AUTO FORMATION SUR hector HRX

hector PROGRAMME

AUTO FORMATION SUR

hector HRX

SOMMAIRE

Vous allez suivre une AUTO-FORMATION en six leçons pour vous permettre de manipuler et programmer en BASIC sur HECTOR HRX.

La composition de chacune des six leçons est détaillée dans ce fascicule dans l'ordre chronologique de l'étude.

Le BASIC 3X est le langage de programmation BASIC spécialement adapté sur HECTOR HRX.

Chaque leçon de l'AUTO-FORMATION a été élaborée grâce à BASIC 3X. Cela nous a permis une totale concordance entre les listings affichés et les exécutions correspondantes et vous permettra d'apprécier les possibilités du langage BASIC 3X (les listings de certains passages de programmes sont édités en dernières pages et illustrent l'effet de quelques mots BASIC).

L'ensemble AUTO-FORMATION sur HRX se compose de 3 cassettes dont chaque face supporte une leçon (1/2, 3/4 et 5/6). Chaque leçon représentant près de 30 K.octets de BASIC, le chargement peut atteindre 2 minutes. Le message à l'écran qui accompagne le chargement et qui précise les numéros de cassette et leçon vous permettra de juger du bon chargement du programme.

BASIC 3X sur HECTOR HRX passe par une pratique assidue et progressive à laquelle le plus doué d'entre nous ne peut se soustraire.

L'AUTO-FORMATION sur HRX ne peut et ne doit constituer que le début de votre découverte du langage BASIC 3X.

PRESENTATION D'UN MOT:

EN ILLUSTRATION DU CHAPITRE PRESENTATION DES LECONS.

MOT: **INIT & VOLUME** PAGE: **268**

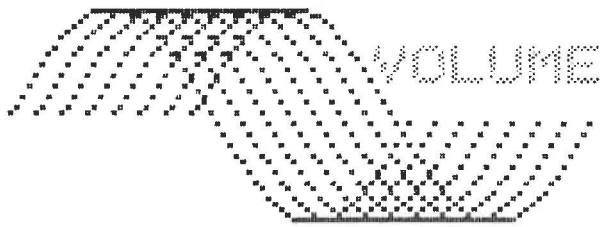
FRANCAIS: INIT & VOLUME

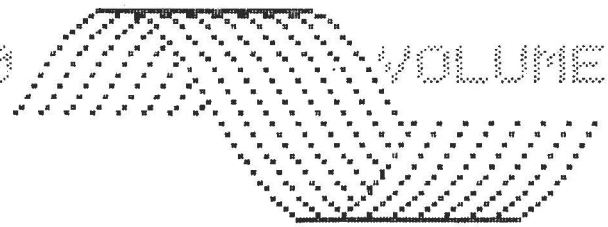
NATURE: INSTRUCTION

MODE: DIRECT ET INDIRECT

SYNTAXE: VOLUME (drapeau):INIT

```
10 VOLUME 1:INIT
20 FOR A=10 TO 40 STEP4
30 FOR U=0 TO PI(2) STEP.1:Z=Z+1
40 PLOT Z+A,50+30*SIN(U)
50 NEXT:NEXT
```





ALORS QUE .. APRES MODIFICATIONS

TOUS LES DESSINS QUI ILLUSTRENT CE MANUEL SONT DES COPIES D'ECRAN IMPRIMES SUR PAPIER GRACE AU PROGRAMME 'COPIE D'ECRAN B3X'.

1/ CHARGEMENT PRÉALABLE DU BASIC 3X

Le BASIC 3X ayant servi de support de création à chacune des leçons d'auto-formation, son chargement préalable est impératif.

3 hypothèses sont à envisager.

a) Vous possédez une cassette sur laquelle est gravé BASIC 3X

- BRANCHER et allumer HECTOR
- OUVRIR la platine du magnéto-cassette à l'aide de la TOUCHE STOP-EJECT
- INSERER la cassette BASIC 3X dans la platine, fermer le couvercle et enfoncer la touche LECTURE.

Des points de vue Electrique et Mécanique, le lecteur est prêt à lire la cassette introduite.

- APPUYER sur la touche 2 qui, dans le texte du menu proposé par HECTOR, désigne la lecture d'une cassette.

Le moteur tourne, le programme se charge ...

b) Vous possédez une cartouche

- Connecter la cartouche au connecteur qui se trouve à la gauche de l'appareil
- Brancher et allumer HECTOR

- Appuyer sur la touche 3 , qui, dans le texte du menu proposé par HECTOR, désigne la lecture d'une cartouche.

Le langage de programmation BASIC 3X se charge ....

c) Vous possédez une unité de disquette et un disque système contenant BASIC 3X

- Connecter l'unité de disque à votre HECTOR HRX par l'intermédiaire du cable plat
- Brancher et allumer HECTOR et DISC II
- Introduire dans le lecteur de gauche le disque système BASIC 3X
- Appuyer sur la touche 4, qui dans le texte du menu proposé par HECTOR HRX, désigne la lecture d'une disquette.
- Après l'affichage des messages de titres et versions du CP/m ,taper le mot B3X.

Le langage de programmation BASIC 3X se charge ...

Quels que soient le support du langage de programmation BASIC 3X et le mode de chargement, l'apparition du message ...

```
BASIC 3X      3.n
COPYRIGHT 1984 par MICRONIQUE, FRANCE
31741 Octets libres
OK
```

... indique la disponibilité de l'interpréteur BASIC 3X.

2/ CHARGEMENT D'UNE CASSETTE AUTO-FORMATION SUR HRX

- Placer la cassette dans le lecteur comme nous l'avons détaillé pour le chargement de BASIC 3X en cassette
- Au clavier, tapez le mot LOAD qui provoque la lecture d'un programme sur cassette.

Le programme une fois entré commence seul.

1/ MENU

Les 5 leçons (2 à 6 : DICTIONNAIRE ILLUSTRÉ) qui constituent la seconde partie d'AUTO-FORMATION sur HRX, se présentent toutes de la même façon.

La première page se résume à l'affichage de tous les mots qui seront détaillés dans la leçon, le dernier étant le mot "tous" destiné à reprendre tous les autres de façon continue.

La sélection du mot désiré peut se faire soit par commande du clavier, soit par manipulation du contrôleur à main, les touches B, H, G et D provoquant respectivement un déplacement du curseur vers les BAS, HAUT, GAUCHE et DROITE de l'écran et imitant à ce titre les quatre directions disponibles sur le contrôleur à main.

Le déplacement du curseur se traduit par l'affichage en rouge (initialement jaune) du numéro d'ordre du mot dans le menu.

La validation du mot, une fois repéré, est obtenue par l'appui de la touche ⏎ au clavier ou du bouton poussoir du contrôleur.

Le mot validé est alors détaillé comme nous le verrons dans le chapitre suivant.

Un mot sélectionné dès le menu implique un retour à ce même menu.

La position du mot déjà étudié est prise en compte et chaque retour au menu replace le curseur à l'endroit qu'il avait quitté.

La couleur du mot initialement blanche est alors jaune de façon à ce que les mots restants soient mis en évidence.

L'appui de la touche R vous permettra de rembobiner la cassette après lecture, la touche S permettant elle d'arrêter le moteur.

2/ PRÉSENTATION DU MOT ÉTUDIÉ

Elle est uniforme quel que soit le mot considéré.

Sont indiqués en première page le libellé du mot (pour orthographe) la page de "DICTIONNAIRE DES BASIC" qui lui fait référence, sa traduction en français, sa syntaxe ainsi que les modes de programmation admissibles (DIRECT, INDIRECT ou les deux).

Tous ces renseignements restent visibles en permanence en haut de page car le développement des explications s'opère dans un écran partiel.

Certains mots trop liés pour être dissociés sont développés simultanément. Le numéro de page indique alors celle qui, du dictionnaire, illustre le plus important de tous.

3/ DÉROULEMENT

Le déroulement des explications d'un mot est interrompu de très nombreuses fois dans l'attente d'une confirmation.

Cet arrêt s'illustre d'une petite animation représentant un doigt appuyant sur une touche, car c'est en fait ce que vous aurez à faire pour poursuivre.

Toutes les touches sont entendues par HECTOR comme une autorisation de continuer. Seule la touche * provoque un retour au menu.

Cette obligation d'appuyer sur une touche et donc de rester devant le clavier pourrait paraître ennuyeuse si nous n'avions pensé à étendre les possibilités de répondre au contrôleur à main. Le bouton poussoir permet également de confirmer à HECTOR que la page d'explication a été assimilée et qu'il peut poursuivre.

Dans le même esprit, rappelons que les deux touches ⇧ (ou SHIFT) suppriment pauses et délais et qu'à ce titre elles peuvent aider à la compréhension en accélérant les passages connus ou déjà vus. Le flux d'explications est donc soumis au bon vouloir de chaque utilisateur dont la disponibilité est toute circonstancielle.

L'étude de chaque mot se termine par la question ...

VOULEZ-VOUS REVOIR LE MOT ...?

... à laquelle la touche Ø du clavier et la direction GAUCHE répondent OUI alors que la touche 1 et la direction DROITE répondent NON.

4/ EXERCICES D'APPLICATION

La plupart des mots étudiés sont illustrés ou d'exemples ou d'exercices.

Il existe deux types d'exercices qui se différencient par la façon d'y répondre.

Si à la question posée il suffit de répondre en donnant une des solutions proposées par HECTOR, il s'agit alors du premier type d'exercices dans lequel aucune aide n'est apportée à l'utilisateur.

Le deuxième type d'exercices auxquels les réponses doivent être données directement sans choix possible s'accompagne d'une possibilité d'aide.

L'appui de la touche A (AIDE) inscrit la réponse à la place de l'utilisateur qui peut donc, quand l'exercice entraîne plusieurs réponses, ne s'attacher qu'à détailler un point précis de l'étude ou plus simplement éviter les exercices dans le cas d'une révision par exemple.

Ainsi, à la suite du mot COLOR, la question suivante est posée ...

Pour obtenir COLOR NOIR, BLANC, ROUGE, VERT
il faut taper COLOR ?
... à laquelle on peut répondre 0, 7, 1, 2 ou 0, A, 1, 2
si l'on a oublié que 7 représente la couleur blanche.

5/ NOTATION

Si le mot est suivi d'exercices, ils sont au nombre de 5, le tout étant noté sur 20.

Chaque recours à l'aide et bien sûr chaque mauvaise réponse enlève un point.

Le premier exercice est noté sur quatre points et autorise quatre mauvaises réponses auquel cas la note reste bloquée à seize sur vingt même si une cinquième ou une sixième vient s'ajouter aux premières.

Le second exercice est noté sur quatre points auxquels on ajoute éventuellement les points gagnés lors du premier.

En conclusion, on ne peut perdre au cours d'un exercice que quatre points et éventuellement ceux que l'on avait réussi à gagner lors des exercices précédents.

Si par exemple, au premier exercice, deux mauvaises réponses ont précédé la bonne, vous serez noté 18/20; mais le prochain exercice met en jeu les quatre points qui lui sont propres et les deux que vous avez gagné précédemment.

