant à 16 et 4 en décimal sont hachurées, et 16+4=20 .

Chaque case remplie ayant une valeur binaire de 1, et chaque case vide une valeur vide une valeur binaire de 0, le dessin de la lettre A donne aussi :

```
001000
010100
100010
100010
111110
100010
100010
000000
```

Sur ce dessin, chaque bit à 1 représente une case pleine.

Vous pouvez alors obtenir directement la valeur de chaque rangée en décimal :

1ère rangée : PRINT %001000 --> 8
2ème rangée : PRINT %010100 --> 20
....

En mode TEXT, le tableau des caractères standards est stocké à partir de l'adresse #B400, et le tableau des caractères alternés à partir de #B800. En mode HIRES, ces 2 tableaux sont déplacés en #9800 (standards) et en #9C00 (alternés).

Chaque caractère occupant 8 données dans le tableau des caractères, il faut multiplier le code ASCII du caractère recherché par 8, et ajouter #B400 (pour un caractère standard en mode TEXT) pour obtenir l'adresse de la 1ère donnée de ce caractère.

Pour A : `PRINT #B400+8*'A'`

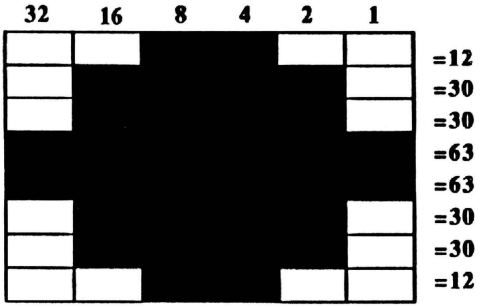
'A' représente le code ASCII de A ('A'=65).

#B800+8*'A' donne 46600 ou encore #B608 en hexadécimal. Cette adresse est celle de la 1ère donnée composant le caractère A. Pour redéfinir le caractère A, il faudra changer les 8 valeurs qui donnent l'affichage d'un 'A' lors de l'appui sur cette touche. (8,20,34,34,62,34,34,0)

Faites en mode direct : `POKE#B400+8*'A',63` et appuyez sur la lettre A. Une barre s'affiche au dessus de la lettre A. Pourquoi ? Vous venez de changer la première donnée définissant la lettre A (normalement 8 = voir tableau du caractère A). POKE#B800+8*'A',63 a mis 63 en 46600. Si vous regardez le tableau du caractère A, vous vous apercevez que 63 équivaut à 32+16+8+4+2+1, ou encore à %111111 en binaire, c'est à dire que toutes les cases de la première rangée du tableau sont remplies.

Pour retrouver le jeu de caractères normaux, appuyez sur RESET. Tout appui sur le bouton RESET, ou tout arrêt du TELESTRAT remettra le jeu de caractères normaux en place.

Vous avez décidé de faire un jeu de football. Malheureusement, le jeu de caractères de l'ORIC ne dispose pas de ballon . Qu'à cela ne tienne, nous allons redéfinir le caractère A.



Maintenant que nous avons les 8 valeurs définissant le ballon, nous allons les mettre en mémoire à la place des valeurs de la lettre A.

```
10   D$="12,30,30,63,63,30,30,12"
20   FOR I=1 TO LEN (D$) STEP 3
30    A=VAL(MID$(D$,I,2))
40    POKE#B400+8*'A'+E,A
50    E=E+1
60   NEXT
```

En ligne 10, on stocke dans la variable chaîne les 8 valeurs (chaque valeur doit occuper 2 caractères. Si une valeur était égale à 0, elle serait stockée sous la forme 00). La ligne 20 dit : pour I égal 1 jusqu'à la fin de la chaîne D$. En 30, A=VAL(MID$(D$,I,2)) extrait une par une les 8 valeurs de la chaîne D$. La ligne 40 stocke les valeurs en mémoire. La variable E sert de 'pointeur'.

La disquette STRATSED fournie avec le TELESTRAT possède un éditeur de caractères. Tapez GENCAR pour le charger. Tous les caractères standards de 32 à 127 apparaissent d'abord en bas de l'écran. Au milieu de l'écran, un damier de 6*8 apparaît, et un 1 se dessine (1er caractère affiché par défaut). En haut et à gauche de ce damier apparaît une croix qui fait office de curseur. Déplacez la avec les 4 touches fléchées. Si vous appuyez sur la barre d'espace et que la case en dessous de la croix est vide, celle-ci se remplie, et elle se vide si elle était auparavant remplie. En haut et à droite de l'écran se trouve le code ASCII ainsi que le caractère qui est en train d'être redéfini. Pour le moment, le code est 49, et la "lettre" 1 (par défaut). Pour changer le code ASCII de la lettre à redéfinir, entrez directement son code ASCII avec les chiffres de 0 à 9, et appuyez sur RETURN ou ESPACE. Entrez par exemple 066 et ESPACE. Le B s'affiche alors.

Une fois votre caractère redéfini, tapez sur la touche ESC. Le message '**Sauvez ?**' apparaît. Répondez '**Y**' si vous voulez sauver le jeu de caractère et entrez son nom. Le catalogue de tous les jeux de caractères présents sur la disquette (extension .CHS) est automatiquement donné.

La disquette STRATSED met à votre disposition 4 styles de caractères redéfinis différents :

SLANT.CHS	= caractères penchés.
BOLD.CHS	= caractères gras.
SQUARE.CHS	= caractères fins.
STD.CHS	= caractères standards.

Les caractères standards sont ceux instaurés à l'allumage du TELESTRAT. Un appui sur le bouton RESET remettra à tout moment le jeu de caractères standards.

Pour changer un des 4 jeux de caractères, il suffit tout simplement de taper leur nom. (Exemple : SQUARE.CHS)

```
10   CLS
20   FOR I=32 TO 127
30    PRINT I;" = ";CHR$(I)
40    IF KEY$ THEN GET A$
50   NEXT
```

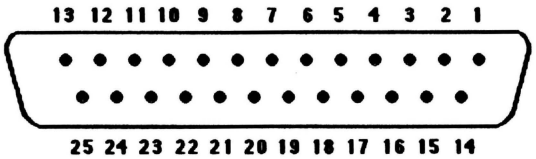
Ce programme affiche les codes ASCII ainsi que les caractères correspondant aux caractères standards compris entre 32 et 127. Un appui sur une touche arrête le défilement, un autre le fait reprendre.

CHAPITRE 23

LES COMMANDES DE GESTION DE LA RS 232

La RS232 est l'entrée/sortie série du TELESTRAT.

Voici le schéma de la prise RS 232:



Elle est composée de 25 broches numérotées de 1 à 25, en allant de la droite vers la gauche.

Seules certaines broches jouent un rôle précis dans la transmission de données entre le TELESTRAT et un périphérique connecté à la RS232, les autres n'étant pas connectées. Les voici :

2	TX	S	Données émises
3	RX	E	Données reçues
4	RTS	S	Demande d'émission
5	CTS	E	Prêt pour émission
7	GND		Masse signaux
8	DCD	E	Détection de porteuse
20	DTR	S	Terminal prêt

De nombreux périphériques ont adoptés le mode de transmission par la RS232. Les imprimantes par exemple. Pour une imprimante avec interface RS232, il faudra un cable spécial pour la relier au TELESTRAT.

Brochage à adopter pour brancher une imprimante série sur un TELESTRAT, via la RS232 :

TELESTRAT		IMPRIMANTE
7 (GND)	<---->	7 (GND)
2 (TX)	----->	3 (RX)
8 (DCD)	<-----	20 (DTR)

SSPEED

Avant d'envoyer ou de recevoir des informations provenant de la RS232, il faut, selon le protocole de réglage de la RS232, définir la vitesse de transmission des données (en Bauds). SSPEED v fixe parmi les valeurs entières standards (75,150,300,600,1200,2400,4800,9600 ou 19200 Bauds) la vitesse de transmission. 1 Baud équivaut à 1 bit par seconde. 1200 Bauds correspond à 1200 bits par seconde, soit 120 caractères par seconde si l'on prend 10 bits pour chaque caractère. (8 bits du caractère et 2 bits de stop.)

La vitesse de transmission par défaut est mise à 9600 Bauds à l'allumage du TELESTRAT.

Exemple : `SSPEED 4800`

SMODE

SMODE EA fixe les paramètres de la transmission série. SMODE est suivi d'une chaîne de caractères qui indiquent chacun un paramètre :

- Le nombre de bits : 5,6,7 ou 8.
- Le bit de stop : 0 = 1 bit de stop
  1 = 2 ou 1/2 bit selon la vitesse

- L'horloge utilisée : B = horloge interne
  C = horloge externe

- La gestion de parité : P = parité autorisée
  N = parité non autorisée

- Le type de parité : O = imparité
  E = parité
  M = Mark
  S = Space

Les valeurs par défaut de SMODE à l'allumage sont :

SMODE "80BN" soit une transmission en 8 bits, 1bit de stop, horloge interne, parité non autorisée.

SPRINT

SPRINT EA (Expression Alphanumérique) ou SPRINT EN (Expression Numérique) agit exactement comme PRINT pour l'écran, ou LPRINT pour la sortie parallèle, et affiche sur le terminal relié la RS232 (imprimante, ordinateur,...) les données suivant SPRINT.

