ordinateur portable

HX-20

MANUEL DE REFERENCE BASIC

EPSON

AVIS

* La reproduction partielle de ce manuel, sous quelque forme que ce soit, sans autorisation expresse écrite d'EPSON, est absolument interdite.
* Le contenu de ce manuel peut être modifié sans avis préalable.
* Tous les efforts ont été faits pour garantir la précision du contenu du présent manuel. Toutefois, au cas ou le lecteur découvrirait quelques erreurs, EPSON apprécierait beaucoup d'en être informé.
* Malgré cela, EPSON ne peut assumer aucune responsabilité au sujet des erreurs qui pourraient subsister dans ce manuel ou de leurs conséquences.

© Copyright 1982 par EPSON CORPORATION.
Nagano, Japon

PREFACE

L'ordinateur portable EPSON HX-20, est le plus perfectionné des ordinateurs personnels - C'est un ordinateur personnel complet qui se pose sur une table et qui est miniaturisé au point de tenir dans une mallette. Les caractéristiques matérielles et logicielles présentées ci-dessous ont été incorporées à l'ordinateur de façon à tirer profit au maximum de ses dimensions.

Ce manuel explique en détail le BASIC développé uniquement par EPSON, et nous souhaitons que l'utilisateur le considère comme une référence pour la programmation aussi bien que pour le développement d'un logiciel pour de futures applications. C'est pourquoi nous avons édité et préparé ce manuel en désirant qu'il soit facile à consulter.

NOTE:
Le clavier du HX-20 varie selon le pays où il est acheté. Dans l'Annexe K de ce Manuel les explications sont données pour les utilisateurs d'un clavier français.

CARACTERISTIQUES

1. Les RAM sont protégées par des batteries, de sorte que les programmes et les données stockées dans les RAM ne risquent pas d'être perdus même lorsque le commutateur général est tourné sur OFF. L'exécution du programme est possible dès la mise sous tension.
2. L'espace mémoire est divisé en cinq zones de programmes capables chacune de stocker un programme BASIC distinct. Chaque programme peut être sélectionné à partir du menu et être exécuté immédiatement.
3. Les cinq programmes du menu sont gérés indépendamment les uns des autres, de telle sorte que la création d'un nouveau programme ou l'édition d'un programme existant n'affecte pas les autres programmes de la mémoire.
4. L'ordinateur HX-20 est équipé d'un écran à cristaux liquides (LCD) de 80 caractères (quatre lignes de 20 caractères chacune). L'utilisateur a la possibilité de spécifier un écran virtuel plus grand que l'écran LCD, au moyen d'une commande WIDTH (largeur) et de décaler librement l'écran dans l'une des quatre directions de façon à procéder à des opérations d'édition sur un écran interne allant au-delà des dimensions matérielles de l'écran LCD.
5. Toutes les opérations effectuées sur des unités à microcassettes, en option, peuvent être contrôlées par le BASIC.
6. En plus des zones de programmes, le HX-20 comprend une zone RAM de fichiers qui facilite le stockage des données ainsi que le transfert des données d'un programme à l'autre.
7. En reliant un terminal à disque souple en option TF-20X au HX-20, le BASIC DISK peut être exécuté.

AVERTISSEMENT

Dans les pays où les contrôleurs d'affichage ne sont pas disponibles, ce manuel pourra être utilisé moyennant les changements suivants.

CHAPITRE 1 - Introduction au HX-20

Les paragraphes suivants devront être lus sans tenir compte des références à un écran extérieur ou à un contrôleur d'affichage.

1.3	Ecran de texte et de graphisme
1.4	Ecran de texte
1.5.2	Déplacement de l'écran matériel
1.5.7	Ecran

CHAPITRE 3 - Commandes et Instructions

COULEUR
Cette commande est inutilisable si le contrôleur d'affichage n'est pas connecté au HX-20.

Toute référence à un écran extérieur ou à un contrôleur d'affichage dans les commandes suivantes devra être négligée.

CLS
GCLS
LINE
PRESET
PSET
SCREEN
SCROLL
WIDTH

Table des matières

Chapitre 1 Présentation du HX-20

1.1	Initialisation et BASIC	1-1
1.2	Utilisation du clavier	1-6
1.3	Ecrans de textes et de graphismes	1-10
1.4	Ecran de texte (écran virtuel)	1-12
1.5	Edition de l'écran	1-14
1.6	Opérations sur les fichiers	1-16
1.7	Equipement périphérique	1-18

Chapitre 2 Présentation de l'EPSON BASIC

2.1	Modes de fonctionnement	2-1
2.2	Instructions	2-2
2.3	Lignes	2-2
2.4	Jeu de caractères	2-2
2.5	Constantes	2-3
2.6	Variables	2-4
2.7	Changement de type	2-5
2.8	Expressions et opérations	2-6
2.9	Messages d'erreurs	2-11
2.10	Réglage du commutateur DIP	2-12
	Comment utiliser les Chapitres 3 et 4	2-13

Chapitre 3 Commandes et Instructions

Chapitre 4 Fonctions

Chapitre 5 Informations complémentaires

5.1	Fichiers RAM	5-1
5.2	Fichiers séquentiels	5-5
5.3	Programmes en langage machine	5-9
5.4	Comment utiliser la porte RS-232C	5-14

ANNEXES

A. Messages d'erreurs
B. Noms des dispositifs
C. Table de Correspondance entre les Noms des dispositifs et les Commandes BASIC.
D. Caractères de Formattage
E. Disposition des claviers et Affectations des touches
F. Tables des codes de caractères
G. Codes de contrôle
H. Configuration de la mémoire
I. Table des mots réservés
J. Liste des commandes et instructions

Commandes et Instructions

AUTO 3-1
CLEAR 3-2
CLOSE 3-3
CLS 3-3
COLOR 3-4
CONT 3-5
COPY 3-5
DATA 3-7
DEFFIL 3-8
DEF FN 3-9
DEFINT/SNG/DBL/STR 3-10
DEF USR 3-12
DELETE 3-13
DIM 3-14
END 3-14
ERASE 3-15
ERROR 3-16
EXEC 3-17
FILES 3-18
FOR...TO...STEP – NEXT 3-19
GCLS 3-21
GET% 3-21
GOSUB – RETURN 3-22
GO TO/GOTO 3-23
IF...THEN...ELSE/
IF...GOTO...ELSE 3-24
INPUT 3-25
INPUT# 3-26
KEY 3-26
KEY LIST/KEY LLIST 3-27
LET 3-27
LINE 3-28
LINE INPUT 3-29
LINE INPUT# 3-30
LIST/LLIST 3-30
LIST<file descriptor> 3-31
LIST"COM0:" 3-31
LOAD 3-32
LOAD"COM0:" 3-32
LOADM 3-33
LOAD? 3-34
LOCATE 3-35
LOCATES 3-36
LOGIN 3-36
MEMSET 3-37
MERGE 3-38
MERGE"COM0:" 3-38
MID$ 3-39
MON 3-40
MOTOR 3-41
NEW 3-42
ON ERROR GOTO 3-42
ON...GOSUB/ON...GOTO 3-44
OPEN 3-45
OPEN"COM0:" 3-46
OPTION BASE 3-48
PCOPY 3-49
POKE 3-50
PRESET 3-52
PRINT/LPRINT 3-53
PRINT USING/LPRINT USING 3-54
PRINT# 3-58
PRINT# USING 3-58
PSET 3-59
PUT% 3-60
RANDOMIZE 3-61
READ 3-61
REM 3-62
RENUM 3-62
RESTORE 3-63
RESUME 3-64
RUN 3-65
RUN"COM0:" 3-65
SAVE 3-66
SAVE"COM0:" 3-67
SAVEM 3-67
SCREEN 3-68
SCROLL 3-69
SOUND 3-70
STAT 3-71
STOP 3-72
SWAP 3-72
TITLE 3-73
TRON/TROFF 3-74
WIDTH 3-75
WIDTH<device name> 3-76
WIND 3-77

Fonctions

ABS 4-1
ASC 4-1
ATN 4-2
CDBL 4-2
CHR$ 4-3
CINT 4-3
COS 4-4
CSNG 4-4
CSRLIN 4-5
DATE$ 4-5
DAY 4-6
EOF 4-6
ERL/ERR 4-7
EXP 4-7
FIX 4-8
FRE 4-8
HEX$ 4-9
INKEY$ 4-10
INPUT$ 4-11
INSTR 4-12
INT 4-13
LEFT$ 4-14
LEN 4-14
LOF 4-15
LOG 4-15
MID$ 4-16
OCT$ 4-17
PEEK 4-17
POINT 4-18
POS 4-19
RIGHT$ 4-19
RND 4-20
SGN 4-21
SIN 4-21
SPACE$ 4-22
SPC 4-22
SQR 4-23
STR$ 4-23
STRING$ 4-24
TAB 4-25
TAN 4-25
TAPCNT 4-26
TIME$ 4-27
USR 4-27
VAL 4-28
VARPTR 4-28

Chapitre 1

Présentation du HX-20

1.1 Initialisation et BASIC

Contrairement aux autres ordinateurs personnels, l'espace mémoire de votre ordinateur portable HX-20 est divisé en cinq zones de programmes dans lesquelles cinq programmes BASIC peuvent être stockés indépendamment. Ces programmes resteront en mémoire sans aucun changement, même si vous tournez le commutateur général de votre HX-20 sur OFF. Le HX-20 comprend une fonction menu qui vous permet d'exécuter l'un quelconque des programmes en mémoire en appuyant sur une seule touche. Tournez le commutateur général sur ON et le menu suivant apparaîtra sur l'écran LCD.

France

```
CTRL/à Initialize
1 MONITOR
2 BASIC
```

Suède

```
CTRL/é Initialize
1 MONITOR
2 BASIC
```

Allemagne

```
CTRL/§ Initialize
1 MONITOR
2 BASIC
```

Danemark, Norvège

```
CTRL/é Initialize
1 MONITOR
2 BASIC
```

Le menu affiche les numéros et les fonctions (c'est à dire les noms des programmes) qui peuvent être exécutées lorsque vous appuyez sur les touches numériques correspondantes. Lorsque le HX-20 est mis sous tension pour la première fois, deux fonctions seulement apparaîssent sur l'écran LCD comme vous pouvez le voir ci-dessus. L'apparition de «CTRL/à » sur la ligne supérieure de l'écran, vous indique que vous devez appuyer sur la touche **à** tout en appuyant sur la touche **CTRL**

NOTE:

Si vous choisissez un caractère différent des caractères prévus pour les Etats-Unis, l'Angleterre, l'Italie ou l'Espagne, les symboles suivants apparaîtront au lieu de "à" ».

France	Allemagne	Suède	Danemark	Norvège	ASCII
à	§	É	É	É	@

Le code du caractère et la fonction de chacune de ces touches sont les mêmes que celles du « à » et vous devez appuyer sur le caractère correspondant de chaque police de caractères tout en appuyant sur la touche **CTRL** *(Voir section 2.4, Les détails des Polices de caractères).*

Si vous faites suivre ces opérations de l'introduction de la date et de l'heure, selon les indications données plus loin, votre HX-20 sera initialisé. Cet état est appelé «démarrage à froid». (Vous devez initialiser votre HX-20 par un démarrage à froid lorsque vous tournez le commutateur général sur ON pour la première fois aprés l'achat). Pour exécuter l'EPSON BASIC, appuyez sur la touche numérique «2».

```
EPSON BASIC V-1.0
Copyright 1982 by
Microsoft & EPSON
P1:          0 Bytes
```

Lorsque le BASIC est exécuté, la zone du programme No 1 est toujours sélectionnée. Le message «0 octet» suivant «P1» indique la longueur du programme stocké dans la zone de programmes considérée. Dans le cas ci-dessus, il n'y a eu aucun programme stocké dans la zone No 1, donc la valeur affichée est «0» octet.

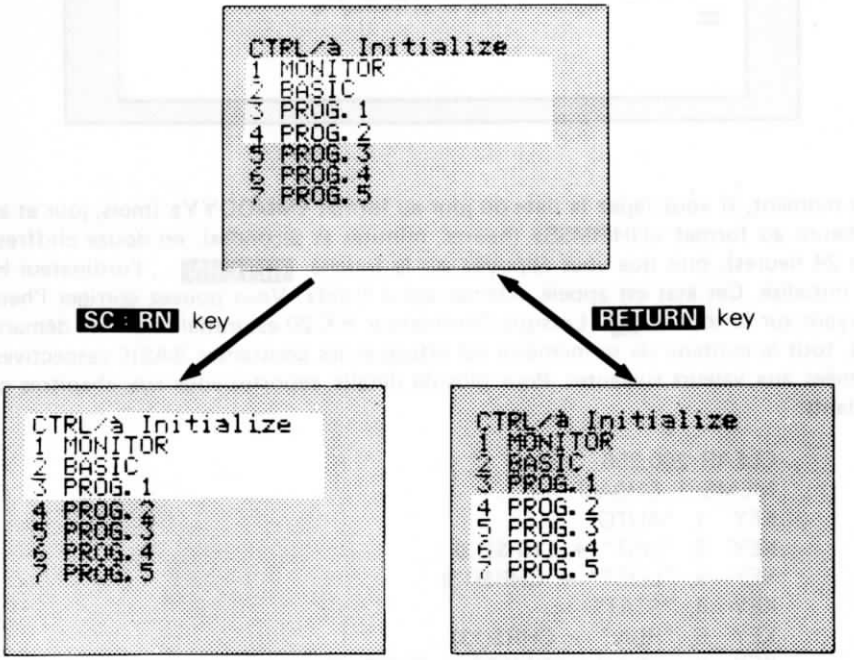
1.1.1 Menu

Si vous désignez un programme que vous avez écrit en BASIC au moyen d'une commande TITLE, le nom du programme apparaîtra sur le menu affiché. Cela signifie que le nom du programme a été enregistré dans le menu.

```
CTRL/à Initialize
1 MONITOR
2 BASIC
3 PROG.1
```

Les cinq programmes enregistrés dans le menu sont respectivement numérotés de 3 à 7. L'un quelconque de ces programmes sera exécuté directement (sans recourir aux commandes LOGIN et RUN) à chaque fois que vous appuierez sur la touche numérique correspondant au numéro du programme.

Lorsque vous appelez le menu, après avoir enregistré tous les noms de programmes, l'écran va se décaler pour vous présenter successivement les noms de programmes, étant donné que tous les noms de programmes ne peuvent pas être affichés en même temps sur l'écran LCD. Après l'affichage de tous les noms de programmes, le premier nom "1 MONITOR", réapparaît pour vous signaler que le système HX-20 est revenu au niveau des commandes. Si, à ce moment, vous appuyez sur la touche **RETURN** l'écran affiche ce qui est en dehors des dimensions del'écran matériel pour vous présenter les autres noms de programmes. En appuyant sur la touche **SC : RN**, vous retrouvez l'écran initial. Si vous appuyez sur les touches **SHIFT** et **SC : RN**, l'écran revient à sa position initiale.



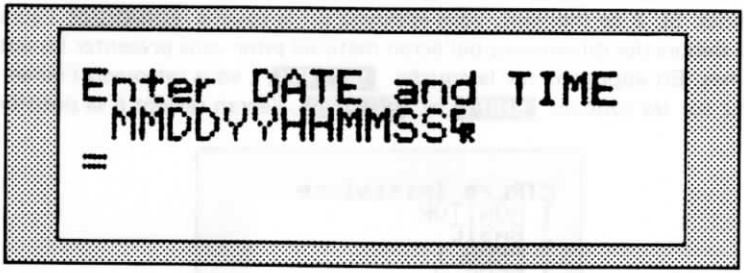
NOTE :

Les numéros affectés au programme servent uniquement à l'écran menu et à la commande K du MONITEUR, et n'ont aucun lien direct avec les numéros des zones de programmes utilisés par une commande LOGIN. Ces numéros peuvent aussi varier lorsque l'on enregistre d'autres programmes dans le menu.

La commande K du MONITEUR prépare le HX-20 à l'exécution directe d'un programme sans nécessiter aucune autre opération dès que le commutateur général est mis sur ON.

1.1.2 Démarrage à froid

Lorsque vous appuyez sur les touches **CTRL** et "@", alors que le menu est affiché, le HX-20 vous demande d'introduire la date et l'heure.



A ce moment, si vous tapez la date du jour au format «MMDDYY» (mois, jour et année) et l'heure au format «HHMMSS» (heures, minutes et secondes), en douze chiffres (système 24 heures), puis que vous appuyez sur la touche **RETURN**, l'ordinateur HX-20 sera initialisé. Cet état est appelé «Démarrage à froid». Vous pouvez corriger l'heure en appuyant sur la touche **INS DEL**. Lorsque l'ordinateur HX-20 est initialisé par un démarrage à froid, tout le contenu de sa mémoire est effacé et les constantes BASIC respectives sont ramenées aux valeurs suivantes. Pour plus de détails, reportez-vous aux chapitres correspondants.

```
CLEAR 200,256
MEMSET &H0A3F
KEY  1, "AUTO"
KEY  2, "LIST" + CHR$(13)
KEY  3, "LLIST" + CHR$(13)
KEY  4, "STAT"
KEY  5, "RUN" + CHR$(13)
KEY  6, "?DATE$:?TIME$" + CHR$(13)
KEY  7, "LOAD"
KEY  8, "SAVE"
KEY  9, "TITLE"
KEY 10, "LOGIN"
```

Comme indiqué ci-dessus, au moment où l'ordinateur HX-20 est démarré à froid, toutes les données se trouvant stockées dans sa mémoire sont effacées. Vous devez donc veiller à ne pas procéder à un démarrage à froid de votre HX-20 sans que cela ne soit nécessaire. Si vous appuyez accidentellement sur les touches **CTRL** et **@**, n'introduisez ni la date ni l'heure, mais appuyez sur la touche **BREAK** ou **MENU**. Cela ramènera le HX-20 à l'affichage du menu.

1.1.3 Démarrage à chaud

Toute exécution du BASIC autrement que par un démarrage à froid, est appelé «Démarrage à chaud». Lorsque le HX-20 est initialisé par un démarrage à chaud, les constantes respectives du BASIC sont ramenées aux valeurs suivantes. (Ces valeurs sont les «valeurs implicites aprés démarrage à chaud»).

```
CLEAR 200
SCREEN 0,0
SCROLL 9,0,10,4 | LCD
WIDTH 40,8,3    |
SCROLL 9,0,16,16 | External display
WIDTH 40,37,5    |
```

Si vous appuyez sur la touche **MENU**, même au cours de l'exécution d'un programme BASIC, le BASIC passera à l'état de démarrage à chaud. Par conséquent, n'oubliez pas que si vous êtes par exemple en train de changer la dimension de l'écran du HX-20, les valeurs implicites seront reprises lorsque le menu aura été affiché en appuyant sur la touche **MENU**. (Cependant, la dimension de l'écran extérieur ne sera pas modifiée par le démarrage à chaud).

1.1.4 Remise à l'état initial

Il existe un commutateur RESET placé dans un renfoncement, à l'arrière droit de votre HX-20. Vous n'avez pas à agir sur ce commutateur pendant une exécution normale en BASIC. N'utilisez jamais le commutateur RESET, sauf dans les cas définis ci-dessous. Vous ne devez utiliser le commutateur RESET que si le HX-20 ne répond pas à la touche **BREAK** ni au commutateur POWER (par suite d'un bloquage du programme). Dans le cas du HX-20 un bloquage du programme ne se produira jamais, sauf si le HX-20 est exploité de façon incorrecte, a savoir:

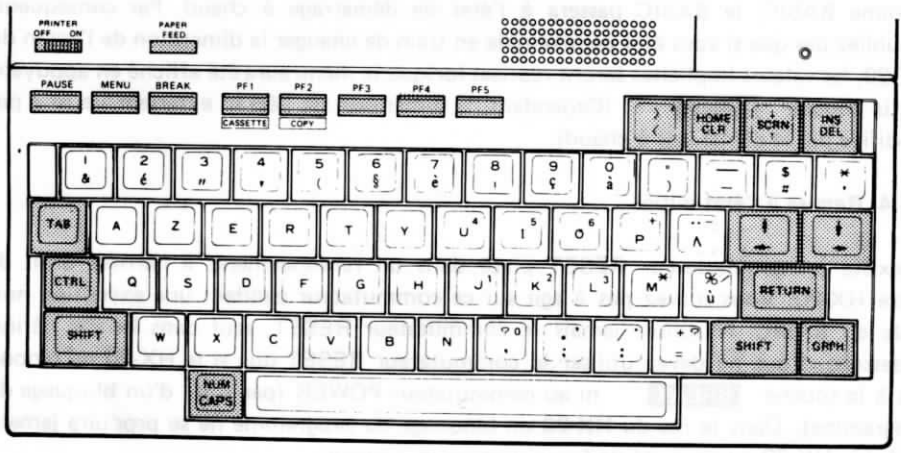
- o Réécriture de données dans la zone système (Adr. mémoire &HOO4D à &HOA3F).
- o Tentative de lecture des données de la zone E/S (Adr. mémoire &HOOOO à HOO4D).
- o Exécution d'un programme en langage machine qui n'a pas été complètement écrit.

Si un bloquage de programme se produisait à la suite de l'une des opérations ci-dessus, il faudrait immédiatement tourner le commutateur POWER sur OFF et puis appuyer sur le commutateur RESET. Lorsque vous tentez une exécution en BASIC aprés avoir appuyé sur le commutateur RESET, il peut arriver qu'une partie de la mémoire a été reécrite par suite du bloquage. Dans ce cas, l'affichage du menu est interrompu et vos tentatives d'exécution en BASIC ne réussissent pas. Si cela se produit, vous devez initialiser le HX-20 par un démarrage à froid pour effacer le contenu de sa mémoire. Avant d'exécuter un programme en langage machine qui n'a pas été écrit complètement, sauvegardez toujours les données et les programmes de la mémoire sur une cassette ou autre mémoire extérieure, de façon à éviter la perte définitive de votre programme, même si la mémoire devait être effacée.

NOTE:
Lorsque vous utilisez le HX-20, il peut arriver que le message «CHARGE BATTERY!» apparaîssent brusquement sur l'écran. Dans ce cas, toutes les opérations du HX-20 sont suspendues et le HX-20 n'accepte plus aucune entrée. Ce message vous signale que la tension de la batterie est faible. Si la tension de la batterie du HX-20 descend au dessous d'un certain niveau, l'appareil s'arrête de fonctionner pour protéger les programmes et les données de sa mémoire. Lorsque ce message apparaît, tournez le commutateur général sur OFF et rechargez les batteries le plus tôt possible.

1.2 Utilisation du clavier

Le clavier du HX-20 est disposé de la façon suivante:



NOTE:
La disposition présentée ici correspond au clavier Français.

1.2.1 Modes du clavier

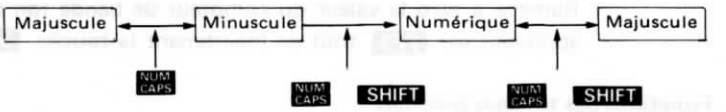
Le HX-20 dispose de trois modes d'introduction des caractères. Les caractères que vous pouvez introduire dans chacun de ces modes sont différents.

(1) Mode majuscule:
Avec ce mode, les entrées normales au clavier sont en caractères majuscules. Par exemple, en appuyant sur la touche marquée "A" du clavier, vous introduisez le caractère "A". Si vous appuyez sur cette touche tout en maintenant la touche **SHIFT** enfonçée, le "a" est affiché. Les touches numériques et les symboles sont le chiffre ou le symbole apparaissant sur la moitié inférieure de la touche est normalement affiché et le nombre ou le symbole de la moitié supérieure est introduit lorsque vous appuyez sur la touche en maintenant la touche **SHIFT** enfonçée. Par exemple, si vous appuyez sur la touche en mode majuscule, le "<" sera normalement introduit et ">" est introduit si vous appuyez sur la touche tout en maintenant la touche **SHIFT** enfonçée. Le mode majuscule est le mode par défaut et le HX-20 est réglé sur ce mode à chaque fois que le BASIC est exécuté.

(2) Mode minuscule:
Le HX-20 passe dans ce mode lorsque vous appuyez sur la touche NUM CAPS . Dans ce mode, les minuscules sont normalement introduites et les majuscules sont introduites en appuyant sur une touche de lettre tout en maintenant la touche **SHIFT** enfonçée. La fonction des touches numériques et des symboles est la même que pour le mode majuscule.

(3) Mode numérique:
Le HX-20 passe en mode numérique lorsque vous appuyez sur les touches NUM CAPS et **SHIFT** .
Les touches qui sont utilisables dans ce mode sont les touches numériques de la rangée supérieure du clavier et les touches des lettres qui portent le numéro 0 à 6 dans l'angle supérieur droit ainsi que les touches des symboles (+), (–), (*), (/), (.), (,) et (?). Les autres touches n'ont alors aucune action .
Vous pouvez voir tous les caractères affectés pour chaque mode en vous reportant à l'Annexe E "Affectation des touches pour chaque mode du clavier".

Vous pouvez changer de mode en appuyant sur l'une des touches de mode du clavier. Mais si vous appuyez sur la touche du mode dans lequel se trouve déjà le HX-20, le mode du clavier reviendra au mode majuscule. Voici un exemple de changement des modes au moyen des touches.



Les touches marquées sur la figure de la page 1-6 sont des touches de fonctions spéciales qui peuvent être introduites quel que soit le mode.

1.2.2 Introduction des caractères graphiques

Vous pouvez introduire des caractères graphiques en mode majuscule en appuyant sur les touches correspondantes, tout en maintenant la touche **GRPH** enfonçée. Pour plus de détails, vous pouvez vous reporter à l'Annexe E "Affectation des touches pour chaque mode du clavier".

1.2.3 Fonctions spéciales de la touche de CONTROLE

La touche **CTRL**, quand elle est employée en liaison avec d'autres touches, introduit des codes de contrôle qui effectuent des fonctions spéciales comme le déplacement du curseur sur l'écran, etc. Reportez-vous à l'Annexe G qui donne des détails au sujet de ces codes de contrôle.
De plus, la touche **CTRL** a deux fonctions spéciales:

CTRL + **PF2** Recopie les données de l'écran LCD sur la micro-imprimante incorporée Cette fonction est la même que l'exécution d'une commande COPY.

CTRL + **PF1** Cette séquence de touches place le dérouleur de microcassette (en option) en mode manuel. Si l'enregistreur n'est pas connecté, il est impossible de passer dans ce mode. Après passage en mode manuel, l'écran s'éteint et la valeur du compteur de bande est affichée dans l'angle supérieur droit de l'écran matériel. En mode manuel de fonctionnement, les opérations de la microcassette peuvent être contrôlées par les touches des fonctions programmables **PF1** à PF6 de la façon suivante:

PF1 Déroulement rapide.

PF2 Déroulement lent.

PF3 Arrêt du réenroulement de la bande, du déroulement rapide ou lent.

PF4 Réenroulement de la bande.

PF5 Fait sortir du mode de fonctionnement manuel.

PF6 Ramène à zéro la valeur du compteur de bande (on introduit PF6 en appuyant sur **PF1** tout en maintenant la touche **SHIFT** enfonçée).

1.2.4 Fonctions des touches spéciales

PAPER FEED Cette touche permet de faire avancer le papier dans la micro-imprimante incorporée.

PAUSE S'emploie pour arrêter momentanément l'exécution et le listage d'un programme. L'opération interrompue est reprise lorsque l'on appuie sur une autre touche quelconque du clavier. Dans ce cas, si l'on appuie sur l'une des touches numériques 0 à 9 pour reprendre l'exécution, il est possible de spécifier la vitesse de défilement.