LEÇON 1 : CLAVIER ET ÉDITEUR DE LIGNE

Les commandes d'HECTOR se résument à 3 types de touches :

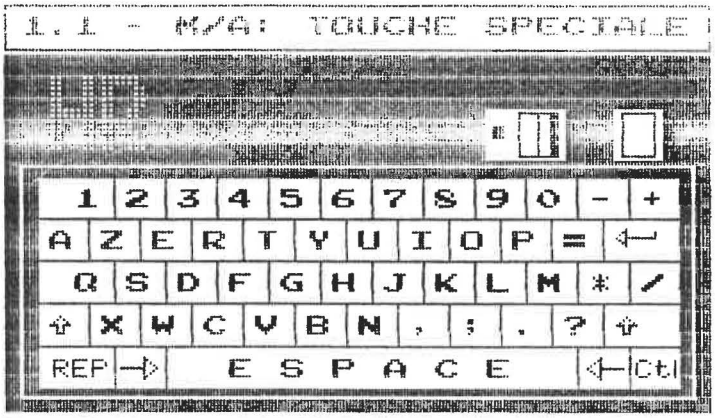
1/ TOUCHES SPÉCIALES (OU ÉLECTRIQUES)

2/ TOUCHES CASSETTES (OU MÉCANIQUES)

3/ TOUCHES CLAVIER

1/ TOUCHES SPÉCIALES

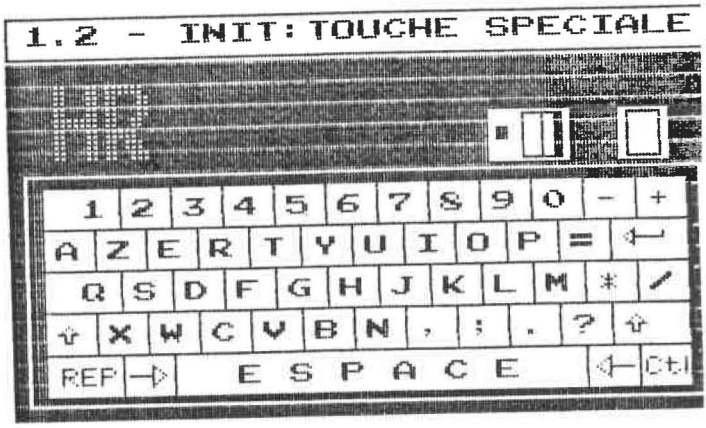
Ce sont des touches dont l'effet est uniquement électrique.



L'interruption de marche arrêt établit ou coupe l'alimentation.

La consommation du HECTOR HRX est d'environ 20 Watts (moteur du magnéto cassette bloqué).

Le bouton poussoir INIT (ou RESET ou RAZ) permet le retour au menu proposé par le HRX. Son emploi ne détruit aucune information contenue dans l'appareil et restitue les programmes dans l'état où ils étaient.



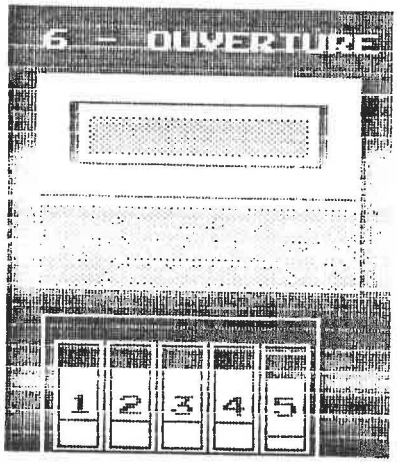
2/ TOUCHES CASSETTES

Des 5 touches disponibles sur le magnéto cassette, 2 seulement s'accompagnent d'effets électriques.

Il s'agit de LECTURE et ÉCRITURE qui soumettent la tête de lecture de la platine aux circuits d'entrée ou de sortie de HECTOR.

2.1 - TOUCHE CASSETTE

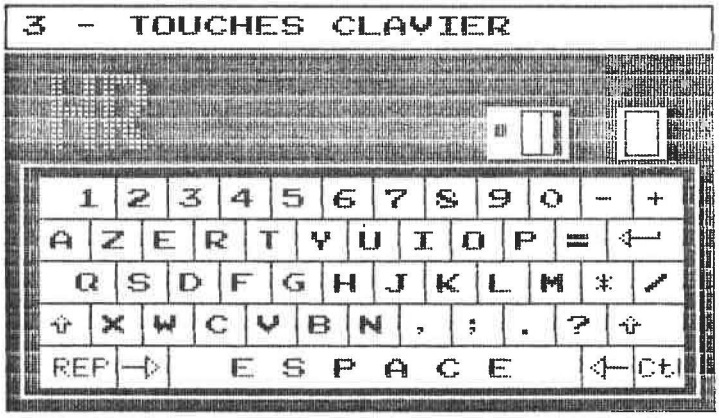
L'OUVERTURE DE LA PLATINE CASSETTE S'OBTIENT EN ENFONCANT LA TOUCHE 5 UNE SECONDE FOIS.



Les instructions BASIC de lecture, écriture ou rembobinage doivent être complétées des actions sur les touches correspondantes.

3/ TOUCHES DE CLAVIER

Pour définir le clavier on distingue 2 familles de touches dans lesquelles 2 groupes seront encore déterminés.



3.1. Touche directe

Il s'agit là de la première famille. Elle regroupe toutes les touches qui génèrent à elles seules caractères ou fonctions. Cette dernière distinction établie les 2 groupes dont nous parlions plus haut.

3.1.1. Touche directe simple

L'appui sur une touche directe simple produit 1 caractère. C'est le cas pour :

- les 26 lettres de l'alphabet (A à Z)
- les 10 chiffres (0 à 9)
- les 4 caractères de ponctuation (, ; . ?)
- les 5 opérations arithmétiques (—, +, *, /, =)
- l'espace ou séparateur.

3.1.2. Touche directe de fonction

L'appui sur une touche directe de fonction produit 1 fonction. C'est le cas pour :

- ↵ (ou RETURN) qui confirme les commandes à HECTOR.
- ⇧ (ou SHIFT) qui permet d'accéder aux caractères situés dans la partie supérieure des cabochons ainsi qu'aux minuscules.
- REP (ou LOCK) qui répète la dernière touche appuyée.
- → (ou TAB) qui déplace d'un cran sur la droite le curseur

- ⟵ (ou BACK SPACE) qui déplace le curseur d'un cran sur la gauche
- CTL (ou CONTROL) qui permet de générer diverses fonctions en association avec d'autres touches.

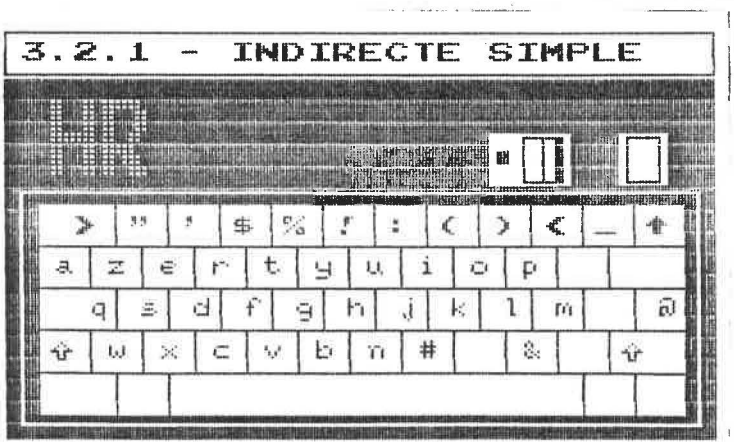
3.2. Touche indirecte

Cette deuxième famille regroupe toutes les touches qui nécessitent l'appui d'une seconde touche (dite touche d'accompagnement) pour produire caractères ou fonctions.

3.2.1. Touche indirecte simple

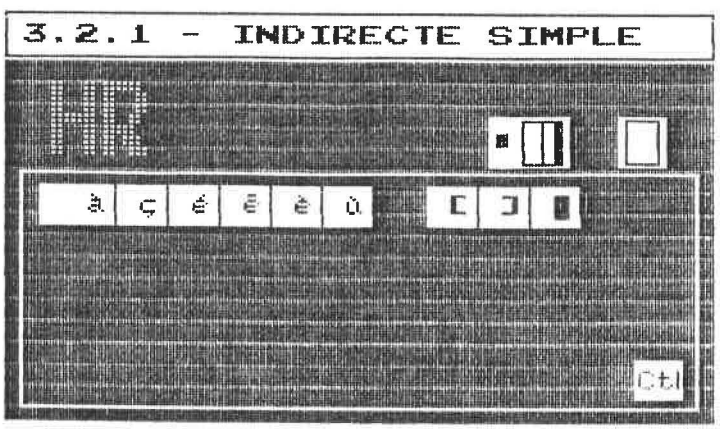
Les touches d'accompagnement sont au nombre de 2.

Touche indirecte simple générée par ⇧ (SHIFT)



La touche SHIFT permet presque d'attribuer à chaque touche un second caractère.

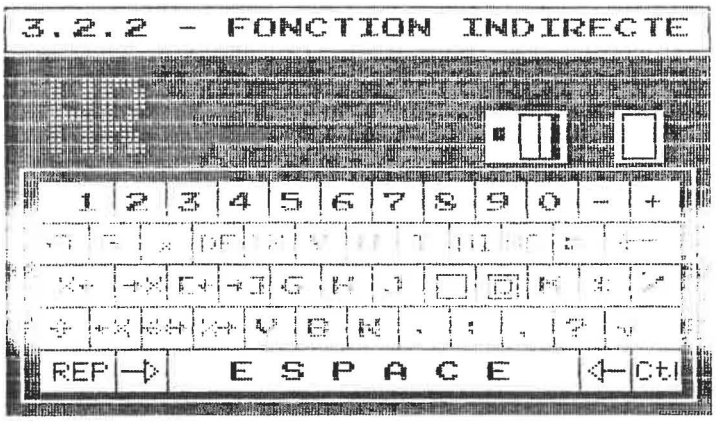
Touche indirecte simple générée par CTL (CONTROL)



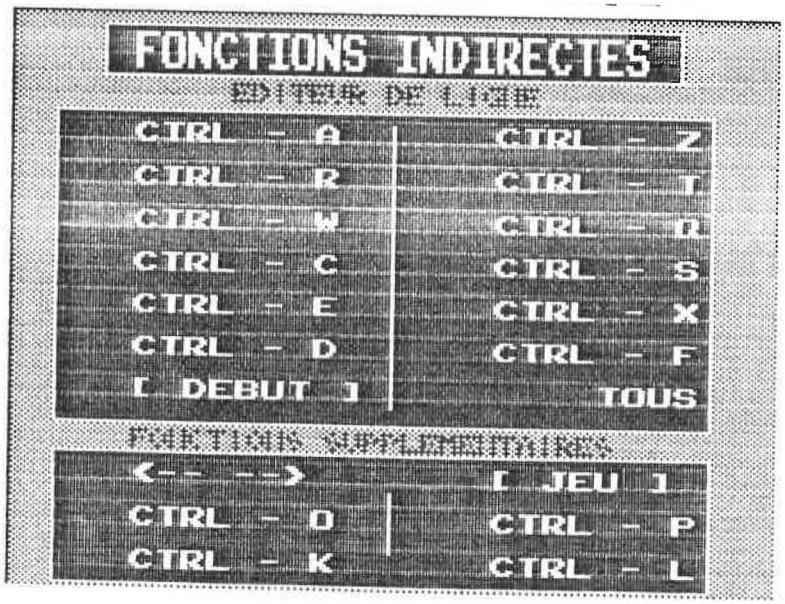
La touche contrôle permet de générer les minuscules accentuées ainsi que 3 autres caractères particuliers ([ ] ■)

3.2.2. Touche indirecte de fonction

La touche contrôle est la seule à pouvoir engendrer des fonctions. Ces dernières correspondent pour la plupart à des fonctions d'éditeur de ligne c'est-à-dire qu'elles ont pour but essentiel d'offrir à l'utilisateur des moyens rapides et performants pour éditer et modifier une ligne de programmation.