N'oubliez jamais que l'envoi de données vers un périphérique relié à la RS232 suppose que les paramètres de la transmission série ont déjà été réglés (SSPEED, SMODE).

```
 10   CLS:PRINT
 20   PRINT "VITESSE DE TRANSMISSION SOUHAITEE ? ";
 30   INPUT VIT
 40   SSPEED VIT
 50   PRINT
 60   PRINT "chaine A ENVOYER : ";
 70   INPUT CHAINE$
 80   SPRINT CHAINE$
 90   PRINT :PRINT "ENCORE (O/N) ? ";
100   GET A$:PRINT A$;
110   IF A$="O"THEN 50
120 END
```

SLIST

SLIST (ligne,ligne) envoie vers le périphérique relié à la sortie série RS232 le listing du programme BASIC en mémoire. SLIST est donc pareil que LIST, sauf que les données sont envoyées vers la RS232.
Les numéros des lignes de départ et de fin peuvent être également spécifiés.

```
SLIST  100,150
```

Attention : les imprimantes reliées à la RS232 acceptent plus ou moins bien les caractères de contrôle (caractères compris entre 0 et 31 = couleurs,...). Un caractère de contrôle peut donner n'importe quoi sur votre imprimante. Une seule solution pour l'éviter : ne pas en mettre dans les listings.

SOUT

SOUT EN envoie vers le périphérique relié à la sortie série le caractère de code ASCII EN.

```
10  SOUT  65:SOUT  13:SOUT  66:SOUT  13:SOUT  67
```

En règle générale, il est préférable de ne pas utiliser les instructions SOUT et SPRINT pour envoyer des données vers la RS232. La meilleure solution consiste à ouvrir la sortie RS232 sur un canal (le 3 par exemple), et d'y envoyer ses données.

Exemple : OPCH 3,16 ouvre la sortie RS232 sur le canal 3.

Et dès maintenant, il suffira d'un `PRINT]3;"chaine de caractères"` pour que celle-ci s'affiche sur le périphérique RS232.

CONSOLE

L'instruction CONSOLE établit une double liaison entre le TELESTRAT et la RS232 : les caractères frappés au clavier sont envoyés vers la sortie RS232, et les caractères reçus par l'entrée RS232 sont affichés à l'écran. Le retour au BASIC se fait par CTRL C.

SDUMP

SDUMP visualise à l'écran les caractères reçus par la prise RS232. Elle permet entre autres, de comprendre pourquoi un cable de liaison ne fonctionne pas, en examinant les données reçues.

Lors d'un SDUMP, les codes de 0 à #1F et de #80 à #FF sont affichés en hexadécimal et en rouge.

Si des erreurs de transmission ont été détectées, un chiffre de #B0 à #B7 est affiché en vidéo inverse :

B1 : erreur de parité.
B2 : il manque un bit de stop.
B4 : un ou plusieurs caractères ont été perdus dans la transmission.

SLOAD

SLOAD charge un fichier à partir du périphérique relié à l'entrée RS232, avec le format spécial "TELESTRAT"
Pour plus de détails, voir l'instruction LOAD.

SLOADA

SLOADA "adresse" charge un fichier sans en-tête à partir du périphérique relié à la RS232. Normalement, le TELESTRAT sauve et charge un programme avec un en-tête qui lui est spécifique, et contenant des informations 'personnelles' (début, fin, type du programme ...). Dans certaines transmissions, comme celle via la RS232, cet en-tête peut devenir inutile, voir gênante. Dans ce cas, utiliser SLOADA qui ne prend pas d'en-tête.

SSAVE

SSAVE "nom de fichier" a exactement le même rôle que SAVE pour le lecteur de disquette, mais sauve un fichier sur le périphérique connecté à la sortie série. SSAVE accepte les mêmes options que SAVE (,A,E,T,AUTO), à l'exception de SAVEU,SAVEO et SAVEM. Voir l'instruction SAVE pour plus de détails sur ces options. Le format est spécifique "TELESTRAT"

Exemple : `SSAVE "FICHIER",A#7000,E#8000,AUTO`

SSAVEA

SSAVEA sauve un fichier du TELESTRAT via la RS232 sans en-tête spécifique au TELESTRAT, ce que fait en temps normal SSAVE. Cette en-tête devient en effet gênante dans certaines applications.

```
SSAVE "PROGRAMME"
```

SEBUF

SEBUF adresse de début, adresse de fin redéfinit la taille du buffer réservé à la réception de données via la RS232. La taille de ce buffer est à l'origine de 1/2 Ko. Pour certaines applications, il pourra être utile d'augmenter la taille de ce buffer grâce à SEBUF.

Les adresses par défaut de SEBUT sont : #C800 (adresse de début inclue) à #CA00 (adresse de fin exclue).

SEBUF #5000,#7000 définit un nouveau buffer de réception de #5000 à #7000 de 8 K octets.

Pour retrouver la taille d'origine du buffer, faire SEBUF #C800,#CA00.

SRBUF

SRBUF adresse de début, adresse de fin définit la taille du buffer d'émission vers la RS232. Ce buffer, de 1/2 K octet à l'origine occupe la portion mémoire #C680,#C800 (exclu).

Si, par exemple, vous ne désirez que recevoir des données par la RS232, pourquoi ne pas récuperer la zone mémoire réservée à l'émission ?

Un SRBUF #C680,#CA00, et le tour est joué !

Cable TELESTRAT-IBM

Pour en terminer avec la RS232, voici le détail du brochage à adopter pour réaliser une liaison TELESTRAT-IBM.

Le TELESTRAT dispose d'une prise RS232 de 25 broches DB25. L'IBM dispose soit d'une DB25, soit d'une DB9.

TELESTRAT			IBM	IBM	
DB25			DB25	DB9	
Rx	3	<-----	2	3	Tx
Tx	2	----->	3	2	Rx
GND	7	<---->	7	5	GND
DCD	8	<-----	20	4	DTR
DTR	20	------>	6	6	DSR

Pour une DB25 (IBM) : si le brochage donné ci-dessus ne fonctionne pas correctement, effectuez les straps suivants :

IBM DB25 : 4 (RTS) ---> 5 (CTS)
IBM DB25 : 6 (DSR) ---> 8 (DCD)

Pour une DB9 (IBM) : si le brochage donné ne fonctionne pas correctemment, effectuez les straps suivants :

IBM DB9 : 7 (RTS) ---> 8 (CTS)
IBM DB9 : 6 (DSR) --->1 (DCD)

Détails des brochages IBM DB9 et DB25:

DB9	DB25
1 DCD	
2 Rx	2 Tx
3 Tx	3 Rx
4 DTR	4 RTS
5 GND	5 CTS
6 DSR	6 DSR
7 RTS	7 GND
8 CTS	8 DCD
9 RI	
	20 DTR

UTILISATION D'UNE IMPRIMANTE SERIE

En tapant POKE #28A,132 tous les accès vers l'imprimante parallèle seront dirigés vers l'imprimante série. Cette astuce vous permet désormais d'utiliser LPRINT, LOUT, LLIST, LWIDTH. Et tous les caractères seront affichés sur l'imprimante RS232. Pour revenir au mode normal, entrez : POKE #28A,128.

INTERFERENCES AVEC LE MINITEL

Vous avez sans doute remarqué que lors de l'envoi de caractères vers la RS232 en serveur ou en mode émulation, l'écran du MINITEL était envahi par des caractères parasites. Pour éviter cela, il faut envoyer les codes suivant au MINITEL :

1/B 3/B 6/0 5/B 5/3

Ce qui donne : `MOUT #1B:MOUT #3B: MOUT #60:MOUT #5B:MOUT #53` ou encore :
`MPRINT CHR$(#1B)CHR$(#3B)CHR$(#60)CHR$(#5B)CHR$(53);`

L'envoi de ces codes a pour effet de couper la prise péri-informatique de l'écran du MINITEL. Résultat : plus rien ne s'affiche sur le MINITEL lors de l'envoi de caractères via la RS232.

Pour revenir en mode normal, entrez les codes suivants:

1/B 3/B 6/1 5/B 5/3

Ce qui donne : `MOUT #1B:MOUT #3B:MOUT #61:MOUT #5B:MOUT #53` ou encore :
`MPRINT CHR$(#1B)CHR$(#3B)CHR$(#61)CHR$(#5B)CHR$(#53);`

CHAPITRE 24

LES COMMANDES DE GESTION DU MINITEL

Ce chapitre ne contient aucun renseignement sur la façon de créer une page ou une arborescence pour un serveur minitel, tout ceci étant largement détaillé dans le manuel des applications télématiques fourni avec le TELESTRAT. En conséquence, il est indispensable de lire ce manuel, et de savoir créer sans problème une arborescence avant d'entamer la lecture de ce chapitre.

Voici les 5 instructions les plus utilisées lors de la conception d'un serveur en BASIC :

APLIC

Dans un programme en BASIC, APLIC EN renvoie le TELESTRAT au menu EN de la cartouche TELEMATIC.

APLIC 0 renvoie au menu général des applications télématiques.
APLIC 1 renvoie au menu du mode émulation minitel.
APLIC 2 renvoie au menu d'édition de page vidéotex.
APLIC 3 renvoie au menu d'édition du serveur.
APLIC 4 renvoie au menu du lancement du serveur.
APLIC 5 relance le TELESTRAT.

MINITEL

MINITEL est l'instruction qui vous permet de revenir à la gestion de l'arborescence et de l'enchaînement des pages par la télématique lorsque vous insérez une page HYPERBASIC dans une arborescence (page rouge) pour faire un retour au BASIC.

En clair, si une page HYPERBASIC est rencontrée dans l'arborescence lors de l'exécution d'un serveur, le TELESTRAT ira directement exécuter le programme BASIC en mémoire.

MINITEL est l'instruction BASIC qui demandera au TELESTRAT de retourner à la gestion du serveur par la télématique.

PAGE$

PAGE$ est une variable qui renvoie le nom de la page de l'arborescence qui est actuellement traitée. Elle permet de se diriger dans un programme BASIC selon la page où l'on se trouve.