MENU Cette touche permet de ramener le HX-20 à l'état dans lequel il était avant l'exécution du BASIC. Le menu est affiché et le HX-20 attend l'introduction de la fonction que vous avez choisie.

BREAK Cette touche s'emploie pour arrêter l'exécution ou le listage d'un programme. L'exécution du programme peut être reprise par l'introduction d'une commande CONT.

PF1–**PF5** PF6–PF10 Vous pouvez définir les fonctions spéciales de ces touches par logiciel (Pour de plus amples informations, reportez-vous à la commande KEY au chapitre 3).

RETURN Cette touche signale au BASIC que l'entrée des données avec les unités nécessaires a été achevée. Lorsque vous appuyez sur cette touche, le curseur se déplace jusqu'au début de la ligne suivante.

NUM **GRPH** Ces touches permettent de sélectionner le mode du clavier.

TAB **SC : RN** Ces touches sont utilisées pour l'édition de l'écran. Vous obtiendrez des détails sur chaque touche en vous reportant au paragraphe 1.5.

Pour la micro-imprimante incorporée, il existe un commutateur (marche/arrêt de l'imprimante) **PRINTER ON/OFF** et une touche PAPER FEED. La micro-imprimante ne peut être utilisée que si le commutateur **PRINTER ON/OFF** est dans la position ON. Par exemple, même si vous exécutez une instruction LLIST, ou que vous appuyez sur la touche PAPER FEED alors que le commutateur **PRINTER ON/OFF** est sur OFF, la micro-imprimante ne fonctionnera pas. La touche PAPER FEED fera avancer le papier uniquement pendant qu'elle est enfoncée.

1.2.5 Fonction de répétition automatique

Si vous appuyez de façon continue sur les touches alphabétiques, elles ont une fonction d'introduction automatique permanente. En outre les touches spéciales suivantes ont la même fonction.

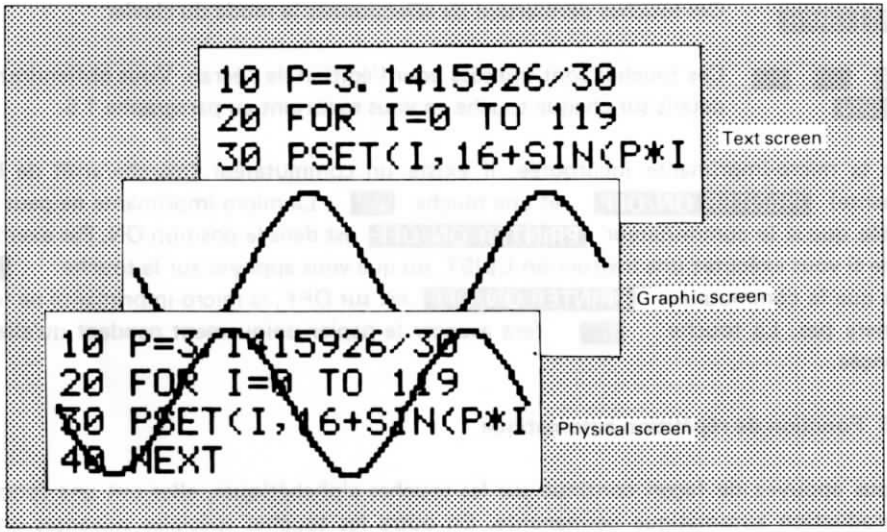
HOME CLR **SC : RN** INS DEL **TAB** **RETURN**

1.3 Ecrans de texte et de graphisme

Le HX-20 dispose de deux écrans distincts que vous pouvez utiliser dans des buts totalement différents. Le premier écran, appelé «écran de texte», est destiné à afficher des caractères et le second, appelé « écran graphique», permet de présenter des graphismes en utilisant des instructions comme LINE et PSET. Ces deux écrans peuvent être sortis sur deux dispositifs de sortie différents: le premier est l'écran LCD incorporé et le second l'écran extérieur, en option.

Pour l'EPSON BASIC, avec une instruction SCREEN, vous pouvez spécifier trois modes d'affichage différents en combinant les deux modes d'écran et les deux dispositifs de sortie. Lorsque le BASIC est exécuté, l'écran de texte, de même que l'écran de graphisme sont spécifiés pour être affichés sur le LCD.

NOTE: En France, l'option d'affichage sur un écran extérieur - moniteur vidéo ou télévision - pourra ne pas être disponible à la date de sortie du manuel.

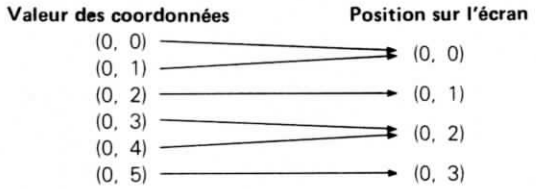


Lorsque vous désirez tracer des lignes et des points sur l'écran graphique, vous devez utiliser des coordonnées rectangulaires en prenant comme origine l'angle supérieur gauche de l'écran. La valeur de la coordonnée horizontale augmente quand un point se déplace sur l'axe des X en direction de la droite, tandis que la valeur de la coordonnée verticale augmente quand le point de l'axe des Y descend. La gamme des coordonnées de l'écran LCD diffère de celle d'un écran extérieur. Dans le cas d'un écran extérieur, la gamme des coordonnées varie en fonction du mode sélectionné pour l'écran. Pour une instruction LINE ou PSET, le HX-20 ne vérifie pas si oui ou non les coordonnées spécifiées sont réellement à l'intérieur de l'écran. Veuillez faire attention à la gamme des coordonnées.

	Mode	Direction horizontale (points)	Direction verticale (points)
LCD		120	32
Ecran extérieur	4 couleurs	128	64
	Haute résolution	128	96

NOTE:
La table ci-dessus indique la résolution de chaque affichage et n'indique pas la gamme des coordonnées. Par exemple, le LCD a une résolution de 120 x 32 points. Mais les valeurs spécifiées des coordonnées doivent se situer dans la gamme de 0 à 119 horizontalement et de 0 à 31 verticalement.

Lorsque l'écran extérieur est réglé en mode graphique à quatre couleurs, la résolution de l'écran est de 64 points verticalement. Mais la gamme des coordonnées que vous pouvez spécifier est située entre 0 et 96, ce qui revient au même qu'un mode à haute résolution. Autrement dit, les valeurs de vos coordonnées spécifiées ne correspondent pas aux points de l'écran.

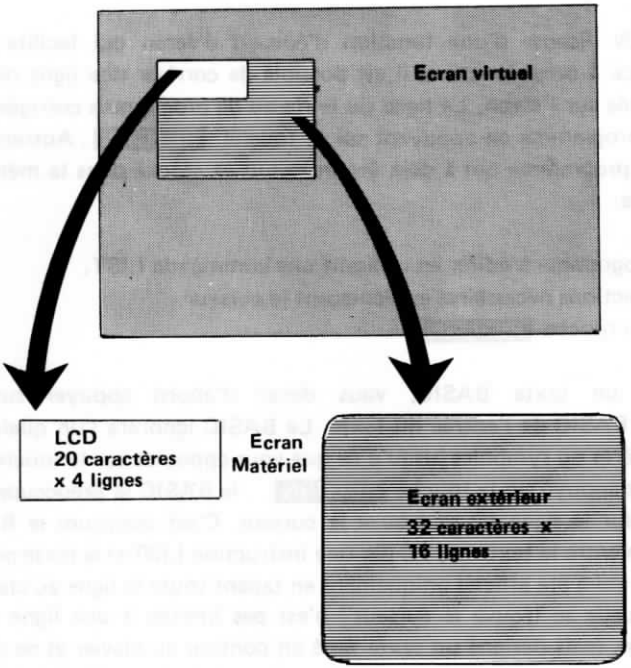


Supposons que la valeur de la coordonnée verticale à spécifier soit N et que la position verticale sur l'écran soit V.
La relation entre les deux peut s'exprimer par la formule suivante:

V = 2N\3 (« \ » indique une division entière)

1.4 Ecran de texte (écran virtuel)

L'écran LCD du HX-20 ne peut afficher que 80 caractères (4 lignes x 20 caractères) à la fois. Mais, en déplaçant le curseur, vous pouvez faire apparaître plusieurs écrans successivement. Vous pouvez utiliser le HX-20 comme vous utiliseriez un grand écran. L'explication en est que le HX-20 a adopté le principe de «l'écran virtuel». Supposons qu'il y ait un très grand écran à l'intérieur du HX-20 ; des commandes comme LIST, PRINT, sont toutes sorties sur l'écran interne. Maintenant, supposons que l'écran LCD ait une «lucarne de visualisation» qui vous permette de voir uniquement une partie de l'écran interne. Cette «lucarne de visualisation» peut se déplacer partout sur le grand écran interne. Mais elle ne peut pas quitter les limites de l'écran interne. Ce même principe de «l'écran virtuel» s'applique aussi à l'écran extérieur. La seule différence, c'est que la dimension de la «lucarne» à travers laquelle vous pouvez voir, est plus grande que la lucarne du LCD. Par comparaison avec l'écran virtuel, cette «lucarne» du LCD et de l'écran extérieur est appelé «écran matériel». Grâce à cet écran virtuel, vous pouvez voir des instructions qui ont été précédemment introduites ou sorties en déplaçant l'écran réel. Mais si l'écran virtuel était rempli de données, l'écran serait remonté d'une ligne pour l'affichage suivant et vous ne pourriez plus voir la ligne débordant de l'écran virtuel en déplaçant l'écran matériel.



La dimension de l'écran virtuel peut être spécifié par une commande WIDTH (largeur). Vous pouvez spécifier la dimension de l'écran virtuel librement dans la gamme de 225 lignes de 255 caractères chacune, en respectant les limitations imposées par la capacité de la mémoire (sur l'écran extérieur, la dimension maximale de l'écran est de 37 lignes de 40 caractères, soit un total de 1480 caractères). Si vous spécifiez un écran virtuel trop grand, la zone disponible pour le programme sera réduite. En vous souvenant de cette particularité, spécifiez la dimension de l'écran virtuel selon votre application. L'écran virtuel n'est valable que pour l'écran de texte et la dimension de l'écran graphique est fixe.

1.5 Edition de l'écran

Le BASIC EPSON dispose d'une fonction d'édition d'écran qui facilite l'édition des programmes. Grâce à cette fonction, il est possible de corriger une ligne de texte ou de programme affichée sur l'écran. La ligne de texte ou de programme corrigée peut être introduite dans le programme en appuyant sur la touche **RETURN**. Autrement dit, vous pouvez éditer un programme qui a déjà été introduit et stocké dans la mémoire selon la procédure suivante:

(1) Affichez le programme à éditer en utilisant une commande LIST.
(2) Faites les corrections nécessaires en déplaçant le curseur.
(3) Appuyez sur la touche **RETURN**

Pour introduire un texte BASIC, vous devez d'abord appuyer sur **RETURN** pour informer le BASIC de l'entrée du texte. Le BASIC ignorera l'un quelconque ou la totalité des caractères ou symboles jusqu'à ce que vous appuyiez sur la touche **RETURN** Lorsque vous avez appuyé sur la touche **RETURN** , le BASIC se préoccupe uniquement du texte affiché sur la ligne où se trouve le curseur. C'est pourquoi le BASIC ne fait aucune distinction entre le texte appelé par une instruction LIST et le texte précédemment introduit ou celui qui a été affiché uniquement en tapant toute la ligne au clavier.
La "ligne sur laquelle se trouve le curseur" n'est pas limitée à une ligne couramment affichée sur l'écran, mais désigne un texte tapé en continu au clavier et ne dépassant pas 255 caractères de longueur. Le BASIC considère un texte écrit sur plusieurs lignes comme une "seule ligne logique". A chaque fois que le curseur est placé sur la "ligne logique unique" lorsque la touche RETURN est enfoncée, cette ligne est introduite jusqu'à sa fin. Lorsque la ligne de l'écran est plus courte que l'unique ligne logique, vous n'êtes quelquefois pas capable de voir en regardant simplement l'écran si le texte affiché fait partie de ll'unique ligne logique ou non. Veuillez faire attention à ce point.

1.5.1 Déplacement du curseur

Les touches suivantes permettent de déplacer librement le curseur seul pendant l'édition d'un programme.

TAB	Fait avancer le curseur jusqu'au prochain tabulateur (toutes les 8 colonnes)
←	Déplace le curseur d'une position vers la gauche
→	Déplace le curseur d'une position vers la droite
SHIFT + ↑	Fait monter le curseur d'une ligne
SHIFT + ↓	Fait descendre le curseur d'une ligne

Si le curseur est à une extrémité de l'écran matériel et doit sortir de l'écran du fait de l'action sur une touche de mouvement du curseur, l'écran matériel se déplace automatiquement à l'intérieur de l'écran virtuel. De cette façon, le curseur ne peut pas quitter l'écran matériel. De même, au cours d'un démarrage à chaud, la BASIC fixe automatiquement la «marge de défilement» à 3. Ceci signifie qu'au moment ou le curseur atteint la troisième position à partir de l'un ou l'autre extrémité de l'écran, l'écran matériel se déplace dans la marge laissée à l'autre extrémité. La marge de défilement peut être spécifiée avec une instruction WIDTH.
Si le curseur se trouve à l'une ou l'autre extrémité de l'écran virtuel, toute tentative de déplacement du curseur dans la même direction le fait passer sur la ligne suivante (ou précédente). L'affichage devra naturellement suivre un tel mouvement du curseur. Mais, le texte avancé de cette façon, à une nouvelle ligne, par l'une des touches de déplacement du curseur n'est pas considéré comme une ligne logique unique. Le curseur ne peut pas être déplacé au-delà de l'angle supérieur gauche ou inférieur droit de l'écran virtuel.

1.5.2 Déplacement de l'écran matériel

CTRL + A	Déplace l'écran matériel jusqu'à l'angle gauche de l'écran virtuel
CTRL + F	Déplace l'écran matériel jusqu'à l'angle droit de l'écran virtuel
CTRL + ← ou CTRL + S	Déplace l'écran matériel vers la gauche du nombre spécifié de colonnes.
CTRL + → ou CTRL + D	Déplace l'écran matériel vers la droite du nombre spécifié de colonnes.
SC ⇕ RN ou CTRL + P	Fait monter l'écran du nombre spécifié de lignes.
SHIFT + SC ⇕ RN CTRL + Q	Fait descendre l'écran matériel du nombre spécifié de lignes.

Ces pas de défilement sont fixés aux valeurs suivantes au moment d'un démarrage à chaud du BASIC : pour le LCD, quatre lignes verticalement et dix colonnes horizontalement et pour l'affichage extérieur, 16 lignes verticalement et 16 colonnes horizontalement.
Le réglage du pas de défilement peut être modifié à l'aide d'une commande SCROLL. Le mouvement de l'écran matériel diffère de celui du curseur. C'est à dire qu'au moment où l'écran matériel atteint l'une ou l'autre extrémité de l'écran virtuel, il s'arrête à cet endroit et ne se déplace pas plus loin.

SHIFT + HOME CLR ou CTRL + K	Déplace le curseur jusqu'à sa position de repos, c'est à dire dans l'angle supérieur de l'écran virtuel.

1.5.3 Insertion et suppression

Au moment de la correction d'un texte affiché sur l'écran, il est possible de modifier le caractère qui se trouve dans la position du curseur en tapant un autre caractère ou un autre symbole par-dessus. Mais pour les corrections importantes, les opérations suivantes sont exécutées au clavier.

SHIFT + INS DEL ou CTRL + R	Quand vous appuyez sur ces touches, le HX-20 passe en mode d'insertion. Tout caractère tapé après cette opération est introduit à gauche de la position du curseur et les caractères situés à la droite du curseur sont déplacés en même temps que le curseur. Ce mode se poursuit jusqu'à ce que vous appuyez sur la touche RETURN, sur l'une quelconque des touches de déplacement du curseur ou sur la touche INS DEL
INS DEL ou CTRL + H	Les caractères se trouvant immédiatement à gauche de la position du curseur sont supprimés et le curseur se déplace vers la gauche. Les fautes de frappe sont normalement corrigées par cette opération.
CTRL + E	Tous les caractères situés à droite de la position du curseur sur la ligne où se trouve le curseur à ce moment sont supprimés. Dans ce cas, le curseur ne se déplace pas.
CTRL + Z	Tous les caractères depuis la ligne sur laquelle le curseur est placé jusqu'à la ligne inférieure de l'écran virtuel sont supprimés. Dans ce cas, le curseur ne se déplace pas.
HOME CLR ou CTRL + L	Supprime tous les caractères de l'écran virtuel et ramène le curseur à sa position de repos en haut à gauche.

1.6 Opérations sur les fichiers

On appelle «enregistrement» une collection d'informations et on appelle «fichier» une collection d'enregistrements. Dans l'EPSON BASIC, les transferts E/S en entrée ou en sortie des périphériques sont effectués sur des unités d'un seul enregistrement et les données E/S sont gérées en unités d'un seul fichier.
En EPSON BASIC, une zone spéciale de RAM est prévue pour les fichiers. Les instructions GET% et PUT% permettent de lire et d'écrire des fichiers en RAM pour assurer un stockage de données et un transfert E/S efficaces.

1.6.1 Numéros des fichiers

Pour renforcer l'efficacité des E/S, le HX-20 dispose dans la mémoire d'une zone spéciale appelée «tampon» dans laquelle des enregistrements sont stockés provisoirement. Les numéros individuels affectés aux placements de la zone tampon sont appelés «numéros des fichiers». Ces numéros de fichiers permettent d'accéder aux fichiers.
Chaque numéro du fichier doit être un entier compris entre 1 et 16.

1.6.2 Descripteurs de fichiers

Etant donné que le principe des fichiers a été adopté pour tous les dispositifs E/S, toutes les opérations E/S peuvent être effectuées au moyen de commandes normalisées. Un dispositif E/S est distingué d'un autre au moyen d'un "descripteur de fichier" qui est une chaîne comprenant les éléments suivants:

"< nom de dispositif >: [< nom de fichier >]"

Un descripteur de fichier doit toujours être indiqué entre guillemets.

1.6.3 Noms des dispositifs

Le nom de dispositif indique le nom d'un dispositif E/S, et en règle générale, il est représenté soit par une chaîne de quatre caractères alphabétiques, soit par une chaîne de trois caractères alphabétiques et d'un caractère numérique suivi par deux points. Les noms des dispositifs suivants sont définis par EPSON BASIC.

Nom de dispositif	Nom de l'équipement	Entrée	Sortie	Remarque
KYBD:	Clavier	○	X	
SCRN:	Ecran	X	○	
LPT0:	Micro-imprimante incorporée	X	○	
COM0:	Porte RS-232C	○	○	
CAS0:	Unité à microcassette	○	○	Option
CAS1:	Cassette audio	○	○	
PAC0:	Cartouche ROM	○	X	Option
A:	Unité à disque souple A	○	○	Noms des
B:	Unité à disque souple B	○	○	dispositifs
C:	Unité à disque souple C	○	○	pour le
D:	Unité à disque souple D	○	○	BASIC DISK

○: Application X : Non applicable

Si le nom du dispositif est omis, l'EPSON BASIC vérifie automatiquement l'équipement périphérique connecté au HX-20. Si un dérouleur à microcassette ou une ROM sur cartouche est connecté, le BASIC donne la priorité à ces dispositifs par rapport à une cassette audio. Si aucun de ces dispositifs n'est connecté, la cassette audio est sélectionnée. Le dispositif sélectionné lorsque le nom du dispositif a été omis est appelé "dispositif implicite".

1.6.4 Noms de fichiers

Le < nom de fichier > est le nom donné à chaque fichier par l'utilisateur. Aucun nom de fichier ne peut être omis si un transfert de données E/S doit être effectué entre le HX-20 et une autre mémoire auxiliaire comme une cassette ou une ROM sur cartouche. Dans certains autres cas, il peut être omis.

Le nom de fichier doit être mentionné avec le format suivant:

< nom de fichier >[. [< type de fichier >]]

La première partie du < nom de fichier > est constituée d'une chaîne de 8 caractères et le < type de fichier > qui suit un point est constitué d'une chaîne de trois caractères. Tous les caractères qui ne sont ni deux points, ni les points, ni les crochets, ni les codes de caractères 0 et 255 peuvent être utilisés comme < nom de fichier > et < type de fichier >. Habituellement, le < nom de fichier > indique le nom d'un fichier et le < type de fichier > son attribut. Mais ceci est laissé à la discrétion de l'utilisateur, le < type de fichier > étant normalement inclus dans toute référence au < nom de fichier >.
Si vous spécifiez un nom de fichier ayant plus de 8 caractères ou un type de fichier ayant plus de 3 caractères, une erreur "FD" apparaît.

1.7 Equipement périphérique

Le concept de "fichier" a été adopté pour effectuer les opérations E/S entre le HX-20 et tous les dispositifs périphériques. C'est pourquoi l'emploi des commandes normalisées est efficace pour presque tous les dispositifs.
La table suivante présente les commandes et instructions en BASIC de traitement des fichiers et les dispositifs pour lesquels les commandes et instructions respectives sont valables.

Descripteur / Commande	KYBD:	SCRN:	LPT0:	COM0:	CAS0:	CAS1:	PAC0:
LOAD	×	×	×	○	○	○	○
LOADM	×	×	×	×	○	○	○
LOAD?	×	×	×	×	○	○	×
RUN "< descripteur de fichier >"	×	×	×	○	○	○	○
MERGE	×	×	×	○	○	○	○
FILES	×	×	×	×	○	○	○
INPUT#	○	×	×	○	○	○	○
INPUT$	○	×	×	○	○	○	○
EOF	–	×	×	○	○	○	○
LOF	–	×	×	○	–	–	○
SAVE	×	○	○	○	○	○	×
SAVEM	×	×	×	×	○	○	×
LIST	×	○	○	○	○	○	×
PRINT# (USING)	×	○	○	○	○	○	×
POS	×	○	○	○	–	–	×
Mode OPEN	I	O	O	I/O	I/O	I/O	I

NOTE:
Dans cette table, ○ ou ✕ indique que si un dispositif est spécifié pour une commande ou une instruction, le dispositif

○: peut être utilisé
✕: ne peut pas être utilisé. Une erreur FC se produit.
—: n'entraîne aucune erreur mais la commande n'est pas valable.

Reportez-vous au Chapitre 3 au sujet de chaque commande ou instruction.

1.7.1 Ecran

Si un dispositif de sortie de fichier est spécifié sous la forme "SCRN:" (écran), il désigne soit le LCD, soit un écran extérieur qui a été spécifié au moyen d'une commande SCREEN pour l'affichage d'un écran de texte. Les caractères à délivrer sont en réalité écrits sur l'écran virtuel et l'écran que vous pouvez voir à un instant donné est limité par la capacité d'affichage de l'écran LCD (ou de l'écran extérieur). Faites attention à l'allocation de sortie et à la vitesse de sortie. L'écran est également utilisable pour le déverminage des programmes. Lorsque des données sont sorties sur cassette ou sur fichier à disque, vous aurez besoin d'un programme distinct pour vérifier la sortie correcte des données. Dans ce cas, en remplaçant le nom de dispositif dans l'instruction OPEN par "SCRN:", vous pouvez vérifier visuellement les données sorties.

1.7.2 Imprimante

Avec le HX-20, vous disposez de deux types d'imprimantes. La première est la micro-imprimante incorporée que vous devez spécifier sous la forme "LPT0:". L'autre est une imprimante extérieure connectée au HX-20 par l'intermédiaire de l'interface RS-232C que vous devez spécifier sous la forme: "COM0.".
Etant donné que vous utilisez plus souvent la micro-imprimante incorporée, les commandes spéciales de sortie sont prévues en plus des commandes générales de sortie.

LPRINT, LPRINT USING
LLIST
KEY LLIST
COPY

En utilisant une commande WIDTH < nom de dispositif >, vous pouvez aussi spécifier la largeur de l'impression. De cette façon, vous pouvez largement simplifier la spécification du format de sortie.

1.7.3 Cassette

En BASIC, vous pouvez utiliser deux types de cassettes comme mémoires auxiliaires.

"CAS1:"
Grâce à un enregistreur à cassette audio disponible dans le commerce, vous pouvez SAVE (sauvegarder) et LOAD (charger) des programmes et des données librement.Lorsque vous chargez un programme avec une commande LOAD, ou lorsque vous ouvrez un fichier de données en lecture, spécifiez le nom de fichier et une opération de recherche de ce fichier sera automatiquement faite sur la bande.

Lorsque vous exécutez une commande LOAD ou OPEN en mode direct, le message suivant est affiché sur l'écran LCD à chaque fois qu'un fichier différent du fichier spécifié est recherché.

Saute: <nom de fichier>

Lorsque le fichier spécifié est trouvé, le message suivant apparaît sur l'écran LCD.

Trouvé: <nom de fichier>

Vous pouvez aussi utiliser une commande LOAD? pour sauter le fichier spécifié. Vous obtiendrez davantage de renseignements au chapitre 3 au sujet de la commande LOAD?.

"CAS0:"
Le nom de dispositif désigne le dérouleur de microcassette en option. "CAS0:" peut être utilisé de la même manière que "CAS1:". Vous pouvez SAVE, VERIFY ou SEARCH les fichiers en utilisant le compteur incorporé et en exécutant une commande WIND (enroulement).

1.7.4 Porte RS-232C

Les communications avec les dispositifs extérieurs sont considérées comme très importantes en EPSON BASIC. C'est pourquoi la porte RS-232C est conçue pour permettre à la programmation en BASIC de prévoir les conditions de communications non seulement avec l'imprimante extérieure mais aussi, avec d'autres dispositifs extérieurs. Ces conditions peuvent être spécifiées non seulement par une commande OPEN (ouverture) mais aussi, par les commandes suivantes:

LOAD	(chargement)
LIST	(listage)
SAVE	(sauvegarde)
RUN	(exécution)
MERGE	(fusion)

Vous trouverez d'autres détails aux chapitres 3 et 5.

1.7.5 ROM sur cartouche (option)

Vous pouvez utiliser la cartouche ROM en option en spécifiant le nom de dispositif: "PAC0:". En dehors du fait que la cartouche ROM soit utilisée exclusivement pour l'entrée et que le chargement des fichiers puisse être effectué plus rapidement, cette cartouche est utilisable essentiellement de la même manière que la microcassette. C'est pourquoi ce dispositif est extrêmement utile lorsque vous utilisez des programmes et/ou des données particulières.

Chapitre 2

Présentation de l'EPSON BASIC

2.1 Modes de fonctionnement

Lorsque votre BASIC est mis en service, votre HX-20 affiche le signe attente ">" aprés le message d'ouverture. Le signe d'interrogation vous indique que le HX-20 est maintenant au niveau des commandes, c'est à dire qu'il attend que vous introduisiez une commande.

```
Copyright 1982 by
Microsoft & EPSON
P1:          0 Bytes
>
```

Si vous introduisez une instruction de programme écrite conformément à la syntaxe établie pour le BASIC mais sans numéro de ligne, l'instruction est exécutée immédiatement, c'est à dire dès que vous appuyez sur la touche **RETURN** . Ce type d'exécution est appelé "Exécution en mode direct". Les commandes et instructions BASIC peuvent être, presque toutes exécutées en mode direct à l'exception des instructions ci-dessous:

IMPUT
LINE INPUT
INPUT #
LINE INPUT #
RANDOMIZE

Si vous introduisez une instruction précédée d'un numéro de ligne (allant de 0 à 63999), cette instruction et son numéro de ligne sont stockées ensemble dans la mémoire considérée comme un programme. Le programme ainsi stocké peut être exécuté soit au moyen d'une commande RUN, soit au moyen d'une instruction GOTO ou GOSUB. Ce type d'exécution est appelé "Exécution en mode programme".
Lorsqu'un programme est exécuté par une commande RUN, toutes les variables (numériques, variables chaînes et réseaux et autres instructions définies) sont effacées avant l'exécution. Au contraire, une instruction GOTO exécute un programme sans changer les valeurs en vigueur de toutes les variables.