Les effets des 16 touches de fonctions sont illustrés dans un petit exemple de modification d'une ligne et détaillés à partir d'un tableau reprenant un à un les rôles des touches indirectes de fonctions.



La sélection de la fonction étudiée s'opére en 2 temps.

- Pointer le curseur devant le mot à l'aide du contrôleur à main ou des touches B, H, D et G qui génèrent respectivement les déplacements BAS, HAUT, DROITE et GAUCHE du curseur.
- Valider le choix en appuyant ou sur le bouton poussoir de contrôleur ou sur la touche ↵ du clavier.

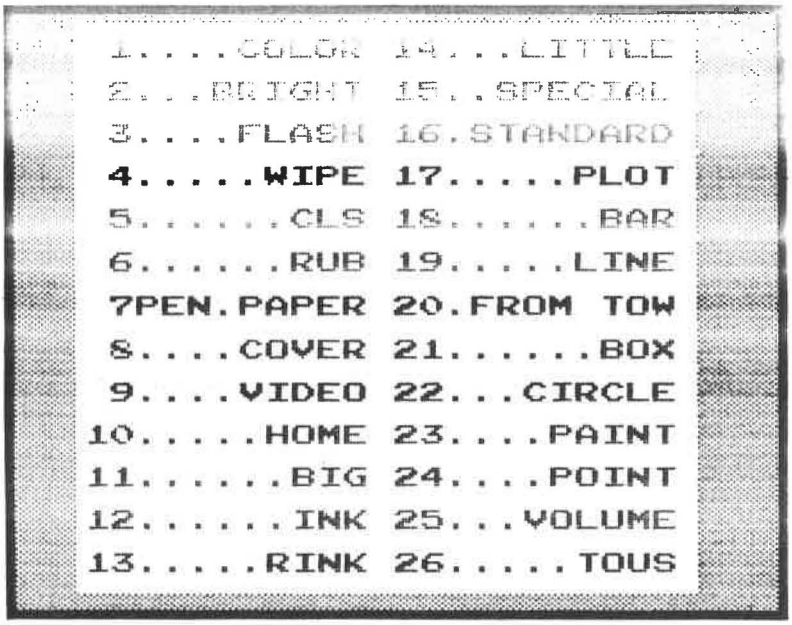
N.B. Bien que les touches ◁— et —▷ ne soient pas des touches indirectes de fonctions, leurs rôles sont tout de même illustrés dans le chapitre éditeur de ligne car elles contribuent en grande partie à rendre efficace les touches indirectes de fonction en plaçant le curseur à l'endroit désiré.

JEU : MÉMO-CLAVIER

Pour ceux qui découvriraient le clavier d'HECTOR, un petit jeu leur est proposé pour les entraîner à lire le clavier avec les mains et non plus avec les yeux.

Une série de lettres s'affiche en haut de l'écran. Un à un les caractères commencent à descendre. Vous devez les en empêcher en appuyant sur la touche correspondante. Très rapidement vous devez taper sur la touche sans la voir pour concentrer sur l'écran toute votre attention visuelle.

COULEURS
ET
GRAPHISME



CASSETTE NUMERO : 1

LECON NUMERO : 2

1/ COLOR

L'instruction COLOR est celle autour de laquelle s'articule toutes les autres relatives à la couleur.

Elle permet, parmi les 8 couleurs de base de HECTOR, de sélectionner les 4 qui seront à l'écran.

Ainsi pour sélectionner les couleurs BLEU, NOIR, VERT et ROUGE, il faudra écrire COLOR 4, 0, 2, 1
où Ø désigne le noir, 2 le vert, 1 le rouge et 4 le bleu.

Le tableau ci-dessous vous aidera à retrouver vos couleurs.

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

Les quatre couleurs une fois sélectionnées sont désignées non plus par le numéro ci-dessus qui leur est propre mais par le rang qu'elles occupent dans la palette.

Ainsi dans le COLOR ci-dessus le BLEU est toujours bien la quatrième couleur de HECTOR mais c'est aussi la couleur de rang Ø de la palette. C'est en tant que couleur numéro Ø que le bleu sera désormais entendu.

(N.B. : Certains programmeurs ont pris pour habitude d'associer à ce numéro de rang une fonction particulière. Ainsi la couleur de rang Ø est considérée comme la couleur de fond d'écran. Ces conventions facilitent les attributions de couleurs aux textes et autres dessins).

2/ BRIGHT

L'instruction BRIGHT n'est active que sur la couleur numéro 2. Elle atténue son intensité et multiplie ainsi par 2 le registre des couleurs d'HECTOR.

3/ FLASH n, x

L'instruction FLASH déclenche x fois l'alternance de la couleur n et de son complément à 7.

Ainsi COLOR 0, 1, 7, 3 : FLASH 2, 50
déclenche 50 fois le passage du BLANC (7) au NOIR (0) de la couleur de rang 2

FLASH 2, 50 peut se traduire

```
10 FOR Z = 1 TO 50
20 COLOR 0, 1, 7, 3 : COLOR 0, 1, 0, 3
30 NEXT
```

4/ WIPE

Efface la totalité de l'écran et replace le curseur en haut de l'écran.

5/ CLS

Efface les écrans partiels définis par l'instruction SCREEN.

6/ RUB

Efface la totalité de l'écran sans en détruire les pointeurs.

7/ PEN - PAPER

L'instruction PEN attribue aux caractères une couleur prélevée dans la palette COLOR.

L'instruction PAPER définit la couleur de fond sur lequel sont affichés les caractères.

La couleur définie par PAPER est aussi celle utilisée par les instructions d'effacement d'écran (WIPE, CLS et RUB).

8/ COVER n

Signifie aux instructions d'affichage de caractères (PRINT, OUTPUT ...) si elles doivent tenir compte de la couleur PAPER.

COVER Ø répond non. Le caractère seul est affiché en superposition avec l'écran (décalcomanie).

COVER 1 répond oui. Le caractère est affiché sur un rectangle de couleur PAPER qui efface l'écran à l'endroit où il est écrit.

9/ VIDEO n

Echange les couleurs PEN et PAPER. L'instruction VIDEO permet d'obtenir des affichages de texte en VIDEO normale (VIDEO Ø) et en VIDEO inverse (VIDEO 1) à l'image de positifs et négatifs de photographie.

10/ HOME

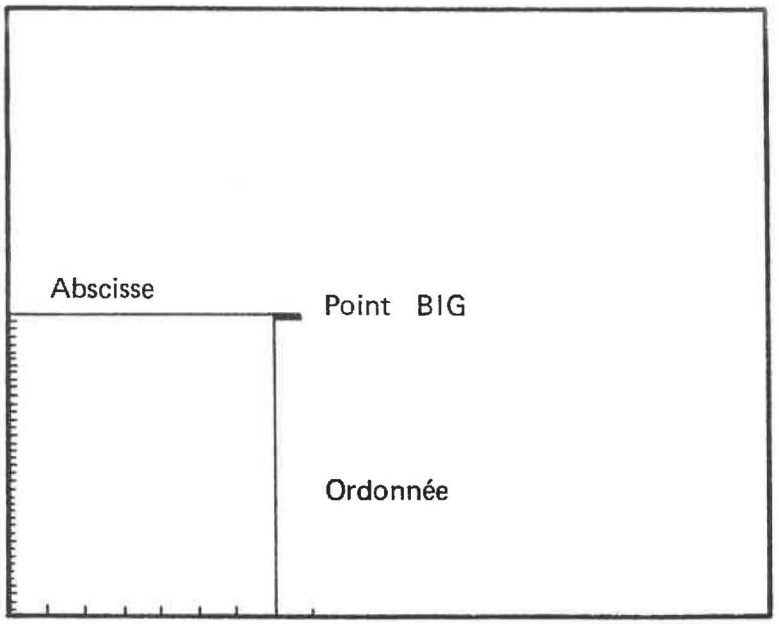
Replace le curseur en haut d'écran. En spécial, l'instruction HOME ramène à 1,1 l'échelle d'affichage de caractères.

11/ BIG

BIG est, avec LITTLE, l'un des 2 modes de résolution d'écran. La figure ci-dessous illustre le mot BIG et rappelle les instructions affectées.

RÉSOLUTION DE L'ÉCRAN EN MODE BIG

Elle est définie par le plus grand nombre de points BIG affichables à l'écran. Elle est de 228 points de hauteur par 64 de largeur.



LISTE DES INSTRUCTIONS DONT LES PARAMETRES SONT LIÉS AU MODE BIG

PLOT
BAR
LINE
FROM.TOWARDS
BOX
CIRCLE
POINT
PAINT
SCREEN

Le point BIG est appelé OCTET c'est-à-dire le point (ou la dernière donnée de façon générale) dont le traitement par HECTOR est le plus souple et le plus rapide. Il se compose en fait de 4 points LITTLE et peut donc prendre une des 256 couleurs composites que forment les différentes combinaisons.

12/ INK n

INK signifie aux instructions graphiques la couleur de la palette à utiliser. n possède 2 domaines de définition selon la résolution (BIG ou LITTLE) adoptée.

En mode LITTLE n varie de Ø à 3 (les 4 couleurs de la palette COLOR).

En mode BIG n varie de Ø à 255 (l'ensemble des possibilités de combinaisons des 4 couleurs de la palette entre-elles).

13/ RINK

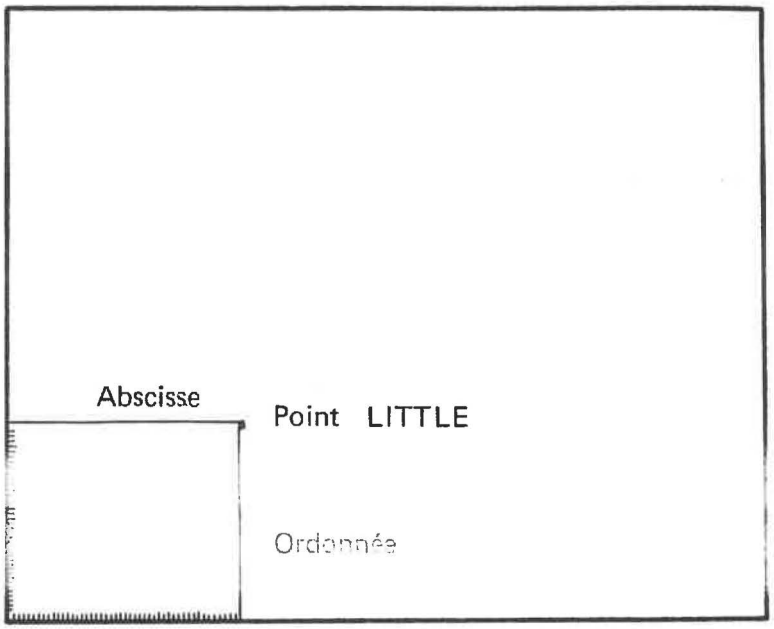
Assure la rotation d'un point BIG pour permettre une homogénéité des couleurs composites de INK.

14/ LITTLE

La figure ci-dessous illustre le mode LITTLE et rappelle les instructions affectées.

RÉSOLUTION DE L'ÉCRAN EN MODE LITTLE

Elle est définie par le plus grand nombre de points LITTLE affichables à l'écran. Elle est de 228 points de hauteur par 256 de largeur.



LISTE DES INSTRUCTION DONT LES PARAMETRES SONT LIÉS AU MODE LITTLE

PLOT
BAR
LINE
FROM.TOWARDS
BOX
CIRCLE
POINT
PAINT
SCREEN (en mode STANDARD*)

* L'instruction SCREEN bien que suivie de coordonnées LITTLE définie un écran dont chaque paramètre (position, hauteur, longueur) est arrondi au point BIG le plus proche.