PAGE$ renvoie toujours une chaîne de 7 caractères.

```
IF PAGE$="MENU    " THEN GOTO ...
```

Va à une partie du programme selon la page où l'on se trouve. N'oubliez jamais que si le nom de la page ne contient pas 7 caractères, il devra être complété par des espaces.

`IF PAGE$="MENU"` ne marche pas (Il n'y a pas 7 caractères).

`IF PAGE$="MENU    "` marche (7 caractères = 4 lettres et 3 espaces).

SERVEUR

L'instruction BASIC SERVEUR EN envoie à une option de la cartouche télématique selon la valeur EN :

SERVEUR 0 : LANCER LE SERVEUR
SERVEUR 1 : TESTER LE SERVEUR
SERVEUR 2 : MODE BORNE DE COMMUNICATION
SERVEUR 3 : TESTER SANS ACCES AU DISQUE

TINPUT

TINPUT (@y,x;)EN;VA saisit une chaîne depuis l'entrée MINITEL. Les coordonnées y et x de la saisie peuvent éventuellement être spécifiés. EN représente le nombre de caractères à saisir.

TINPUT gère automatiquement la deconnexion au bout de 2 minutes 30, ce qui permet d'éviter le 'plantage' du serveur, si le correspondant vient de se déconnecter.

TINPUT ne rend la main au BASIC que lorsqu'une des touches suivantes est pressée: SOMMAIRE / ANNULATION / RETOUR / REPETITION / GUIDE / SUITE / RETOUR / CORRECTION

```
TINPUT  @10,12;4,A$
```

Place le curseur de saisie en x=12 et y=10, et attend qu'une chaîne de 4 caractères soit entrée. Cette chaîne sera affectée à la variable A$.

REALISATION D'UN MINI-SERVEUR A TITRE D'EXEMPLE.

Nous allons vous apprendre ci-dessous comment réaliser un serveur à partir du BASIC.

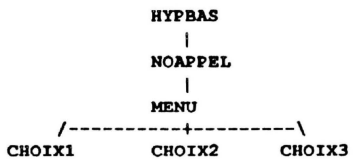
ATTENTION : la première chose à faire avant de créer un serveur en BASIC est de taper :

```
NEW:HIMEM  #4000
```

Ceci est impératif.

Le serveur que nous allons créer aura 6 pages vidéotex que vous allez devoir créer. Son rôle sera d'afficher le nombre de personnes ayant déjà accédé au serveur, puis de proposer 3 choix. L'accès au choix n°1 conduira directement au choix n°2. Une fois qu'un des 3 choix aura été choisi, le programme vous demandera votre nom, l'heure, et les minutes.

Commencez par créer l'arborescence suivante :



Les pages NOAPPEL, CHOIX1, CHOIX2 et CHOIX3 comportent un accès au BASIC et doivent apparaître en rouge dans l'arborescence. La page MENU qui propose 3 choix apparaît en bleu.

Sauvez l'arborescence, et nommez la ARBO.

Créez maintenant les 6 pages de l'arborescence (peu importe ce que contiennent ces pages) :

PAGE HYPBAS : marquez y par exemple : `"ESSAI DE SERVEUR EN HYPERBASIC"`.

PAGE NO APPEL : entrez : `Nombre d'appels:`
PAGE MENU : faites un menu comme ceci :

```
    *** MENU ***
1 - ACCES AU CHOIX 1
2 - ACCES AU CHOIX 2
3 - ACCES AU CHOIX 3

    Votre choix :
```

PAGES:CHOIX1,CHOIX2,CHOIX3 : marquez en double hauteur souligné au milieu de l'écran : `CHOIX1, CHOIX2, CHOIX3`.

Sauvez les 6 pages vidéotex créées avec leur nom donné dans l'arborescence .

Retournez au BASIC, et copiez le programme suivant :

(Tapez `NEW:HIMEM #4000` avant de le copier)

```
 10   IF FLAG=TRUE THEN 100 ' Teste si arbo en memoire.
 11   ENVOI=#A0:SOMMAIRE=#A5:SUITE=#A7
 20   LOAD "ARBO.SRV" ' Charge l'arborescence
 30   FLAG=TRUE ' Met variable FLAG à 1.
 40   SERVEUR 1 ' Mode 'tester le serveur'
100   IF PAGE$="NOAPPEL" THEN NOAPPEL
102   IF PAGE$="CHOIX1 " THEN CHOIX1
104   IF PAGE$="CHOIX2 " THEN CHOIX2
106   IF PAGE$="CHOIX3 " THEN CHOIX3
108 ]NOAPPEL
110   OPEN 0,1,"COMPTEUR" ' Ouvre fichier sequentiel
120   IF FST(1)=0 THEN CP=1 ELSE STAKE 1,CP 'Prend valeur du
compteur
125   FOR I=1 TO 5  ' Saute
126    PRINT ]1;     '   5
127   NEXT          ' lignes
130   PRINT ]1;"Vous etes l'appel no";CP
132   PRINT @0,13;"APPEL NO ";CP
135   WAIT 50 ' Attend 5 secondes.
140   REWIND 1' Place pointeur sequentiel en debut de fichier.
150   CP=CP+1 ' Augmente le compteur d'appels.
160   SPUT 1,CP:CLOSE 1 ' Sauve compteur.
170   POKE 255,SUITE: ' Simule appui sur la touche SUITE.
180   MINITEL ' Retour à la TELEMATIC.
200 ]CHOIX1
210   PRINT ]1;@7,4;"VOUS VENEZ D'ACCEDER AU CHOIX NO 1":WAIT 20
220   POKE #590,"CHOIX2 ":POKE 255,ENVOI:MINITEL
300 ]CHOIX2
310   PRINT ]1;@7,4;"VOUS VENEZ D'ACCEDER AU CHOIX NO 2":WAIT 20
320   GOTO QUESTIONS
400 ]CHOIX3
410   PRINT ]1;@7,4;"VOUS VENEZ D'ACCEDER AU CHOIX NO 3":WAIT 20
500 ]QUESTIONS
```

```
505   PRINT ]1;@15,2;"VOTRE NOM SVP :";
510   TINPUT 15,NOM$ ' Attend l'entree d'un nom.
511   S=PEEK(255)
512   IF S=SOMMAIRE THEN MINITEL
513   IF S<>ENVOI THEN 505
520   PRINT ]1;@17,2;"HEURE : ";
530   TINPUT 2,HEURE$ ' Saisit l'heure.
531   S=PEEK(255)
532   IF S=SOMMAIRE THEN MINITEL
533   IF S<>ENVOI THEN 520
540   PRINT ]1;@19,2;"MINUTES : ";
550   TINPUT 2,MINUTE$ ' Saisit les minutes.
551   S=PEEK(255)
552   IF S=SOMMAIRE THEN MINITEL
553   IF S<>ENVOI THEN 540
555   PRINT ]1;CHR$(12); ' efface l'ecran minitel
560   COUNT 10
570    PRINT ]1;"MERCI ";NOM$;" IL EST ";HEURE$;":";MINUTE$
580   UNCOUNT
590   POKE 255,0:MINITEL
```

Sauvez ce programme par : `SAVE "DEMO",AUTO`.

Lancement du programme : le lancement d'un serveur comprenant un programme BASIC se fait toujours par RUN, et jamais en mode minitel par l'option 'Lancer le serveur', qui aboutit à un plantage de la machine.

Tapez : `RUN`

Le numéro de l'appel s'affiche sur le minitel, ainsi que sur l'écran 'espion' (écran). Choisissez un des 3 menus.

Explications du programme :

Un programme BASIC lié à la TELEMATIC doit toujours commencer par :

```
10    IF FLAG=TRUE THEN 50
```

A chaque fois que l'arborescence trouve une page avec accès au BASIC, elle retourne au début du programme BASIC en mémoire.

La première chose à faire dans un programme BASIC est de charger et de lancer l'arborescence. Or, étant donné qu'à chaque fois qu'une page de l'arborescence avec accès au BASIC est rencontrée, le TELESTRAT retourne au début du programme BASIC en mémoire, l'arborescence risque de se charger et de s'exécuter à l'infini.

C'est pour éviter cela que l'on place toujours la ligne IF FLAG=TRUE THEN ...

Dans le programme ci-dessus : lorsque vous lancez pour la première fois le programme (RUN), les variables sont mises à 0. La variable FLAG vaut donc 0 (FALSE). La condition en ligne 10 n'étant pas vérifiée, les lignes 11, 20 et 30 sont exécutées. L'arborescence est chargée (ligne 20), et la variable FLAG est mise à 1 (ligne 30). Le serveur est alors lancé.

Et maintenant, si une page de l'arborescence fait appel au BASIC, FLAG valant 1 (TRUE), l'arborescence ne sera pas chargée et lancée. Tout ceci est dû au fait que la TELEMATIQUE retourne toujours au début du programme BASIC en mémoire.

Les lignes 100 à 106 envoient le déroulement du programme à une ligne spéciale, selon la page du serveur où l'on se trouve.

page NOAPPEL : la page NOAPPEL a pour fonction d'afficher le nombre d'appels qui ont déjà eu lieu. Ce nombre d'appels est stocké dans un fichier séquentiel nommé COMPTEUR. La ligne 110 ouvre ou crée selon le cas ce fichier sur disquette.

Vous remarquez qu'il n'y a pas de FILE EN au début de ce programme BASIC, alors que l'on ouvre un fichier séquentiel. Il faut savoir que la TELEMATIC exécute automatiquement un FILE 2. Conclusion : on ne peut pas ouvrir plus de deux fichiers en TELEMATIQUE, et un programme TELEMATIQUE ne doit jamais comporter de FILE EN.