2.2 Instructions

Une instruction de programme est une définition des expressions commandes, instructions, fonctions, etc... qui sont exécutées par le BASIC. Il est possible de spécifier plusieurs instructions reliées entre elles par deux points. Ces instructions connectées sont appelées "instructions multiples".

2.3 Lignes

Une ligne de programme en BASIC commence toujours par un numéro de ligne (un entier compris entre 0 et 63999), suivi d'une ou plusieurs instructions et se terminant par un retour de chariot. Une ligne de programme peut contenir 255 caractères au maximum. Les numéros de lignes indiquent l'ordre dans lequel les lignes de programme sont stockées dans la mémoire. De la même manière, l'exécution du programme suit cet ordre en commençant par la ligne dont le numéro est le plus faible. Les numéros de lignes permettent aussi d'accéder au programme pour le branchement et l'édition. Un point peut remplacer un numéro de ligne après des commandes comme LIST et AUTO pour instruire le BASIC d'agir sur la dernière ligne, c'est à dire la ligne sur laquelle une erreur s'est produite au cours de l'exécution du programme ou la dernière ligne introduite dans le programme.

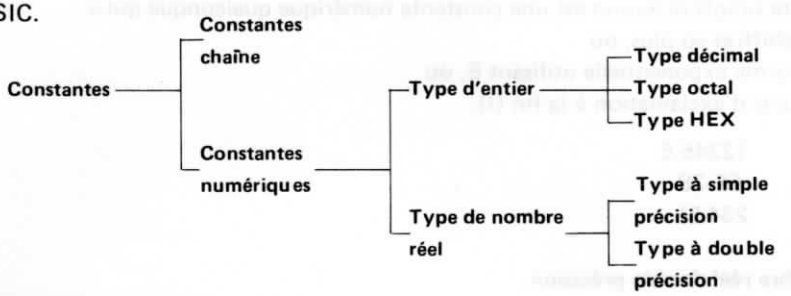
Exemples: LIST.
AUTO.

2.4 Jeu de caractère

Le jeu de caractères utilisable en BASIC est constitué de caractères alphabétiques (majuscules et minuscules), de caractères numériques (0 à 9), de symboles spéciaux et de caractères graphiques. Il existe aussi un jeu de caractères spéciaux de contrôle qui exécutent les fonctions spéciales définies par leurs codes.
Vous pouvez vous reporter à l'annexe F, "table des codes de caractères" et à l'annexe G "codes de contrôle" qui donnent des détails au sujet de ces caractères.
Reportez-vous également au paragraphe 2.10 au sujet du réglage du commutateur DIP et à la commande POKE du chapitre 3 pour le choix du caractère.

2.5 Constantes

Les constantes sont des valeurs utilisées par l'EPSON BASIC sans aucun changement pour l'exécution du programme. Les types de constantes suivants sont employés en EPSON BASIC.



2.5.1 Constantes chaînes

Une constante chaîne est une séquence d'un maximum de 255 caractères alphanumériques et symboles présentée entre guillemets (").

Exemples: "HX-20"
"Programme-1"

Les guillemets et les codes de contrôle ne peuvent pas être inclus dans les constantes chaînes. Utilisez la fonction CHRS pour traiter ces repères et codes comme des caractères.

2.5.2 Constantes numériques de type entier

(1) Type décimal
Tous les nombres compris entre -32768 et 32767 ou les nombres suivis par le signe "%". Un point décimal ne peut pas être ajouté à ces nombres.

Exemples: 12345
-9999
45678%

(2) Type octal
Nombres en octal de 0 à 7, préfixés par "&0" ou "&".
Les nombres en octal sont situés dans la gamme de &0 à &177777.

Exemples: &0123
&77777

(3) Type HEX
Les nombres hexadécimaux de 0 à F préfixés par "&H". Les nombres HEX sont situés dans la gamme allant de &H0 à &HFFFF.

Exemples: &H1F
&HABCD

2.5.3 Nombre réel simple précision

Les constantes simple précision sont stockées dans 7 chiffres significatifs. Six de ces 7 chiffres sont affichés après avoir arrondi le septième chiffre.
Une constante simple précision est une constante numérique quelconque qui a:

(1) sept chiffres au plus, ou
(2) une forme exponentielle utilisant E, ou
(3) un point d'exclamation à la fin (!).

Exemples: 12345.6
-5E-38
234.5!

2.5.4 Nombre réel double précision

Les constantes double précision sont stockées avec la précision de 16 chiffres significatifs. 16 chiffres significatifs, au maximum, sont affichés. Une constante double précision est une constante numérique quelconque qui a:

(1) huit chiffres au plus, ou
(2) une forme exponentielle utilisant D, ou
(3) un signe (#).

Exemples: 3.141592653
1.23 D 22
888.8 #

2.6 Variables

Les variables constituent une sorte d'emplacement qui contient les valeurs utilisées dans les programmes BASIC. Chaque variable correspond à un nom de variable constitué de caractères alphanumériques. Si une variable numérique est référencée avant qu'une valeur ne lui ait été affectée, 0 lui est affecté comme valeur numérique. Pour une variable chaîne, une chaîne nulle c'est à dire une chaîne de longueur 0 est affectée comme valeur de chaîne.

2.6.1 Noms de variables et caractères de déclaration

(1) Un nom de variable est représenté par une séquence de 255 caractères au maximum commençant par un caractère alphabétique. Seuls les 16 premiers caractères d'un nom sont identifiés. Un nom de variable peut contenir un mot réservé, mais ne peut pas commencer par un mot réservé. (les mots réservés sont des mots clés comme des commandes, des instructions et des fonctions). Les variables chaines commençant par les caractères alphanumériques "FN" ne sont pas admises. Dans les noms de variables, les lettres majuscules et minuscules sont considérées comme identiques.

(2) Les types de variables sont déterminés par le type de caractère de déclaration. Un caractère de déclaration choisi comme dernier caractère d'un nom de variable, définit le type de variable. Si le caractère de déclaration est omis, le type de variable est défini comme un nombre réel type à simple précision.

(3) Le BASIC distingue différents types de variables sous le même nom de variable.

Types de caractères de déclaration
- % variable entière
- ! variable simple précision
- # variable double précision
- $ variable chaîne

2.6.2 Variables tableaux

Un tableau correspond à une variable dont plusieurs éléments peuvent être référencés par un seul nom de variable. Chaque élément du tableau est référencé par une "variable tableau" ayant en indice un nombre entier. Les dimensions d'une variable tableau et la valeur maximale d'un indice dépendent de la capacité de la mémoire (voir DIM au Chapitre 3).

2.7 Changement de type

Lorsque cela est nécessaire, l'EPSON BASIC fait automatiquement passer une constante numérique d'un type à un autre.

(1) Si vous affectez des données numériques d'un type à une variable numérique d'un autre type, la constante numérique est changée en type déclaré dans le nom de variable et stockée sous cette forme. Si vous tentez d'affecter une variable numérique à une variable chaîne ou inversement, une erreur de «type» se produit.

Exemple: A%-32.34
Dans cet exemple, 32 sera stocké en A%

(2) Au cours de l'évaluation d'une expression, toutes les opérandes d'une opération arithmétique ou relationnelle sont converties, et le résultat sera envoyé avec la précision de l'opérande la plus précise.

Exemple: 10 #/3
10 #/3 #

Les deux expressions ci-dessus sont évaluées avec la même précision. Mais si vous ne faites pas attention au nombre de chiffres significatifs de la variable, ni à l'ordre d'exécution des opérations, il peut en résulter une erreur.

A # = 10/3 * 3 #
B # = 10 # / 3 * 3

Dans l'exemple ci-dessus, 9.999999761581421 sera stockée dans la variable A # et 1.0E + 01 sera stocké en B #

(3) Dans les opérations logiques, toutes les opérandes logiques sont converties en entiers, et un résultat entier est renvoyé.
Une erreur de «dépassement» interviendra si les opérandes ne sont pas dans la gamme: -32768 à 32767 lorsqu'elles ont été converties en entiers.

Exemple: A = NOT 123.456
Dans cet exemple, –124 sera stocké en A.

(4) Lorsqu'un nombre réel est converti en entier, la partie fractionnaire est tronquée. Si le nombre réel après conversion n'est pas un entier de la gamme -32768 à 32767, une erreur intervient aussi.

Exemple: A% = 777.7
778 sera stocké en A.

(5) Si une variable double précision est affectée à une variable simple précision, seuls les sept premiers chiffres arrondis seront stockés.

Exemple: P! = 3.141592653589793
3.141593 sera stocké en P! et affiché sous la forme
3.14159.

2.8 Expressions et opérations

Une expression désigne (1) une constante chaîne ou une constante numérique, (2) une variable chaîne ou une variable numérique, et (3) une combinaison de constantes et de variables chaînes ou numériques reliées par des opérateurs.

Exemple: "ABCD"
1.41421356
2 * 3
A + B/C

Le BASIC emploie les cinq types d'opérateurs suivants pour exécuter des opérations mathématiques ou logiques.

(1) Opérateurs arithmétiques
(2) Opérateurs relationnels
(3) Opérateurs logiques
(4) Opérateurs fonctionnels
(5) Opérateurs chaînes

2.8.1 Opérateurs arithmétiques

(1) Les opérateurs arithmétiques suivants sont employés en EPSON BASIC.

Opérateur	Opérations	Exemple d'expression
^	Elévation à une puissance	A^B
–	Négation	– A
*, /	Multiplication et division de nombres réels	A*B, A/B
+, –	Addition et soustraction	A +B, A–B

Les parenthèses permettent de changer l'ordre des opérations ; les opérations entre parenthèses étant effectuées les premières.
Voici les représentations en BASIC d'expressions algébriques.

EPSON BASIC	Expression algébrique
3*X+Y	3X+Y
X/Y–Z	(X÷Y)–Z
X^2+Y*3+4	X^2+3Y+4
Y^Y^Z	$(X^Y)^Z$
X*(–Y)	X(–Y)

(2) Division de nombres entiers et modulo-arithmétique
La division de nombres entiers est désignée par le symbole (\ ). Si les opérandes sont réelles, elles sont d'abord converties en entier avant l'exécution. Les quotients sont tronqués à des valeurs entières.

Exemple: A% = 10 \ 3
B% = 9.5 \ 3.3
La valeur stockée en A% et B% sera 3.

Le modulo-arithmétique désigné par l'opérateur MOD, renvoie la valeur entière du reste d'une division de nombres entiers.

Exemple: A%=10 MOD 3
1 sera stocké en A%.

(3) Division par zéro et débordement

Si une division par zéro intervient dans l'évaluation d'une expression, une erreur de «division par zéro» apparaît et l'exécution de l'opération s'arrête. Si le résultat d'une évaluation ou une valeur affectée dépasse le nombre des chiffres qui peuvent être admis par la variable, un «débordement» se produit et l'exécution s'arrête.

2.8.2 Opérateurs relationnels

Les opérateurs relationnels permettent de comparer deux valeurs. Le résultat de la comparaison est, soit «vrai» (-1), soit «faux» (0). Ce résultat servira ensuite à prendre une décision concernant le déroulement d'un programme.

Opérateur	Relation testée	Exemple d'expression
=	Egalité	X=Y
< >, ><	Inégalité	X<>Y, X><Y
<	Inférieur à	X<Y
>	Supérieur à	X>Y
<=, =<	Inférieur ou égal à	X<=Y, X=>Y
>=, =>	Supérieur ou égal à	X<=Y, X=>Y

Exemple: X = A = 0

Si A = 0, –1 sera stocké en X et

si A <> 0, 0 sera stocké en X.

2.8.3 Opérateurs logiques

Les opérateurs logiques servent à effectuer des tests sur des relations multiples, à faire des manipulations de bits ou des opérations booléennes. Les opérateurs logiques renvoient pour chaque bit un résultat qui peut être vrai (1) ou faux (0).

NOT (Négation)	
X	NOT X
1	0
0	1

AND (Produit logique)		
X	Y	X AND Y
1	1	1
1	0	0
0	1	0
0	0	0

OR (Addition logique)		
X	Y	X OR Y
1	1	1
1	0	1
0	1	1
0	0	0

XOR (Exclusive-OR)		
X	Y	X XOR Y
1	1	0
1	0	1
0	1	1
0	0	0
IMP (Implication)		
X	Y	X IMP Y
1	1	1
1	0	0
0	1	1
0	0	1
EQV (Equivalence)		
X	Y	X EQV Y
1	1	1
1	0	0
0	1	0
0	0	1

Les opérateurs logiques permettent de connecter deux ou plusieurs opérateurs relationnels pour prendre des décisions dans des cas complexes avec une seule instruction. Si la valeur indiquée par l'opérateur logique est -1 (vraie) ou 0 (faux) le résultat obtenu doit être, soit -1 soit 0.

Exemple: Si A < 0 ET B = 0 alors 100
Dans cet exemple, si A est négatif et B est égal à 0, le contrôle du programme fait un branchement à la ligne 100.

Les opérateurs logiques convertissent leurs opérandes en entiers compléments à deux dans la gamme -32768 à 32767 avant toute opération. Si les opérateurs ne sont pas dans cette gamme, une erreur intervient. L'opération indiquée est effectuée bit par bit sur ces entiers.

Exemple. A=21 AND 13
$21=(10101)_2$, $13=(01101)_2$
$(10101)_2$ AND $(01101)_2=(00101)_2$
$(00101)_2=5$
La valeur introduite en A est donc 5.

2.8.4 Opérateurs fonctionnels

Une fonction s'emploie dans une expression pour appeler une opération prédéterminée qui doit être exécutée sur un argument donné.
Le BASIC possède des "fonctions intrinsèques" constituées de fonctions numériques comme SIN, SQR, etc... et des fonctions chaînes comme RIGHTS , STRS , etc.. Vous trouverez des détails sur ces fonctions en vous reportant au chapitre 4.

Comme vous le verrez plus loin, au Chapitre 3 au sujet de DEFFN, le BASIC admets aussi des fonctions '"définies par l'utilisateur". Si un nombre réel est affecté à une fonction dont l'argument est normalement uniquement entier, la partie fractionnaire du nombre réel est arrondie à l'entier le plus proche et l'opération fonctionnelle est effectuée. Un nombre réel double précision, peut être affecté comme argument d'une fonction numérique. Mais dans ce cas, l'opération est effectuée en simple précision.

2.8.5 Opérations sur des chaînes

Les chaînes peuvent être concaténées en utilisant un signe (+).

Exemple: A$ = "ABC" + CHRS (34)
"ABC" sera stocké en A$.

Les opérateurs opérationnels suivants permettent aussi de comparer des chaînes :

=, <, >, <>, ><, <=, =<, >=, =>

Les chaînes sont comparées en prenant, à la fois, un caractère dans chaque chaîne. Si deux chaînes comparées sont identiques, les deux chaînes sont estimées "égales". Si un des codes diffère, le numéro de code de caractère inférieur précède le numéro supérieur. Dans une comparaison de chaînes, la chaîne la plus courte est considérée comme la plus petite. Les blancs du début et de la fin sont significatifs dans la comparaison des chaînes.

Exemple: "AAA" = "AAA"
"X" < "Y"
"XYZ" > "XY"
"ABC" > "A BC"

2.8.6 Ordre de priorité des opérations

Les opérations mathématiques et logiques sont effectuées dans l'ordre suivant:

1. Opérations entre parenthèses
2. Fonctions
3. Puissance (^)
4. Négation (-)
5. Multiplication et division de nombres réels (*, /)
6. Division d'entiers (\ )
7. Modulo de division d'entiers (MOD)
8. Addition et soustraction (+, —)
9. Opérandes relationnelles
10. NOT
11. AND
12. OR
13. XOR
14. IMP
15. EQV

2.9 Messages d'erreurs

Si l'EPSON BASIC détecte une erreur qui entraîne l'arrêt de l'exécution d'un programme, un message d'erreur est imprimé et le HX-20 retourne au niveau des commandes. Le format des messages d'erreurs en mode direct est:

XX Erreur

Le format en mode de programme est :

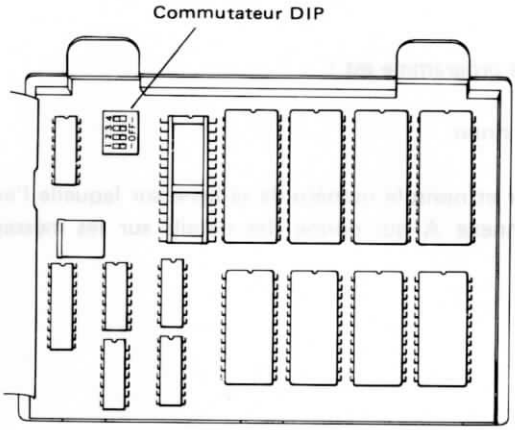
XX Erreur en nnnn

XX est le code erreur et nnnn le numéro de la ligne sur laquelle l'erreur a été détectée. Reportez-vous à l'Annexe A qui donne des détails sur les messages d'erreur BASIC.

2.10 Réglage du commutateur DIP

Vous pouvez changer le réglage du commutateur DIP (à double rangée de bornes) en ouvrant le capot arrière du HX-20 selon les indications ci-dessous.

NOTE:
Assurez-vous d'avoir débrancher l'appareil avant de régler le commutateur DIP.

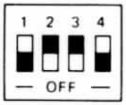


Emplacement du commutateur DIP

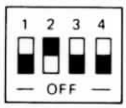
Les réglages du commutateur DIP sont les suivants pour les codes des caractères de chaque pays.

Clavier Français

France

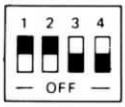


France ASCII

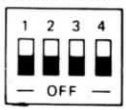


Clavier Danois

Danemark

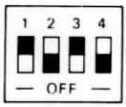


Suède ASCII

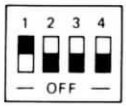


Clavier Allemand

Allemagne

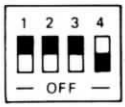


Allemagne ASCII

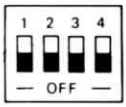


Clavier Norvégien (option)

Norvège

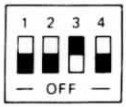


Suède ASCII

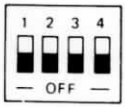


Clavier Suédois

Suède



Suède ASCII



NOTES :

1. *La broche No 4 du commutateur DIP peut rester en position ON ou OFF car elle n'influe pas sur le type de caractère choisi. Toutefois, cette broche doit être en position ON si l'on utilise un BASIC résident sur disque.*
2. *Il est déconseillé d'utiliser des réglages autres que ceux indiqués ci-dessus. Reportez-vous à l'Annexe D du manuel d'utilisation du HX-20 "Mode d'utilisation des claviers de chaque pays".*

Voici les différents jeux de caractères internationaux:

Country / Dec.code	ASCII	France	Allemagne	Danmark	Suède	Norvège
35 (23)	#	#	#	#	#	#
36 (24)	$	$	$	$	O	O
64 (40)	@	à	§	É	É	É
91 (5B)	[	°	Ä	Æ	Ä	Æ
92 (5C)	\	ç	Ö	Ø	Ö	Ø
93 (5D)	]	§	Ü	Å	Å	Å
94 (5E)	^	^	^	Ü	Ü	Ü
96 (60)	`	`	`	é	é	é
123 (7B)	{	é	ä	æ	ä	æ
124 (7C)	¦	ù	ö	ø	ö	ø
125 (7D)	}	è	ü	å	å	å
126 (7E)	~	¨	ß	ü	ü	ü

NOTE: Les chiffres entre parenthèses sont des codes hexadécimaux.

Vous pouvez, également, sélectionner chaque type de caractère par logiciel. (Reportez-vous à l'instruction POKE pour les informations complémentaires).

Comment utiliser les Chapitres 3 et 4

Toutes les commandes et instructions de l'EPSON BASIC sont expliquées dans l'ordre alphabétique pour faciliter la consultation des Chapitres 3 et 4. Toutes les commandes E/S concernant les périphériques sont expliquées globalement.
Vous trouverez des détails sur les opérations et les caractéristiques particulières des différents périphériques en vous reportant aux Chapitres 1 et 5.
Les commandes et instructions utilisant comme préfixe l'astérisque " " sont particulières à ce BASIC. Etant donné que ces commandes et instructions agissent différemment des commandes et instructions BASIC classiques, lisez attentivement les explications de chacune de ces commandes. Les descriptions des commandes et instructions BASIC sont présentées avec le format suivant :

FORMAT Explique comment écrire la commande ou l'instruction. Suivre la syntaxe indiquée ci-dessous lorsque vous introduisez chaque commande ou instruction.

1. Introduisez les éléments présentés en majuscules en utilisant soit des majuscules, soit des minuscules. Faites cette distinction, en particulier pour les éléments indiqués entre guillemets (noms de fichiers, etc)
2. Les éléments présentés entre "< >" sont spécifiés par l'utilisateur.
3. Les éléments présentés entre crochets "[ ]" sont facultatifs et peuvent être omis. Si vous omettez ces éléments, le HX-20 introduira des valeurs implicites ou des valeurs préalablement spécifiées.
4. Les symboles différents de ceux qui viennent d'être décrits, par exemple les parenthèses, deux points, point virgule, trait d'union, signe égal, etc, doivent être introduits exactement à l'endroit où ils sont présentés dans le manuel.
5. Les éléments en option présentés par "..." peuvent être répétés autant de fois qu'il est nécessaire jusqu'à un maximum de 255 caractères.
   Exemples: <variable>[, <variable>...]
   A, BS, C!, D, etc... peuvent être répétés.
6. Lorsque deux ou plusieurs éléments apparaissent entre deux barres verticales, cela signifie que vous devez spécifier la fonction désirée en choisissant l'un des éléments que vous allez introduire.

   Exemples : **THEN** **<instruction>**
   <numéro de ligne>
   GOTO **<numéro de ligne>**
   Dans cette exemple, les choix possibles sont :
   THEN **<instruction>**
   THEN **<numéro de ligne>**
   GOTO **<numéro de ligne>**

7. En règle générale, le BASIC ignore les espaces. Mais les espaces ne sont pas admis à l'intérieur des noms de variables et des mots clés. Si une variable est suivie d'un mot clé, un espace doit être prévu pour les séparer afin que l'opération soit correcte.

OBJET Définit rapidement comment la commande ou l'instruction fonctionne.

EXEMPLE Ceci donne un exemple simple d'entrée réelle.

REMARQUES Donne des conseils sur l'emploi correct de chaque commande ou instruction, ainsi qu'une définition de leurs fonctions.

EXEMPLE DE PROGRAMME Présente des exemples de programmes utilisant la commande ou l'instruction à titre d'exercice.

Dans le HX-20, cinq programmes sont gérés indépendamment l'un de l'autre dans leurs zones respectives. Par conséquent, pour éviter la destruction ou la perte accidentelle de données ou de programmes importants dans la mémoire, veillez à vérifier la zone des programmes en utilisant une commande STAT ou LIST avant de modifier un programme quelconque.

Chapitre 3

Commandes et Instructions

AUTO

FORMAT **AUTO [<numéro de ligne>][, [<incrément>]]**

OBJET Créer un numéro de ligne automatiquement.

EXEMPLE
AUTO 100, 10
AUTO 200,
AUTO 300
AUTO

REMARQUES La commande AUTO génère automatiquement un numéro de ligne qui sera incrémenté de la quantité <incrément> à chaque fois que vous appuierez sur la touche **RETURN**, ce qui vous aidera à créer vos programmes efficacement. La valeur implicite du <numéro de ligne> et de <incrément> est 10. Si <numéro de ligne> est suivi d'une virgule mais que <incrément> n'est pas spécifié, le dernier incrément spécifié dans une commande AUTO sera implicitement adopté.

Si AUTO génère un numéro de ligne déjà utilisé dans votre programme, un astérisque sera imprimé après le numéro de ligne pour vous avertir que toute entrée ultérieure suivie par la touche **RETURN** remplacera la ligne existante. Mais si la touche **RETURN** est tapée immédiatement après l'astérisque, sans aucune autre entrée, la ligne sera sauvegardée.

AUTO est terminé en tapant **CTRL** + C ou la touche **BREAK**. Lorsque vous avez appuyé sur la touche **CTRL** + C ou sur la touche **BREAK** le BASIC revient au niveau des commandes. L'écran peut être édité alors que AUTO est en vigueur. Donc, si vous déplacez le curseur sur la ligne affichée et que vous appuyez sur **RETURN**, cette ligne va être introduite. Si l'instruction ainsi introduite est une ligne de programme (c'est à dire qu'elle a un numéro de ligne), la prochaine ligne générée aura le numéro de cette ligne auquel sera ajouté l' <incrément>. Si l'instruction introduite n'a pas de numéro de ligne, le programme quittera le mode AUTO après exécution de cette instruction.

CLEAR

FORMAT **CLEAR [< mémoire pour chaînes de caractères >] , < mémoire pour fichier RAM >]]**

OBJET Initialiser des variables et fixer la dimension de la mémoire des chaînes de caractères et des fichiers RAM.

EXEMPLE **CLEAR 200, 256**

REMARQUES La commande CLEAR remets toutes les variables numériques à 0 et toutes les variables chaînes nulles. CLEAR ferme aussi tous les fichiers ouverts par OPEN et annule toutes les données définies par les instructions DEF (DEFFIL, DEF FN, DEF USR, FEFINT, etc...). < la mémoire pour chaîne de caractères > est la dimension en octets de la zone de mémoire utilisée par le BASIC pour le traitement des chaînes de caractères.

< la mémoire pour chaînes de caractères > est fixée à 200 octets à chaque démarrage à chaud. Pour l'exécution d'opérations sur un grand nombre de caractères, spécifiez une < mémoire pour chaînes de caractères > suffisamment grande, sinon une erreur "OS" apparaîtra.

< la mémoire pour fichiers RAM > indique en octets la zone de mémoire à utiliser pour le stockage des fichiers RAM. < la zone pour fichier RAM > est fixée à 256 octets à chaque démarrage à froid et n'est pas modifiée par un démarrage à chaud.

Lorsque vous utilisez une commande CLEAR pour fixer < la mémoire des fichiers RAM > , vous ne pouvez pas omettre la mémoire pour chaînes de caractères. Notez aussi que si vous modifiez la dimension d'un fichier RAM dans lequel des données sont stockées, et que la nouvelle dimension est inférieure à la précédente, les données en surplus de la nouvelle dimension seront annulées.