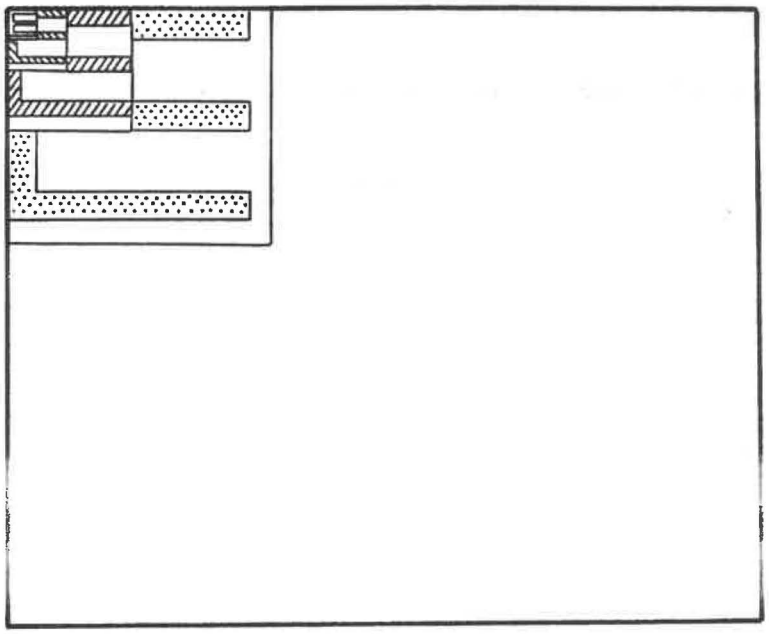
Le point LITTLE est appelé PIXEL c'est-à-dire le plus petit point affichable à l'écran. Il ne peut prendre qu'une des 8 couleurs disponibles sur HECTOR.

15/ SPECIAL

SPECIAL est, avec STANDARD, l'un des 2 modes de définition de caractère. La figure ci-dessous illustre le mode SPECIAL et rappelle les instructions affectées.

DÉFINITION DE L'ÉCRAN EN MODE SPÉCIAL

La déclaration du mode SPÉCIAL intéresse par les instructions graphiques car leurs paramètres ne sont liés qu'au choix de résolution BIG ou LITTLE seules les instructions d'édition de texte sont affectées.



LISTE DES INSTRUCTIONS DONT LES PARAMETRES SONT LIÉS AU MODE SPÉCIAL

OUTPUT
CURSOR
TAB
POS
PRINT
SCREEN

Note : Les matrices de lettres en mode SPÉCIAL sont toujours affichables en mode papier (COVER 1). Une instruction COVER Ø déclarée en mode SPÉCIAL n'aura d'effet qu'en retour au mode STANDARD.

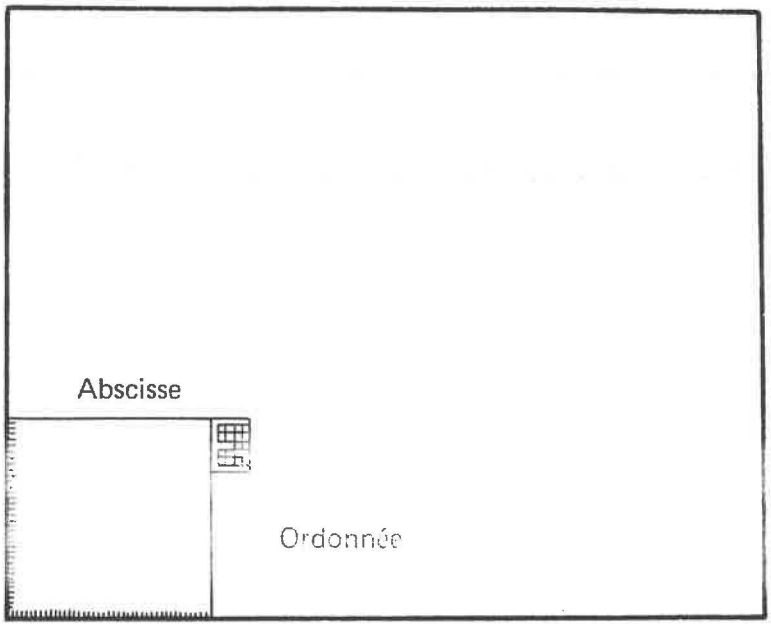
L'instruction SCALE est indissociable du mode SPÉCIAL. C'est elle qui fixe l'échelle des coordonnées. Ainsi les 4 lettres "E" affichées à l'écran le sont toutes aux coordonnées 0,0 mais avec une échelle différente.

16/ STANDARD

La figure ci-dessous illustre le mode STANDARD et rappelle les instructions affectées.

DÉFINITION DE L'ÉCRAN EN MODE STANDARD

La déclaration du mode STANDARD intéresse peu les instructions graphiques car leurs paramètres ne sont liés qu'au choix de résolution BIG ou LITTLE. Seules les instructions d'édition de texte sont affectées.



LISTE DES INSTRUCTIONS DONT LES PARAMETRES SONT LIÉS AU MODE STANDARD

OUTPUT
CURSOR
TAB
POS
PRINT

INSTRUCTIONS DE TRACES GRAPHIQUES

Elles sont au nombre de 7 : PLOT, BAR, LINE, FROM, TOWARDS, BOX et CIRCLE. Leurs paramètres qui désignent des coordonnées d'écran sont liés à la résolution adoptée (voir les résolutions BIG et LITTLE). La couleur du tracé est définie par INK.

17/ PLOT

Affiche un point.

18/ BAR

Trace une ligne horizontale.

19/ LINE

Trace une ligne entre 2 points spécifiés.

20/ FROM

Enregistre l'origine d'une ligne tracée par TOWARDS.

21/ TOWARDS

Trace une ligne entre le point désigné par FROM et celui spécifié après TOWARDS. Après exécution le dernier TOWARDS est considéré comme un FROM.

TOWARDS permet donc de tracer une ligne dont l'extrémité est le début d'une autre.

22/ BOX

Trace un rectangle plein entre 2 points opposés.

23/ CIRCLE

Trace une ellipse ou un cercle de rayons horizontaux et verticaux définis. Les figures obtenues sont constituées de points joints. Elles peuvent donc être peintes grâce à l'instruction PAINT.

24/ PAINT

Peint à partir du point spécifié jusqu'à la rencontre d'un obstacle. PAINT ne doit donc opéré qu'à l'intérieur de surfaces fermées.

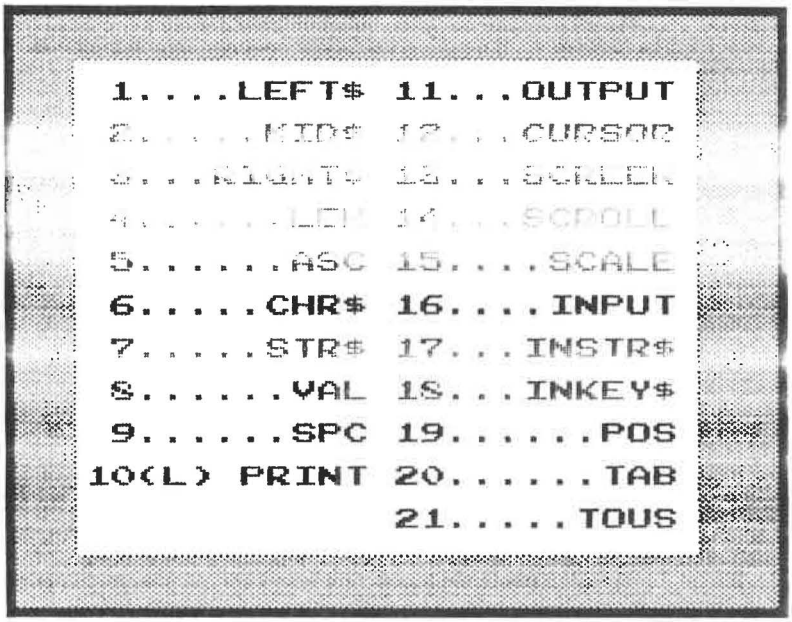
25/ POINT

Renvoie le numéro de rang de la couleur du point dont les coordonnées sont spécifiées.

26/ VOLUME & INIT

Permet de rendre aux tracés graphiques leur troisième dimension en éliminant les lignes cachées.

CHAINES



CASSETTE NUMERO : 2

LECON NUMERO : 3

1/ LEFT$

Permet d'extraire la partie gauche d'un texte.

2/ MID$

Permet d'extraire une partie d'un texte.

3/ RIGHT$

Permet d'extraire la partie droite d'un texte.

4/ LEN

Renvoie la longueur d'une chaîne de caractères.

5/ ASC

Renvoie le code ASCII du caractère spécifié.

6/ CHR$

Renvoie le caractère dont le code ASCII est spécifié.

7/ STR$

Permet de traiter un élément numérique en élément caténique pour, par exemple, le soumettre aux fonctions LEFT$, MID$, RIGHT$ ou LEN.

8/ VAL

Permet de traiter un élément caténique en élément numérique pour, par exemple, le soumettre aux opérateurs de calcul (+, —, x, :).

9/ SPC n

Remplace n espaces par une fonction beaucoup moins gourmande que l'écriture des n espaces.

10/ PRINT

Instruction d'affichage simple. L'instruction PRINT ne permet pas de positionner le texte à l'écran. Le message est affiché à l'endroit pointé par le curseur (endroit qui peut être redéfini par l'instruction CURSOR).

L'instruction PRINT supporte dans sa syntaxe des variantes :

PRINT " "; : affiche le texte spécifié et laisse le curseur pointé en fin de texte.

PRINT " ", " " : Tabule l'écran de 14 caractères en 14 caractères.

11/ OUTPUT

Affichage de texte spécifié aux coordonnées spécifiés.

12/ CURSOR

Permet de positionner le curseur avant l'affichage d'un message par PRINT.

13/ SCREEN

Permet de déclarer des écrans partiels dont les dimensions sont spécifiées. Les affichages de texte sont limités à cet écran sauf si l'on utilise l'instruction OUTPUT. L'instruction CLS permet d'effacer cet écran sans rien détruire du reste de l'écran.

14/ SCROLL

Permet de déplacer l'écran dans toutes les directions.

Les paramètres du SCROLL sont toujours entendus comme des paramètres de résolution BIG.

Les déplacements horizontaux sont des rotations car l'écran est entièrement conservé.

Les déplacements verticaux sont des décalages car les parties d'écran qui ne sont plus visibles sont détruites.

15/ SCALE

N'est opérationnelle qu'en mode SPECIAL.

de 1 à 28, l'instruction SCALE détermine la taille d'un caractère par rapport à une matrice minimale de 8 points sur 8.

16/ INPUT

Permet de saisir un texte au clavier. Les fonctions de l'éditeur de ligne y sont opérationnelles.

17/ RECTIFY

Restitue à l'écran un texte dans le but d'une modification.

18/ INSTR$ (n)

Attend la saisie de n touches au clavier. Les caractères des touches appuyées ne sont pas affichées. Le programme ne peut continuer qu'à la saisie du nombre de touches spécifié.

19/ INKEY$ (n)

Autorise n scrutations du clavier dans l'attente d'une saisie. Le programme peut continuer après les n scrutations même si aucune touche n'a été appuyée.