La ligne 120 dit que si le fichier est vide, alors le nombre d'appels vaut 1. Sinon, STAKE1,CP charge le nombre d'appels dans la variable CP. La ligne 130 affiche le nombre d'appels sur le MINITEL, et la ligne 132 sur votre téléviseur. (Pour vous renseigner sur le nombre d'appels).

Les lignes 150 et 160 augmentent le compteur, et le sauvent sur disquette.

La ligne180 (Instruction MINITEL) retourne à la gestion de l'arborescence par la TELEMATIQUE.

page MENU : cette page sélectionne votre choix (entre 1 et 3).

page CHOIX1 : la page CHOIX1 renvoie au BASIC, et le sous-programme CHOIX1 est exécuté. Il y a affichage de '`Vous venez d'acceder au choix no 1`' (ligne 210). Ensuite, le programme se rend à la page CHOIX 2.

Le retour direct à une page est possible en exécutant ces 3 instructions :

1 - `POKE #590,"CHOIX2 "` : faire un POKE #590,"nom de la page à accéder (7 caractères)"

2 - `POKE 255,ENVOI` : simule l'appui sur la touche ENVOI.

3 - `MINITEL` : retour à la TELEMATIC

(Voir le manuel des applications télématiques page 74 pour plus de détails.)

page CHOIX2 : la page CHOIX2 retourne au BASIC et va au sous-programme CHOIX2 par la ligne 104.

La ligne 310 affiche "`Vous venez d'acceder au choix no 2`", et la ligne 320 dirige le programme vers le label QUESTIONS.

page CHOIX3 : la page CHOIX3 fait exactement la même chose que la page CHOIX2.

LABEL QUESTIONS : une fois que vous avez accédé à un des trois choix, des questions vous sont posées :

- En ligne 505 et 510, votre nom :

Remarquez par la même occasion que toutes les questions sont gérées par TINPUT.

Lors d'un TINPUT, la dernière touche de commande (SUITE, RETOUR,...) appuyée est mise à l'adresse 255. En ligne 511, `S=PEEK(255)` charge la variable S avec cette valeur qui est :

ENVOI = #A0
RETOUR = #A1
REPETITION = #A2
GUIDE = #A3
SOMMAIRE = #A5
SUITE = #A7
RETOUR = #A9

La ligne 532 teste si l'utilisateur a appuyé sur la touche SOMMAIRE. Si c'est le cas, alors on retourne à la gestion de l'arborescense par MINITEL, et le sommaire est affiché. (Car le programme de gestion de l'arborescence exécute la touche correspondant à la valeur située en 255 après en retour du BASIC. Ainsi : POKE 255,#A0 : MINITEL et la touche ENVOI sera appuyée (par simulation). POKE 255,0:MINITEL donnera un retour à la télématique sans qu'aucune touche soit appuyée.)

La ligne 533 teste si la dernière touche est différente de la touche ENVOI . Si oui, on retourne à la ligne 505 qui redemande le nom. (Ceci pour éviter les éventuels GUIDE, SUITE,RETOUR ... qui ne sont pas gérés.)

Le reste du programme agit exactement pareil que les lignes 505 à 513, et vous demande l'heure et les minutes.

L'écran est ensuite effacé (ligne 555), et le nom de la personne et l' heure sont affichés.

La ligne 590 retourne au programme de gestion de l'arborescence par la cartouche TELEMATIQUE.

Si vous possédez une extension horloge temps réel , tapez ces deux lignes en plus :

```
 515   GOTO 555
 570   CC=PEEK(#360):PRINT "MERCI ";NOM$;"IL EST:";PEEK(#361);":"
;PEEK(#362);":";PEEK(#363)
```

Et sauvez le programme par `SAVEU "DEMO",AUTO`. L'heure de l'horloge temps réel s'affichera désormais à l'écran.

Pour des raisons logicielles, la première page d'un serveur ne peut pas avoir accès au BASIC. Si vous appuyez sur la touche H sur cette première page, le TELESTRAT vous ignorera tout simplement !

Ne pas pouvoir accéder au BASIC dès la première page de l'arborescence oblige parfois l'utilisateur à créer une page 'bidon'. Il existe pourtant une solution à ce problème. La voici :

- Depuis le BASIC, tapez `APLIC 3`.
- Validez le choix 1 (accès au disque), et chargez votre arborescence .
- Validez ensuite le choix 6 (menu principal). Vous êtes de nouveau en BASIC.
- Tapez en mode direct : `POKE #400C,8`
- Tapez ensuite `APLIC 3` pour revenir en télématique.
- Choisissez l'option 2. La première page est maintenant en rouge.
- Sauvez votre arborescence une fois pour toute.

BORNE DE COMMUNICATION EN HAUTE RESOLUTION

Il est possible d'utiliser les possibilités graphiques du Télestrat pour intégrer des images hautes-résolutions dans les bornes de communication. Ce qui est bien sûr impossible en mode serveur à cause de la faible résolution du Minitel.

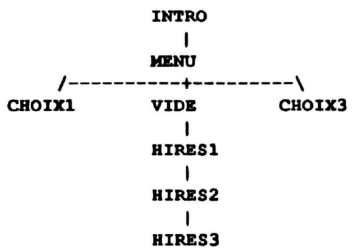
La première étape est de créer ces pages hautes-résolutions, et de les sauver avec un nom de 7 caractères maximum, et une extension .VDT, comme les pages vidéotex.

Il faut ensuite créer leur emplacement dans l'arborescence, là encore comme pour les pages vidéotex normales.

Il faut ensuite créer une page vidéotex vide, et surtout sans effacement préalable (Funct E).
Enfin, il faut placer cette dernière page de telle sorte que ce soit la dernière page vidéotex "normale" rencontrée par l'utilisateur avant les pages hautes résolutions.

En effet, après l'affichage de la page haute résolution, le Télestrat affichera automatiquement la dernière page vidéotex rencontrée. Si celle-ci était vide et sans effacement, ceci n'apparaitra pas.

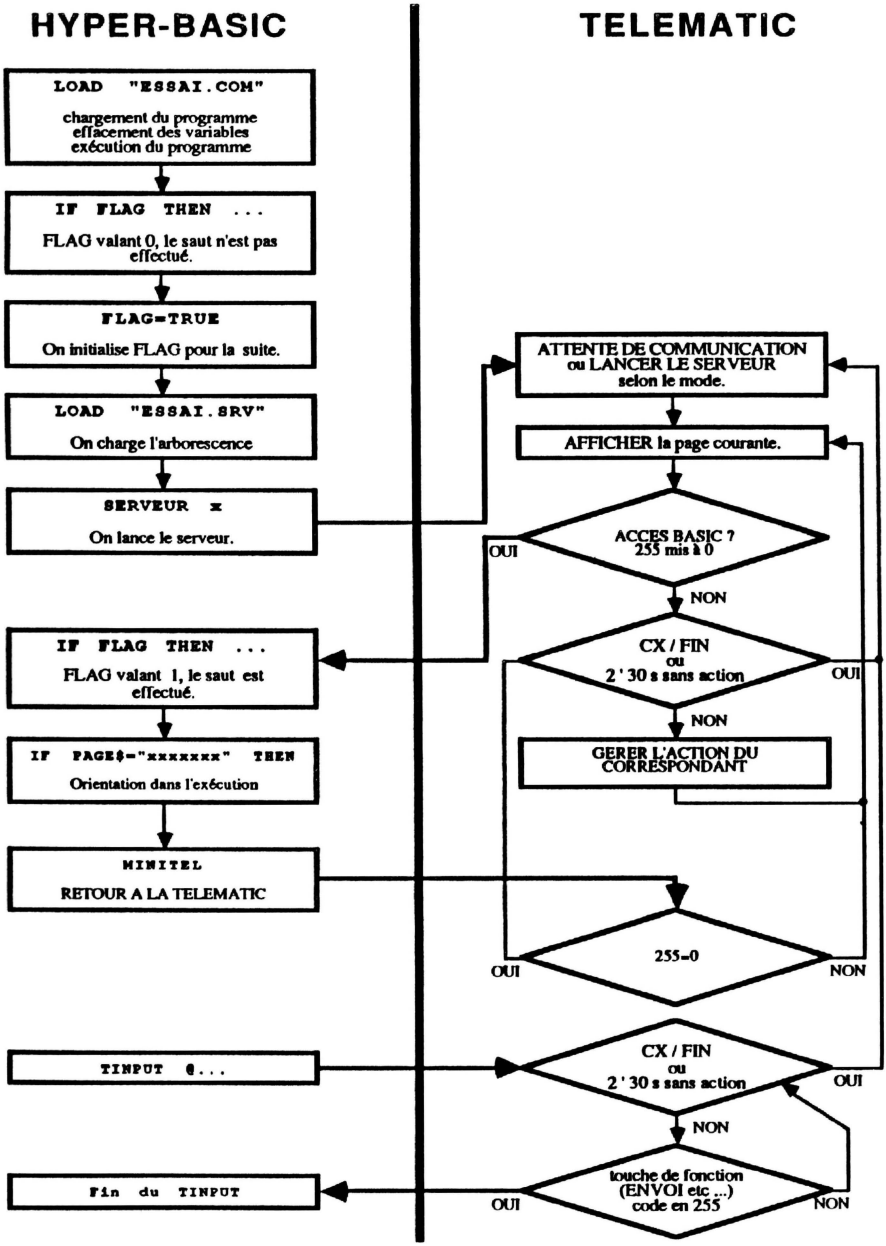
Voici un exemple d'arborescence comportant des pages hautes-résolutions (appelées ici "HIRESx", mais ce n'est bien entendu pas une obligation. La page vide sans effacement est appelée "VIDE", c'est original !