EXEMPLE DE PROGRAMME

```
LIST
100 FOR I=1 TO 10
110 A(I)=I : NEXT I
120 GOSUB 200
130 CLEAR
140 GOSUB 200
150 END
200 FOR I=1 TO 10
210 PRINT USING"####";A(I);
220 NEXT I
230 PRINT
240 RETURN
RUN
   1   2   3   4   5   6   7   8   9  10

   0   0   0   0   0   0   0   0   0   0

>
```

CLOSE

FORMAT **CLOSE [[#]< numéro de fichier >, [#]< numéro de fichier >...]]**
OBJET Fermer des fichiers
EXEMPLE **CLOSE # 3**

REMARQUES CLOSE ferme un fichier spécifié par <numéro de fichier>. Le <numéro de fichier > est le numéro sous lequel le fichier a été ouvert. Le fichier peut être réouvert en utilisant le même numéro ou un numéro différent ; ce numéro de fichier peut, de la même manière, être réutilisé par la suite pour ouvrir n'importe quel fichier.
Une instruction CLOSE sans <numéro de fichier> ferme tous les fichiers ouverts au moment de l'exécution de l'instruction. Le symbole # avant < numéro de fichier > peut être omis. Notez que les instructions END et NEW ferment toujours automatiquement tous les fichiers mais qu'une instruction STOP ne ferme aucun fichier.
Lorsqu'un fichier de sortie a été ouvert, il doit être refermé pour terminer correctement le traitement de sortie des données restant dans le tampon. Après l'exécution de CLEAR, LOGIN, NEW, DELETE, WIDTH, LOAD, RUN ou MERGE, ou l'édition d'un programme, tous les fichiers ouverts à ce moment seront fermés .

(Voir END, OPEN et 5.2, fichier séquentiel)

CLS

FORMAT **CLS**
OBJET Effacer un écran texte
EXEMPLE **CLS**

REMARQUES CLS efface uniquement l'écran texte (écran virtuel) du LCD ou d'un écran extérieur et ramène le curseur à sa position de repos (c'est à dire dans l'angle supérieur gauche de l'écran virtuel). Mais, lors de l'exécution de CLS, si un écran texte et un écran graphique sont tous deux affichés sur le LCD, l'instruction CLS annulera également l'écran graphique. CLS fonctionne de la même façon que PRINT CHRS (12).

(Voir GCLS)

COLOR

FORMAT **COLOR [< couleur texte >][, [< couleur fond >] [, < numéro de l'ensemble >]]**

OBJET Spécifier les couleurs de l'écran de l'affichage extérieur.

EXEMPLE **COLOR 0, 3, 0**

REMARQUES COLOR désigne la couleur des textes (c'est à dire la couleur des lignes et des caractères) et la couleur du fond d'un affichage extérieur. Cette commande est efficace uniquement si le HX-20 est connecté à un affichage extérieur (par exemple, téléviseur) par l'intermédiaire du contrôleur d'affichage en option.

Vous pouvez spécifier les couleurs en utilisant les codes 0 et 3. La couleur des textes et la couleur du fond peuvent être spécifiées indépendamment. Vous disposez d'un total de 8 couleurs d'affichage. Quatre couleurs maximum peuvent être obtenues simultanément et vous pouvez choisir entre deux ensembles de couleurs en utilisant les codes " 0 " ou " 1 ". Les codes des couleurs sont les suivants:

	Code de couleur	
Ensemble 0	0:	vert
	1:	jaune
	2:	bleu
	3:	rouge
Ensemble 1	0:	blanc
	1:	bleu clair
	2:	violet
	3:	orange

Les valeurs implicites au démarrage à chaud sont les suivantes:
< couleur de texte > : 1
< couleur du fond > : 0
< ensemble > : 0

***NOTE** : Le contrôleur d'affichage pourra ne pas être disponible à la date de parution du manuel et cette commande est inopérante sur l'écran LCD.*

(Voir SCREEN)

CONT

FORMAT **CONT**

OBJET Reprendre l'exécution d'un programme qui a été arrêté.

EXEMPLE **CONT**

REMARQUES Si vous appuyez sur la touche **BREAK** au cours de l'exécution d'un programme ou si un STOP est exécuté dans un programme, l'exécution du programme s'arrête et le message «interruption en XXXX» est affiché. A ce moment, l'exécution du programme reprend si vous introduisez une commande CONT.

CONT s'emploie habituellement avec une instruction STOP pour le déverminage d'un programme.

Si vous appuyez sur la touche **BREAK** pendant l'opération E/S sur une imprimante ou une cassette, le message «abandon» sera affiché. Dans ce cas, l'exécution du programme ne pourra pas être reprise par une commande CONT. Cette commande est aussi invalide si le programme est édité après avoir été arrêté par la touche **BREAK** ou l'instruction STOP.

COPY

FORMAT **COPY**

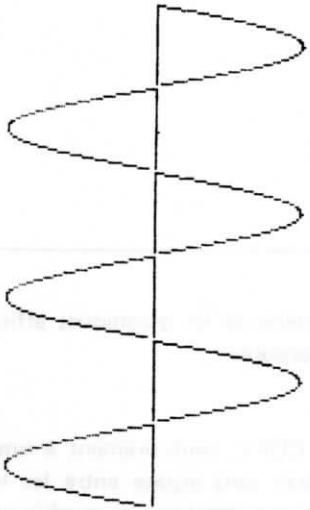
OBJET Sortir les caractères et les graphiques affichés par le LCD sur la micro-imprimante incorporée.

EXEMPLE **COPY**

REMARQUES La commande COPY, contrairement à une instruction LPRINT, délivre une copie d'écran sans espace entre les lignes. Donc, avec cette commande, vous pouvez imprimer les graphiques tracés en utilisant les instructions LINE et PSET exactement comme elles sont affichées sur l'écran.

EXEMPLE DE PROGRAMME

```
100 PI=3.141592
110 DX=PI/64
120 CLS
130 PRINTCHR$(23)
140 LINE(60,31)-(60,0),P
SET
150 FOR Y=0 TO 63
160 A=SIN(X)*59
170 X=X+DX
180 LINE-(A+60,Y/2),PSET
190 NEXT Y
200 COPY
210 GOTO 120
220 END
```



DATA

FORMAT **DATA <constante>[,<constante>...]**

OBJET Stocker les constantes numériques et les chaînes auxquelles on accède par la ou les instructions(s) READ.

EXEMPLE **DATA HX, 20, EPSON**

REMARQUES Les instructions données ne sont pas exécutables et peuvent être placées n'importe où dans le programme et vous pouvez utiliser un nombre quelconque d'instructions DATA dans un programme.

Vous pouvez utiliser un type de constante quelconque pour la constante (exemple: simple précision, double précision, entier), tandis qu'aucune expression numérique (exemple. 3 4) n'est admise. Le type de variable (numérique ou chaîne) indiqué dans l'instruction READ doit correspondre à la constante correspondante de l'instruction DATA.

Chaque espace <constante> dans une instruction DATA doit être séparée par des virgules. Les constantes chaînes doivent être présentées entre guillemets si elles contiennent des virgules, des points ou des espaces significatifs en début ou à la fin. Sinon, les guillemets ne sont pas nécessaires.

(Voir READ et RESTORE)

EXEMPLE DE PROGRAMME

```
10 READ A,B,C
20 PRINT A;B;C
30 DATA 1,2,3,4,5,6,7,8,
9,10
40 READ A,B,C
50 PRINT A;B;C
60 READ A,B,C
70 PRINT A;B;C
80 END
```

```
 1  2  3
 4  5  6
 7  8  9
```

DEFFIL

FORMAT DEFFIL <longueur d'enregistrement>, <adresse relative>

OBJET Définir l'adresse relative de l'enregistrement 0 d'un fichier RAM et la longueur d'enregistrement.

EXEMPLE DEFFIL 20, 200

REMARQUES L' <adresse relative> est le nombre d'octets depuis l'adresse de début de zone de fichier RAM, spécifiée par une instruction CLEAR. La <longueur d'enregistrement> indique la longueur en octets à utiliser pour un enregistrement du fichier RAM. Au moment du démarrage à chaud du BASIC, la <longueur d'enregistrement> est fixée à 255 octets et l'adresse relative à 0. L'exécution d'une instruction en CLEAR va aussi annuler la <longueur d'enregistrement> et <l'adresse relative> définies auparavant par l'instruction DEFFIL et les valeurs implicites deviendront respectivement 255 et 0 au moment du démarrage à chaud.

(Voir GET%, PUT% et 5.1 Fichiers RAM).

EXEMPLE DE PROGRAMME

```
LIST

100 DEFINT I,J,K
110 DEFFIL 2,0
120 FOR I=0 TO 15
130 PUT% I,I
140 PRINT USING "  & &";HEX$(I);
150 NEXT I:PRINT
160 DEFFIL 2,1
170 FOR J=0 TO 15
180 GET% J,K
190 PRINT USING "  & &";HEX$(K);
200 NEXT J
210 END
RUN
  0    1    2    3    4    5    6    7
  8    9    A    B    C    D    E    F

  0    100  200  300  400  500  600  700
  800  900  A00  B00  C00  D00  E00  F00
>
```

DEF FN

FORMAT **DEF FN <nom>[(<paramètre>[,<paramètre>...])]= <définition de fonction>**

OBJET Définir une fonction créée par l'utilisateur

EXEMPLE **DEF FNZ (X,Y) =X*2 +Y*3+A**

REMARQUES <nom> est le nom d'une fonction et doit commencer par des caractères alphabétiques, sauf les mots réservés (il est donc soumis à la même limitation que les noms de variables). La liste des paramètres comprend les noms de variables de la définition de fonctions qui doivent être remplaçés lorsque la fonction est appelée. Les éléments de la liste sont séparés par des virgules. La < définition de fonction > est une expression qui indique l'opération de la fonction. Elle est limitée à une ligne.

Les noms de variables qui apparaissent dans l'expression servent uniquement à définir la fonction ; ils n'affectent pas les variables du programme qui ont le même nom. Si un nom de variable utilisé dans la < définition de fonction > n'apparaît pas dans la liste des arguments, la valeur en cours de la variable est utilisée.

Les fonctions utilisateurs peuvent être des fonctions numériques, chaînes ou une association des deux. Mais, le type d'argument défini doit correspondre à la variable réellement appelée. L'instruction DEF FN doit être exécutée pour définir une fonction avant qu'elle ne soit appelée. Si une fonction est appelée avant d'être définie, une erreur UF (fonction utilisateur non définie) apparaît.

EXEMPLE DE PROGRAMME

```
100 DEFFNA(B,C,R)=SQR(B^
2+C^2-2*B*C*COS(R/180*3.
1415926))
110 INPUT"SIDE B=";B
120 INPUT"SIDE C=";C
130 INPUT"ANGLE R=";R
140 A=FNA(B,C,R)
150 PRINT "SIDE A= ";A
160 GOTO 110
170 END
```

DEFINT/SNG/DBL/STR

FORMAT DEF | INT | SNG | DBL | STR | <gamme(s) de lettres>

OBJET Déclarer que les variables sont du type entier, simple précision, double précision et chaînes.

EXEMPLE DEFSTR A, X – Z

REMARQUES DEFINT, DEFSNG, DEFDBL et DEFSTR déclarent que les noms de variables commençant par la ou les lettres spécifiées par < gamme(s) de lettres > seront du type entier, simple précision, double précision et chaîne, respectivement. Pour déclarer une variable ou une gamme de variables, une seule lettre ou deux lettres reliées par un trait d'union seront employées comme lettres à spécifier par < gamme(s) de lettres >. Pour déclarer deux ou plusieurs gammes de variables, la ou les lettres spécifiant chaque gamme de lettres seront séparées par une virgule. Mais, un caractère de déclaration de type a toujours la priorité sur une instruction de définition de type dans la frappe d'une variable. Si le BASIC ne rencontre aucune instruction de déclaration, il suppose que toutes les variables sans caractère de déclaration sont des variables simple précision.

EXEMPLE DE PROGRAMME

```
LIST

100 DEFINT A-C
110 DEFSNG E-G
120 DEFDBL I-K
130 A =123.456789
140 B#=123.456789
150 D =123.456789
160 E =789.123456
170 F%=789.123456
180 H#=789.123456
190 I =0.98765432
200 J%=0.98765432
210 L =0.98765432
220 PRINT A ,B#,D
230 PRINT E ,F%,H#
240 PRINT I ,J%,L
250 END
RUN
 123            123.456789     123.457
 789.124        789            789.123456

 .98765432      1              .987654
>
```

DEF USR

FORMAT **DEF USR [<chiffre>] = <adresse de départ>**

OBJET Spécifier l'adresse de départ d'un sous-programme en langue machine

EXEMPLE **DEF USR6 = &H0C00**

REMARQUES DEF USR spécifie l'adresse de départ d'un sous-programme en langue machine appelé par la fonction USR. <chiffre> peut être un chiffre quelconque de 0 à 9. Au total, 10 fonctions USR sont admises. Si <chiffre> est omis, 0 est pris comme valeur implicite.

(Voir USR et 5.3, Programmes en Langage Machine)

EXEMPLE DE PROGRAMME

```
MEMSET &H0A41

LIST

100 POKE &H0A40,&H39
110 DEFUSR0=&H0A40
120 ' &H39 = RETURN
130 A=USR(0)
140 PRINT A
150 END
RUN
 0
>
```

DELETE

FORMAT **DELETE [<numéro de ligne de départ>][—[<numéro de ligne de fin>]]**

OBJET Supprimer des lignes spécifiées d'un programme.

EXEMPLE
DELETE 100–200
DELETE 100–
DELETE –200
DELETE 100

REMARQUES La commande DELETE supprime la totalité ou une partie du programme avec lequel on travaille, délimité par le <numéro de ligne de départ> et le <numéro de ligne de fin>. Si , seul le <numéro de ligne de départ> est spécifié, seule cette ligne du programme sera supprimée. Le numéro de ligne de départ suivi par un trait d'union supprime cette ligne ainsi que toutes les lignes de numéros supérieurs. Si le <numéro de ligne de fin> précédé par un trait d'union est spécifié, toutes les lignes depuis le début du programme jusqu'à la ligne en question sont supprimées. Lorsque la commande DELETE est exécutée, tous les fichiers ouverts par la commande OPEN à l'instant de l'exécution sont fermés.

(Voir LOGIN)

EXEMPLE DE PROGRAMME

```
100 'DELETE ELIMINATES
110 'UNEEDED LINES
120 'DELETE LINES 120 TO
 140
130 '****UNEEDED LINE***
140 '****EXCESS LINE***
150 '##IMPORTANT LINE##
160 '##NECESSARY LINE##

DELETE 120-140

100 'DELETE ELIMINATES
110 'UNEEDED LINES
150 '##IMPORTANT LINE##
160 '##NECESSARY LINE##
```

DIM

FORMAT **DIM < variable > (< variable maximale d'indice >[, < valeur maximale d'indice > ...])[,...]**

OBJET Déclarer la dimension des éléments d'une variable réseau

EXEMPLE **DIM A(40, 10), B$ (50)**

REMARQUES DIM spécifie la valeur maximale des indices des variables tableau et alloue un espace mémoire correspondant.

Si un nom de variable tableau est utilisé sans instruction DIM, la valeur maximale de son ou ses indices est supposée être 10. La valeur minimale d'un indice est toujours 0, sauf si 1 a été spécifié dans l'instruction OPTION BASE. Si un indice supérieur au maximum spécifié est utilisé, une erreur BS ("mauvais indice") apparaît. L'instruction DIM mets tous les éléments du tableau numérique spécifié à la valeur initiale de zéro et ceux du tableau de caractères spécifié à la valeur nulle.

(Voir ERASE et OPTION BASE)

END

FORMAT **END**

OBJET Fermer tous les fichiers et achever l'exécution d'un programme.

EXEMPLE **END**

REMARQUES END achève l'exécution d'un programme après avoir fermé tous les fichiers et ramené le BASIC au niveau des commandes.
Une instruction END peut apparaître n'importe où et aussi souvent qu'il le faut dans un programme. L'instruction END est facultative à la fin d'un programme. Dans ce cas, aucun fichier ne sera fermé.

ERASE

FORMAT ERASE <variable tableau>[,<variable tableau>...]

OBJET Eliminer les tableaux d'un programme.

EXEMPLE ERASE A,B

REMARQUES ERASE élimine les tableaux préalablement dimensionnés et spécifiés par la liste des variables tableaux. Lorsque les tableaux ont été effacés, ils peuvent être à nouveau dimensionnés au moyen d'une instruction DIM. Si une tentative est faite pour redimensionner un tableau sans l'effacer d'abord, une erreur DD ("double définition") apparaît.

(Voir DIM)

EXEMPLE DE PROGRAMME

```
LIST

100 OPTION BASE 1
110 D=10
120 FOR I=1 TO 10
130 A(I)=11-I
140 NEXT I
150 GOSUB 300
160 ERASE A
170 D=20
180 DIM A(20)
190 FOR I=1 TO 20
200 A(I)=I
210 NEXT I
220 GOSUB 300
230 END
300 FOR I=1 TO D
310 PRINT USING"####";A(I);
320 NEXT I
330 PRINT
340 RETURN
RUN
  10   9   8   7   6   5   4   3   2   1

   1   2   3   4   5   6   7   8   9  10
  11  12  13  14  15  16  17  18  19  20

>
```

ERROR

FORMAT ERREUR <expression entière>

OBJET Simuler l'apparition d'une erreur ; ou permettre à l'utilisateur de définir des codes d'erreur.

EXEMPLE ERREUR 225

REMARQUES Lorsqu'une instruction ERREUR est exécutée sans ON ERROR GOTO, le message du code d'erreur correspondant à l'<expression entière>sera délivré et l'exécution du programme s'arrêtera.

Si une instruction ERREUR spécifie un code pour lequel aucun message d'erreur n'a été défini, le HX-20 répondra par un message" UP Error".

Dans tous les cas, la valeur du code d'erreur sera affectée à la variable ERR et le numéro de ligne sur laquelle l'instruction ERROR est apparue sera affecté à la variable ERL.

Un code d'erreur qui n'a pas été prédéfini, peut être défini par l'utilisateur au moyen d'une instruction ERROR et ce code peut à son tour être utilisé dans un programme de traitement d'erreurs avec une instruction ON ERROR GOTO. <expression entière>doit être comprise entre 1 et 255.

(Voir ON ERROR GOTO, RESUME, ERL/ERR et Annexe A, Messages d'erreurs).

EXEMPLE DE PROGRAMME

```
LIST

100 INPUT "ERROR NUMBER"
;A
110 ERROR A
120 END

RUN
ERROR NUMBER? 78
UP Error In 110

RUN
ERROR NUMBER? 4
OD Error In 110

RUN
ERROR NUMBER? 24
WE Error In 110
>
```

EXEC

FORMAT **EXEC** [< **adresse de départ** >]

OBJET Démarrer l'exécution d'un sous-programme en langage machine

EXEMPLE **EXEC &H0C00**

REMARQUES EXEC exécute un sous-programme en langage machine à partir de l'<adresse de départ >. Mais avant l'exécution d'une commande EXEC le sous-programme en langage machine doit être chargé dans la mémoire par l'exécution d'une commande LOADM ou d'une instruction du même type.

Lorsque EXEC est utilisé dans un programme, l'exécution du programme BASIC se poursuit avec l'instruction BASIC qui suit immédiatement l'instruction EXEC lorsque le sous-programme en langage machine a été terminé. Lorsque l' < adresse de départ >n'est pas spécifiée, l'exécution commence à l' <adresse de départ > de la commande LOADM ou de la commande EXEC exécutée au préalable. Les commandes BASIC sont invalidées pendant l'exécution d'un sous-programme en langage machine. Le programme BASIC et le fichier RAM peuvent être modifiés par un sous-programme en langage machine et il faut prendre beaucoup de précautions pour éviter leur destruction.

(Voir LOADM, USR et paragraphes 5.3 Programmes en langage machine)

FILES

FORMAT FILES [":<nom de dispositif>"]

OBJET Afficher les noms de tous les fichiers résidents dans une mémoire spécifiée.

EXEMPLE FILES "CAS1:"

REMARQUES Les informations suivantes sont affichées pour tous les fichiers résidant dans la mémoire auxiliaire définie par le <nom de dispositif> : nom de fichier, type de fichier, classification, format d'enregistrement. La classification est affichée avec un seul chiffre.

0: fichier programmes BASIC
1: fichier de données
2: fichier programmes en langage machine

Le format d'enregistrement est présenté sous la forme "A" ou" B"

A: format ASCII
B: format binaire

Si le <nom de dispositif>est omis, une recherche est faite parmi tous les périphériques actuellement connectés et le dispositif ayant la priorité la plus élevée (c'est à dire le dispositif implicite) est adopté. Si aucun périphérique n'est connecté, "CAS1:" est adopté. En BASIC, il existe un dispositif de détection des fins de fichiers, mais il n'en existe pas pour la détection des fins de bandes. C'est pourquoi, lorsque "CAS1:" ou "CAS0:" est spécifié comme <nom de dispositif>, la cassette de bande sera utilisée indéfiniment. Vous devez déterminer la fin de la bande et appuyer sur la touche **BREAK** au moment approprié pour arrêter l'exécution.

FOR...TO...STEP — NEXT

FORMAT **FOR <variable>=<valeur initiale>TO <valeur finale>**
[STEP <incrément>]
•
•
•

OBJET **NEXT [<variable>[, <variable>...]]**
Répéter un certain nombre de fois les instructions placées entre les instructions FOR et NEXT.

EXEMPLE
```
FOR I=0 TO 100 STEP 5
•
•
•
NEXT I
```

REMARQUES La < variable > est utilisée comme un compteur. Le compteur pointe d'abord sur la valeur spécifiée par la <valeur initiale>, le BASIC exécute les lignes de programmes qui suivent l'instruction FOR jusqu'à l'instruction NEXT. Ensuite, le compteur est incrémenté de l'<incrément> spécifié par STEP et une vérification est effectuée pour s'assurer que la valeur du compteur n'est pas supérieure à la < valeur finale >. Si elle n'est pas supérieure le BASIC revient à la ligne de programme qui suit l'instruction FOR pour répéter le même traitement. Si cette valeur est supérieure, l'exécution se poursuit avec l'instruction qui suit l'instruction NEXT.
Des expressions numériques peuvent être utilisées pour les <valeur initiale>, < valeur finale > et < incrément >. Si STEP n'est pas spécifié, l' < incrément > sera supposé être égal à un. Si < incrément > est négatif, la <valeur finale> du compteur est fixée à une valeur inférieure à la <valeur initiale>.

Si l' < incrément > est positif et la <valeur initiale> du compteur est supérieure à la < valeur finale > ou si l'< incrément > est négatif et la <valeur initiale> du compteur n'est pas supérieure à la <valeur finale>, aucune boucle FOR...NEXT ne sera exécutée, mais la <valeur initiale> sera affectée à la <variable>.
Les boucles FOR...NEXT peuvent être imbriquées, c'est à dire qu'une boucle FOR...NEXT peut être plaçée à l'intérieur d'une autre boucle FOR...NEXT. Lorsque des boucles sont imbriquées, chaque boucle doit avoir un nom de variable unique qui lui sert de compteur. L'instruction NEXT de la boucle intérieure doit apparaître avant celle de la boucle extérieure.
Si l'instruction NEXT se rencontre avant l'instruction FOR qui lui correspond, un message d'erreur "NEXT sans FOR" est émis et l'exécution du programme s'arrête.

Si des boucles imbriquées ont le même point final, une instruction NEXT unique peut être utilisée pour toutes les boucles, à condition de séparer les variables à représenter par des virgules (Exemple: NEXT I,J,K).

EXEMPLE DE PROGRAMME

```
LIST

100 FOR I=30 TO 0 STEP -2
110 FOR J=1 TO I
120 PRINT" ";
130 NEXT J
140 FOR K=1 TO 30-I
150 PRINT"*";
160 NEXT K
170 PRINT
180 NEXT I
190 END
RUN
                              **
                             ****
                            ******
                           ********
                          **********
                         ************
                        **************
                       ****************
                      ******************
                     ********************
                    **********************
                   ************************
                  **************************
                 ****************************
>
```

GCLS

FORMAT GCLS
OBJET Effacer un écran graphique
EXEMPLE GCLS

REMARQUES GCLS efface uniquement l'écran graphique affiché sur l'écran LCD ou extérieur. Si un écran graphique et un écran texte sont affichés à la fois sur le LCD, l'écran texte restera tel quel.

(Voir CLS)

GET%

FORMAT **GET% < numéro d'enregistrement >, < nom de variable >[, < nom de variable > ...]**
OBJET Lire des données d'un fichier RAM pour les utiliser comme variables.
EXEMPLE **GET% 0, A!, B # , C$**

REMARQUES L'instruction GET% lit des données de l'enregistrement d'un fichier RAM spécifié par le < numéro d'enregistrement > et les introduit dans les variables spécifiées par la liste des noms de variables. Les < numéros d'enregistrement > commencent à zéro.
Avant d'exécuter une instruction GET%, la longueur d'un seul enregistrement et la position du fichier contenant cet enregistrement doivent être définies par une instruction DEFFIL. Les types de noms de variables de l'instruction PUT%. Egalement lorsque vous utilisez des noms de variables chaînes, ils doivent être ajoutés à la fin de la liste des variables. Lorsque GET% et PUT% doivent être utilisés avec des instructions multiples DEFFIL, il faut bien faire attention à la longueur et au numéro d'enregistrement de chaque enregistrement, ainsi qu'à la méthode de stockage des valeurs des variables.

(Voir DEFFIL, PUT% et au 5.1 Fichiers RAM)

GOSUB — RETURN

FORMAT **GOSUB <numéro de ligne >**

•
•
•

OBJET **RETURN**

EXEMPLE Branchement à un sous-programme et retour de sous-programme.
GOSUB 500

REMARQUES GOSUB entraîne une interruption inconditionnelle de l'exécution d'un programme en transférant le contrôle à un sous-programme défini par le <numéro de ligne> de la première ligne du sous-programme. Le contrôle du programme revient à la ligne qui suit l'instruction GOSUB la plus récente lorsqu'une instruction RETURN est exécutée dans le sous-programme. Les sous-programmes sont des portions indépendantes d'un programme principal comprenant chacune une instruction RETURN à la fin du programme. L'un quelconque de ces sous-programmes peut être appelé un nombre quelconque de fois dans un programme par une instruction GOSUB. Un sous-programme peut aussi être appelé depuis un autre sous-programme Une telle imbrication des sous-programmes n'est limitée que par la mémoire disponible. Un sous-programme peut contenir plusieurs instructions RETURN, si la logique entraîne un retour à des points différents du sous-programme et si chaque instruction RETURN correspond correctement à une instruction GOSUB.

EXEMPLE DE PROGRAMME

```
10 INPUT "X=";X
20 INPUT "Y=";Y
30 Z=X
40 GOSUB 200
50 X1=ANS
60 Z=Y
70 GOSUB 200
80 X2=ANS
90 PRINT "X=";X,"Y=";Y
100 PRINT "(2*X+3)^2+(2*
Y+3)^2"
110 PRINT "=";X1+X2
120 END
200 ANS=(2*Z+3)^2
210 RETURN
```

GO TO/GOTO

FORMAT (1) **GO TO <numéro de ligne>, ou**
(2) **GOTO <numéro de ligne>**

OBJET Brancher l'exécution d'un programme à un numéro de ligne particulier.

EXEMPLE **GOTO 300**

REMARQUES Lors de l'exécution d'une instruction GOTO, le contrôle du programme se branche inconditionnellement à un numéro de ligne spécifiée.
Les formats (1) et (2) exécutent exactement la même fonction.

IF...THEN...ELSE/IF...GOTO...ELSE

FORMAT

IF <expression logique>
| THEN | <instruction> | [ELSE | <instruction> |]
| | <numéro de ligne> | | <numéro de ligne> |
| GOTO | <numéro de ligne> |

OBJET

Choisir une voie particulière pour l'exécution d'un programme d'après des conditions établies dans une expression logique.