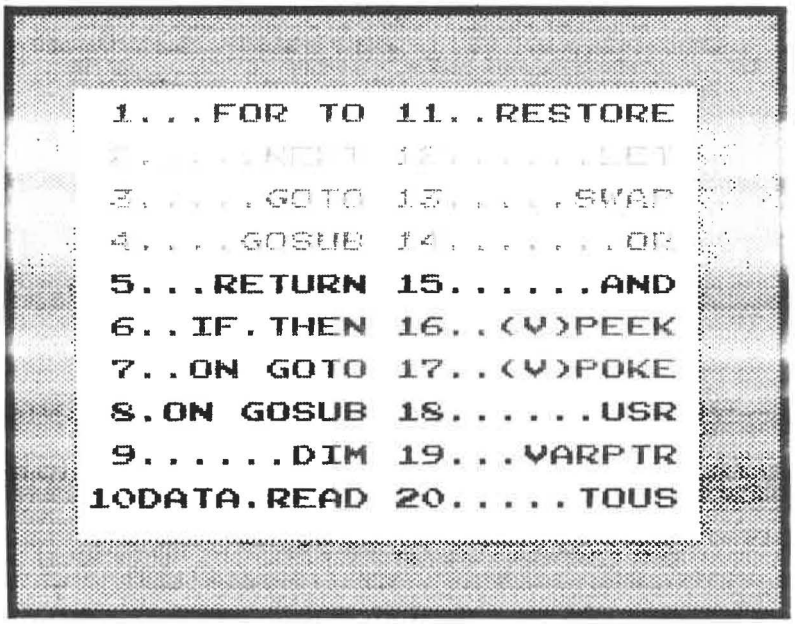
20/ POS

Renseigne sur la position du curseur. Les valeurs retournées sont propres au mode d'écriture (SPECIAL ou STANDARD) considéré.

21/ TAB

Permet, associée à l'instruction PRINT, d'afficher des textes en respectant une tabulation précise.

BOUCLES



CASSETTE NUMERO : 2

LECON NUMERO : 4

1/ FOR TO 2/ NEXT

Permet de répéter une tâche. La séquence doit s'inclure entre les instructions FOR TO et NEXT.

```
10 FOR X = 1 TO 10
20 PRINT "BONJOUR"
30 NEXT
```

La ligne 20 se répéte 10 fois, autant de fois que met X partant de 1 pour égaler 10.

Certaines séquences utilisent le compteur de Boucle pour assurer des tâches différentes.

```
10 FOR X = 0 TO 3
20 FLASH X, 50
30 NEXT
```

Les boucles peuvent s'inclure l'une dans l'autre, le premier NEXT renvoyant au dernier FOR TO.

```
10 FOR Y = 10 TO 50 STEP 10
20 FOR X = 0 TO 3
30 FLASH X, Y
40 NEXT : NEXT
```

le mot STEP signifie au compteur de Boucle de s'incrémenter à chaque passage de l'argument précisé.

3/ GOTO

Assure le branchement au numéro de ligne spécifié.

4/ GOSUB 5/ RETURN

Assure le branchement à un sous-programme dont la première ligne est spécifiée.

Le sous-programme est terminé par un RETURN qui renvoie l'exécution à l'instruction qui suit le GOSUB.

6/ IF ... THEN ELSE

Permet de soumettre à une condition des traitement différents. Si la condition est remplie, les instructions suivant le THEN sont exécutées sinon ceux sont celles qui suivent ELSE qui le sont.

7/ ON GOTO 8/ ON GOSUB

Permet d'envisager une distribution des tâches selon le contenu d'une variable.

9/ DIM

Dimensionne un tableau. Les variables indicées du type A (n) sont autorisées jusqu'à A (10). Au-dessus de 10, il faut dimensionner le tableau A (à une seule dimension), grâce à l'instruction DIM A (20) après laquelle 21 valeurs (0 à 21) sont autorisées.

Les variables indicées dont l'argument est inférieur à 10 peuvent également être dimensionnées pour éviter une perte de place.

DIM A (5) limite à 5 les valeurs admissibles.

10/ DATA READ 11/ RESTORE

L'instruction DATA permet de constituer un stock d'informations. Les informations numériques ou caténiques sont écrites à la suite de l'instruction DATA, séparées par des virgules et éventuellement encadrées de guillemets si elles contiennent elles-mêmes une virgule.

```
10 DATA BONJOUR, "3.50 F", 16, AUREVOIR
```

L'instruction READ attribue à la variable spécifiée la première information de la ligne DATA.

Dans le cas fréquent de plusieurs lignes de DATA, l'instruction RESTORE permet de signifier à READ dans quelle ligne il doit piocher.

A titre d'exemple, regardez le programme d'illustration du mot TONE.

12/ LET

L'instruction LET est facultative et permet seulement d'attribuer un contenu à une variable.

LET A = 10 peut s'écrire A = 10

13/ SWAP

Echange les contenus de 2 variables. La plupart des méthodes de tri et classement s'articule autour de l'instruction SWAP.

Parmi les plus connues on note :

La méthode BUBBLE SORT qui consiste à comparer d'abord le premier élément à tous les autres (en inversant à chaque fois que l'un de ceux-ci remplit la condition de tri mieux que le premier élément). L'opération est répétée en comparant le second élément et ainsi de suite.

Exemple : soit à classer dans l'ordre croissant les nombres A (1) à A (10)

```
10 FOR I = 1 TO 9
20 FOR J = I + 1 TO 10
30 IF A (J) < A (I) THEN SWAP A (I), A (J)
40 NEXT
50 NEXT
```

La méthode SHELL METZNER

```
10 T = 10 : P = T
20 P = INT (P/2) : IF P<1 THEN END
30 J = 1 : K = T - P
40 I = J
50 L = I + P
60 IF A (I) < A (L) THEN GOTO 90
```

```
70 SWAP A (L), A (I)
80 I = I - P : IF I < 1 THEN GOTO 90 : ELSE GOTO 50
90 J = J + 1 : IF J > K THEN GOTO 20 : ELSE GOTO 40
```

la méthode du RIPPLE amélioré dans laquelle les nombres consécutifs sont comparés et interchangés si nécessaires.

Le programme signale si il a procédé à un échange en notant une variable Z = 0.

Cette méthode peut être longue si le nombre le plus petit est le dernier.

```
10 K = 10
20 Z = Ø
30 FOR I = 1 TO K - 1
40 IF A (I+1) < A (I) THEN SWAP A (I+1), A (I) : Z = 1
50 NEXT
60 IF Z = 1 THEN K = K - 1 : GOTO 20
```

14/ OR 15/ AND

Permettent d'établir un test à 2 conditions.

OR validant le test si l'une des deux conditions est remplie. AND le validant si les 2 sont remplies.

```
10 IF A = 10 OR A = B THEN ...
```

Si A vaut 10 ou B le traitement est assuré.

```
10 IF A = 10 AND C = B THEN ...
```

Si A vaut 10 et si C vaut B le traitement est assuré.

OR et AND peuvent aussi être entendus comme des opérateurs logiques. Les opérandes sont alors obligatoirement traduit en binaire.

16/ (V)PEEK

Les 64 K de ram d'HECTOR sont découpées en 4 pages de 16 K dont deux ont les mêmes adresses.

Ces deux pages sont sélectées par une bascule et ne peuvent être obtenues simultanément.

Ces deux pages ont pour nom page vidéo et page programme. PEEK est l'instruction qui permet de lire le contenu d'une adresse en page programme. VPEEK elle, agit en page VIDEO.

Les adresses des pages communes (jusqu'à & CØØØ) peuvent être lues indifféremment par PEEK et VPEEK.

17/ (V)POKE

Permet de stocker une valeur à l'adresse spécifiée.

18/ USR

Permet d'exécuter à partir du BASIC des sous-programmes en langage machine.

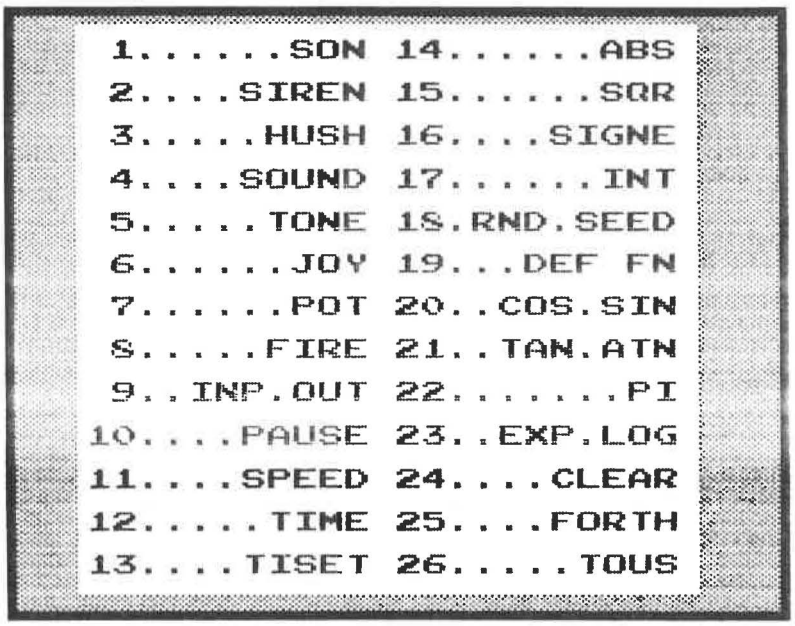
Essayez USR & 19DA, USR & 19D0, USR & C057

19/ VARPTR

Renvoie l'adresse de stockage de la variable numérique ou caténique spécifiée.

```
10 A$ = "AAAAAAA"
20 POKE VARPTR(A$),2
30 PRINT A$
```

DIVERS



CASSETTE NUMERO : 3

LECON NUMERO : 5

1/ SON

Dans ce chapitre sont regroupés 4 sons particuliers (LASER, BELL, BEEP et SHOT) dont les durées sont fixes et finies.

L'instruction SHOT n génére un bruit différent selon que n vaut Ø (Un tir d'arme à feu) ou 1 (Rafale).

2/ SIREN

L'instruction SIREN génére un bruit de SIRENE dont la durée est illimitée. Le recours à l'instruction HUSH est alors nécessaire pour arrêter le son.

3/ HUSH - 4/ SOUND

L'instruction SOUND génére tout une série de sons divers dont la plupart ont une durée illimitée. Comme pour l'instruction SIREN le recours à HUSH est alors nécessaire.

Pour arrêter un son on peut également déclarer un nouveau sound qui remplacera le précédent.

REMARQUES IMPORTANTES : HECTOR ne fait que provoquer l'émission de sons qu'il a préalablement paramétré. Dès que le son est émis, HECTOR peut retourner à l'exécution du reste du programme.

5/ TONE

TONE est un son "SOFT" c'est-à-dire qu'il est fabriqué par le micro-processeur lui-même et non pas par un circuit périphérique comme c'est le cas pour SOUND. HECTOR est donc immobilisé tout le temps que dure le tone demandé.

Certains tone correspondent par leur fréquence à des notes de musique. Voici le sous-programme qui a servi à illustrer le mot tone.