Bien entendu, les pages hautes-résolutions peuvent être de tous les types habituels (BASIC, BAL, SOMMAIRE, JOURNAL etc.) mais il faut prendre garde à ce qu'une page vide soit rencontrée avant.

Dernière remarque : lors de la composition des pages hautes-résolutions, ne pas oublier que la TELEMATIC effacera les 8 premières lignes du haut. Ne composer donc que les lignes 8 à 199.

ORGANIGRAMME RESUME D'UN SERVEUR EN BASIC



Les autres fonctions de gestion du MINITEL :

MPRINT

MPRINT EA (même syntaxe que PRINT) affiche la chaîne alphanumérique EA sur le minitel.

MPRINT @y,x;EA affiche la chaîne EA aux coordonnées x et y sur le MINITEL.

```
10   MPRINT CHR$(12)' Efface l'ecran
20   MPRINT "COMMENT ALLEZ VOUS DONC ?"
30   WAIT 40
50   MPRINT CHR$(12);
```

MOUT

MOUT EN envoie un caractère de code ASCII EN vers le MINITEL.

```
10   MOUT 12 ' CLS
20   COUNT 10
30    MOUT 96
40   UNCOUNT
```

Lorsque vous devez envoyer un grand nombre de données vers le MINITEL, il ne faut pas utiliser les instructions MPRINT et MOUT.

```
10   COUNT 500
20    MPRINT "A";
30   UNCOUNT
```

Ou :

```
10   COUNT 500
20    MOUT 65
30   UNCOUNT
```

Ces deux programmes utilisant MPRINT et MOUT vont saturer l'émission des données, et les caractères ne seront pas transmis correctement.

Une seule solution : ouvrir un canal (le 3 par exemple) sur la sortie MINITEL, et y envoyer ses données, ce qui donne :

```
10   OPCH 3,15 ' Ouvrir sortie MINITEL sur le canal 3.
20   COUNT 500
30    PRINT ]3;"A";
40   UNCOUNT
50   CLCH 3,15 ' Fermer la sortie MINITEL sur le canal 3.
```

MLIST

MLIST (ligne, ligne) envoie vers le MINITEL la liste des programmes BASIC résidant en mémoire. (Pratique si l'on veux faire profiter quelqu'un de son dernier programme : on l'envoie par la ligne téléphonique).

MLOAD, MLOADA

MLOAD "fichier" charge un fichier depuis le MINITEL (exactement comme LOAD pour le lecteur de disquette).

MLOADA ADR charge un fichier depuis le MINITEL, à l'adresse ADR, mais sans l'en-tête spécifique du TELESTRAT. Le fichier à charger doit être au format standard, c'est à dire 7 bits sans en-tête. Tapez CTRL-C lorsque la transmission est terminée. L'adresse de fin du fichier s'affiche alors.

MSAVE, MSAVEA

MSAVE "fichier" sauve un fichier vers le MINITEL. La syntaxe de MSAVE est exactement la même que SAVE (,A,E,T,AUTO). Seuls les suffixes U,O,M (MSAVEU, MSASEO, MSAVEM) ne sont pas acceptés.

MSAVEA "nom de fichier" (,A)(,E)(,T)(,AUTO) sauve un fichier sans en-tête vers le MINITEL.

CONNECT

Cette instruction effectue un retournement du Modem du MINITEL et une prise de ligne (transmission en 1200 Bauds).

UNCONNECT

UNCONNECT coupe la communiction du MINITEL, le déconnecte, et raccroche la ligne.

RING

L'instruction RING attend une sonnerie, et ne continue le déroulement d'un programme que lorsque celle-ci a été reçue.

WCXFIN

Instruction qui attend la connexion du correspondant avant de continuer l'exécution d'un programme.

Voici un exemple de programme reproupant CONNECT, UNCONNECT, RING, WCXFIN, et M SAVE.

```
10 ' PROGRAMME DE TRANFERT D'UN PROGRAMME A UNE AUTRE PERSONNE
20 ' L'AUTRE PERSONNE DOIT VOUS APPELER
30   RING ' ATTEND UNE SONNERIE
40   CONNECT ' RETOURNEMENT DE LIGNE
50   WCXFIN ' ATTENTE DE CONNECTION
60   MSAVE "JEU",A#4000,E#6000 ' ENVOI DU PROGRAMME EN MEMOIRE.
70   UNCONNECT ' DECONNEXION
```

ANNEXE 1

LISTE DES ERREURS D'EXECUTION

0 ? mode TEXT

Vous venez d'utiliser des instructions non exécutables en mode TEXT. Exemple : CURSET 120,100,1 n'est pas possible en TEXT.

1 ? mode HIRES

Vous venez d'utiliser des instructions non exécutables en mode HIRES. Exemple : PLOT 10,10,"A" n'est pas possible en HIRES.

2 ? mode minitel

Vous avez entré une instruction qui ne marche pas avec le mode minitel. Exemple : HIRES

4 fichier absent

Le nom de fichier à charger, copier, renommer, détruire ... n'a pas été trouvé sur la disquette. Vous avez peut-être fait une faute en tapant un mot Basic.

5 erreur d'E/S

Ce message apparaît lorsqu'il y a une erreur de lecture ou d'écriture sur le lecteur de disquettes. Dans ce cas, le TELESTRAT affiche le numéro du lecteur, la piste, le secteur, le type de l'erreur qui a été trouvée. Il vous laisse ensuite le choix entre : (A)bandonner, (R)ecommencer, (I)gnorer (l'erreur). Si vous décidez d'abandonner, le message 'Erreur d'E/S' (d'Entrée/Sortie) est affiché. N'utilisez Ignorer que pour les backup, ne pas utiliser pour les autres commandes (COPY, SAVE,...) sous peine de grave dommage pour la disquette.

6 fichier existant

Le nom de fichier que vous avez essayé de sauver/copier (SAVE, COPY) existe déjà sur la disquette. Une disquette n'accepte pas deux noms de fichiers identiques (Utiliser SAVEU).

7 disquette saturee

La disquette présente dans le lecteur en cours ne possède plus assez de secteurs libres pour sauver le fichier demandé. Une seule solution : changer de disquette, ou 'faire le ménage' sur sa disquette, en enlevant des fichiers inutiles.

8 disquette protegee

Vous avez essayé d'enregistrer un fichier sur une disquette protégée en écriture (taquet abaissé). Remontez le taquet, ou changez de disquette.

9 mauvais type de fichier

Cette erreur se produit lors d'une tentative de chargement d'un fichier à accès séquentiel ou d'un fichier à accès direct sans l'instruction OPEN, c'est à dire comme un fichier normal. Exemple : **FICHIER.DAT** renverra le message '**Mauvais type de fichier**'. Cette erreur a aussi lieu si vous tentez d'ouvrir un fichier à accès direct avec un OPEN 0, ... ouvrant un fichier séquentiel, et vice-versa.

10 disquette non STRATSED

La disquette présente dans le lecteur courant n'est pas au format STRATSED : toute tentative de lecture/écriture est impossible.

11 pas de STRATSED

Le STRATSED (système d'exploitation du DISK) n'a pas été chargé en mémoire lors de l'initialisation du TELESTRAT : tout accès au lecteur de disquette est impossible.

12 nom de fichier incorrect

Le nom de fichier entré est invalide : soit le nom ou l'extension contiennent trop de caractères (9 au maximum pour le nom, et 3 pour l'extension), soit le nom ou l'extension contiennent des caractères non alphanumérique (!,à,#,$,î,*...)

13 lecteur absent

Le lecteur auquel vous avez essayé de faire appel n'est pas branché au TELESTRAT. Un lecteur branché après l'initialisation de l'ORIC ne sera pas reconnu comme présent, sauf si vous tapez en mode direct : POKE#208+no du lecteur (entre 0 et 3),128 (si double face)+nbre de pistes. Pour connecter un lecteur B double face avec 42 pistes par face après initialisation, entrez : POKE#208+1,128+42. Et le message '**lecteur absent**' n'apparaîtra plus.

14 fichier ouvert

L'ordre OPEN "fichier" vient d'être rencontré, alors que le fichier est déjà ouvert.

15 fichier ferme

Vous avez entré une instruction de lecture ou d'écriture dans un fichier de données alors que ce fichier n'a pas été préalablement ouvert avec OPEN.

16 fin de fichier

Ce message signale que le pointeur d'un fichier séquentiel est à la fin de ce fichier. Il n'est donc plus possible de lire (STAKE) une valeur.

17 mauvais type d'expression

L'expression rencontrée par le TELESTRAT n'est pas du type attendu. Par exemple : STAKE 1,A au lieu de STAKE 1,A$

18 valeur incorrecte

Ce message se produit en général lorsque vous n'avez pas respecté la valeur minimum ou maximum possible dans la syntaxe d'une instruction. Exemple : CURSET 100,300,1 est impossible, car la coordonnée Y varie entre 0 et 199. Ce message se produit aussi lorsque l'on tente d'ouvrir un fichier de données (OPEN) sans avoir auparavant déclaré (FILE) le nombre de fichiers à ouvrir.

19 execution arretee

Ce message apparaît lors de l'appui sur CTRL-C pendant le déroulement d'un programme, ou quand l'instruction STOP est rencontrée.

20 nombre trop grand

La valeur donnée à un paramètre d'une instruction dépasse le nombre maximum qu'accepte ce paramètre. Exemple : FILE 100, PRINT FST(4), WIDTH 200...

21 division par 0

Il est impossible de diviser une valeur par 0.