EXEMPLE

IF A>10 THEN A=0 ELSE 200

REMARQUES

IF contrôle l'exécution du programme d'après les conditions établies dans l'<expression logique>. Si le résultat de l'<expression logique>n'est pas zéro (c'est à dire vrai), l'instruction THEN ou GOTO est exécutée. Si le résultat est zéro (c'est à dire faux), l'instruction ELSE est exécutée. Si l'instruction ELSE est omise, l'exécution continue avec la prochaine instruction exécutable. A l'intérieur de l'instruction IF... THEN... ELSE, les instructions IF distinctes peuvent être imbriquées pour créer une instruction multiplexée, qui se limite toutefois à une seule ligne.

EXEMPLE DE PROGRAMME

```
100 DEFSTR A-Z
110 DEF FNS=RIGHT$(TIME$
,1)
120 DEF FND=MID$(TIME$,7
,1)
130 CLS
140 OD=FND
150 OS=FNS
160 IF OS=FNS THEN 160
170 LOCATE 0,0,0
180 PRINT TIME$
190 IF OD<>FND THEN SOUN
D 12,4 ELSE SOUND 19,1
200 GOTO 140
210 END
```

INPUT

FORMAT **INPUT ["<chaîne d'interrogation>"] | ; | , | <variable> [, <variable>...]**

OBJET Permettre l'entrée au clavier d'une variable spécifiée pendant l'exécution d'un programme.

EXEMPLE **INPUT "NAME" ; A$**

REMARQUES Lorsqu'elle rencontre une instruction INPUT, l'exécution du programme fait une pause et le HX-20 attend une introduction au clavier. Si la <chaine d'interrogation > est suivie d'un point virgule, la < chaine d'interrogation > est affichée sur l'écran LCD suivie d'un point d'interrogation "?" avec un espace significatif.
Si la <chaîne d'interrogation > est suivie d'une virgule, elle apparaît sur l'écran LCD sans être suivie d'un point d'interrogation.
Les données sont introduites an appuyant sur la touche **RETURN** et affectées à une variable spécifiée. Lorsque des variables multiples séparées par des virgules sont spécifiées, les données doivent aussi être séparées par des virgules pour l'entrée et doivent correspondre aux variables en nombre et en type. Si elles ne correspondent pas, le système va faire apparaître un message "?Refaire" et revenir à un état d'attente d'introduction des données.
Les constantes chaînes ne sont pas nécessairement présentées entre guillemets pour l'entrée sauf si elles contiennent des virgules, des points virgules ou des espaces significatifs. INPUT n'est pas valable en mode direct.

(Voir LINE INPUT)

EXEMPLE DE PROGRAMME

```
100 INPUT"STRING";A$
110 INPUT"NUMBER";A
120 PRINT"THE STRING AND
 THE NUMBER ARE ";A$A

STRING? HX
NUMBER? 20

>
THE STRING AND THE NUMBE
R ARE HX 20
```

INPUT

FORMAT INPUT# <numéro de fichier>, <variable>[, <variable>...]

OBJET Lire les éléments de données dans un fichier spécifié et les affecter aux valeurs d'un programme.

EXEMPLE INPUT# 1, A, B, C$

REMARQUES A part le fait que les données soient lues dans un fichier spécialisé et qu'il n'y ait pas de point d'interrogation délivré, INPUT# fonctionne exactement comme l'instruction INPUT. Le < numéro de fichier > doit être le numéro utilisé au moment où le fichier a été ouvert pour l'entrée. Les éléments de données du fichier doivent être ceux qui sont nécessaires à une instruction INPUT.

Lorsque vous utilisez une instruction INPUT# pour lire des données dans un fichier de données, ces données doivent être déjà préparées dans le fichier. Si l'exécution de l'instruction INPUT# continue lorsque toutes les données du fichier ont été lues, une erreur IE ("entrée dépassant la fin") apparaît. Si vous tenez compte de cela, toutes les données qui ont été écrites dans le fichier des données utilisant plusieurs instructions PRINT# peuvent être lues avec une seule instruction INPUT#.

(Voir INPUT, LINE, INPUT#, OPEN, et PRINT#)

KEY

FORMAT KEY <numéro de touche>, <chaîne>

OBJET Définir les touches de fonctions programmables

EXEMPLE KEY 1, "LIST"

REMARQUES Le clavier du HX-20 comprend 5 "touches de fonction programmable" chacune de ces touches pouvant être affectée à la fonction définie dans une chaîne quelconque. En associant ces touches avec le mode SHIFT, il est possible de définir un total de 10 chaînes. La longueur de la chaîne de caractères comprenant un code de contrôle doit avoir 15 caractères au plus Les caractères qui ne peuvent pas être introduits au clavier peuvent être spécifiés en utilisant la fonction CHRS ajoutée à la chaîne avec un signe (+).

(Voir KEY LIST)

KEY LIST/KEY LLIST

FORMAT (1) **KEY LIST**
(2) **KEY LLIST**

OBJET Afficher les chaînes affectées aux touches de fonctions programmables sur l'écran et la micro-imprimante respectivement.

EXEMPLE **KEY LLIST**

REMARQUES (1) KEY LIST fait afficher une liste complète des chaînes affectées aux touches de fonctions avec les numéros des touches. Chaque code de contrôle est affiché en tapant la flèche supérieure " ^" et une lettre.
(2) KEY LLIST est identique à KEY LIST mais elle entraîne la sortie d'une liste complète des chaînes sur la micro-imprimante incorporée. Au moment du lancement du BASIC, les chaînes suivantes sont affectées aux touches de fonctions respectives.

LET

FORMAT [**LET**] < **variable** > = < **expression** >

OBJET Affecter la valeur d'une expression à une variable.

EXEMPLE **LET A = 3.141592**

REMARQUES LET affecte une valeur ou la valeur d'une < expression > à une < variable >. L'affectation réelle est effectuée par le signe égal " = " et LET peut être omis. L' < expression > peut être une constante numérique ou une constante chaîne, mais l'affectation d'une variable chaîne à une variable numérique et l'inverse ne sont pas autorisées. En affectant des types de variables numériques qui ne correspondent pas, le type de variable qui se trouve à droite du signe égal est converti en type indiqué à gauche du signe égal avant d'effectuer l'affectation.

LINE

FORMAT **LINE [(< coordonnée horizontale 1 >, < coordonnée verticale 1 >)]–(< coordonnée horizontale 2 >, < coordonnée verticale 2 >),**
| PSET | [,< couleur >]
| PRESET |

OBJET Tracer une ligne droite entre deux points définis.

EXEMPLE **LINE (0,0) – (119,31), PSET**

REMARQUES Cette instruction permet de tracer une ligne droite entre deux points quelconques de l'écran graphique et spécifiés par l'instruction SCREEN. Si vous utilisez le contrôleur d'affichage en option, vous pouvez également spécifier la ou les couleurs, en mettant l'écran graphique sur l'écran d'affichage extérieur au moyen d'une instruction SCREEN. Dans ce cas, vous devez faire attention à la gamme dans laquelle vous pouvez spécifier des coordonnées, étant donné que la configuration de l'écran LCD est différente de celle de l'écran extérieur.

PRESET est utilisé pour effacer une ligne entre deux points. Avec l'écran LCD, la ligne est effacée simplement. Mais, avec l'écran graphique réalisé sur l'écran extérieur, la ligne est tracée avec la couleur de fond spécifiée par l'instruction COLOR.
Si les coordonnées 1 sont omises, les coordonnées 2 de l'instruction LINE précédente ou les coordonnées spécifiées par PSET ou PRESET sont adoptées.
NOTE : les instructions LINE risquent de ne pas fonctionner si la valeur spécifiée pour l'une des deux coordonnées dépasse 4000.

Le contrôleur d'affichage sur écran extérieur pourra ne pas être disponible à la date de parution de ce manuel.

(Voir COLOR, PRESET, et PSET)

EXEMPLE DE PROGRAMME

```
10 CLS
20 FOR I=5 TO 30
30 LINE (5,I)-(30,I),PSE
T
40 NEXT I
50 LINE (0,0)-(50,30),PS
ET
60 LINE (0,15)-(100,15),
PSET
70 COPY
80 END
```

LINE INPUT

FORMAT **LINE INPUT [" <chaîne d'interrogation> ";] <variable chaîne>**

OBJET Introduire une ligne entière comme variable chaîne.

EXEMPLE **LINE INPUT " WHAT?" ; A$**

REMARQUES LINE INPUT introduit une chaîne entière d'un maximum de 255 caractères sans utiliser les séparateurs à partir du clavier et l'affecte à une <variable chaîne>. Une <chaîne d'interrogation> est une chaîne littérale affichée sur l'écran LCD avant d'être introduite. Le point d'interrogation n'est pas affiché, sauf s'il fait partie de la chaîne d'interrogation. Toutes les entrées au clavier depuis la fin de la chaîne d'interrogation jusqu'à la touche **RETURN** sont affectées à la <variable chaîne>. Par conséquent, vous pouvez introduire les séparateurs comme des virgules, deux points, etc..., qui habituellement ne sont pas autorisées avec une instruction INPUT.

Une LINE INPUT peut être supprimée en appuyant sur la touche **BREAK** et le BASIC reviendra au niveau des commandes. Dans ce cas, introduisez une commande CONT pour reprendre l'exécution du programme.

(Voir INPUT)

EXEMPLE DE PROGRAMME

```
10 INPUT "NO. OF CUSTOME
RS";N
20 FOR I=1 TO N
30 LINE INPUT "CUSTOMER
DATA?";C$(I)
40 PRINT C$(I)
50 NEXT I
60 END

RUN
NO. OF CUSTOMERS? 2
CUSTOMER DATA? TOM
JONES, FIELDING AVE.
 TOM  JONES, FIELDING AV
E.
CUSTOMER DATA? DAVID
COPPERFIELD, DICKENS
ST.
 DAVID COPPERFIELD, DICK
ENS ST
```

LINE INPUT

FORMAT **LINE INPUT #< numéro de fichier >, < variable chaîne >**

OBJET Lire une ligne entière d'un fichier séquentiel de données et l'introduire dans une variable chaîne.

EXEMPLE **LINE INPUT # 1, A$**

REMARQUES LINE INPUT # introduit une ligne entière de caractères (255 caractères au plus) terminée par la frappe de la touche RETURN et prélevée dans un fichier séquentiel sans utiliser de séparateur et l'affecte à une <variable chaîne>
Le < numéro de fichier > est le numéro sous lequel le fichier a été ouvert par une instruction OPEN. La < variable chaîne > est le nom de la variable à laquelle la ligne va être affectée.
LINE INPUT # est particulièrement utile si chaque ligne d'un fichier de données a été séparée en zone, ou si un programme BASIC sauvegardé en mode ASCII est lu et constitue des données pour un autre programme.

(Voir INPUT #)

LIST/LLIST

FORMAT (1) **LIST** [< **numéro de ligne de départ** >][–[< **numéro de la dernière ligne** >]]
(2) **LLIST** [< **numéro de ligne de départ** >][– [< **numéro de la dernière ligne** >]] .

OBJET Sortir un listage de programme (1) sur l'écran LCD ou sur l'écran extérieur ou (2) sur la micro-imprimante.

EXEMPLE
LIST 100 – 200
LIST – 200
LIST 100 –
LIST 200
LIST.
LIST

REMARQUES LIST sort la totalité ou une partie du programme sur lequel on travaille selon les spécifications du < numéro de ligne de départ > et < numéro de dernière ligne >. Si seul le < numéro de ligne de départ > a été spécifié, cette seule ligne du programme sera sortie. Si le < numéro de ligne de départ > est suivi d'un trait d'union, cette ligne et tous les numéros de lignes suivants sont listés. Si le < numéro de dernière ligne > précédé par un trait d'union est spécifié, toutes les lignes depuis le début du programme et y compris cette ligne seront listées. Si les deux numéros de lignes sont omis, le programme sera totalement listé. Après l'apparition d'une erreur, ou après édition du programme, le numéro de ligne peut être remplacé par un point. Le listage peut être interrompu en appuyant sur la touche **BREAK** ou en l'arrêtant momentanément au moyen de la touche **PAUSE**. Une commande LIST peut être écrite dans un programme pour le listage, mais la commande suivant la commande LIST ne sera pas exécutée et le système retournera au niveau des commandes. L'emploi de LLIST est le même que celui de LIST, mais le listage est établi sur la micro-imprimante.

(Voir LOGIN)

LIST <file descriptor>

FORMAT **LIST <descripteur de fichier >[, [< numéro de ligne >] [−[< numéro de ligne >]]]**

OBJET Sortir un listage de programme sur un fichier spécifié.

EXEMPLE **LIST "COM0:**

REMARQUES Lorsqu'une cassette est spécifiée comme nom de dispositif, cette commande fonctionne comme SAVE en format ASCII. Si le < descripteur de fichier > est spécifié avec une expression chaîne ou une variable chaîne, il commence toujours par des guillemets (").

Les numéros de lignes sont spécifiés de la même manière que pour LIST, mais les numéros de lignes sont toujours précédés d'une virgule.

(Voir LIST)

LIST"COM0:"

FORMAT **LIST" COM0: [(< BLPSC >)] " [, [< numéro de ligne >][−[< numéro de ligne >]]]**

OBJET Spécifier les conditions d'interfaces de la porte RS − 232C et exécuter LIST.

EXEMPLE **LIST" COM0: (270 1B)".**

REMARQUES L'emploi de cette commande est le même que celui de LIST mais le dispositif de sortie est la porte RS − 232C.

Vous trouverez des détails au sujet de < BLPSC > qui spécifie les conditions d'interface en vous reportant à OPEN " COM0:".

LOAD

FORMAT **LOAD [< descripteur de fichier >[,R]]**

OBJET Charger un fichier programme dans la mémoire.

EXEMPLE **LOAD " CAS 1: PROG1.ASC"**

REMARQUES Cette commande charge le fichier programme spécifié par le <descripteur de fichier >dans la mémoire. Lorsque LOAD est exécuté, tous les fichiers ouverts sont fermés et toutes les variables en cours sont supprimées. Mais, si l'option <R>(c'est à dire chargement et exécution) est indiquée après LOAD, tous les fichiers ouverts restent ouverts et le programme est exécuté immédiatement après son chargement.

Si le <descripteur de fichier >est omis, le premier fichier du dispositif implicite est chargé. LOAD garde les programmes résidant actuellement dans la mémoire jusqu'à ce que le fichier spécifique soit trouvé et que le chargement réel commence.

Avant d'exécuter LOAD, utilisez la commande STAT pour vérifier la zone dans laquelle vous vous trouvez (cf. LOGIN).

Une tentative de chargement (LOAD) dans une zone qui a déjà été nommée par une instruction TITLE fera apparaître une erreur PP ("programme protégé").

(Voir LOGIN, SAVE, STAT et TITLE)

LOAD"COM0:"

FORMAT **LOAD " COM0: [(<BLPSC >)] "**

OBJET Spécifier les conditions d'interface de la porte RS – 232C et l'exécution de LOAD.

EXEMPLE **LOAD "COM0: (68N2B)" [,R]**

REMARQUES L'emploi de cette commande diffère de la commande normale LOAD uniquement du fait que le programme est lu par l'intermédiaire de la porte RS – 232C . Mais le programme transmis doit être au format ASCII sinon une erreur se produira et le programme ne sera pas lu. En ce qui concerne <BLPSC> qui spécifie les conditions d'interface, reportez-vous à OPEN "COM0:".

LOADM

FORMAT **LOADM [<descripteur de fichier>][, [valeur de décalage] [, R]]**

OBJET Charger un fichier programme en langage machine dans la mémoire.

EXEMPLE **LOADM "CAS 1: ABC"**

REMARQUES Le fichier à charger par la commande LOAD doit être un fichier programme en langage machine créé par la fonction moniteur ou par la commande SAVEM. Si le <descripteur de fichier> est omis, le premier fichier du dispositif implicite sera chargé. La <valeur de décalage> est ajoutée à l'adresse la plus élevée spécifiée par la commande SAVEM et le chargement commence à l'adresse résultante.

Si l'option <R> est spécifiée, lorsque le sous-programme en langage machine aura été chargé dans la mémoire, l'exécution du programme commencera à l'<adresse de début d'exécution> spécifiée par la commande SAVEM. Si le suffixe <R> n'a pas été spécifié, le HX-20 revient au niveau des commandes BASIC lorsque le programme en langage machine a été chargé.
Mais, avec une commande SAVEM, le contenu de la mémoire peut être créé sous forme de fichier. Si <R> est utilisé avec une commande LOADM et qu'il ne s'agit pas d'un fichier programme en langage machine, la CPU interprètera ce fichier comme un fichier programme en langage machine et exécutera le chargement en conséquence. Si cela se produit, les programmes BASIC et les fichiers RAM risquent d'être détruits. Faites attention lorsque vous spécifiez l'option <R>.
La <valeur de décalage> ne peut pas être spécifiée si le sous-programme en langage machine ne peut être changé de position, même s'il est chargé à un emplacement différent de celui à partir duquel il a été sauvegardé. Mais, étant donné que le microprocesseur ne fait aucune vérification pour déterminer si la <valeur de décalage> est permise ou non, il peut y avoir des cas où un fichier est déplacé jusqu'à une adresse différente par la <valeur de décalage> même si le fichier contient les données mémoires.
Avec LOADM, "COM0: " ne peut pas être spécifié comme nom de dispositif.

(Voir SAVEM et 5.3 "Programmes en Langage Machine")

LOAD?

FORMAT **LOAD? [<descripteur de fichier >]**
OBJET Vérifier les fichiers
EXEMPLE **LOAD? "CAS 1: PROG 1.ASC"**

REMARQUES LOAD? vérifie si oui ou non le fichier de données spécifié par le <descripteur de fichier > a été correctement enregistré. Si le <descripteur de fichier > est omis, la machine supposera qu'il s'agit du "CAS0:" à condition que la microcassette soit connectée au HX-20, et du "CAS 1:" si la microcassette n'est pas connectée.
La commande LOAD? effectue la vérification du CRC en lisant le fichier (programmes et données) enregistré sur la mémoire auxiliaire pour confirmer que le fichier des données a été correctement enregistré. Si la vérification du CRC détecte une erreur, une erreur E/S ("dispositif E/S") sera affichée. Etant donné que le fichier des données n'est pas véritablement chargé pendant la vérification CRC, il n'y a aucun risque de perdre les programmes stockés dans la mémoire. Cette commande n'est pas destinée à vérifier le contenu de la cassette de bande par rapport au contenu de la mémoire. C'est pourquoi, LOAD? peut être exécuté même s'il n'y a pas de programme dans l'emplacement de programme actuel.

Lors de l'exécution de LOAD? par rapport aux fichiers (CAS 1: CAS 0:) stockés sur la cassette de bande, une recherche sera faite sur la bande jusqu'à ce que le fichier spécifié par le <descripteur de fichier > soit trouvé et que la vérification réelle commence. Après l'exécution de cette commande, la position de la tête de l'enregistreur magnétique sera à la fin du fichier spécifié ; autrement dit, au début du fichier suivant.

NOTE : Le CRC est un code qui est calculé à partir des données enregistrées et qui est sauvegardé avec les données. Il permet de détecter toute modification ou destruction de données. Lors de la lecture, ce code est automatiquement recalculé et comparé au code CEC enregistré.

LOCATE

FORMAT LOCATE <**coordonnée horizontale**>, <**coordonnée verticale**>
[, <**commutateur du curseur**>]

OBJET Spécifier la position du curseur sur l'écran.

EXEMPLE **LOCATE 10, 10, 0**

REMARQUES LOCATE positionne le curseur en un endroit spécifié de l'écran virtuel. Les deux <coordonnée horizontale> et <coordonnée verticale> doivent être spécifiées. Le curseur ne peut pas être positionné en dehors des limites de l'écran virtuel spécifié par une commande WIDTH.

Si le curseur est positionné à l'extérieur de l'écran matériel en exécutant une commande LOCATE, l'écran matériel sera automatiquement déplacé pour suivre le curseur, de façon à ce que ce dernier soit placé dans l'angle supérieur gauche de l'écran. Mais, si l'écran matériel est à l'extrémité droite ou en bas de l'écran virtuel, le curseur peut ne pas se situer dans l'angle supérieur droit. Même si l'écran a été placé à l'extrémité gauche de l'écran virtuel (mode 0), l'exécution d'une commande LOCATE fera quitter cette position à l'écran matériel (mode 1). Lorsque le système revient au niveau des commandes après l'exécution d'un programme, l'écran matériel est placé en mode 0.

Le curseur est supprimé si l'option <commutateur du curseur> est spécifié à "0", puis il est remis en service si on spécifie "1".

(Voir SCROLL)

LOCATES

FORMAT **LOCATES <coordonnée horizontale> , <coordonnée verticale>**
OBJET Spécifier la position de l'écran matériel
EXEMPLE **LOCATES 0, 0**

REMARQUES LOCATES déplace l'écran matériel de façon à ce que son angle supérieur gauche se trouve à l'endroit de l'écran virtuel spécifié par la <coordonnée horizontale> et par la <coordonnée verticale>. L'écran matériel ne peut pas sortir des limites de l'écran virtuel.
De même qu'avec une commande LOCATE, après l'exécution d'une commande LOCATES, l'écran est toujours en mode 1. Même après l'exécution d'une commande LOCATES, la position du curseur sur l'écran matériel restera inchangée.

NOTE : LOCATE déplace le curseur tandis que LOCATES déplace seulement l'écran matériel.

(Voir LOCATE)

LOGIN

FORMAT **LOGIN <expression> [, R]**
OBJET Commuter les zones de programmes
EXEMPLE **LOGIN 3**

REMARQUES En BASIC, l'espace mémoire est subdivisé en cinq zones qui peuvent chacune stocker un programme distinct. Une commande LOGIN spécifie les zones de programmes à utiliser (pour l'exécution des programmes, l'édition des programmes, etc...).
<expression> doit être un entier compris entre 1 et 5. Toutes les commandes d'exécution et de modification des programmes (NEW, LIST, LOAD, SAVE, etc...), ne sont effectives que dans les zones qui ont été spécifiées par les commandes LOGIN. Si l'option < R > est spécifiée, l'exécution du programme commence dès que la zone du programme a été commutée. Après l'exécution d'une commande LOGIN, toutes les variables sont effacées.

(Voir STAT).

MEMSET

FORMAT **MEMSET [<adresse la plus faible de la mémoire>]**

OBJET Spécifier la limite inférieure de la mémoire.

EXEMPLE **MEMSET &H0D00**

REMARQUES En BASIC, les programmes écrits en langage machine sont placés avant la zone de texte des programmes BASIC. Pour valider le stockage des programmes en langage machine et en BASIC en même temps, la limite inférieure de la mémoire à utiliser par le BASIC doit être fixée en utilisant une commande MEMSET. Ceci définit les emplacements dans la mémoire des programmes en langage machine.

Au moment d'un démarrage à froid, <l'adresse la plus faible de la mémoire> est fixée à &H0A3F. Le démarrage à chaud ne modifiera pas cette <adresse la plus faible de la mémoire.>

Si < l'adresse la plus faible de la mémoire > est omise, la machine suppose que cette adresse est &H0A3F. Au moment de l'exécution d'une commande MEMSET, toutes les variables sont effacées.

Les adresses 0 à &H0A3F sont affectées comme zones système et d'E/S et ne sont pas admises pour le stockage des programmes en langage machine.

EXEMPLE DE PROGRAMME

```
100 ON ERROR GOTO 200
110 FOR I=&H0A3F TO &H0B
00
120 POKE I,0
130 NEXT I
140 END
200 LPRINT "BASIC PROGRA
MME AREA. TO PROTECT BAS
IC, POKE CANNOT BE EXECU
TED."
210 RESUME 140
MEMSET &H0AFF
RUN

BASIC PROGRAMME AREA. TO
 PROTECT BASIC, POKE CAN
NOT BE EXECUTED.
```

MERGE

FORMAT **MERGE [<descripteur de fichier>[, R]**

OBJET Fusionner un fichier programme spécifié avec le programme se trouvant à ce moment dans la mémoire.

EXEMPLE **MERGE "CAS 1: PROG 3.ASC"**

REMARQUES La commande MERGE fusionne le fichier programme spécifié par le <descripteur de fichier> avec le programme qui se trouve dans la zone de mémoire actuellement adressée. Le fichier spécifié doit être sauvegardé en format ASCII. Sinon, une erreur BF ("mauvais mode de fichier") apparaîtra. Si le <descripteur de fichier> est omis, le premier fichier du dispositif implicite sera lu. Si certaines lignes du fichier ont les mêmes numéros de lignes que les lignes du programme présent dans la mémoire, les lignes du fichier seront remplacées par les lignes correspondantes à la mémoire. Si l'option R est spécifiée le programme fusionné sera exécuté après l'opération MERGE. Le BASIC retourne toujours au niveau des commandes après l'exécution d'une commande MERGE. Lorsqu'une commande MERGE est exécutée, tous les fichiers ouverts à ce moment sont fermés et toutes les variables sont annulées. Mais, si < R > a été spécifié, la commande MERGE sera exécutée en laissant les fichiers ouverts.

(Voir SAVE)

MERGE"COM0:"

FORMAT **MERGE "COM0: [(<BLPSC>)" [, R]]**

OBJET Spécifier les conditions d'interface de la porte RS – 232C et exécuter la commande MERGE.

EXEMPLE **MERGE "COM0": (68N2B)" ,R**

REMARQUES Cette commande est essentiellement la même commande normale MERGE, mais le programme est lu par l'intermédiaire de la porte RS – 232C.
Vous trouverez les détails au sujet de < BLPSC > spécifiant les conditions d'interface de la porte RS – 232C en vous reportant à OPEN "COM0:".

MID$

FORMAT **MID$ (< expression chaîne 1 >, < n > [, < m >]) = < expression chaîne 2 >**
où n et m sont des expressions entières et < expression chaîne 1 > et < expression chaîne 2 > sont des expressions chaînes.

OBJET Remplacer une portion de chaîne par une autre.

EXEMPLE **MIDS (A$,2) = "BASIC"**

REMARQUES MID$ remplace les caractères de l'< expression chaîne 1 > en commençant à la position < n >, par les caractères de l'< expression chaîne 2 >. Le < m > en option indique le nombre de caractères de l'< expression chaîne 2 > qui seront utilisés dans le changement. Si < m > est omis, la totalité de l'< expression chaîne 2 > est utilisée. Mais, que < m > soit omis ou inclus, le remplacement des caractères ne dépasse jamais la longueur originale de l'< expression chaîne 1 >. Comme la longueur de l'< expression chaîne 1 > ne varie jamais, la valeur de < n > ne peut pas dépasser le nombre de caractères contenus dans (expression chaîne 1) et < n > ne peut pas, non plus, avoir une valeur négative. L'< expression chaîne 1 > ne peut pas être une chaîne nulle.

(Voir Fonction MID$)

EXEMPLE DE PROGRAMME

```
100 FOR I=1 TO 20 STEP 4
110 A$="----------------
----"
120 MID$(A$,I)="*"
130 PRINT A$:NEXT
140 END

*-------------------
----*---------------
-------*-----------
------------*-------
----------------*---
```

MON

FORMAT MON

OBJET Transférer la commande du programme au moniteur en langage machine.

EXEMPLE MON

REMARQUES MON s'emploie pour transférer le contrôle du programme du BASIC au moniteur incorporé en langage machine. Les commandes du moniteur en langage machine sont les suivantes:

S < **adresse** > Change le contenu de la mémoire. L'introduction d'un point (".") termine cette commande.

D < **adresse** > affiche le contenu de la mémoire.

G < **adresse d'exécution** > , < **point d'interruption** >. Fait exécuter un programme jusqu'au < point d'interruption >. Le HX-20 retourne ensuite au moniteur.