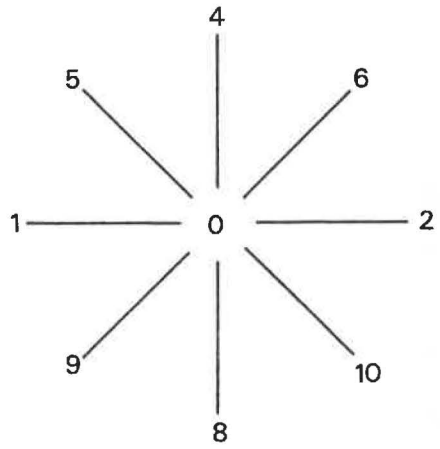
```
10 A=0
18 DATA97,24,97,12,131,24,131,12,97,36,131,36,131,24,131,12,124,24,124,12,
   124,12,110,12,124,12,131,36
20 DATA131,12,124,12,131,12,131,24,148,12,148,12,131,12,148,12,148,24,168,
   12,168,12,148,12,168,12,168,24,179,12,179,12,168,12,179,12,200,36
21 RESTORE18:A=A+1:FORI=1TO33
22 GOSUB50:NEXT
26 IFA=2THENGOTO34:ELSEGOTO18
34 DATA224,12,200,12,179,12,168,12,179,12,168,12,131,12,139,12,131,12,110,
   24,110,12,110,12,97,12,110,12,110,24,124,12,124,12,110,12,124,12,124,24,
   131,12
36 DATA224,12,200,12,179,12,168,12,179,12,168,12,131,12,139,12,131,12,110,
   24,110,12,110,12,117,12,110,12,85,12,97,12,110,12,124,12,131,12,148,12,
   168,36
37 RESTORE34:FORI=1TO42
38 GOSUB50:NEXT:END
50 READC,D
52 TONEC,D*1000/C:RETURN
```

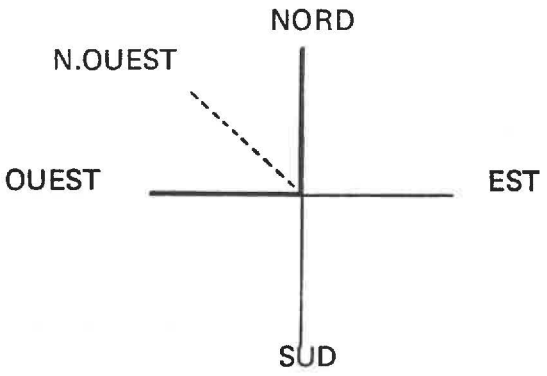
** INSTRUCTIONS DE LECTURE DES CONTROLEURS A MAIN **

Les indices (n) Ø ou 1 désignent respectivement le contrôleur gauche et celui de droite.

6/ JOY (n)

Lecture des directions du manche





Mécaniquement les contacts assurant les positions diagonales n'existent pas. Ils sont entendus comme les appuis simultanés des contacts qui les entourent.

7/ POT (n)

Lecture de la valeur du potentiomètre.

8/ FIRE (n)

Interprétation de l'état du poussoir. Si la valeur retournée est Ø le poussoir est enfoncé. Si elle vaut 1 le poussoir est relâché.

** INSTRUCTIONS D'ENTRÉE/SORTIE SUR PORT PARALLELE **

9/ INP

Renvoie la valeur de l'octet présent sur le port spécifié.

OUT

Ecrit l'octet spécifié sur le port précisé.

** INSTRUCTIONS EN RELATION AVEC L'HORLOGE DE HECTOR **

HECTOR possède une horloge interne dont la capacité est de 24 h qui s'incrémente toutes les 20 ms.

4 registres sont donc disponibles :

Celui des 50ième de seconde : TIME (Ø)
des secondes : TIME (1)
des minutes : TIME (2)
des heures : TIME (3)

10/ PAUSE (n)

L'argument n spécifie le nombre de seconde et donc la durée de la pause.

L'appui sur la touche ⇧ (ou SHIFT) annule les pauses.

11/ SPEED n

Permet de préciser la vitesse d'exécution d'un programme par l'intercalage de pause entre chaque instruction.

12/ TIME (n)

L'argument (n) spécifie le registre de l'unité de mesure du temps. Les 4 registres sont disponibles à tout moment.

13/ TISET

Remet à zéro chacun des 4 registres de l'horloge interne.

** INSTRUCTIONS ET FONCTIONS MATHEMATIQUES OU ASSIMILEES **

14/ ABS (N)

Renvoie la valeur absolue du nombre N

15/ SQR (N)

Renvoie la racine carrée positive du nombre N. Si N est négatif une erreur de code 5 (IV : Illegal value) est générée.

16/ SIGNE (N)

Renvoie un chiffre représentant le signe du nombre N. (Ce chiffre est en fait le rapport du nombre N sur sa propre valeur absolue).

17/ INT (N)

Retourne la partie entière du nombre N.

18/ RND (A, B)

Génére un nombre aléatoire selon une règle de calcul précisée par SEED.

Le domaine de définition d'une fonction RND (A, B) est [A, B[

18/ SEED n

Etablit une règle de calcul à partir de laquelle RND génére des nombres aléatoires.

(Les nombres générés par RND sont dits pseudo-aléatoires puisqu'ils sont déterminés à partir d'une règle mathématique).

19/ DEF FN

Permet de définir une fois pour toutes une fonction précise dont les variables ne seront rencontrées qu'en cours de programme.

20/ COS - SIN 21/ TAN. ATN 22/ PI (n)

Ceux sont les fonctions trigonométriques COSINUS, SINUS, TANGENTE et ARCTANGENTE dont les angles sont exprimés en radians. PI (n) est une fonction de n qui affecte à la variable précisée la valeur n x π

23/ EXP - LOG

Ceux sont les fonctions exponentielle et logarithme de base e.

24/ CLEAR D, n

Cette seule instruction a trois effets bien distincts :

1/ Remise à zéro des variables et tableaux

Si le BASIC 3X rencontre l'instruction CLEAR même sans paramètres, il remet à zéro, les variables déjà utilisées et supprime les emplacements de variables alloués par l'instruction DIM.

2/ Détermination de la taille de l'espace caténique

L'espace caténique est une zone mémoire réservée aux chaînes de caractères créées dans un programme BASIC. Cet espace caténique se remplit à l'image d'un barillet de révolver.

Toutes les chaînes de caractères y sont stockées sous références les unes derrière les autres.

(ATTENTION : Dans le cas de la redéfinition d'une même chaîne, BASIC considère en un premier temps la seconde chaîne comme une nouvelle. Ce n'est qu'après qu'il supprime la première de l'espace caténique.

Ainsi une chaîne de caractères de longueur 129 peut très bien ne pas pouvoir s'inclure dans l'espace caténique restant (et donc générer une erreur de code (OS : débordement d'espace caténique)) si la chaîne qu'elle vient remplacer est de longueur 128).

Exemple : Soit un espace caténique initialement dimensionné à 255 dont 248 octets sont déjà occupés par différentes chaînes de caractères dont A$= "BONJOUR".

Si la chaîne A$ est redéfinie A$= "AUREVOIR", le BASIC ne peut mémoriser à la fois BONJOUR et AUREVOIR sans dépasser l'espace qui lui est imparti.

Dans ce cas il n'existe qu'une seule solution, redimensionner l'espace caténique grâce à l'instruction CLEAR.

De façon générale, l'espace caténique doit pouvoir contenir l'ensemble des chaînes déclarées lors du programme (Pour deux chaînes de même nom, la plus longue est à considérer) auquel on rajoute la plus longue de toutes.

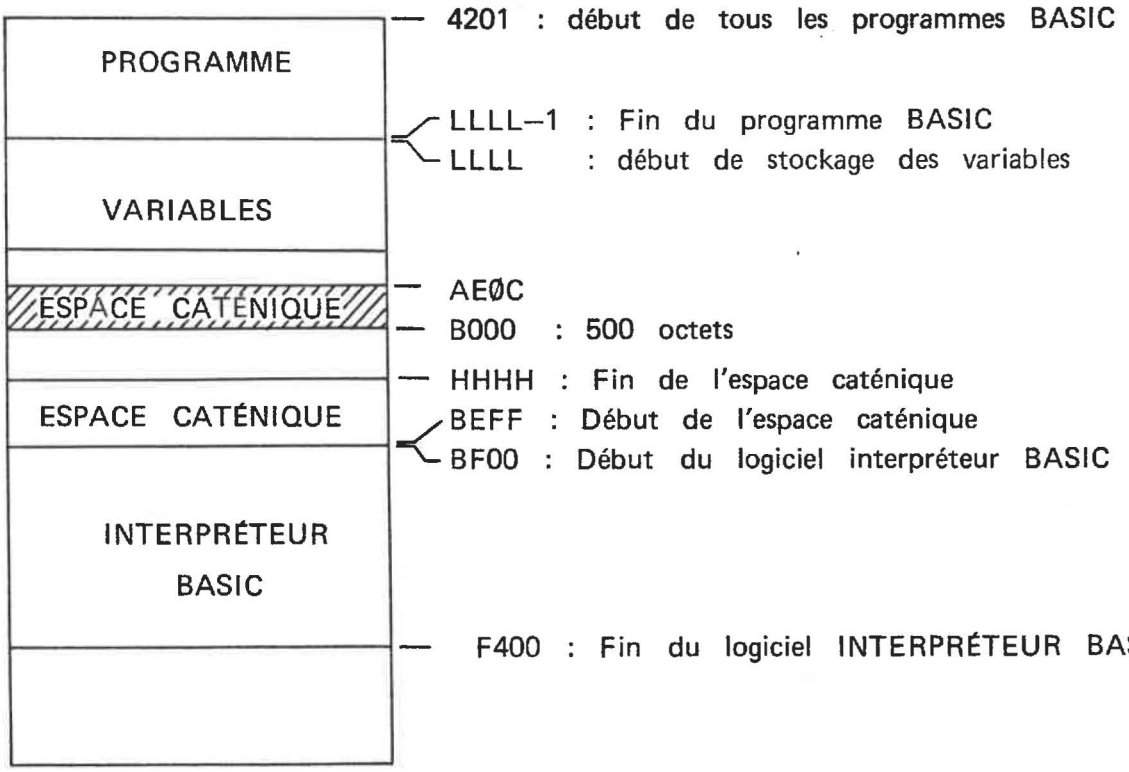
3/ Réservation d'un espace mémoire

Il est parfois nécessaire de réserver un espace dans la mémoire pour "mettre à l'abri" des programmes en langage machine ou autres tables de redéfinition de caractères.

C'est encore l'instruction CLEAR qui s'en charge en amputant l'espace utilisateur BASIC d'autant de place que vous l'aviez spécifié en argument.

REMARQUE IMPORTANTE : Les instructions RUN et EDIT provoquent un CLEAR. Lorsque vous recherchez une erreur liée à une variable, il est plus commode de lister les lignes soupçonnées que de les éditer de façon à conserver tous les paramètres disponibles dans l'état où ils étaient quand l'erreur s'est manifestée.

Pour bien comprendre le rôle de CLEAR, regardons la ventilation de l'interpréteur BASIC 3X dans la ram HECTOR.



(NOTE : Les 4 paramètres principaux de structure du programme utilisateur BASIC (42Ø1, LLLL, HHHH et BEFF) sont stockés respectivement aux adresses F720, F722, F724 et F726).

L'instruction CLEAR permet donc de dimensionner l'espace caténique et de le déplacer en le rapprochant de la zone variable de façon à créer une zone libre entre le début de l'espace caténique et celui de l'interpréteur BASIC.

Exemple : CLEAR 500, BØØØ

dimensionne et déplace l'espace caténique comme le montre la figure en créant ainsi un espace libre entre B000 et BEFF ▨

25' FORTH COLD

HECTOR HRX est doté d'un langage puissant : le FORTH.

Le BASIC 3X vous offre la possibilité d'interpeller le FORTH à partir du BASIC.

Pour cela il faudra :

- Calculer la place mémoire qui sera utilisée par le supplément au Dictionnaire FORTH que vous désirez créer.
- Amputer le BASIC d'autant de place que nécessite votre programme FORTH.
- Communiquer au FORTH l'endroit et la taille de l'espace que vous lui avez réservé.