22 LOG, LN, SQR, ou ^, Parametre <=0

Ces fonctions n'admettent qu'un paramètre positif.

23 mauvais NEXT

Vous avez mis plus de NEXT que de FOR, ou vous vous êtes trompés de variable, ou encore vous êtes sorti d'une boucle FOR / NEXT par un GOTO, alors qu'il est obligatoire de sortir par le NEXT.

24 index de tableaux trop grand

Vous avez utilisé un tableau avec plus d'éléments que vous n'en avez dimensionné. Par exemple : DIM A(12):A(14)=1. Ou vous avez tout simplement oublié de dimensionner votre tableau.

25 RETURN sans GOSUB

Le programme a cherché à exécuter une instruction RETURN avant d'être passé sur un GOSUB.

26 chaine trop longue

Lors d'une addition de chaîne, la longueur maximum de 255 caractères par chaîne a été dépassée.

27 trop de GOSUB

Le Télestrat n'accepte qu'un maximum de sous-programmes imbriqués (environ une centaine). Vous avez dépassé cette limite. Ou plus probablement, vous vous êtes trompés, en écrivant une structure du type : 10 GOSUB 10, qui boucle indéfiniment.

ANNEXE 2

LISTE DES ERREURS DE COMPILATION

Les erreurs de compilation (compilation = action de transformer un programme BASIC en un programme en langage machine) ont lieu lors du lancement d'un programme (instruction RUN), ou lors de son apparition à l'écran (instruction LIST).

ligne/label (numero de la ligne ou nom du label) absent

Ce message signale que l'utilisateur a fait appel à un numéro de ligne ou à un label après GOSUB, GOTO ou THEN qui n'existait pas.

UNCOUNT sans COUNT

Le programme a rencontré l'instruction UNCOUNT sans que l'instruction COUNT ait été auparavant trouvée.

NEXT sans FOR

Le programme a rencontré l'instruction NEXT sans avoir auparavant trouvé la boucle de comptage FOR ... TO.

UNTIL sans REPEAT

L'instruction UNTIL a été trouvée sans l'instruction REPEAT.

WEND sans WHILE

L'instruction WEND a été trouvée sans l'instruction WHILE.

label (nom du label) defini plusieurs fois

Un label a été défini à plusieurs lignes différentes, ce qui est impossible.

ANNEXE 3

LISTE DES ERREURS D'INTERPRETATION

Les erreurs d'interprétation sont affichées sur la première ligne en haut de l'écran, appelée ligne de status. Tous les messages d'erreurs d'interprétation commencent par '**Attention :**'. Ces messages sont délivrés si la syntaxe d'une instruction n'est pas correcte.

plus de parametre

Vous avez donné trop de paramètres à une instruction. Le TELESTRAT place le curseur à un certain endroit de la ligne, et vous signale qu'il n'a pas besoin des données placées après le curseur. Exemple : DRAW 100,100,1,10. Ce message est également délivré lorsque vous oubliez les deux points séparant deux instructions sur une même ligne. Exemple : **PINGPRINT**. Le TELESTRAT place le curseur après PING et affiche '**plus de paramètre**' : il manque deux points entre PING et PRINT.

SET ou OFF

Les instructions CURSOR, TALK, TRACE nécessitent l'apport de SET ou OFF. Si vous entrez CURSOR, TALK, ou TRACE sans SET ou OFF, le message '**SET ou OFF**' apparaît.

expression numerique

Une expression numérique (nombre ou variable) est attendue.

expression alphanumerique

Une expression alphanumérique (chaîne de caractères) est attendue.

No de ligne ou label

Message affiché avec un THEN,GOTO,GOSUB ... qui doivent être suivies d'un n° de ligne ou d'un label.

label

Le TELESTRAT attend l'entrée d'un label (après]).

expression

Après IF par exemple, le TELESTRAT attend une expression, que celle-ci soit numérique (IF A) ou alphanumérique (IF A$).

"(" ?

Il manque la parenthèse "(".

")" ?

Il manque la parenthèse ")".

**"," **

Il manque une virgule avant ou après un paramètre.

nombre trop grand

Le nombre entré est en dehors des limites possibles (trop grand). Exemple : #FF054

variable alphanumerique

Attend l'entrée d'une variable alphanumérique. Exemple : GET.

variable numerique

Attend l'entrée d'une variable numérique.

variable

Attend l'entrée d'une variable numérique ou d'une variable alphanumérique. Exemple : INPUT

"=" ?

Attend l'entrée du signe '='. Exemple : FOR A

"TO" ?

Attend l'entrée du mot clé TO dans la boucle FOR .. TO ...NEXT. Exemple : FOR I =1

"THEN" ?

Attend l'entrée de THEN dans une structure de condition IF ... THEN ... ELSE. Exemple : IF A=1

"ELSE" ?

Attend l'entrée de ELSE dans la structure de condition IF ... THEN ... ELSE

caractere

Attend l'entrée d'un caractère. Exemple : PRINT '

2eme "'" ?

Attend l'entrée d'un 2ème '. Exemple : ?'A

"STEP" ?

Attend l'entrée de STEP dans une boucle FOR ... TO ... STEP ... NEXT.

ANNEXE 4

TABLE D'EQUIVALENCE
DECIMAL / HEXADECIMAL / BINAIRE / ASCII

Rappel :

PRINT HEX$(N) donne l'équivalent hexadécimal du nombre N.

PRINT BIN$(N) donne l'équivalent binaire de N.

PRINT N donne l'équivalent décimal de N.

PRINT ASC("CAR") donne le code ASCII du caractère CAR.

PRINT 'CAR' donne le code ASCII du caractère CAR.

PRINT CHR$(N) donne le caractère équivalent à N.

DECIM	HEXA	BINAIRE	ASCII	DECIM	HEXA	BINAIRE	ASCII
000	#00	%00000000	NUL	001	#01	%00000001	SQH
002	#02	%00000010	STX	003	#03	%00000011	ETX
004	#04	%00000100	EOT	005	#05	%00000101	ENQ
006	#06	%00000110	ACK	007	#07	%00000111	BEL
008	#08	%00001000	BS	009	#09	%00001001	HT
010	#0A	%00001010	LF	011	#0B	%00001011	VT
012	#0C	%00001100	FF	013	#0D	%00001101	CR
014	#0E	%00001110	SO	015	#0F	%00001111	SI
016	#10	%00010000	DLE	017	#11	%00010001	DC1
018	#12	%00010010	DC2	019	#13	%00010011	DC3
020	#14	%00010100	DC4	021	#15	%00010101	NAK
022	#16	%00010110	SYN	023	#17	%00010111	ETB
024	#18	%00011000	CAN	025	#19	%00011001	EM
026	#1A	%00011010	SUB	027	#1B	%00011011	ESC
028	#1C	%00011100	FS	029	#1D	%00011101	GS
030	#1E	%00011110	RS	031	#1F	%00011111	US
032	#20	%00100000		033	#21	%00100001	!
034	#22	%00100010	"	035	#23	%00100011	#
036	#24	%00100100	$	037	#25	%00100101	%
038	#26	%00100110	&	039	#27	%00100111	'
040	#28	%00101000	(	041	#29	%00101001	)
042	#2A	%00101010	*	043	#2B	%00101011	+
044	#2C	%00101100	,	045	#2D	%00101101	-
046	#2E	%00101110	.	047	#2F	%00101111	/
048	#30	%00110000	0	049	#31	%00110001	1
050	#32	%00110010	2	051	#33	%00110011	3
052	#34	%00110100	4	053	#35	%00110101	5
054	#36	%00110110	6	055	#37	%00110111	7
056	#38	%00111000	8	057	#39	%00111001	9
058	#3A	%00111010	:	059	#3B	%00111011	;

060	#3C	%00111100	<	061	#3D	%00111101	=
062	#3E	%00111110	>	063	#3F	%00111111	?
064	#40	%01000000	@ à	065	#41	%01000001	A
066	#42	%01000010	B	067	#43	%01000011	C
068	#44	%01000100	D	069	#45	%01000101	E
070	#46	%01000110	F	071	#47	%01000111	G
072	#48	%01001000	H	073	#49	%01001001	I
074	#4A	%01001010	J	075	#4B	%01001011	K
076	#4C	%01001100	L	077	#4D	%01001101	M
078	#4E	%01001110	N	079	#4F	%01001111	O
080	#50	%01010000	P	081	#51	%01010001	Q
082	#52	%01010010	R	083	#53	%01010011	S
084	#54	%01010100	T	085	#55	%01010101	U
086	#56	%01010110	V	087	#57	%01010111	W
088	#58	%01011000	X	089	#59	%01011001	Y
090	#5A	%01011010	Z	091	#5B	%01011011	[ä
092	#5C	%01011100	\ ç	093	#5D	%01011101	]
094	#5E	%01011110	^	095	#5F	%01011111	£
096	#60	%01100000	© ù	097	#61	%01100001	a
098	#62	%01100010	b	099	#63	%01100011	c
100	#64	%01100100	d	101	#65	%01100101	e
102	#66	%01100110	f	103	#67	%01100111	g
104	#68	%01101000	h	105	#69	%01101001	i
106	#6A	%01101010	j	107	#6B	%01101011	k
108	#6C	%01101100	l	109	#6D	%01101101	m
110	#6E	%01101110	n	111	#6F	%01101111	o
112	#70	%01110000	p	113	#71	%01110001	q
114	#72	%01110010	r	115	#73	%01110011	s
116	#74	%01110100	t	117	#75	%01110101	u
118	#76	%01110110	v	119	#77	%01110111	w
120	#78	%01111000	x	121	#79	%01111001	y
122	#7A	%01111010	z	123	#7B	%01111011	{ é
124	#7C	%01111100	| à	125	#7D	%01111101	} è
126	#7E	%01111110	ä	127	#7F	%01111111	
128	#80	%10000000		129	#81	%10000001	
130	#82	%10000010		131	#83	%10000011	
132	#84	%10000100		133	#85	%10000101	
134	#86	%10000110		135	#87	%10000111	
136	#88	%10001000		137	#89	%10001001	
138	#8A	%10001010		139	#8B	%10001011	
140	#8C	%10001100		141	#8D	%10001101	
142	#8E	%10001110		143	#8F	%10001111	
144	#90	%10010000		145	#91	%10010001	
146	#92	%10010010		147	#93	%10010011	
148	#94	%10010100		149	#95	%10010101	
150	#96	%10010110		151	#97	%10010111	
152	#98	%10011000		153	#99	%10011001	
154	#9A	%10011010		155	#9B	%10011011	
156	#9C	%10011100		157	#9D	%10011101	
158	#9E	%10011110		159	#9F	%10011111	
160	#A0	%10100000		161	#A1	%10100001	
162	#A2	%10100010		163	#A3	%10100011	
164	#A4	%10100100		165	#A5	%10100101	
166	#A6	%10100110		167	#A7	%10100111	
168	#A8	%10101000		169	#A9	%10101001	
170	#AA	%10101010		171	#AB	%10101011	
172	#AC	%10101100		173	#AD	%10101101	
174	#AE	%10101110		175	#AF	%10101111	
176	#B0	%10110000		177	#B1	%10110001	