K < **chaîne** > ^ : @ Ecrit un numéro de menu au début de la < chaîne > puis, écrit une liste de chaînes jusqu'à ^ :@. Lorsque l'appareil est mis sous tension, le programme spécifié par le numéro de menu est automatiquement sélectionné et les opérations qui sont effectuées sont les mêmes que si les chaînes de caractères étaient introduites au clavier. La < chaîne > peut avoir 17 caractères au maximum. Cette fonction est annulée par l'introduction de K^.@, (ce qui définit une chaîne nulle pour < chaîne >.

B Fait sortir le HX-20 du moniteur et retourne au BASIC.

X Change le contenu des registres. A chaque fois que vous appuyez sur la touche **RETURN** le contenu de chaque registre est affiché. Si vous appuyez sur la touche **RETURN** après avoir introduit une valeur numérique, le contenu du registre peut être changé. Cette commande se termine par l'introduction d'un point.

R < **dispositif** > , < **nom de fichier** >, **R** Charge un fichier spécifié dans la mémoire à partir d'un dispositif spécifié. Si R est utilisé avec cette commande le programme est exécuté dès le chargement du fichier spécifié.

V < **dispositif** >, < **nom de fichier** >. Effectue une vérification CRC d'un fichier dans un dispositif spécifié.

W < **dispositif** > , < **nom de fichier** > . Ecrit un fichier sur un dispositif spécifié.

A Précise les adresses lorsque la commande R, V ou W est utilisée.

A chaque fois que vous appuyez sur **RETURN** , des données à deux octets sont nécessaires dans l'ordre suivant.

T Adresse supérieure de la mémoire.

L Adresse inférieure de la mémoire.

O Décalage.

E Adresse de début d'exécution.

Pour le < dispositif >, spécifiez C (cassette), M (microcassette) ou P (cartouche ROM). Mais, avec une commande W, P (cartouche ROM) ne peut pas être spécifié. Le < nom de fichier > est constitué d'un nom de fichier de 8 caractères au plus et d'un type de fichier à 3 caractères.
< nom de fichier > = < nom de fichier >, < type de fichier >.
Pour de plus amples détails, se reporter au Chapitre 10, " Comment Utiliser le Moniteur du HX-20 dans le Manuel de l'Opérateur ".

MOTOR

FORMAT **MOTOR [<commutateur>]**

OBJET Mettre en marche ou arrêter le moteur du dérouleur à cassette audio extérieur

EXEMPLE **MOTOR ON**

REMARQUES MOTOR met en service ou hors service le terminal à distance constitué par un dérouleur à cassette audio extérieur, connecté au HX-20. En spécifiant la valeur du < commutateur > ,ON ou OFF, il est possible de commander le moteur du dérouleur audio extérieur. Si la valeur du < commutateur > est omise, l'état du système est inversé. Autrement dit, si le terminal à distance est à l'état OFF, le moteur passe de OFF à ON et inversement.

NEW

FORMAT NEW

OBJET Supprimer le programme se trouvant en mémoire et effacer toutes les variables

EXEMPLE NEW

REMARQUES NEW supprime le programme qui se trouve dans l'emplacement de programme spécifié. Lorsque cette commande apparaît dans un programme, le programme de cette zone est effacé et le BASIC retourne au niveau des commandes. Si le programme est dans une zone qui a été nommée par une instruction TITLE, l'exécution d'une commande NEW aboutira à une erreur PP ("Programme protégé"). L'exécution d'une commande NEW fait fermer tous les fichiers actuellement ouverts.

(Voir LOAD, LOGIN, et TITLE)

ON ERROR GOTO

FORMAT ON ERROR GOTO <numéro de ligne>

OBJET Permettre un branchement sur une erreur et spécifier la première ligne du sous-programme de traitement des erreurs.

EXEMPLE ON ERROR GOTO 1000

REMARQUES Une instruction ON ERROR GOTO transfère le contrôle du programme à un sous-programme de traitement d'erreur spécifié si une erreur se produit pendant l'exécution d'un programme. Si une erreur est détectée et que cette instruction est dans le programme, le BASIC exécute ce programme commençant à la ligne spécifiée par le <numéro de ligne> et n'affiche pas de message d'erreur. Par conséquent, en permettent le traitement des erreurs avec ce sous-programme, vous pouvez éviter d'arrêter l'exécution du programme du fait de l'apparition d'une erreur.
Pour empêcher le traitement d'erreurs, exécutez ON ERROR GOTO 0. En cas de rencontre d'une erreur pour laquelle il n'y a pas de traitement d'erreur défini, faites exécuter un ON ERROR GOTO 0 dans le sous-programme de traitement d'erreur . Un message d'erreur apparaît et l'exécution du programme s'achève.

(Voir RESUME, ERL/ERR et Annexe A, Messages d'erreur).

EXEMPLE DE PROGRAMME

```
100 ON ERROR GOTO 200
110 INPUT "A=",A
120 IF A<0 THEN ERROR 25
0
130 IF FIX(A)<>A THEN ER
ROR 255
140 B#=1#
150 FOR I=2 TO A
160 B#=B#*I:NEXT I
170 PRINT A;"!"
180 PRINT "IS ";B#
190 GOTO 100
200 '***ERROR***
210 IF ERR=255 AND ERL=1
30 GOTO 250
220 IF ERR=6 AND ERL=160
 GOTO 270
230 IF ERR=250 AND ERL=1
20 GOTO 290
240 ON ERROR GOTO 0
250 PRINT "INPUT AN INTE
GER "
260 RESUME 110
270 PRINT "VALUE TOO LAR
GE "
280 RESUME 110
290 PRINT "NEG NUMBER NO
T ACCEPTED "
300 RESUME 110
310 END
```

```
RUN
A=10
 10 !
IS  3628800
A=-30
NEG NUMBER NOT ACCEPTED
A=17
 17 !
IS  355687428096000
A=45
VALUE TOO LARGE
```

ON...GOSUB/ON...GOTO

FORMAT **ON <expression> | GOSUB / GOTO | <numéro de ligne> [, <numéro de ligne>...]**

OBJET Brancher l'exécution à l'un des numéros de ligne spécifiés.

EXEMPLE
ON A GOSUB 100, 200, 300, 400
ON B GOTO 100, 200, 300, 400

REMARQUES L'instruction ON . . . GOSUB/GOTO détermine le < numéro de ligne > auquel un programme doit se brancher en fonction de la valeur renvoyée lorsque l'<expression >est évaluée.
Si par exemple, la valeur est 2, le second < numéro de ligne >de la liste des numéros de ligne sera la destination du branchement. La valeur de l'<expression >doit se situer entre zéro et 255.
Si la valeur de l'< expression > est négative, une erreur FC ("appel de fonction non valable") apparaît. Si la valeur de l'< expression > est 0 ou supérieure au nombre des numéros de lignes, l'instruction ON GOSUB/GOTO est ignorée et la prochaine instruction du programme est exécutée. Dans ce cas, aucune erreur ne se produit. Dans l'instruction ON GOSUB, chaque <numéro de ligne > de la liste doit être le premier numéro de ligne d'un sous-programme.

(Voir GOSUB... RETURN et GOTO)

EXEMPLE DE PROGRAMME

```
100 INPUT "DATE OR TIME
(D/T) ";A$
110 I=INSTR("DT",A$)
120 ON I GOSUB 300, 200
130 PRINT: GOTO 100
200 PRINT TIME$:RETURN
300 PRINT DATE$:RETURN
```

```
13:21:46

13:22:31

07/21/82
```

OPEN

FORMAT **OPEN "mode" , [#] < numéro de fichier >, < descripteur de fichier >**

OBJET Ouvrir un fichier spécifié pour E/S.

EXEMPLE **OPEN "O" , # 1 "CAS0:TEST.BAS"**

REMARQUES OPEN ouvre un fichier désigné par un < descripteur de fichier > sous un < numéro de fichier >.

L'entrée et la sortie des fichiers ouverts à la suite seront effectuées en spécifiant le < numéro de fichier >.

Le < mode > est une expression chaîne dont le premier caractère est soit "I" signifiant mode d'entrée séquentiel, soit "O" spécifiant mode de sortie séquentiel . Le < numéro de fichier > est une expression entière dont la valeur est comprise entre 1 et 16, cependant un nombre associé à un fichier qui a déjà été ouvert ne peut pas être réutilisé.

(Voir CLOSE et 5.2, fichiers séquentiels)

OPEN"COM0:"

FORMAT **OPEN "<mode>",[#]<numéro de fichier>,"COM0:[(<BLPSC>)]"**

OBJET Spécifier les conditions d'interface d'une porte RS – 232C et exécuter OPEN

EXEMPLE **OPEN "0",#1,"COM0: (68N2B)".**

REMARQUES Cette commande est essentiellement identique à la commande normale OPEN mais elle spécifie les conditions d'interface de la porte RS – 232C. <BLPSC> est constitué de cinq caractères qui spécifient une des conditions d'interface de la porte RS – 232C de la façon suivante:

B (vitesse de transfert en nombre de bauds)

La vitesse est spécifiée par les chiffres de 0 à 6 (vitesse de transfert des données).

0: 110 bps
1: 150 bps
2: 300 bps
3: 600 bps
4: 1,200 bps
5: 2,400 bps
6: 4,800 bps

L (longueur des mots)

Utilisez 7 ou 8 pour préciser la longueur des mots des données à un caractère.

7: 7 bits/caractère
8: 8 bits/caractère

P (parité)

N, E ou O permet de spécifier le type de contrôle de parité.

N: pas de contrôle de parité
E: contrôle de parité paire
O: contrôle de parité impaire

S (bits d'arrêt)

1 ou 2 pour spécifier le nombre de bits d'arrêt.

1: 1 bit
2. 2 bits

C (ligne de contrôle active)

Les lignes de contrôle actives (signal) sont déterminées en utilisant les nombres hexadécimaux de 0 à F qui correspondent à 4 bits binaires, chaque bit correspondant à une ligne de contrôle. Dans le tableau ci-dessous, les lignes de signaux actives sont représentées par "O" et les lignes inactives (à ignorer) par "X". Pour le signal RTS, le signe " +" indique que le signal de potentiel positif est actif et le signal " – " indique que le signal de potentiel négatif est actif.

Hexadecimal number \ Control line	CTS	DSR	RTS	CD
0	O	O	–	O
1	O	O	–	×
2	O	O	+	O
3	O	O	+	×
4	O	×	–	O
5	O	×	–	×
6	O	×	+	O
7	O	×	+	×
8	×	O	–	O
9	×	O	–	×
A	×	O	+	O
B	×	O	+	×
C	×	×	–	O
D	×	×	–	×
E	×	×	+	O
F	×	×	+	×

<BLPSC>peut être omis. Dans ce cas, le HX-20 utilise par défaut les valeurs indiquées la dernière fois. Au moment du démarrage à chaud, le HX-20 est placé dans les conditions suivantes:

Vitesse :	4 800 bps
Longueur des mots:	8 bits/caractère
Parité:	Pas de contrôle de parité
Bits d'arrêt:	2 bits
CTS:	A ignorer
DSR:	Actif
RTS:	+ potentiel/actif
CD:	A ignorer

En utilisant la porte RS–232C, deux fichiers (pour l'entrée et la sortie) peuvent être ouverts en même temps. Dans ce cas, différentes conditions d'interface ne peuvent pas être spécifiées pour les fichiers à ouvrir. Les conditions dans lesquelles le premier fichier a été ouvert restent en vigueur.

OPTION BASE

FORMAT **OPTION BASE** $\left|\begin{matrix}0\\1\end{matrix}\right|$

OBJET Déclarer la valeur minimale des indices des variables tableaux.

EXEMPLE **OPTION BASE 1**

REMARQUES OPTION BASE est utilisé pour déclarer que la valeur minimale des indices des variables tableaux est soit 0 soit 1. La base implicite est 0. Mais si une OPTION BASE 1 est exécutée, la valeur minimale d'un indice est fixée à 1. Ensuite, si un élément de tableau est référencé par un indice spécifié à 0, une erreur BS ("mauvais indice") apparaît. Cette déclaration ne peut pas être faite lorsque des variables tableaux ont été déclarées dans un programme ou référencées. Egalement, une fois déclarée, la valeur minimale ne peut pas être modifiée par une nouvelle déclaration qui entraînera alors une erreur DD ("définition en double").

Une fois l'instruction OPTION BASE exécutée, la déclaration par OPTION BASE ne peut être ni modifiée ni annulée avant qu'une commande RUN ou CLEAR n'ait été exécutée.

(Voir CLEAR et DIM)

EXEMPLE DE PROGRAMME

```
100 ON ERROR GOTO 160
110 OPTION BASE 1 : DIM
A(9)
120 FOR I=0 TO 9
130 A(I)=I:PRINT A(I);
140 NEXT I
150 OPTION BASE 0:END
160 IF ERR=9 THEN PRINT
"MINIMUM SUBSCRIPT IS 1"
 : RESUME 140
170 IF ERR=10 THEN PRINT
 "OPTION BASE CANNOT BE
CHANGED"
180 RESUME NEXT
RUN
MINIMUM SUBSCRIPT IS 1
 1  2  3  4  5  6  7  8
 9
OPTION BASE CANNOT BE CH
ANGED
```

PCOPY

FORMAT PCOPY <expression>

OBJET Copier un programme BASIC dans une autre zone de programme.

EXEMPLE PCOPY 3

REMARQUES La commande PCOPY copie les programmes de la zone de programmes en cours d'utilisation dans une autre zone de programme spécifiée par < expression >. L' <expression > doit être comprise entre 1 et 5. S'il y a déjà dans la zone de programme, un programme spécifié par l'<expression > ou s'il n'y a aucun espace dans la zone de programme actuellement adressée, l'exécution d'une commande PCOPY entraîne une erreur FC ("appel de fonction non valable"). Si le programme à copier est trop grand, une erreur OM ("débordement de mémoire") apparaît et la commande PCOPY n'est pas exécutée.

(Voir LOGIN, STAT et TITLE)

POKE

FORMAT **POKE <adresse>, <expression numérique>**

OBJET Ecrire un octet dans un emplacement spécifié de la mémoire.

EXEMPLE **POKE &H0C00, &H39**

REMARQUES La commande POKE écrit un octet (8bits) de données spécifiées comme < expression numérique > dans un emplacement de mémoire désigné par l'<adresse>.

L'<expression numérique > doit être une expression entière en octets comprise entre 0 et 255 (&H0 à &HFF), POKE arrondit les chiffres les moins significatifs à l'entier le plus proche. (par exemple, 1.5 à 2 et 1.4 à 1).

<adresse > doit être une expression entière de doubles octets de la gamme 0 à 65535 (&H0 et &HFFFF). POKE réécrit les données en cours dans la mémoire. Noter que l'emploi sans précaution de cette commande peut être une source de défaut de fonctionnement. Etant donné que &H0 à &H4D constituent une zone spéciale allouée aux entrées/sorties, une erreur d'exécution peut se produire uniquement par une lecture.

Des programmes en BASIC sont stockés dans la zone au-dessus de l'adresse spécifiée par une commande MEMSET. C'est pourquoi, toute tentative d'exécution d'une instruction POKE dans la zone spéciale mentionnée ci-dessus entraîne une erreur FC. Pour réécrire l'une quelconque de ces adresses, vous devez écrire &H80 à l'adresse H7E. Dans la zone comprise entre &H4E et &HA3F, l'exécution de l'instruction POKE ne provoque aucune erreur. Mais, étant donné que cette zone est utilisée par le BASIC comme zone de travail, la réécriture d'une quelconque valeur dans ces emplacements de mémoire peut entraîner un défaut de fonctionnement du BASIC. Par conséquent, vous devez limiter l'exécution des instructions POKE à la zone du langage machine, c'est à dire entre l'adresse &HA40 et l'adresse spécifiée comme adresse inférieure de la mémoire par une instruction MEMSET. La fonction complémentaire de POKE est PEEK. C'est une fonction qui lit une < expression numérique > dans un emplacement spécifié de la mémoire. POKE et PEEK sont utiles pour un stockage efficace des données en chargeant des sous-programmes en langage machine, et en passant des arguments et des résultats en entrée et en sortie des sous-programmes en langage machine.

(Voir PEEK, USR et 5.3 "Les Sous-Programmes en Langage Machine")

Choix des jeux de caractères

Le choix des jeux de caractères peut se faire au moyen des instructions POKE et EXEC. Pour choisir le jeu des caractères, pointez l'adresse &H7F et écrivez les données correspondantes ci-dessous.

Jeu des caractères du pays à sélectionner	Données à écrire
Espagne	&H10
Italie	&H11
Suède	&H12
Danemark	&H13
Angleterre	&H14
Allemagne	&H15
France	&H16
Etats-Unis	&H17

L'adresse de départ de l'instruction EXEC est &HFF6A.

Pour revenir au jeu de caractères implicites (c'est à dire le jeu sélectionné par le commutateur DIP), exécutez les instructions suivantes:

```
10 POKE &H7F,0
20 EXEC &HFF6A
```

Exemple: Espagne

```
10 POKE &H7F,&H10
20 EXEC &HFF6A
```

PRESET

FORMAT **PRESET (<coordonnée horizontale>, <coordonnée verticale>)**

OBJET Effacer un point sur un écran graphique.

EXEMPLE **PRESET (40, 25)**

REMARQUES PRESET ramène la couleur de l'un quelconque des points tracés sur l'écran graphique par une instruction PSET ou LINE à la couleur du fond.
Sur l'écran LCD, cette commande efface simplement le point spécifié de l'écran graphique. Si un écran extérieur est connecté au HX-20 en utilisant le contrôleur d'affichage en option, cette commande ramène le point spécifié à la couleur de fond définie par l'instruction COLOR au moment ou l'écran extérieur est placé en mode graphique couleur par une instruction SCREEN. Etant donné que l'écran LCD et l'écran extérieur sont différents l'un de l'autre par leur configuration, veuillez faire attention à la gamme dans laquelle vous spécifiez les <coordonnées>.

(Voir LINE, COLOR, PSET, et SCREEN)

PRINT/LPRINT

FORMAT | **PRINT** / **LPRINT** | [<**expression**>[, <**expression**>...]]

OBJET Sortir des données sur l'écran ou sur la micro-imprimante incorporée.

EXEMPLE **PRINT "EPSON"**

REMARQUES PRINT fait sortir sur l'écran les données spécifiées par <expression>, tandis que la commande LPRINT fait sortir les mêmes données sur la micro-imprimante incorporée. L'<expression> peut être une expression numérique ou une expression chaîne. Si vous utilisez des expression chaînes, les chaînes de caractères résultant des expressions sont sorties. Si vous utilisez des expressions numériques, les valeurs résultants de ces expressions sont sorties. Mais, dans ce dernier cas, un espace blanc est laissé avant et après chaque valeur sortie. Si la valeur est négative, un signe"–"est sorti dans le blanc qui précède la valeur. Lorsque vous écrivez une liste d'expressions, les expressions doivent être séparées par une virgule, un point virgule ou un blanc. Les séparateurs peuvent être omis entre une variable et une constante chaîne et entre une constante chaîne et une autre. Dans ce cas, l'effet sera le même que si chaque expression était séparée par un point virgule. Un point d'interrogation ("?") peut être utilisé en remplacement de PRINT dans une instruction PRINT, tandis que LPRINT ne peut être ni remplacé ni omis.

Format de sortie et ponctuation.

- Lorsque les articles d'une liste d'expressions sont séparés par des blancs ou des points virgules, la valeur suivante sera affichée ou imprimée immédiatement après la précédente. Lorsque les expressions de la liste sont séparées par des virgules, le BASIC divise la ligne affichée ou les zones d'impression en 14 espaces chacune et délivre la valeur de chaque expression à chaque zone. Si le résultat d'une expression déborde sur 2 ou plusieurs zones, la valeur suivante sera sortie au début de la zone qui suit la dernière valeur.
- Si la liste des expressions se termine sans virgule ou sans point virgule, un retour de chariot est effectué à la fin de la ligne. Si une virgule ou un point virgule termine la liste des expressions, la prochaine instruction PRINT commence la sortie sur la même ligne en suivant le format de sortie spécifié.
- Si une chaîne à sortir est plus longue qu'une ligne spécifiée pour le terminal, la sortie se poursuit sur la ligne suivante. Si toutes les chaînes ou valeurs à sortir ne sont pas acceptées sur une seule ligne (entre la position du curseur et la fin de la ligne), un retour de chariot sera effectué et la sortie continuera sur la ligne suivante.

(Voir PRINT USING/LPRINT USING, SPC et TAB)

PRINT USING/ LPRINT USING

FORMAT | PRINT | USING <"format chaîne">; [<expression>[;]
| LPRINT |
<expression>....]]

OBJET Sortir des chaînes ou des valeurs numériques en utilisant un format déterminé

EXEMPLE PRINT USING "####" ; A,B

REMARQUES PRINT USING/LPRINT USING détermine la zone et le format de l'<expression> à sortir par le <"format chaîne">. PRINT USING permet d'afficher des chaines ou des nombres sur l'écran, tandis que LPRINT USING permet de sortir les mêmes éléments sur la micro-imprimante.

Zones chaîne

! Spécifie que seul le premier caractère d'une chaîne donnée sera sorti.

\ (n blancs) \ . . . Spécifie que (n + 2) caractères depuis le début de la chaîne sont sortis. Si la chaîne est plus longue que la zone, les caractères supplémentaires sont ignorés.
Si la zone est plus longue que la chaîne, la chaîne est justifiée à gauche dans la zone et comblée par des espaces à droite. (Voir note ci-dessous).

& Utilisé pour sortir des chaînes de caractères. Lorsqu'un nombre & est spécifié, la valeur d'une variable affectée à chaque & sera délivrée. Si le nombre ("&") est supérieur au nombre des chaînes de la liste des expressions, les "&" supplémentaires seront ignorés.

Zones numériques

Un signe numérique est utilisé pour représenter chaque position de chiffre. Les positions de chiffres sont toujours remplies. Si le nombre à délivrer a moins de chiffres que de positions spécifiées, le nombre est justifié à droite (et est alors précédé par des espaces) dans la zone. (Voir note ci-dessous).

. Un point décimal peut être introduit en une position quelconque de la zone. Si le format de la zone spécifie qu'un chiffre doit précéder un point décimal, le chiffre est toujours sorti sous la forme 0. Les nombres sont arrondis selon la nécessité. Dans le dernier exemple, trois espaces ont été placés à la fin du format de la chaîne pour séparer les valeurs imprimées sur la ligne.

+ Un signe plus au début ou à la fin d'une chaîne de format fait sortir le signe du nombre (plus ou moins) avant ou après le nombre. Si deux ou plusieurs signes sont placés successivement, les signes plus supplémentaires sont traités comme des «caractères autres que les caractères de formattage» décrits ultérieurement.

– Un signe moins à la fin d'une chaîne de format fait délivrer les nombres négatifs avec un signe moins à la fin. Si un signe moins est placé au début d'une chaîne de format, ou si deux ou plusieurs signes moins sont indiqués successivement, les signes moins sont traités comme des "caractères autres que des caractères de formattage" décrits plus loin

** Un double astérisque au début d'une chaîne de format fait combler les espaces se trouvant au début de la chaîne numérique par des astérisques. Les ** spécifient aussi les positions de deux chiffres supplémentaires.

$$ Un double signe dollar au début d'une chaîne de format fait sortir un signe dollar immédiatement à gauche du nombre formatté. Les SS spécifient deux positions de chiffres supplémentaires, dont l'une est le signe dollar. Le format exponentiel (" ^^^^ ") défini ci-dessous ne peut pas être utilisé avec SS. (voir la note ci-dessous).

**$ Les **$ au début d'une chaîne de format associent les effets des deux symboles ci-dessus (" ** " et "$ $ "). Les espaces du début seront comblés par des astérisques et un signe dollar sera délivré avant le nombre. **$ spécifient trois positions de chiffres supplémentaires, dont l'une est utilisée comme zone de sortie pour un signe dollar.

, Une virgule qui se trouve à gauche du point décimal dans une chaîne de format fait sortir une virgule à gauche de chaque troisième chiffre situé à gauche du point décimal. Si une virgule est placée à droite du point décimal dans une chaîne de format, une virgule est délivrée à la fin du nombre formatté.

^^^^ Quatre accents circonflexes (ou flèches vers le haut) peuvent être placés après les caractères de position des chiffres (" # ") pour spécifier un format exponentiel. (Voir note ci-dessous).

—— Utilisé pour représenter l'un quelconque des caractères de formattage ci-dessus comme un caractère littéral. Un souligné dans une chaîne de format fait sortir le caractère suivant comme un caractère qui n'a pas de fonction de formattage.

% Si le nombre à sortir est plus grand que la zone spécifiée, un signe % est délivré en avant du nombre. Si l'arrondi conduit à un nombre qui dépasse la zone, un signe % sera imprimé en avant du nombre arrondi.

Caractères autres que des caractères de formattage

Si des caractères différents des caractères de formattage ci-dessus (c'est à dire alphanumériques, symboles graphiques, etc...), sont placés au début ou à la fin d'une chaîne de format, ces caractères sont délivrés avant ou après le nombre formatté.

NOTE: Les caractères de formattage indiqués ci-dessous concernent le jeu des caractères ASCII. Si vous avez choisi un jeu de caractères différent du jeu ASCII, certains des caractères de formattage sont délivrés différemment comme indiqué ci-dessous.

USASCII	France	Allemagne	Angleterre	Danemark	Suède	Italie	Espagne
#	#	#	£	#	#	#	Pt
$	$	$	$	$	¤	$	$
\	ç	Ö	\	ø	Ö	\	Ñ
^	^	^	^	^	Ü	^	^

(Voir chapitre 2.4, Jeux de Caractères, pour avoir des informations détaillées).

EXEMPLE DE PROGRAMME

```
100 A$="EPSON HX-20"
110 B=123.456
120 C=-123.456
130 D=1234.56
140 E=123456
150 PRINT USING "!";A$
160 PRINT USING "\       \
";A$
170 PRINT USING "THIS IS
A &";A$
180 PRINT USING "###";B
190 PRINT USING "###.#";
B
200 PRINT USING "+##.#";
B
210 PRINT USING "###.#-"
;C
220 PRINT USING "**#.#";
B
230 PRINT USING "$$#.#";
B
240 PRINT USING "**$.#";
B
250 PRINT USING "###.#^
^^^";D
260 PRINT USING "$$#.#-"
;D
270 PRINT USING "###";E
280 END
RUN
E
EPSON H
THIS IS A EPSON HX-2
0
123
123.5
%+123.5
123.5-
123.5
%$123.5
%$123.5
123.5E+01
123.5E+01
%$1235.-
%123456
>
```

PRINT

FORMAT **PRINT #<numéro de fichier>, [<expression>...]**

OBJET Ecrire des données dans un fichier séquentiel.

EXEMPLE **PRINT #1, 1, B**

REMARQUES Le <numéro de fichier> est le numéro utilisé au moment où le fichier a été ouvert avec l'instruction OPEN pour la sortie. L'<expression>est l'expression numérique ou chaîne qui a été écrite dans le fichier. Pour CAS0: et CAS1: , PRINT # , contrairement au cas de la sortie sur l'écran d'affichage, délimite automatiquement les données avant de les écrire dans le fichier.
Par conséquent, que l'on utilise des virgules ou des points virgules comme séparateur dans la liste des expressions, les données sont sorties avec le même format.

(Voir INPUT #, PRINT/LPRINT USING et PRINT #USING)

PRINT # USING

FORMAT **PRINT #<numéro de fichier>, USING <"chaîne de format">;**
[<expression>[; | , <expression>...]]