1/ Calcul de la place mémoire

La longueur que vous accordez au DICTIONNAIRE FORTH dépend du nombre de mots, de variables ou d'images que vous voulez créer sous FORTH.

La longueur réservée à deux buffers d'édition, elle, est fixe et correspond à 674 octets en hexadécimal, c'est-à-dire 1652 octets en décimal.

Supposons que vous souhaitiez réserver 1 K octets pour le dictionnaire du forth, la place totale est donc :

Hexa : 674 + 400 = A74 H

Décimal : 1652 + 1024 = 2676

2/ Abaissement du sommet mémoire du BASIC

Il faut maintenant restreindre l'espace adressable par BASIC de manière à placer la zone de Forth au-delà. Reprenons l'exemple ci-dessus, dans lequel nous réservons 1 K pour le dictionnaire et l'espace de deux buffers d'édition. Le sommet mémoire devient :

Hexa : BFØØ – A74 = B48C H

Décimal : 48896 – 2676 = 46220

Pour le modifier il suffit donc d'écrire en mode direct ou indirect :

CLEAR 255,46220 ou CLEAR &FF,&B48C

3/ Initialisation des paramètres avec COLD

Après avoir réservé la place mémoire, il convient d'informer Forth des adresses que l'on lui a réservé.

C'est le rôle de l'instruction COLD. Le premier paramètre, qui précise l'adresse du dictionnaire, doit être l'adresse du sommet mémoire plus un.

Le deuxième paramètre, qui précise l'adresse des buffers d'édition, doit correspondre au calcul suivant :

Hexadécimal : BFØØ − 674 = B88C H

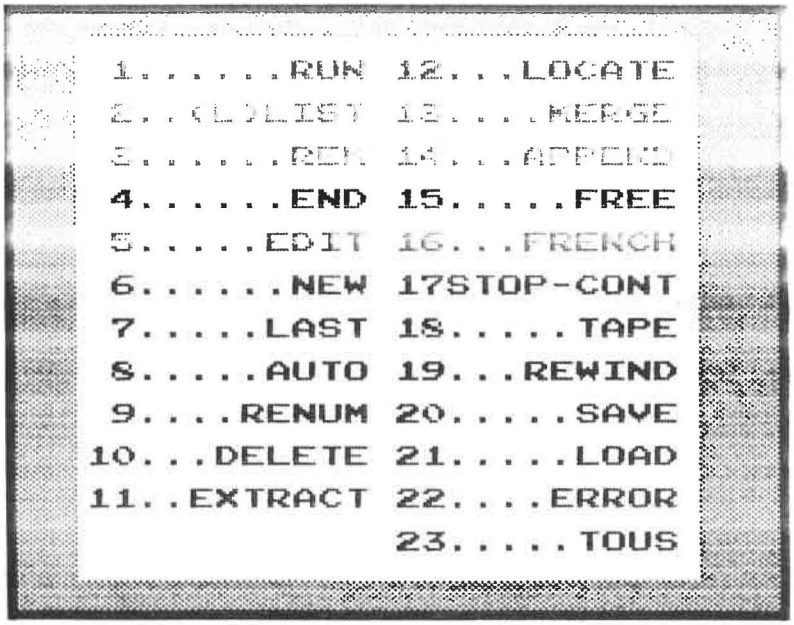
Décimal : 48896 − 1652 = 47244

Dans l'exemple évoqué, cela s'écrit :

COLD 46221,47244 ou bien COLD &B48D,&B88C

Il est indispensable que la réservation par CLEAR ait eu lieu avant l'utilisation de COLD.

EDITEUR DE PROGRAMME



CASSETTE NUMERO : 3

LECON NUMERO : 6

1/ RUN (n)

Permet l'exécution du programme à partir de la n'ième ligne ou à défaut de la première.

L'instruction RUN effectue un CLEAR implicite. Toutes les variables sont donc nulles.

2/ (L) LIST A, B

Affiche les lignes de A à B. Des nuances de syntaxe permettent plusieurs applications.

LLIST imprime sur papier les lignes spécifiées.

3/ REM ou '

Permet d'inclure dans vos programmes des lignes de commentaires. L'instruction REM peut se situer n'importe où dans la ligne : Tout ce qui suit n'est pas interprété par BASIC 3X.

4/ END

Souvent facultative, l'instruction END est parfois indispensable pour séparer programme principal et sous-programmes. Elle évite que l'interpréteur, en fin de programme, ne débouche sur des lignes de sous-programmes qui ne peuvent être appelées que par un GOSUB et dont l'exécution n'aurait chronologiquement aucun sens.

5/ EDIT n

Permet de modifier le contenu d'une ligne n, en la soumettant à l'éditeur de ligne.

L'emploi de EDIT est souvent liée à la détection par l'interpréteur d'une erreur dans une ligne.

Si cette erreur porte sur des contenus incorrects de variables, il est préférable de lister la ligne, le temps de trouver votre erreur, ce qui vous garantie le maintien des contenus de vos variables.

L'instruction EDIT remet à zéro les variables déjà initialisées.

6/ NEW

Permet d'écraser le programme en mémoire.

7/ LAST

Permet de connaître le numéro et le contenu de la dernière ligne.

8/ AUTO

Utile dans l'élaboration de vos programmes. Elle permet la numérotation des lignes que vous allez frappé.

9 / RENUM

Permet de renuméroter les lignes de programmes. L'instruction RENUM rectifie les adresses des GOTO, GOSUB, RESTORE qui auraient changé.

10/ DELETE

Permet d'effacer des parties entières de programme.

11/ EXTRACT A, B

Permet d'effacer toutes les lignes d'un programme sauf celles placées entre les lignes A et B.

12/ LOCATE

Permet de localiser une variable ou une instruction dans le programme. L'instruction LOCATE liste toutes les lignes contenant l'expression demandée.

N.B. : Il peut être pratique d'imprimer les lignes désignées par un LOCATE. Pour se fabriquer ce "LLOCATE" on procéde ainsi :

- On relève le contenu de l'adresse FEC9 en page vidéo.
- On le traduit en binaire en ne s'intéressant qu'aux 2 bits de poids faible, le bit Ø désignant l'écran, le bit 1 l'imprimante

  On place à l'adresse FEC9 le contenu prélevé en ayant adapté les 2 bits de poids faible au résultat que l'on escompte.

Ex : PRINT VPEEK (& FEC9)

& 89 ..... 1000 1001

Le bit 0 est à 1 : L'écran est donc sélectionné

VPOKE &FEC9, & 8A

Le bit 1 est à 1 : L'imprimante est donc sélectionnée.

Tapez ensuite l'instruction LOCATE suivie de l'expression recherchée. Les lignes retournées seront imprimées.

13/ MERGE

Fusionne le programme en mémoire avec celui se trouvant sur cassette. Les lignes du programme sur bande s'imbriquent à leurs places parmi celles du programme en mémoire. Une ligne du programme sur bande remplace la ligne en mémoire. qui porte le même numéro.

14/ APPEND

Ajoute à la suite du programme en mémoire le programme contenu dans la cassette que vous aurez placé dans le lecteur. Les lignes du programme sur bande sont renumérotées à partir du dernier numéro du programme en mémoire.

15/ FREE

L'instruction FREE renseigne sur l'occupation statique de la mémoire. 2 nombres sont retournés.

Le premier désigne le nombre d'octets occupés par les lignes de programme, le second le nombre d'octets disponibles.

La somme des 2 nombres est toujours égale à la place totale disponible (31999) à laquelle on retire une valeur correspondant à la taille de l'espace caténique défini par CLEAR (l'espace caténique définit par CLEAR.

Pour un CLEAR 255, la somme est toujours égale à 31744.

Les fonctions FREE (Ø) et FREE (A$) renseignent elles sur l'occupation des secteurs réservés aux variables du même type que l'indique l'argument de la fonction.

Ainsi FREE (Ø) indique le nombre d'octets réservés aux variables numériques et encore disponibles. FREE (A$) indique le nombre d'octets réservé aux variables caténiques et encore disponibles à l'intérieur de l'espace qui leur est imparti (défini par CLEAR).

Les nombres retournés par les fonctions FREE promettent d'évaluer la place dynamiquement utilisée par un programme. Leurs valeurs étant liées à l'exécution du programme, leur affichage en fin de programme ne permet de connaître la place utilisée que dans la dernière partie du programme, ce qui ne peut suffir à évaluer les disponibilités des espaces numériques et caténiques.

Une évaluation correcte doit correspondre à un ensemble de relevés, répartis tout le long du programme, dont la densité peut être multipliée dans les zones à forte consommation.

16/ FRENCH

Les mots (instructions ou fonctions) anglais sont remplacés par leurs équivalents français.

17/ STOP-CONT

Alors que STOP interrompt le programme, CONT permet de reprendre l'exécution à l'endroit de l'interruption.

18/ TAPE n

n précise dans quel état doit être le moteur du magnéto-cassette. Si n = 0 le moteur est à l'arrêt et si n = 1 il est en marche.

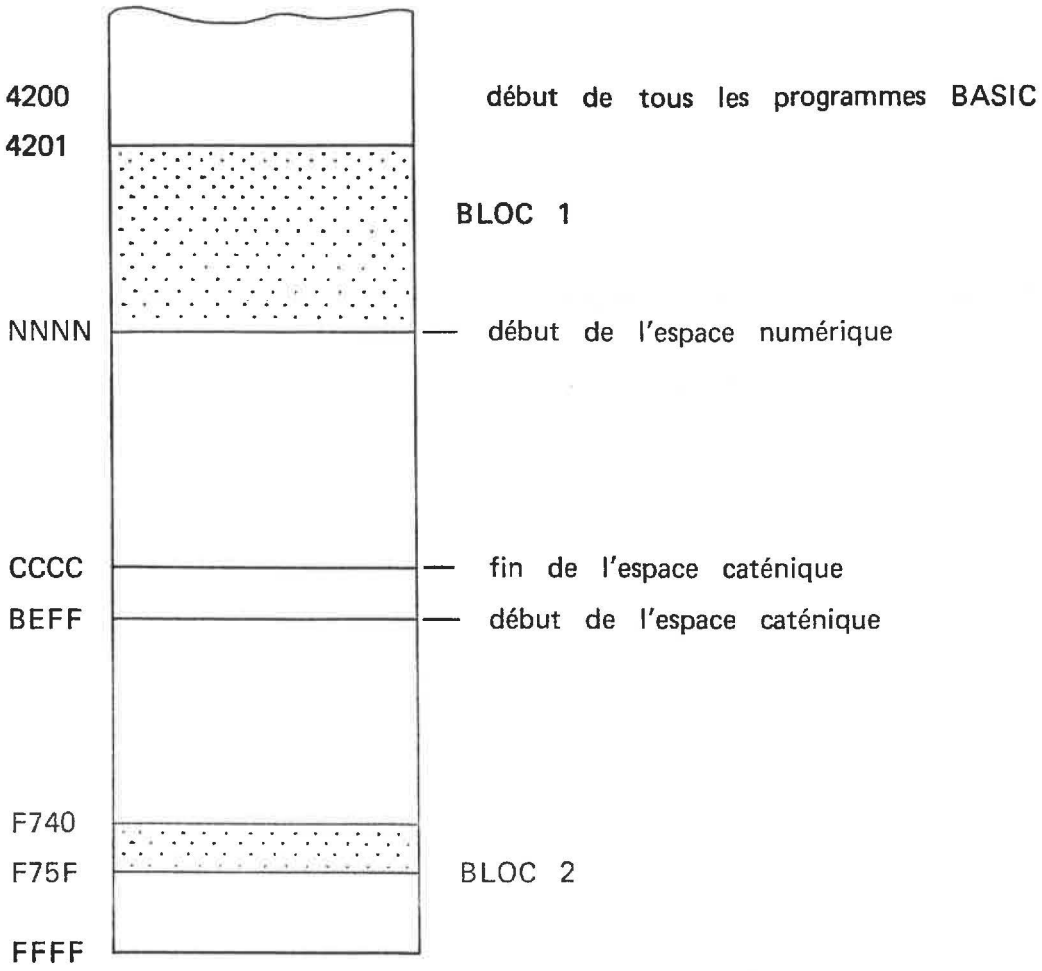
19/ REWIND

Actionne le moteur du magnéto-cassette. Contrairement à TAPE, l'instruction REWIND immobilise le programme tant qu'une touche n'a pas été appuyée lui indiquant l'arrêt du moteur. REWIND peut s'écrire :

```
10 TAPE 1
20 A$ = INSTR$ (1)
30 TAPE Ø
```

20/ SAVE

Permet la sauvegarde sur cassette du programme en mémoire. Cette sauvegarde consiste en l'écriture de 2 blocs bien distincts qui sont représentés sur la figure suivante :



BLOC 1 : Il correspond à la partie de l'espace utilisateur dans laquelle sont écrites les lignes de programme. Il s'étend de l'adresse 4201 (en hexadécimal) à l'adresse NNNN propre à chaque programme.