178	#B2	%10110010	179	#B3	%10110011
180	#B4	%10110100	181	#B5	%10110101
182	#B6	%10110110	183	#B7	%10110111
184	#B8	%10111000	185	#B9	%10111001
186	#BA	%10111010	187	#BB	%10111011
188	#BC	%10111100	189	#BD	%10111101
190	#BE	%10111110	191	#BF	%10111111
192	#C0	%11000000	193	#C1	%11000001
194	#C2	%11000010	195	#C3	%11000011
196	#C4	%11000100	197	#C5	%11000101
198	#C6	%11000110	199	#C7	%11000111
200	#C8	%11001000	201	#C9	%11001001
202	#CA	%11001010	203	#CB	%11001011
204	#CC	%11001100	205	#CD	%11001101
206	#CE	%11001110	207	#CF	%11001111
208	#D0	%11010000	209	#D1	%11010001
210	#D2	%11010010	211	#D3	%11010011
212	#D4	%11010100	213	#D5	%11010101
214	#D6	%11010110	215	#D7	%11010111
216	#D8	%11011000	217	#D9	%11011001
218	#DA	%11011010	219	#DB	%11011011
220	#DC	%11011100	221	#DD	%11011101
222	#DE	%11011110	223	#DF	%11011111
224	#E0	%11100000	225	#E1	%11100001
226	#E2	%11100010	227	#E3	%11100011
228	#E4	%11100100	229	#E5	%11100101
230	#E6	%11100110	231	#E7	%11100111
232	#E8	%11101000	233	#E9	%11101001
234	#EA	%11101010	235	#EB	%11101011
236	#EC	%11101100	237	#ED	%11101101
238	#EE	%11101110	239	#EF	%11101111
240	#F0	%11110000	241	#F1	%11110001
242	#F2	%11110010	243	#F3	%11110011
244	#F4	%11110100	245	#F5	%11110101
246	#F6	%11110110	247	#F7	%11110111
248	#F8	%11111000	249	#F9	%11111001
250	#FA	%11111010	251	#FB	%11111011
252	#FC	%11111100	253	#FD	%11111101
254	#FE	%11111110	255	#FF	%11111111

ANNEXE 5

CARTE MEMOIRE DU TELESTRAT

0000 005F	Variables systèmes TELEMON
0060 008B	Variables systèmes TELEMON, usages spécifiques (flottant...)
008C 00CF	Variables systèmes HYPERBASIC ou langage
00D0 00FF	Variables systèmes TELEMATIC ou applications
0100 01FF	Pile 6502
0200 0300	Variables système moniteur
0300 032F	E/S internes
0330 037F	E/S extension ORIC
0380 03FF	E/S utilisateur
0400 04FF	Routines moniteur
0500 055F	Variables systèmes STRATSED
0560 058F	Variables systèmes HYPERBASIC ou langage
0590 05FF	Buffer éditeur de texte
0600 07FF	Buffers, piles ... pour langage application
0800 97FF	(HIRES) Zone utilisateur
0800 B3FF	(TEXT) Zone utilisateur
9800 9FFF	(HIRES) caractères
A000 BFDF	(HIRES) Ecran
B400 BB7F	(TEXT) caractères
BB80 BFDF	(TEXT) Ecran
C000 FFFF	Banque 0 Ram interne
C000 FFFF	Banque 4-7 port cartouche gauche (Reset sur le 7)
C000 FFFF	Banque 1-4 port cartouche droit

ANNEXE 6

TABLEAU DES ENTREES/SORTIES DU TELESTRAT

0 à 7 = Entrées :

0 Clavier, Joystick, Souris
2 MINITEL en entrée
3 RS232 en entrée

8 à 20 = Sorties :

8 Ecran fenêtre 0
9 Ecran fenêtre 1
10 Ecran fenêtre 2
11 Ecran fenêtre 3
14 Sortie Parallèle CENTRONICS
15 MINITEL en sortie
16 RS232 en sortie
17 Ecran en mode émulation VIDEOTEX

ANNEXE 7

CONTENU DE LA DISQUETTE STRATSED 2.0 FOURNIE AVEC LE TELESTRAT

ANIBAL.BAS : vous déplacez une balle avec les 4 touches flèchées. La balle rebondit sur les côtés de l'écran. Le but de ce programme est de montrer la rapidité de l'HYPERBASIC. Ceux qui possédaient un ORIC 1 ou un ATMOS verront tout de suite la différence de vitesse !

CARRE.BAS : programme de démonstration de sous-programme.

DCOUNT.BAS : exécute une boucle 10000 fois en 1/10 ème de seconde. L'horloge en témoigne !

DEVIN.BAS : le but du jeu est de deviner une lettre tirée au hasard entre A et Z.

DEVIN2.BAS : pareil que DEVIN, mais il faut cette fois-ci découvrir un nombre tiré au hasard entre 1 et 100.

DIA.BAS : programme de simulation d'un dialogue local entre un MINITEL et le TELESTRAT.

MDIA.BAS : pareil que DIA.BAS, mais par l'intermédiaire de la ligne téléphonique.

DTELE.BAS : exemple de téléchargement d'un fichier.

TRANSPA.BAS : exemple de transmission d'un fichier.

RECEPA.BAS : exemple de réception d'un fichier.

STPAGE : ce programme demande le nom d'une page MINITEL (ne pas entrer l'extension .VDT) et l' affiche sur l'écran du MINITEL. Des renseignements sur cette page vous sont ensuite donnés :

- Les coordonnées X,Y du début de la page.

- Les coordonnées X,Y de la fin de la page.

- Les coordonnées X,Y où se trouve le curseur.

- Effacement préalable ou non de l'écran.

- Affichage du curseur ou non.

- Adresse de début de la page en mémoire.

- Longueur de la page en nombre d'octets.

TRI.BAS : programme de tri alphanumérique. Affiche et classe en une vingtaine de secondes 51 chaînes de 4 caractères chacune.

TRIFAST.BAS : programme de tri rapide. Tri 51 chaînes de 4 caractères chacune en 3 secondes !

TRINUM.BAS : programme de tri numérique. Tri 51 nombres en une vingtaine de secondes.

SLANT.CHS : caractères penchés.

BOLD.CHS : caractères gras.

SQUARE.CHS : caractères fins.

STD.CHS : caractères standards.

CREATE.COM et **CREATE25.COM** : programme de création des champs d'un fichier à accès direct. Entrez le nom du fichier à créer/lire. Entrez ensuite le nom des champs (avec $ à la fin du nom pour un champ alphanumérique). Appuyez sur RETURN. La taille des champs alphanumériques est définissable, tandis que celle des champs numériques est toujours de 5 caractères. CTRL S sauvegarde le fichier, CTRL L lit le fichier. La touche ESC permet de sortir de CREATE.

BUILD.COM : permet de créer des fichiers séquentiels à partir du clavier. Entrez d'abord le nom du fichier à créer, puis le texte que contiendra ce fichier. Arrêtez le programme par ESC.

TYPEBUILD.COM : visualise le fichier créé par l'utilitaire BUILD.COM.

TYPE.COM : visualise le contenu d'un fichier séquentiel. Entrez le nom du fichier. Le défilement des informations s'interrompt momentanément et reprend par l'appui sur une touche.

K7DSK.COM : programme de transfert d'un fichier ATMOS sur disquette TELESTRAT.

BASEQ.COM : programme de transfert d'un fichier BASIC ATMOS dans un fichier séquentiel.

SEQHYP.COM : convertit un programme BASIC ATMOS stocké dans un fichier séquentiel en un programme HYPERBASIC.

Les autres programmes de la disquette STRATSED ayant une extension .VDT ou .SRV font partie du serveur de démonstration DEMO.SRV présent sur la disquette.

ANNEXE 8

TABLE DES CODES ESCAPE ET FUNCT

En appuyant sur ESC ou FUNCT + une touche donnée, on obtient directement un attribut à l'écran. Ces attributs sont directement intégrables dans des lignes BASIC. On obtient le même résultat en tapant PRINT CHR$(27) suivi du caractère entre guillemets.