OBJET Ecrire des chaînes et des nombres dans un fichier séquentiel en utilisant un format particulier.

EXEMPLE **PRINT #1, USING "###" ; A**

REMARQUES PRINT # USING écrit des expressions chaînes ou numériques dans le fichier séquentiel spécifié par le <numéro de fichier>en utilisant le format spécifié par la <" chaîne de format" >. Reportez-vous à PRINT USING au sujet de la <"chaîne de format" >.
PRINT # USING délivre des données dans un format presque identique à celui qui sort sur l'écran d'affichage. Donc, lors de la lecture d'un fichier de données de sortie par PRINT USING, les données ne sont pas séparées, sauf si vous utilisez des séparateurs ; des virgules pour les expressions numériques et des guillemets pour les expressions chaînes.

(Voir OPEN, PRINT USING et PRINT #)

PSET

FORMAT **PSET (<coordonnée horizontale>, <coordonnée verticale>)[, <couleur>]**

OBJET Tracer des points sur un écran graphique spécifié.

EXEMPLE **PSET (30,20)**

REMARQUES PSET trace un point en un endroit spécifié de l'écran graphique désigné par une instruction SCREEN. PSET n'affecte pas les points déjà tracés sur l'écran. Si le contrôleur d'affichage en option est utilisé, prenez des précautions quand vous spécifiez les coordonnées étant donné que la configuration de l'écran extérieur est différente de celle de l'écran LCD. Si vous indiquez l'écran extérieur en mode graphique en couleur, à l'aide d'une instruction SCREEN, <couleur> peut également être spécifié dans ce cas. Si la couleur des points existants est différente de la <couleur> que vous avez sélectionnée, elle changera au profit d'une nouvelle couleur spécifiée. Si vous n'indiquez pas la <couleur>, la couleur du point tracé sur l'écran sera la même que celle des tracés, spécifiée dans une instruction COLOR.

(Voir COLOR, PRESET et SCREEN)

EXEMPLE DE PROGRAMME

```
LIST

100 CLS:PI=3.14159
110 DEFINT X,Y
120 FOR I=0 TO 2*PI STEP PI/54
130 X=COS(I)*15+64
140 Y=SIN(I)*15+16
150 PSET(X,Y)
160 NEXT:COPY
>
```

PUT%

FORMAT PUT% <numéro d'enregistrement>, <variable> [, <variable>...]

OBJET Ecrire les valeurs des variables dans un fichier RAM

EXEMPLE PUT%0 , A!, B # , C$

REMARQUES PUT% écrit les valeurs des variables dans l'enregistrement spécifié par le <numéro d'enregistrement>. Les variables chaînes doivent être écrites à la fin de la liste des variables et une seule variable chaîne est admise par enregistrement.

Si toutes les valeurs des variables chaînes ne peuvent pas être écrites dans l'enregistrement spécifié, la portion des variables chaînes en excès est ignorée. S'il reste une capacité supplémentaire dans l'enregistrement après l'écriture de la dernière variable, l'enregistrement sera comblé avec des espaces.

(Voir DEFFIL, GET% et fichiers RAM)

EXEMPLE DE PROGRAMME

```
100 DEFFIL 20,0
110 A=1.2:B%=3:C$="ABCDE
FG"
120 PUT%0,A,B%,C$
130 D=0.123:E$=C$+C$
140 PUT%1,A,B%,D,E$
150 'GET% CAN BE EXECUTE
D EVEN FOR TYPES DIFFERE
NT THAN THE PUT% STATEME
N BUT THE VALUE WILL BE
REFUSED
160 FOR I=0 TO 1
170 PRINT
180 GET% I,J,K%,L$
190 PRINT I;J;K%;L$
200 GET% I,J,K%,M,M$
210 PRINT I;J;K%;M;M$
220 NEXT I
230 END

 0  1.2  3
ABCDEFG
 0  1.2  3  8.22735E-20
EFG

 1  1.2  3
3 mABCDEFGABC
 1  1.2  3  .123
ABCDEFGABC
```

RANDOMIZE

FORMAT RANDOMIZE [<expression>]

OBJET Réensemencer le générateur de nombres aléatoires.

EXEMPLE RANDOMIZE

REMARQUES RANDOMIZE change la séquence des nombres aléatoires en donnant une valeur d'initialisation au générateur de nombres aléatoires. La valeur d'initialisation doit être spécifié par < expression > dans la gamme –32768 à 32767. Si vous omettez l'<expression > un message "valeur d'initialisation?" est affiché au moment de l'exécution de RANDOMIZE pour demander la valeur d'initialisation. Si la valeur introduite est en dehors de la gamme indiquée ci-dessus, une erreur apparaît. Les instructions RANDOMIZE ne peuvent pas être exécutées en mode direct.

READ

FORMAT READ <variable >[, <variable > ...]

OBJET Lire les valeurs d'une instruction DATA et les affecter aux variables.

EXEMPLE READ A, I, C$

REMARQUES Une instruction READ doit toujours être utilisée avec une instruction DATA. Les instructions READ affectent les variables aux valeurs de l'instruction DATA avec une correspondance une à une. Les variables d'instruction READ peuvent être numériques ou chaînes et les valeurs lues doivent correspondre aux types de variables spécifiés. Si elles ne correspondent pas, une "erreur de syntaxe" en résulte.

Si le nombre des variables dans la liste des variables des instructions READ dépasse le nombre des éléments des instructions DATA, une erreur OD ("données épuisées") apparaît.

Si le nombre des variables spécifié est inférieur au nombre des éléments des instructions DATA, les instructions READ suivantes commencent à lire des données dans le premier élément qui n'a pas été lu. S'il n'y a pas une autre instruction READ, les données supplémentaires sont ignorées.

(Voir DATA et RESTORE)

REM

FORMAT **REM [<remarque>]**

OBJET Faire des remarques explicatives qui sont insérées dans un programme.

EXEMPLE **REM COMMENT MESSAGE**

REMARQUES Une instruction REM est une instruction non exécutable c'est à dire qu'elle ressort exactement comme elle a été rentrée dans le programme.
Dans une instruction REM, le mot "REM" peut être remplacé par une apostrophe " ' ".
Dans une instruction REM, un caractère deux points (":") est considéré comme une partie de la remarque. Par conséquent, aucune instruction ne peut être placée après l'instruction REM.

RENUM

FORMAT **RENUM [<Nouveau numéro de ligne>][, [<ancien numéro de ligne>][, [<incrément>]]**

OBJET Renuméroter les lignes de programmes.

EXEMPLE **RENUM**

REMARQUES Le < nouveau numéro de ligne > est le premier numéro de ligne à utiliser dans la nouvelle séquence. La valeur implicite est 10. L'<ancien numéro de ligne > est la ligne du programme en cours au moment ou la renumérotation doit commencer. La valeur implicite est la première ligne du programme. L' < incrément > est l'incrément à utiliser dans la nouvelle séquence. La valeur implicite est 10. RENUM change aussi toutes les références aux numéros de lignes qui suivent les instructions GOTO, GOSUB, THEN, ON...GOTO, ON...GOSUB et ERL pour tenir compte des nouveaux numéros de lignes. Si un numéro de ligne inexistant dans le programme apparaît après une de ces instructions, le message d'erreur " numéro de ligne nnnn non défini dans mmmm" apparaît. La référence au numéro de ligne correcte (nnnn) n'est pas changé par RENUM, mais le numéro de ligne mmmm peut être changé. Une commande RENUM ne peut pas être utilisée pour changer l'ordre des lignes de programme ni pour créer des numéros de ligne supérieurs à 64000. Une erreur FC ("appel de fonction non valable") résulte de l'emploi de RENUM en infraction aux règles ci-dessus. Dans ce cas, les numéros de lignes du programme resteront inchangés. L'instruction de renumérotation n'est valable que dans la zone de programme actuellement utilisée.

RESTORE

FORMAT **RESTORE** [<**numéro de ligne**>]

OBJET Permettre de relire des instructions DATA à partir d'un point spécifié.

EXEMPLE **RESTORE 1000**

REMARQUES RESTORE fait accéder l'instruction READ suivante au premier élément de l'instruction DATA de la ligne spécifiée par le <numéro de ligne>. Si le <numéro de ligne> n'est pas indiqué, l'instruction READ suivante lit le premier élément DATA du programme. S'il n'y a pas d'instruction DATA à la ligne spécifiée par le <numéro de ligne>, l'instruction READ lit à partir de l'instruction DATA au numéro de ligne, après le <numéro de ligne> spécifié.

Si la ligne spécifiée par le <numéro de ligne> n'existe pas dans le programme, une erreur UL ("numéro de ligne non définie") apparaîtra.

(Voir DATA et READ)

EXEMPLE DE PROGRAMME

```
LIST

100 DATA "*************","   HX -",20
110 READ A$
120 PRINT A$
130 READ A$,B%
140 PRINT A$;B%
150 RESTORE 100
160 READ A$
170 PRINT A$
180 END
RUN
*************
   HX - 20
*************
>
```

RESUME

FORMAT RESUME [| NEXT / <numéro de ligne> |]

OBJET Continuer l'exécution d'un programme après exécution d'une procédure de traitement d'erreur.

EXEMPLE **RESUME 100**

REMARQUES Avec RESUME, vous pouvez spécifier l'instruction ou la ligne de programme à laquelle l'exécution du programme doit reprendre après l'application d'une procédure de reprise sur erreur. Les trois formats en option sont les suivants:

RESUME	L'exécution reprend à l'instruction qui a provoqué l'erreur.
RESUME NEXT	L'exécution reprend à l'instruction qui suit immédiatement celle qui a provoqué l'erreur.
RESUME <numéro de ligne>	L'exécution reprend à la ligne spécifiée par le <numéro de ligne>.

(Voir ON ERROR GOTO)

RUN

FORMAT (1) **RUN [<numéro de ligne>]** ou
(2) **RUN <descripteur de fichier> [, R]**

OBJET Démarrer l'exécution du programme.

EXEMPLE (1) **RUN 300**
(2) **RUN "CAS0: PROG4.ASC"**

REMARQUES Dans le format (1), si le <numéro de ligne> est spécifié, l'exécution du programme présent dans la mémoire commence à cette ligne. Si le <numéro de ligne> n'est pas indiqué, l'exécution commence au numéro deligne le plus faible.
Le format (2), un programme spécifié par le <descripteur de fichier> est chargé dans la mémoire et alors l'exécution du programme commence. L'exécution d'une commande RUN annule toutes les variables et ferme tous les fichiers ouverts avant le chargement du programme désigné. Mais, si <R> est utilisée avec cette commande, tous les fichiers de données restent ouverts. Avant d'exécuter une commande RUN en format (2), utilisez une commande STAT pour vérifier la zone de programme actuellement utilisée.

(Voir LOGIN et STAT)

RUN"COM0:"

FORMAT **RUN "COM0: [(<BLPSC>)]" [, R]**

OBJET Spécifier les conditions d'interface de la porte RS-232C, et exécuter RUN.

EXEMPLE **RUN"COM0: (68N2B)"**

REMARQUES En dehors du fait que le programme est chargé dans la mémoire par l'intermédiaire de la porte RS-232C, cette commande est essentiellement la même que la commande normale RUN. Mais, pour être transmis par l'intermédiaire de cette porte, le programme doit être en format ASCII. Vous aurez des détails au sujet de <BLPSC>, avec spécification de l'interface de la porte RS-232C en vous reportant à OPEN"COM0:".

SAVE

FORMAT SAVE <description de fichier> [, | A |]
| V |

OBJET Sauvegarder un programme EPSON BASIC sur un fichier spécifié.

EXEMPLE SAVE "CAS0:ABC"

REMARQUES Cette commande permet de sauvegarder des programmes BASIC sur le fichier spécifié par le <descripteur de fichier>. S'il existe déjà un fichier sous le nom de fichier spécifié, le programme peut être sauvegardé en un autre endroit sous le même nom de fichier. Dans ce cas, deux programmes sauvegardés différents ne peuvent pas se distinguer l'un de l'autre étant donné qu'ils sont sous le même nom de fichier. Veillez à ne pas utiliser le même nom de fichier. L'option <A> sauvegarde un fichier en format ASCII. Si un <A> n'est pas spécifié, le fichier est sauvegardé en format binaire comprimé. Le format ASCII demande plus d'espace que le format binaire, mais l'accès à certains fichiers exige que le fichier soit en format ASCII (par exemple, lorsque l'on utilise la commande MERGE pour l'édition d'un programme). Un fichier sauvegardé en format ASCII peut aussi être lu comme fichier de données.

L'option < V > peut être utilisée lorsque l'on emploie un enregistreur à microcassette comme mémoire auxilliaire. Dans ce cas, lorsque la commande SAVE est exécutée, la bande est automatiquement réenroulée jusqu'au début du programme et la vérification (vérification CRC) est effectuée. Si le <dispositif> spécifié n'est pas l'enregistreur à microcassette, l'option <V> est ignorée.

(Voir LOAD et MERGE)

SAVE"COM0:"

FORMAT **SAVE"COM0: [(<BLPSC>)] ", A**

OBJET Spécifier les conditions d'interface de la porte RS-232C et exécuter la commande SAVE.

EXEMPLE **SAVE"COM0: (68E13)" , A**

REMARQUES En dehors du fait que le programme est sorti par l'intermédiaire de la porte RS-232C, cette commande est essentiellement la même que la commande SAVE. Si A est omis, aucune erreur n'intervient. Mais, étant donné que les données en format binaire condensées ne peuvent pas être lues correctement par la porte RS-232C, une tentative de transmission de ces données n'a aucune signification. Reportez-vous à OPEN"COM0:" pour avoir des détails au sujet de <BLPSC>, et des spécifications sur les conditions d'interface du RS-232C.

SAVEM

FORMAT **SAVEM <descripteur de fichier>, <adresse supérieure>, <adresse inférieure>, <adresse de début d'exécution> [, V]**

OBJET Sauvegarder le contenu dans la mémoire sur un fichier spécifié.

EXEMPLE **SAVEM"CAS1:ABC" , &H0C00, &H0B00, &H0B00**

REMARQUES SAVEM sauvegarde un programme ou un contenu de mémoire en langage machine sur un fichier spécifié.

L'< adresse supérieure > et l'< adresse inférieure > indiquent la gamme du contenu de la mémoire à sauvegarder sur un fichier spécifié.

Si l'option <V > est spécifiée, la vérification du programme (vérification CRC) est effectuée après l'exécution de SAVEM. Si vous utilisez un autre dispositif que l'enregistreur à microcassette, l'option <V > sera ignorée.

L'exécution du programme en langage machine chargé dans la mémoire par la commande LOADM commencera à l'< adresse de début d'exécution >. Même si les données sauvegardées ne correspondent pas à un programme en langage machine, l'< adresse de début d'exécution > ne peut être omise et la même valeur que l'<adresse supérieure > doit être indiquée.

(Voir LOADM)

SCREEN

FORMAT SCREEN <texte>, <mode graphique>
OBJET Spécifier le mode de l'écran : texte ou graphique.
EXEMPLE SCREEN 0,2

REMARQUES SCREEN désigne l'écran LCD ou l'écran extérieur en mode d'écran de texte et/ou en mode d'écran graphique. (Le contrôleur d'affichage en option est nécessaire pour connecter le HX-20 à un affichage extérieur).
La valeur du <texte> doit être soit 0, soit 1. Spécifiez 0 pour afficher un écran texte (écran matériel) sur l'écran LCD et 1 pour afficher un écran texte sur l'écran extérieur.

La valeur du <mode graphique> doit être comprise entre 0 et 2. Spécifier 0 pour afficher un écran texte sur le LCD. Si vous spécifiez 1, l'écran extérieur passe en mode graphique couleur (128 x 64 points, 4 couleurs). Si vous spécifiez 2, l'écran extérieur est réglé en mode à haute résolution (128 x 96 points, en noir et blanc). En cas de démarrage à chaud, le LCD est lancé à la fois en mode de texte et en mode graphique.

SCREEN 0,0 . . . Affiche à la fois des écrans textes et graphiques sur le LCD.
SCREEN 0,1 . . . Affiche un écran texte sur le LCD et un écran graphique en couleur sur l'affichage extérieur.
SCREEN 0,2 . . . Affiche un écran texte sur le LCD et un écran graphique en mode haute résolution sur l'affichage extérieur.
SCREEN 1,0 . . . Affiche un écran texte sur l'affichage extérieur et un écran graphique sur le LCD.

Etant donné que les possibilités du contrôleur d'affichage sont limitées, il n'est pas possible de visualiser simultanément un écran texte et un écran graphique sur l'affichage extérieur.

NOTE : L'option contrôleur d'affichage pourra ne pas être disponible à la date de parution de ce manuel.

SCROLL

FORMAT **SCROLL [< vitesse >][, [< mode >][, < pas de défilement >, < pas de défilement Y >]]**

OBJET Spécifier le défilement de l'écran matériel.

EXEMPLE **SCROLL 9,0,10,4**

REMARQUES La < vitesse > spécifie la vitesse à laquelle défile l'écran LCD en utilisant une valeur de la gamme 0 à 9, 9 indiquant la vitesse la plus élevée possible et 0, la plus faible. Mais la vitesse de défilement de l'affichage extérieur est fixe et non réglable.

Le < mode > spécifie le type de mouvement de l'écran matériel et sa valeur doit être soit 0, soit 1. Si vous spécifiez < mode >0, l'écran matériel est fixe du côté gauche de l'écran virtuel et ne peut se déplacer qu'en haut et en bas. Avec ce mode, si la position du curseur est en dehors des limites de l'écran matériel, notez qu'il ne reviendra pas dans les limites sauf si l'écran matériel est déplacé, même si une commande d'affichage est exécutée.

Si vous spécifiez le < mode >1, l'écran matériel se déplacera automatiquement pour suivre le curseur. Quand le système revient au niveau des commandes, après l'exécution d'un programme, le < mode > est fixé à 0.

Le < pas de défilement X > indique le nombre de caractères dont l'écran se déplace lorsque vous tapez **CTRL** + ↵ ou **CTRL** + ← . La valeur du < pas de défilement X > doit être comprise entre 1 et 20 pour le LCD et entre 1 et 32 pour l'affichage extérieur. Le < pas de défilement Y > indique le nombre de lignes dont l'écran doit se déplacer verticalement à la fois lorsque vous appuyez sur **CTRL** + P, **CTRL** + Q ou **SC : RN**

La valeur du < pas de défilement Y > doit être dans la gamme de 1 à 4 pour le LCD et de 1 à 16 pour l'écran extérieur. Les valeurs implicites au démarrage à chaud sont les suivantes:

LCD	SCROLL 9, 0, 10, 4
Ecran extérieur	SCROLL 9, 0, 16, 16

SOUND

FORMAT SOUND <tonalité>, <durée>

OBJET Faire retentir une tonalité spécifiée.

EXEMPLE SOUND 10, 10

REMARQUES SOUND fait délivrer une < tonalité > spécifiée par le haut parleur piezoélectrique incorporé pendant une <durée > déterminée.

La < tonalité > doit être une valeur de la gamme 0 à 56, 0 indique une pause et la gamme de 1 à 28 correspond à l'échelle de la tonalité C (do) à la tonalité B (si) qui est de 4 octaves au dessus du premier C. 13 équivaut à 880 Hz. La gamme de 29 à 56 correspond à l'échelle des tonalités dont chacune est à un demi ton au-dessus de celles de l'échelle représentée par 1 à 28.

La <durée > peut être une valeur de la gamme de 0 à 255. Si la <durée > est spécifiée, le haut-parleur émet un son pendant une durée spécifiée par la <durée > multipliée par 0,1 seconde.

EXEMPLE DE PROGRAMME

```
LIST

100 FOR I=1 TO 68
110 READ A,B
120 SOUND A,B
130 NEXT
140 END
200 DATA 8,8,8,4,9,4,10,8,10,4,11,4,12,8
,13,4,12,4,10,16
210 DATA 12,8,11,4,10,4,9,16,11,8,10,4,9
,4,8,16
220 DATA 8,8,8,4,9,4,10,8,10,4,11,4,12,8
,13,4,12,4,10,16
230 DATA 12,8,11,4,10,4,9,8,10,4,9,4,8,3
2
240 DATA 12,8,11,4,10,4,9,16,11,8,10,4,9
,4,8,16
250 DATA 12,8,11,4,10,4,9,16,11,8,10,4,9
,4,8,16
260 DATA 8,8,8,4,9,4,10,8,10,4,11,4,12,8
,13,4,12,4,10,16
270 DATA 12,8,11,4,10,4,9,8,10,4,9,4,8,2
4
>
```

STAT

FORMAT STAT [ALL | <expression>]

OBJET Afficher l'état de chaque zone de programme

EXEMPLE STAT 3

REMARQUES STAT affiche le nom et la dimension du programme stocké dans chaque zone de programme. L'<expression>est le numéro de la zone de programme (1 à 5) Si vous utilisez l'< expression >, les données de la zone spécifiée par le numéro de zone de programme sont affichées. Sinon, les données de la zone de programme en cours sont affichées. Si ALL est spécifié, les données de toutes les zones de programmes sont affichées simultanément avec la dimension de la zone du fichier RAM, les adresses MEMSET et la dimension de la zone de texte inutilisée. Cependant, comme la zone de texte inutilisée comprend une zone de travail pour l'exécution des programmes BASIC, toute cette zone n'est pas nécessairement disponible complètement pour le stockage des programmes.

(Voir MEMSET et TITLE)

STOP

FORMAT STOP
OBJET Terminer l'exécution d'un programme et retourner au niveau des commandes.
EXEMPLE STOP

REMARQUES Les instructions STOP peuvent être utilisées n'importe où dans un programme pour en achever l'exécution et pour ramener le BASIC au niveau des commandes. Lors de l'exécution d'une instruction STOP, le message suivant est délivré:

Interruption en nnn (nnn est le numéro de la ligne à laquelle l'instruction STOP a été exécutée).

Contrairement à l'instruction END, l'instruction STOP ne ferme pas les fichiers. Lorsque le BASIC est retourné au niveau des commandes, l'exécution du programme peut reprendre au moyen d'une commande CONT.

(Voir CONT et END)

SWAP

FORMAT SWAP <variable 1>, <variable 2>
OBJET Echanger les valeurs de deux variables
EXEMPLE SWAP A$, B$

REMARQUES Avec une instruction SWAP, vous pouvez échanger n'importe quel type de variable (entière, simple précision, double précision ou chaîne), mais les deux variables doivent être du même type, sinon une erreur TM ("non concordance des types") apparaît.
Lorsque le type de la <variable 2> est une variable simple, et que la valeur de la variable n'est pas indiquée, une erreur FC ("appel de fonction non valable") apparaît.

EXEMPLE DE PROGRAMME

```
LIST

100 A=123.456
110 B=999999
120 PRINT "A=";A
130 PRINT "B=";B
140 SWAP A,B
150 PRINT
160 PRINT "SWAP!!"
170 PRINT
180 PRINT "A=";A
190 PRINT "B=";B
200 END
RUN
A= 123.456
B= 999999

SWAP!!

A= 999999
B= 123.456
>
```

TITLE

TITLE <nom de programme>
Nommer les programmes
TITLE "TEST1"

TITLE nomme un programme de la zone de programme en cours. Le <nom de programme > peut comprendre 8 caractères. Une fois affecté, ce nom apparaît sur le menu lorsque la commande STAT est exécutée ou lorsque le HX-20 est mis sous tension.
Le programme nommé par l'instruction TITLE peut être exécuté directement par la sélection du menu immédiatement après application de la tension au HX-20 et il est protégé contre un effacement accidentel par une commande NEW ou LOAD. Si vous essayez d'exécuter l'une ou l'autre commande dans une zone de programme nommée par la commande TITLE, une erreur PP ("programme protégé") apparaît.

(Voir STAT)

TRON/TROFF

FORMAT TRON
TROFF

OBJET Suivre l'exécution des instructions du programme.

REMARQUES L'exécution d'une instruction TRON (soit en mode direct, soit en mode indirect) permet le déverminage d'un programme en générant une suite de références qui comprend chaque numéro de ligne de programme au moment où elle est exécutée. Les numéros apparaissent entre crochets. La suite de références est invalidée lorsqu'une instruction TROFF ou une commande NEW est exécutée.

EXEMPLE DE PROGRAMME

```
LIST

10 FOR I=1 TO 5
20 PRINT I
30 NEXT
40 END
TRON
RUN
[10][20] 1
[30][20] 2
[30][20] 3
[30][20] 4
[30][20] 5
[30][40]
TROFF
RUN
 1
 2
 3
 4
 5
>
```

WIDTH

FORMAT **WIDTH < caractères par ligne >, < nombre de lignes >[, < marge de défilement >]**

OBJET Fixer la dimension de l'écran virtuel.

EXEMPLE **WIDTH 20, 25, 5**

REMARQUES WIDTH spécifie la dimension de l'écran virtuel en < caractères par ligne > et en < nombre de lignes >. Les < caractères par ligne > et < nombre de lignes > peuvent être fixés arbitrairement dans les gammes respectives de 20 à 255 et de 4 à 255. La dimension de l'écran virtuel est limitée par la capacité de la mémoire. Si vous demandez une largeur d'écran trop grande, une erreur OM ("en dehors de la mémoire") apparaît. Les valeurs minimales sont les dimensions de l'écran LCD.
La < marge de défilement > peut être fixée dans la gamme 1 à 10.

La < marge de défilement > est une tolérance sur les marges que vous pouvez prévoir à gauche et à droite de l'écran d'affichage de façon à pouvoir lire facilement l'écran sans qu'il soit déplacé latéralement.
Lorsque le curseur arrive au bord de l'écran, le caractère suivant est en dehors de l'écran d'affichage et par conséquent, n'est pas visible. C'est pourquoi, quand le curseur s'est déplacé sur une certaine valeur fixée, il est conseillé de déplacer également l'écran pendant que le caractère suivant est encore visible. De cette façon, le caractère situé à gauche du curseur est toujours sur l'écran. Lorsque le curseur s'est déplacé sans quitter la marge de défilement, cela indique que le curseur a atteint l'extrémité de l'écran virtuel.

Au moment du démarrage à chaud, la dimension de l'écran virtuel est fixée à WIDTH 40, 8, 3 pour le LCD et WIDTH 40, 37, 5 pour l'écran externe. Lorsqu'une commande WIDTH est exécutée, tous les fichiers ouverts à ce moment sont fermés et toutes les variables, les zones de caractères et les données définies par les instructions DEF sont effacées.
Vous pouvez fixer librement la dimension de l'écran virtuel.
Lorsque vous effectuez des corrections sur les programmes, plus l'écran virtuel est grand, plus vous pouvez l'utiliser facilement. Mais, avec un écran très large vous risquez de ne pas voir les résultats des instructions. Même si les instructions sortent correctement sur l'écran virtuel, il risque de n'y avoir aucun changement visible sur l'écran d'affichage.

WIDTH <device name>

FORMAT WIDTH | "LPT0:" | "COM0:" | ,<nombre de chiffres>

OBJET Régler la largeur d'impression de l'imprimante.

EXEMPLE WIDTH "LPT0:" , 20

REMARQUES WIDTH fixe la valeur maximale du nombre de caractères par ligne sortie sur l'imprimante. Normalement, pour les instructions PRINT (y compris LPRINT et PRINT #) le signe LINE FEED (LF) est envoyé sur l'imprimante par un point virgule à la fin de chaque instruction de ce type. En spécifiant la largeur d'impression avec une commande WIDTH, le signe LF est envoyé automatiquement à l'imprimante lorsque le <nombre de chiffres> qui définit le nombre des caractères à imprimer par ligne est atteint. Le <nombre de chiffres> doit être compris entre 1 et 255. Mais 255 est considéré comme un nombre infini et aucun changement de ligne automatique n'interviendra selon la largeur d'impression spécifiée.