BLOC 2 : Il correspond à la partie de l'interpréteur BASIC 3X dans laquelle sont notées les informations suivantes :

- (F740 - F741) début du programme (initialement 4201)
- (F742 - F743) début de l'espace numérique

- (F744 - F745) fin de l'espace caténique
- (F746 - F747) début de l'espace caténique ou RAMTOP
- nom du programme
- F758 drapeau d'AUTORUN

Le BLOC 2 est physiquement écrit le premier sur la cassette de façon à faciliter le chargement d'un programme dont le nom est spécifié.

Le BLOC 2 permet en réalité de définir la configuration du programme sauvegardé. Nous avons vu que cette configuration pouvait être modifiée par l'instruction CLEAR qui place et dimensionne l'espace caténique.

Le programme chargé est alors placé dans les mêmes conditions de fonctionnement qui le régiraient avant la sauvegarde.

21/ LOAD

Permet la lecture des blocs 1 et 2 d'un programme sur cassette. Si l'instruction LOAD ne peut reconnaître en tête de chargement le bloc numéro 2, le programme sur cassette est alors considéré comme un bloc autonome et chargé aux adresses qu'il précise.

22/ ERROR n

Permet de substituer au traitement d'erreur exécuté à la rencontre de chacune d'entre elles par l'interpréteur BASIC 3X, un traitement particulier dont la première ligne est spécifiée. L'instruction CLEAR possède un témoin qui assure son exécution. Ce témoin, validé dès la rencontre de l'instruction ERROR, est invalidé par chaque traitement d'erreur. Il convient donc que l'organigramme du programme prévoit la lecture du mot ERROR après chaque traitement.

```
10 ' ILLUSTRATION DES MOTS
12 ' COLOR,PEN ET PAPER
14 '
16 ' APPARITION POINT PAR POINT
18 ' D'UN TEXTE A L'ECRAN
20 '
22 WIPE:COLOR0,1,0,7:INK2
24 SPECIAL:HOME:PEN2
26 OUTPUT"HECTOR",7,12:PEN3
28 '
30 INK3:FOR X=1 TO 1000
32 X=RND(50,110):Y=RND(122,130)
34 INKPOINT(X,Y)+1
36 IFPOINT(X,Y)=3THENGOTO42
38 IFPOINT(X,Y)=1THENGOTO42
40 PLOTX,Y
42 NEXT
```

```
10 ' ILLUSTRATION DES MOTS
12 ' LINE,BOX ET PAINT
14 '
16 ' COLORIAGE D'UN COULOIR
18 '
20 WIPE:INK3:BOX10,220,230,10
22 INK0:BOX11,219,229,11
24 '
26 INK3
28 FORY=13TO227STEP8:A=0
30 FORX=YTOY+4STEP4
32 LINEX,11+A,X,217+A:A=2
34 NEXT:NEXT
36 '
38 INK1:PAINT11,11
```

```
10 ' ILLUSTRATION DES MOTS
12 ' FROM,TOWARDS ET DEFFN
14 '
16 ' DESSIN DANS UN CERCLE
18 '
20 DEFFNB(X)=100-80*COS(X)
22 DEFFNA(X)=100-80*SIN(X)
24 '
26 WIPE:FORC=1TO4
27 FROM100,180
28 INKC:FORX=0TO500STEP3
30 TOWARDSFNA(X),FNB(X)
32 NEXT:NEXT
```

```
10 ' ILLUSTRATION DES MOTS
12 ' BIG,INK ET RINK
14 '
16 ' MOSAIQUE 64 COULEURS
18 '
19 WIPE
20 FORU=0TO15
22 FORV=0TO15
23 BIG:INKU+16*V:LITTLE
24 FORY=U*13TO(U+1)*13
27 FROM100,180
28 BAR16+V*13,16+(V+1)*13,Y
30 RINK:NEXT
32 NEXT:NEXT
```

```
10 ' ILLUSTRATION DES MOTS
12 ' VOLUME ET INIT
14 '
16 ' TOLE ONDULEE
18 '
19 WIPE:VOLUME1:INIT
20 FORD=40TO80STEP4
24 DEFFNB(X)=D+30*SIN(X)
26 FORC=.1TO20STEP.1
28 PLOTC*10+D-30,FNB(C)
32 NEXT:NEXT
```

```
10 ' ILLUSTRATION DES MOTS
12 ' MID$ ET OUTPUT
14 '
16 ' LETTRES DE CREDITS
18 '
20 WIPE:A$="CREDITS"
22 '
24 FORX=1TO7:GOSUB30:NEXT
26 X=INT(RND(1,8)):GOSUB34:GOSUB30
28 GOTO26
30 FORY=150TO50STEP-3
32 GOSUB38:NEXT:RETURN
34 FORY=50TO150STEP3
36 GOSUB38:NEXT:RETURN
38 OUTPUTMID$(A$,X,1),100+X*10,Y
40 TONEY+X*10,2
42 RETURN
```

```
10 ' ILLUSTRATION DU MOT
12 ' SCROLL
14 '
16 ' LIMITES ECRAN
18 '
20 WIPE:INK3:BOX10,10,20,1
22 '
24 FORX=1TO54:SCROLL256:NEXT
26 FORX=1TO54:SCROLL-1:NEXT
28 FORX=1TO54:SCROLL-256:NEXT
30 FORX=1TO54:SCROLL1:NEXT
```

```
10 ' ILLUSTRATION DES MOTS
12 ' SPECIAL ET SCALE
14 '
16 ' TEXTE ELASTIQUE
18 '
20 SPECIAL:HOME:WIPE
22 FORX=7TO28
24 GOSUB32:NEXT
26 FORX=28TO7STEP-1
28 GOSUB32:NEXT
30 GOTO22
32 SCALE2,X
34 OUTPUT"ELASTIQUE",1,0
36 RETURN
```

```
10 ' ILLUSTRATION DES MOTS
12 ' DATA,READ ET RESTORE
14 '
16 ' DESSIN DU DOIGT
18 ' APPUYANT SUR UNE TOUCHE
20 '
22 DATA3,15,28,59,227,65,227,224
24 DATA63,127,0,227,67,65,227,224
26 DATA0,0,0,0,0,0,0,0
28 DATA251,251,251,123,59,251,243
30 DATA0,0,0,0,0,0,0,0
32 DATA28,24,24,16,0,48,48,0
34 DATA108,108,36,72,0,0,0,0
36 DATA108,254,108,108,108,254,108
38 '
40 RESTORE22:GOSUB54:SPECIAL
42 HOME:VPOKE&FEAD,0
44 OUTPUT"!#",25,4:PAUSE.5
46 OUTPUT" #",25,4:PAUSE.5
48 IFVPEEK(&FEAD)=0THENGOTO42
50 RESTORE30:GOSUB54:END
52 '
54 FORX=&F 00TO&F91E:READ U
56 VPOKEX,U:NEXT:RETURN
```

```
10 'ILLUSTRATION DES MOTS
12 'POINT ET PLOT
14 '
16 'DUPLICATION D'UN DESSIN
18 '
20 WIPE:SPECIAL:HOME
22 PEN2:OUTPUT"TEXTE RECOPIE",3,14
24 INK2:BAR26,127,105
26 '
28 FOR X=20 TO 130
30 FOR Y=104 TO 114
32 IF POINT(X,Y)=2 THEN PLOT X,Y+50:PLOT250-X,Y+50
34 NEXT:NEXT
```

```
10 ' ILLUSTRATION DES MOTS
12 ' POKE ET USR
14 '
16 ' DEFILEMENT D'UN BANDEAU
18 ' EN BAS DE L'ECRAN
20 '
22 WIPE
24 DATA243              :' DI
26 DATA223              :' RST  18H
28 DATA33,0,246 :'DEBUT    LD   HL,F600H
30 DATA17,0,246         :' LD   DE,F600H
32 DATA6,9              :' LD   B,09H
34 DATA197       :'STO     PUSH BC
36 DATA126              :' LD   A,(HL)
38 DATA35               :' INC  HL
40 DATA1,63,0           :' LD   BC,0039H
42 DATA237,176          :' LDIR
44 DATA18               :' LD   A,(DE)
46 DATA19               :' INC  DE
48 DATA193              :' POP  BC
50 DATA16,243           :' DJNZ STO
52 DATA1,211,12         :' LD   BC,0CD3H
54 DATA205,246,7        :' CALL 07F6H
56 DATA1,1,0            :' LD   BC,0001H
58 DATA58,0,56    :'ST1    LD   A,(3800H)
60 DATA60               :' INC  A
62 DATA192              :' RET  Z
64 DATA11               :' DEC  BC
66 DATA120              :' LD   A,B
68 DATA177              :' OR   C
70 DATA32,246           :' JR   NZ,ST1
72 DATA195,2,64         :' JP DEBUT
74 '
76 RESTORE24:FORX=1TO45
78 READA:POKE&3FFF+X,A:NEXT
80 '
82 OUTPUT"ILLUSTRATION DES MOTS POKE ET USR",20,10
84 '
86 FOR X=1 TO 100
88 USR&4000
90 NEXT
```

```
10 ' ILLUSTRATION DU MOT
12 ' SAVE
14 '
16 ' SAUVEGARDE D'UN PROGRAMME
18 ' AVEC BANIERE
20 '
22 CLEAR255,&B000
24 DATA0,0,16,0,0,0,0
26 DATA255,64,247,32,0,32,247
28 DATA254,0,8,1,0,0,0
30 DATA254,0,192,0,57,0,0
32 DATA251,192,219,0,3,0,192
34 DATA254,8,8,1,0,0,0
36 DATA255,1,66,0,5,1,66
38 DATA253,0,0,0,0,0,0
40 '
42 RESTORE24:FORX=1TO56
44 READA:POKE&AFFF+X,A:NEXT
46 '
48 DATA221,33,0,176
50 DATA62,1
52 DATA205,40,3
54 DATA201
56 '
58 RESTORE48:FORX=1TO10
60 READA:POKE&B0FF+X,A:NEXT
62 '
64 WIPE:PEN3
66 OUTPUT"ESSAI BANIERE",80,228
68 POKE&B02D,PEEK(&F722)
70 POKE&B02E,PEEK(&F723)-PEEK(&F721)+1
72 USR&B100
```


silicium