ESCAPE @	0	ENCRE NOIRE
ESCAPE A	1	ENCRE ROUGE
ESCAPE B	2	ENCRE VERTE
ESCAPE C	3	ENCRE JAUNE
ESCAPE D	4	ENCRE BLEUE
ESCAPE E	5	ENCRE MAGENTA
ESCAPE F	6	ENCRE CYAN
ESCAPE G	7	ENCRE BLANCHE
ESCAPE H	8	TEXTE STANDARD
ESCAPE I	9	TEXTE ALTERNE
ESCAPE J	10	HAUTEUR DOUBLE (STANDARD)
ESCAPE K	11	HAUTEUR DOUBLE (GRAPHIQUE)
ESCAPE L	12	CLIGNOTEMENT (STANDARD)
ESCAPE M	13	CLIGNOTEMENT (ALTERNE)
ESCAPE N	14	CLIGNOTEMENT HAUTEUR DOUBLE STANDARD
ESCAPE O	15	CLIGNOTEMENT HAUTEUR DOUBLE ALTERNE
ESCAPE P	16	PAPIER NOIR
ESCAPE Q	17	PAPIER ROUGE
ESCAPE R	18	PAPIER VERT
ESCAPE S	19	PAPIER JAUNE
ESCAPE T	20	PAPIER BLEU
ESCAPE U	21	PAPIER MAGENTA
ESCAPE V	22	PAPIER CYAN
ESCAPE W	23	PAPIER BLANC

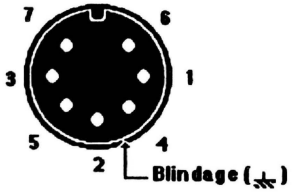
La couleur du papier à l'écran s'obtient aussi par :

FUNCT 0	PAPIER NOIR
FUNCT 1	PAPIER ROUGE
FUNCT 2	PAPIER VERT
FUNCT 3	PAPIER JAUNE
FUNCT 4	PAPIER BLEU
FUNCT 5	PAPIER MAGENTA
FUNCT 6	PAPIER CYAN
FUNCT 7	PAPIER BLANC

ANNEXE 9

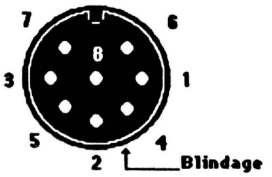
BROCHAGE DES CONNECTEURS

PRISE CASSETTE:



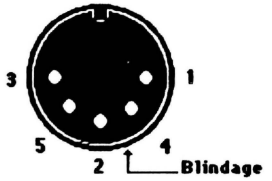
1. Sortie des données
2. Masse
3. Entrée des données
4 - 5 : Son (reliées à l'intérieur)
6 - 7 : Relai télécommande cassette
Blindage = Masse

PRISE MINITEL:



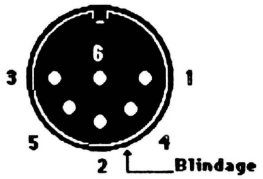
1. Non connecté
2. Sortie des données
3. Masse
4. N.C.
5. Entrée des données
6, 7, 8 : N.C.
Blindage=Masse

PRISE M.I.D.I:



1. CA1
2. + 5 V.
3. PA3
4. CB2
5. PA 6
Blindage=Masse

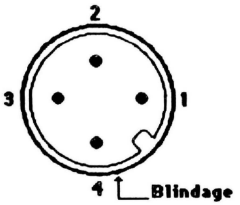
PRISE R.V.B. (pour télé ou moniteur couleur):



1. Rouge (Blanc)
2. Vert (Orange)
3. Bleu (Vert)
4. Synchronisation (Bleu)
5. Son (Jaune)
6. + 12 V (Commnunication Péritel) (Rouge)

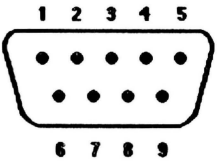
Blindage = Masse

PRISE D'ALIMENTATION:



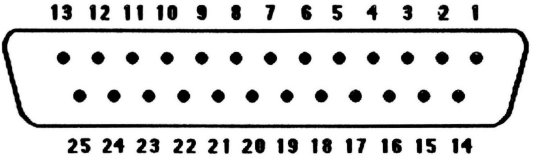
1. + 12 V.
2. + 5 V.
3. - 12 V.
4. Masse

PRISE JOYSTICK:



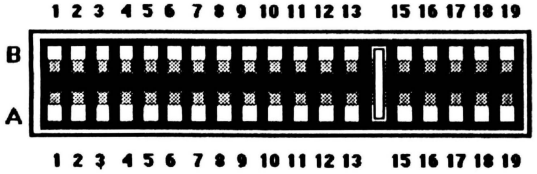
1. ⇑
2. ⇓
3. ⇐
4. ⇒
5. Souris bouton n°3
6. Souris bouton n°1
7. + 5 V.
8. Masse
9. Souris bouton n°2

PRISE RS 232:



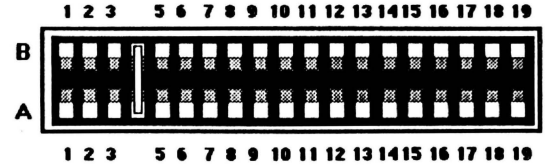
2. Données émises (TX)
3. Données reçues (RX)
4. Demande d'émission (RTS)
5. Prêt pour émission (CTS)
7. Masse signaux
8. Détection de porteuse (DCD)
20. Terminal prêt (DTR)

PORT CARTOUCHE APPLICATION (droite):



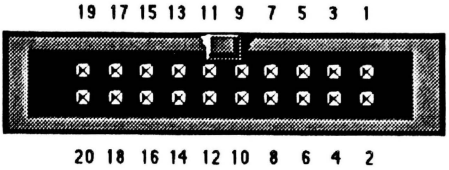
A1	CS5	**A2**	CS6	**A3**	CS3	**A4**	N.C.	**A5**	DØ
A6	D2	**A7**	D4	**A8**	D6	**A9**	Masse	**A10**	AØ
A11	A2	**A12**	A4	**A13**	A6	**A14**	Détrompeur	**A15**	A8
A16	A10	**A17**	A12	**A18**	N.C.	**A19**	+ 5 V.		
B1	Ø2	**B2**	CS4	**B3**	RW	**B4**	N.C.	**B5**	D1
B6	D3	**B7**	D5	**B8**	D7	**B9**	N.C.	**B10**	A1
B11	A3	**B12**	A5	**B13**	A7	**B4**	Détrompeur	**B15**	A9
B16	A11	**B17**	A13	**B18**	N.C.	**B19**	N.C.		

PORT CARTOUCHE LANGAGE (gauche)



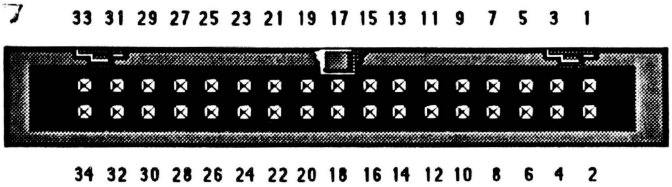
A1	CS1	**A2**	CS1	**A3**	CS3	**A4**	Détrompeur	**A5**	DØ
A6	D2	**A7**	D4	**A8**	D6	**A9**	Masse	**A10**	AØ
A11	A2	**A12**	A4	**A13**	A6	**A14**	N.C.	**A15**	A8
A16	A10	**A17**	A12	**A18**	N.C.	**A19**	+ 5 V.		
B1	Ø2	**B2**	CSØ	**B3**	N.C.	**B4**	Détrompeur	**B5**	D1
B6	D3	**B7**	D5	**B8**	D7	**B9**	N.C.	**B10**	A1
B11	A3	**B12**	A5	**B13**	A7	**B4**	N.C.	**B15**	A9
B16	A11	**B17**	A13	**B18**	N.C.	**B19**	+ 5 V.		

PRISE IMPRIMANTE PARALLELE:



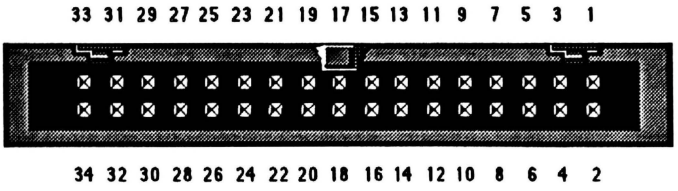
3.	DØ	1.	Strobe ($\overline{STR}$)
5.	D1	19.	Acknoledge ($\overline{ACK}$)
7.	D2	2.4.6...20.	toutes bornes paires reliées à la masse
9.	D3		
11.	D4		
13.	D5		
15.	D6		
17.	D7		

PORT D'EXTENSION:



1.	MAP	**2.**	ROMDIS	**3.**	Ø2	**4.**	RESET	**5.**	I/0
6.	N.C.	**7.**	R/W	**8.**	IRQ	**9.**	D2	**10.**	DØ
11.	A3	**12.**	D1	**13.**	A0	**14.**	D6	**15.**	A1
16.	D3	**17.**	A2	**18.**	D4	**19.**	D5	**20.**	A4
21.	A5	**22.**	D7	**23.**	A6	**24.**	A15	**25.**	A7
26.	A14	**27.**	A8	**28.**	A13	**29.**	A9	**30.**	A12
31.	A1Ø	**32.**	A11	**33.**	+5 V.	**34.**	Masse		

PORT DISQUETTES:



6.	DS3	**8.**	Index	**10.**	DS0	**12.**	DS1	**14.**	DS2
16.	Motor ON	**18.**	DIR	**20.**	STEP	**22.**	W Data	**24.**	W Gate
26.	TRACK 00	**28.**	WPROT	**30.**	R Data	**32.**	Side Select		

2,4,34 : Non connecté **1,3,5,... 33.** : Masse