NOTE:

Au démarrage à chaud, le <nombre de chiffres> est fixé à 80 pour la porte RS-232C ("COM0:"). Par conséquent, si une longue chaîne dépassant 80 caractères est délivrée, le changement de ligne est effectué automatiquement après la sortie des 80 caractères. Le résultat est le même si un dispositif différent de l'imprimante est connecté à la porte RS-232C du HX-20. C'est pourquoi, lorsque vous utilisez la porte RS-232C pour communiquer avec un dispositif extérieur, vous ne pouvez pas transmettre correctement une chaîne de plus de 80 caractères avec le <nombre de chiffres> fixé à la valeur implicite. Lorsque les programmes doivent être transférés du HX-20 à un autre HX-20, vous devez d'abord exécuter WIDTH "COM0:" , 255 pour invalider l'opération de changement de ligne automatique et puis démarrer la transmission.

WIND

FORMAT **WIND** [< **valeur du compteur** >]

OBJET Commander le moteur de la microcassette pour le déroulement rapide et le réenroulement.

EXEMPLE **WIND 0**

REMARQUES WIND fait dérouler la bande de la microcassette rapidement ou la fait réenrouler selon la < valeur du compteur >. La < valeur du compteur > doit être comprise dans la gamme –32768 à 32767.

La valeur du compteur de l'enregistreur à microcassette peut être lue par la fonction TAPCNT.

Lorsque la valeur est stockée en mémoire, une recherche peut être effectuée dans le fichier objet, quelle que soit la position de la bande en indiquant la valeur au moyen de la commande WIND.

Si la < valeur du compteur > n'est pas indiquée, l'enregistrement à microcassette réenroule la bande jusqu'au début et ramène le compteur à zéro.

Lorsqu'un fichier est recherché à l'aide d'une commande WIND, vous devez exécuter d'abord une opération WIND sans < valeur du compteur > pour bien positionner la cassette à zéro.

(Voir TAPCNT)

EXEMPLE DE PROGRAMME

```
LIST

100 COUNT=TAPCNT
110 OPEN "O",#1,"CAS0:TEST.DAT"
120 FOR A=1 TO 10
130 PRINT#1,A:SQR(A)
140 NEXT
150 CLOSE #1
160 WIND COUNT
170 OPEN "I",#1,"CAS0:TEST.DAT"
175 PRINT :PRINT" A ","SQR(A)"
180 IF EOF(1) GOTO 220
190 INPUT#1,A,B
200 PRINT A,B
210 GOTO 180
220 CLOSE #1
230 END
RUN

 A              SQR(A)
 1               1
 2               1.41421
 3               1.73205
 4               2
 5               2.23607
 6               2.44949
 7               2.64575
 8               2.82843
 9               3
 10              3.16228
>
```

Chapitre 4

Fonctions

ABS

FORMAT **ABS (< expression numérique >)**
OBJET Renvoyer la valeur absolue d'une expression numérique.
EXEMPLE **A = ABS (– 1.6)**

REMARQUES ABS renvoie la valeur absolue de l'< expression numérique >.

ASC

FORMAT **ASC (< chaîne >)**
OBJET Renvoyer le code de caractère d'un caractère
EXEMPLE **A = ASC ("A")**

REMARQUES ASC renvoie une valeur numérique qui correspond au code ASCII du premier caractère de la < chaîne >. Voir à l'Annexe F, la table des "Codes de caractères" indiquant les relations entre les caractères et les codes de caractères ASCII. Si la < chaîne > est nulle, une erreur FC ("appel de fonction illégal") est renvoyée.

(Voir CHR$, et l'Annexe F, "Table des Codes de Caractères)

ATN

FORMAT **ATN (<expression numérique >)**

OBJET Renvoyer l'arc tangente d'une expression numérique.

EXEMPLE **A = ATN (0.5)**

REMARQUES ATN renvoie l'arc tangente de l'< expression numérique > en radians. Le résultat de l'opération est situé dans la gamme $-\pi/2$ ou $\pi/2$.

CDBL

FORMAT **CDBL (<expression numérique >)**

OBJET Convertir les entiers et les nombres simple précision en nombres double précision.

EXEMPLE **A# = CDBL (B! /2)**

REMARQUES CDBL convertit la valeur de l'< expression numérique >en un nombre double précision. Seule la conversion du type est effectuée et il n'y a aucun changement du nombre des chiffres significatifs.

CHR$

FORMAT **CHR$ (<expression numérique >)**

OBJET Renvoyer le caractère correspondant à un code de caractères spécifiés.

EXEMPLE **A$ = CHR$ (&H41)**

REMARQUES CHR$ renvoie le caractère ASCII dont le code est la valeur de l'<expression numérique >. Si la valeur de l'<expression numérique > n'est pas dans la gamme de 0 à 255, une erreur FC ("appel de fonction non valable") apparaît. Vous pouvez inclure des nombres réels dans l'<expression numérique >, mais la partie fractionnaire du nombre réel doit être arrondie avant qu'il soit utilisé comme valeur de l'<expression numérique >.

(Voir ASC et Annexe F: "Table des Codes de Caractères")

CINT

FORMAT **CINT (<expression numérique >)**

OBJET Convertir des nombres simple et double précisions en entiers.

EXEMPLE **A% = CINT (B# /2)**

REMARQUES CINT convertit la valeur de l'<expression numérique > en un entier, en arrondissant la partie fractionnaire. Si la valeur de l'<expression numérique > n'est pas dans la gamme de –32768 à 32767, une erreur OV ("débordement") apparaît.

(Voir CDBL, FIX et INT)

COS

FORMAT **COS (<expression numérique>)**

OBJET Renvoyer le cosinus d'une expression numérique

EXEMPLE **A = COS (3.1415926/2)**

REMARQUES COS renvoie le cosinus d'une <expression numérique> indiquée en radians.

CSNG

FORMAT **CSNG (<expression numérique>)**

OBJET Convertir des entiers et des nombres double précision en des nombres simple précision.

EXEMPLE **A! = CSNG (B#)**

REMARQUES CSNG convertit la valeur de l'<expression numérique> en un nombre simple précision à six chiffres significatifs.
Si la valeur de l'<expression numérique> n'est pas dans la gamme de −1.70141 E +38, une erreur OV ("débordement") apparaît.

(Voir CDBL et CINT)

CSRLIN

FORMAT **CRSLIN**

OBJET Renvoyer la position verticale du curseur sur l'écran virtuel.

EXEMPLE **Y– CSRLIN**

REMARQUES CSRLIN renvoie la valeur de la position verticale du curseur sur l'écran virtuel. La valeur de cette position doit se situer dans la gamme de 0 (jusqu'au nombre de lignes de l'écran virtuel –1).

(Voir POS)

DATE$

FORMAT **DATES** [= **MM/DD/YY**]

OBJET Introduire la date en cours et renvoyer la date gardée par l'horloge du calendrier interne.

EXEMPLE **PRINT DATE$**

REMARQUES DATES affiche la date gardée par l'horloge du calendrier interne dans le HX-20. Avec cette instruction, vous pouvez indiquer la date sous la forme d'une chaîne du type MM/DD/YY (par exemple "08/15/82") où MM représente le mois, DD le jour et YY l'année.

La date est affichée avec le format dans lequel elle a été introduite. Lorsque vous avez introduit la date correcte avec DATES, vous n'avez plus besoin de recommencer étant donné que l'horloge du HX-20 reste à l'heure et à la date.

(Voir DAY et TIME$)

DAY

FORMAT DAY [=<numéro du jour>]

OBJET Introduire le jour de la semaine et afficher le jour de la semaine gardé par l'horloge du calendrier interne.

EXEMPLE **PRINT DAY**

REMARQUES Dans le HX-20, le jour de la semaine est gardé par l'horloge du calendrier interne sous forme d'entiers variant de 1 à 7 en correspondance avec les sept jours de la semaine. Avec cette instruction, vous pouvez fixer le jour de la semaine, par exemple, DAY = 7 (samedi). Le jour de la semaine est affiché par l'un des entiers de 1 à 7. Lorsque vous indiquez le jour correct de la semaine au moyen de DAY, vous n'avez plus besoin de le refaire.

(Voir DATES et TIMES)

EOF

FORMAT EOF (<numéro de fichier>)

OBJET Renvoyer le code de fin de fichier

EXEMPLE **IF EOF (3) THEN CLOSE #1 ELSE GOTO 100**

REMARQUES Le fichier spécifié par le <numéro de fichier> doit avoir été ouvert pour le mode d'entrée. EOF vérifie si le fichier spécifié par le <numéro de fichier> a atteint son extrémité. EOF renvoie −1 (vrai) si la fin du fichier a été atteint et 0 (faux) dans le cas contraire.

Si le fichier spécifié est la porte RS-232C ("COM0:"), EOF renvoie −1 lorsque le tampon est vide, et 0 lorsque le tampon n'est pas vide. La fonction EOF renvoie toujours 0 (faux) pour le fichier affecté au clavier.

ERL/ERR

FORMAT ERL
ERR

OBJET Renvoyer le code d'erreur d'une erreur qui s'est produite et le numéro de la ligne sur laquelle l'erreur s'est produite.

EXEMPLE A = ERL
B = ERR

REMARQUES Lorsqu'une erreur se produit, le code d'erreur est stocké dans la variable ERR et le numéro de ligne sur laquelle l'erreur s'est produite est stocké dans la variable ERL. Si l'instruction qui a provoqué l'erreur est exécutée en mode direct, le numéro de ligne 65535 est stocké dans la variable ERL.
Normalement, les variables ERL et ERR sont utilisées dans la routine de branchement sur erreur spécifiée par une instruction ON ERROR GOTO pour contrôler le cheminement de l'exécution.

(Voir ON ERROR GOTO)

EXP

FORMAT EXP (< expression numérique >)

OBJET Renvoyer la valeur d'une fonction exponentielle de base

EXEMPLE A = EXP(1)

REMARQUES La valeur de l'< expression numérique > doit être le résultat d'une fonction exponentielle. Si la valeur de l'< expression numérique > est supérieure à 88.02969, une erreur OV ("débordement") se produit.

FIX

FORMAT FIX (<expression numérique>)

OBJET Renvoyer la partie entière tronquée d'une expression numérique.

EXEMPLE A = FIX (–B/3)

REMARQUES FIX renvoie la valeur de l'< expression numérique > tronquée à sa partie entière.

(Voir CINT et INT)

FRE

FORMAT FRE (<expression>)

OBJET Renvoyer la dimension d'une zone de mémoire inutilisée.

EXEMPLE PRINT FRE (0)
PRINT FRE ("A$")

REMARQUES Si l'< expression > est une expression numérique, FRE renvoie le nombre d'octets libres dans la zone de texte BASIC. Si l'< expression > est une expression chaîne, FRE renvoie le nombre d'octets de la zone chaîne BASIC. L'< expression > est simplement factice. Les arguments de FRE importent peu dans la mesure où ce sont des expressions numériques ou des expressions chaînes. Etant donné que la dimension de la zone de mémoire inutilisée affichée comprend un espace de travail pour l'exécution des programmes BASIC, utilisez la capacité de mémoire affichée uniquement à titre indicatif.

HEX$

FORMAT **HEX$ (<expression numérique>)**

OBJET Renvoyer une chaîne qui représente la valeur hexadécimale de l'argument décimal.

EXEMPLE **A$ = HEX$ (65535)**

REMARQUES HEX$ convertit la valeur décimale d'une <expression numérique> en une valeur hexadécimale et la renvoie sous forme d'une chaîne. La valeur de l'<expression numérique> doit se situer dans la gamme –32768 à 65535. Si la valeur de l'<expression numérique> comprend une fraction décimale, elle est arrondie à une valeur entière avant l'évaluation HEXS (< expression numérique >).

(Voir OCT$ et VAL)

EXEMPLE DE PROGRAMME

```
LIST

100 PRINT " DEC   OCT HEX"
110 FOR I=5 TO 16
120 PRINT USING"####";I;
130 PRINT USING"&  &";" ",OCT$(I),HEX$(I
)
140 NEXT I
RUN
 DEC    OCT HEX
   5     5   5
   6     6   6
   7     7   7
   8     10  8
   9     11  9
  10     12  A
  11     13  B
  12     14  C
  13     15  D
  14     16  E
  15     17  F
  16     20  10
>
```

INKEY$

FORMAT INKEY$

OBJET Renvoyer une chaîne à un caractère de la touche de caractères sur laquelle vous avez appuyé ou une chaîne nulle si vous n'avez appuyé sur aucune touche de caractère.

EXEMPLE AS = INKEY$

REMARQUES INKEY$ renvoie une chaîne si le tampon du clavier est vide. Si le tampon du clavier contient une quelconque entrée de touche de caractère, INKEYS lit le caractère dans le tampon et le renvoie comme chaine à un caractère. Toutes les touches qui ne font pas partie de la "Table des Codes de Caractères" comme la touche **SHIFT** etc... , sont ignorées.

(Voir Annexe F, "Table des Codes de Caractères")

INPUT$

FORMAT **INPUT$ (<nombre de caractères> [, [# [<numéro de fichier>)**

OBJET Renvoyer une chaîne de caractères lue dans un fichier spécifié.

EXEMPLE **A$ = INPUT$ (5, # 3)**

REMARQUES INPUT$ lit une chaîne de caractères du <nombre de caractères> spécifié par le <nombre de caractères> dans le fichier spécifié par le <numéro de fichier> Si le <numéro de fichier> est omis, des caractères peuvent être introduits depuis le clavier ; mais les caractères introduits au clavier ne sont pas contrôlés par écho, c'est à dire qu'ils ne sont pas affichés sur l'écran, contrairement à l'exécution d'une instruction INPUT.

INPUT$ est en état d'attente jusqu'à ce qu'une < chaîne de caractères > spécifiée par le nombre de caractères soit entièrement introduite. Mais, si une donnée quelconque se trouve déjà dans le tampon d'entrée, INPUT$ lit les caractères du tampon.

Avec INPUT$, tous les caractères sauf la touche **BREAK** sont lus tels quels. Par conséquent, INPUT$ permet l'introduction de caractères comme le Retour de Chariot (code de caractère 13), etc..., qui ne peuvent pas être introduits par les instructions INPUT et LINE INPUT.

INSTR

FORMAT **INSTR ([< expression numérique >,] < chaîne 1 >, < chaîne 2 >)**

OBJET Rechercher la première apparition d'une chaîne dans une autre chaîne et renvoyer la position de la chaîne recherchée.

EXEMPLE **B = INSTR (A$, "XYZ")**

REMARQUES INSTR cherche la première apparition de la < chaîne 2 > dans la < chaîne 1 > et renvoie la position à laquelle la coïncidence a été trouvée. Si la < chaîne 2 > ne peut pas être trouvée, INSTR renvoie 0.
l'< expression numérique > est la position de départ de la recherche. Si l'< expression numérique > n'est pas indiquée, la recherche commence au début de la < chaîne 1 >. Si une chaîne nulle est spécifiée comme < chaîne 2 >, INSTR renvoie la même valeur que la valeur spécifiée par l'< expression numérique >.

EXEMPLE DE PROGRAMME

```
LIST

100 A$="ZXCVBNM,./AS FG JKL :"
110 B$=INPUT$(1)
120 C=INSTR(A$,B$)
130 IF C=0 GOTO 110
140 IF C>10 THENC=C+17
150 SOUND C+5,2
160 GOTO110
>
```

INT

FORMAT INT (<expression numérique>)

OBJET Renvoyer la valeur entière la plus grande (tronquée).

EXEMPLE PRINT INT (–B/3)

REMARQUES INT renvoie la valeur entière la plus grande qui est égale ou inférieure à la valeur de l'<expression numérique>.

(Voir FIX et CINT)

EXEMPLE DE PROGRAMME

```
LIST

100 PRINT "     I      CINT    INT     FIX"
110 FOR I=2.4 TO -2.4 STEP -0.3
120 A%=CINT(I)
130 B%=INT(I)
140 C%=FIX(I)
150 PRINT USING "  ##.#";I;
160 PRINT USING "  #### ";A%;B%;C%
170 NEXT I
180 END
RUN
     I      CINT    INT     FIX
   2.4       2       2       2
   2.1       2       2       2
   1.8       2       1       1
   1.5       2       1       1
   1.2       1       1       1
   0.9       1       0       0
   0.6       1       0       0
   0.3       0       0       0
   0.0       0       0       0
  -0.3       0      -1       0
  -0.6      -1      -1       0
  -0.9      -1      -1       0
  -1.2      -1      -2      -1
  -1.5      -1      -2      -1
  -1.8      -2      -2      -1
  -2.1      -2      -3      -2
  -2.4      -2      -3      -2
>
```

LEFT$

FORMAT **LEFT$ (<chaîne>, <expression numérique>)**

OBJET Renvoyer une longueur arbitraire de chaîne à partir des caractères les plus à gauche d'une chaîne.

EXEMPLE **B$ = LEFT$ (A$, 4)**

REMARQUES La valeur de l'<expression numérique> doit être située entre 0 et 255. Si l'<expression numérique> est supérieure au nombre total des caractères de la <chaîne>, la chaîne totale sera renvoyée. Si l'<expression numérique> est 0, la chaîne nulle (longueur zéro) est renvoyée.

(Voir MID$ et RIGHT$)

LEN

FORMAT **LEN (<chaîne>)**

OBJET Renvoyer le nombre total des caractères contenus dans une chaîne.

EXEMPLE **A = LEN (A$)**

REMARQUES LEN renvoie le nombre total des caractères de la <chaîne>. Si la <chaîne> comprend des caractères qui ne sont pas imprimables comme des codes de contrôle et des blancs, ils ne sont pas réellement sortis mais ils sont comptés comme caractères.

(Voir Annexe F, Table des Codes de Caractères)

LOF

FORMAT LOF (<numéro de fichier>)
OBJET Renvoyer la dimension d'un fichier spécifié
EXEMPLE A = LOF (3)

REMARQUES Le fichier spécifié par le <numéro de fichier> doit avoir été ouvert en mode d'entrée. Si le <numéro de fichier> spécifié est sur une cartouche ROM, LOF renvoie la longueur restante du fichier en octets. Si le <numéro de fichier> spécifié est sur la porte RS-232C, LOF renvoie le nombre des données stockées dans le tampon en octets.

LOG

FORMAT LOG (<expression numérique>)
OBJET Renvoyer le logarithme népérien d'une expression numérique.
EXEMPLE PRINT LOG (2.7812818)

REMARQUES LOG renvoie le logarithme népérien de la valeur indiquée par l'<expression numérique>.

MID$

FORMAT MID$ (<chaîne>, <expression 1> [, <expression 2>])

OBJET Renvoyer une longueur arbitraire de chaîne à partir d'une chaîne

EXEMPLE B$ = MID$ (A$, 2, 3)

REMARQUES MID$ renvoie une chaîne de caractères de la longueur spécifiée par l'<expression 2> à partir de la <chaîne> commençant avec l'<expression 1> ième caractère. Les valeurs de <expression 1> et <expression 2> doivent se situer dans la gamme de 1 à 255 et de 0 à 255 respectivement.
Si <expression 2> est omis, ou s'il y a moins de caractères que <expression 2> à la droite de <expression 1> ième caractère, tous les caractères situés le plus à droite en commençant par l'<expression 1> ième caractère sont renvoyés. Lorsque le nombre des caractères de la <chaîne> est inférieur à l'<expression 1>, MID$ renvoie une chaîne nulle.

(Voir LEFT$ et RIGHT$)

EXEMPLE DE PROGRAMME

```
LIST

100 A$="ABCDEFGHIJKLMN"
110 PRINT "        MID$        LEFT$
RIGHT$ "
120 FOR I=1 TO 7
130 M$=MID$(A$,I,7)
140 L$=LEFT$(A$,I)
150 R$=RIGHT$(A$,I)
160 PRINT USING "     &        &";M$,L$,R$
170 NEXT I
RUN
     MID$        LEFT$        RIGHT$
    ABCDEFG      A            N
    BCDEFGH      AB           MN
    CDEFGHI      ABC          LMN
    DEFGHIJ      ABCD         KLMN
    EFGHIJK      ABCDE        JKLMN
    FGHIJKL      ABCDEF       IJKLMN
    GHIJKLM      ABCDEFG      HIJKLMN
>
```

OCT$

FORMAT **OCT$ (<expression numérique>)**
OBJET Renvoyer une chaîne qui représente la valeur octale de l'argument décimal.
EXEMPLE **PRINT OCT$ (123 +456)**

REMARQUES OCT$ convertit la valeur décimale d'une <expression numérique> en une valeur octale et la renvoie comme chaîne. La valeur de l'<expression numérique> doit être située dans la gamme de –32768 à 65535. Si la valeur de l'<expression numérique> comprend une fraction décimale, elle est d'abord arrondie à un entier avant que l'expression OCT$ (<expression numérique>) soit évaluée.

(Voir HEX$)

PEEK

FORMAT **PEEK (<adresse>)**
OBJET Renvoyer l'octet lu dans un emplacement spécifié de la mémoire.
EXEMPLE **A = PEEK (&H0C00)**

REMARQUES PEEK renvoie l'octet lu dans l'emplacement de la mémoire précisé par l'<adresse>, l'<adresse> doit être dans la gamme de 0 à 65535 (&H0 à &HFFFF) Si la valeur de l'<adresse> comprend une fraction décimale, elle est d'abord arrondie à un entier.
Etant donné que l'espace mémoire entre les adresses &H0 et &H4D est une zone spéciale allouée aux Entrées /Sorties, un débordement peut se produire simplement en lisant l'une quelconque de ces adresses.
En EPSON BASIC, lorsqu'une opération PEEK est exécutée à l'une quelconque des adresses ci-dessus, une erreur FC (''appel de fonction non valable'') apparaît. Pour lire ces adresses, &H80 doit être écrit en &H7E (pour positionner le bit le plus significatif).

(Voir POKE)

POINT

FORMAT POINT (<coordonnée horizontale>,<coordonnée verticale>)

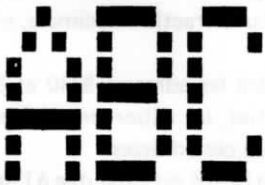
OBJET Renvoyer l'état d'un point situé à un emplacement précisé de l'écran graphique

EXEMPLE PRINT POINT (100, 10)

REMARQUES POINT vérifie si oui ou non un point a été tracé à l'emplacement spécifié par la <coordonnée horizontale> et la <coordonnée verticale> sur l'écran graphique. Dans le cas du LCD, POINT renvoie 1 si un point a été placé à l'emplacement indiqué et un 0 s'il n'y a pas de point. Dans le cas de l'écran extérieur, les codes de couleur 0 à 3 sont renvoyés. Veuillez noter que le LCD est différent de l'écran extérieur par la gamme dans laquelle vous pouvez spécifier des coordonnées graphiques.

EXEMPLE DE PROGRAMME

```
100 CLS:DEFINT A-Z
110 DIM P(17,7)
120 PRINT "ABC"
130 FOR Y=0 TO 7
140 FOR X=0 TO 17
150 P(X,Y)=POINT(X,Y)
160 NEXT X,Y
170 FOR Y=0 TO 7
180 PRINT
190 FOR X=0 TO 17
200 IF P(X,Y) THEN A$="
" ELSE A$=" "
210 PRINT A$;
220 NEXT X,Y
```



POS

FORMAT POS (<nombre>)

OBJET Renvoyer la position horizontale du curseur sur l'écran virtuel ou la position horizontale de la tête d'imprimante.

EXEMPLE X = POS (0)

REMARQUES La valeur du <nombre>, doit être comprise entre 0 et 16. Si 0 est spécifié, POS renvoie la position horizontale du curseur sur l'écran virtuel. Si un entier quelconque de 1 à 16 est spécifié, POS renvoie le nombre des caractères stockés dans la mémoire tampon (c'est à dire le nombre de caractères qui ont été envoyés après le dernier caractère LF jusqu'à l'exécution de cette fonction), sur le fichier ouvert en utilisant l'entier comme numéro de fichier. Ces nombres correspondent aussi aux positions horizontales de la tête d'imprimante.

(Voir CSRLIN et LOF)

RIGHT$

FORMAT RIGHT$ (<chaîne>, <expression numérique>)

OBJET Renvoyer une longueur de chaîne arbitraire à partir des caractères les plus à droite d'une chaîne.

EXEMPLE PRINT RIGHT$ ("ABC", 3)

REMARQUES La valeur de l'<expression numérique> doit être comprise entre 0 et 255. Si l'< expression numérique > dépasse le nombre total des caractères de la <chaîne>, la chaîne complète sera renvoyée. Si l'<expression numérique> est 0 la chaîne nulle (longueur zéro) est renvoyée.

(Voir LEFT$ et MID$)

RND

FORMAT RND (<expression numérique>)

OBJET Renvoyer un nombre aléatoire

EXEMPLE A = RND (1)

REMARQUES RND renvoie un nombre aléatoire compris entre 0 et 1. Le nombre aléatoire formé varie avec la valeur de l'<expression numérique> comme ci-dessous.

- Si <expression numérique> est négative, une nouvelle séquence de nombres aléatoires est créée.
- Si <expression numérique> est 0, le dernier nombre aléatoire créé est répété.
- Si <expression numérique> est positive, le nombre aléatoire suivant de la séquence est généré.

Si <expression numérique> est omise, le nombre aléatoire suivant de la séquence est aussi généré. La même séquence de nombres aléatoires est généré à chaque fois qu'une instruction RUN ou CLEAR est exécutée, si le générateur de nombres aléatoires n'est pas réensemencé par une instruction RANDOMIZE.

(Voir RANDOMIZE)

EXEMPLE DE PROGRAMME

```
LIST

100 DEFINT N,B
110 DIM B(120)
120 INPUT "NUMBER OF REP
ETITIONS ";A
130 CLS
140 FOR I=0 TO A
150 N=RND*120

160 B(N)=B(N)+1
170 PSET (N,31-B(N))
180 NEXT

>
RUN
NUMBER OF REPETITION
S ?  2000
```

SGN

FORMAT **SGN (<expression numérique>)**

OBJET Renvoyer le signe de la valeur d'une expression numérique.

EXEMPLE **B = SGN (A)**

REMARQUES Si la valeur de l'<expression numérique> est positive, SGN renvoie 1. Si la valeur de l'<expression numérique> est 0, SGN renvoie 0. Si la valeur de l'<expression numérique> est négative, SGN revient à −1.

SIN

FORMAT **SIN (<expression numérique>)**

OBJET Renvoyer le sinus d'une expression numérique.

EXEMPLE **PRINT SIN (3.1415926/2)**

REMARQUES SIN renvoie le sinus de l'<expression numérique> exprimée en radians.

(Voir COS et TAN)

SPACE$

FORMAT **SPACE$ (<expression numérique >)**

OBJET Renvoyer une chaîne d'espaces de longueur spécifiée.

EXEMPLE **A$ = "A" + SPACE$ (10) + "C"**

REMARQUES SPACE$ renvoie une chaîne d'espaces de la longueur spécifiée par l'<expression numérique >. La valeur de l'<expression numérique >doit se situer dans la gamme de 0 à 255.

(Voir SPC et TAB)

SPC

FORMAT **SPC (<nombre >)**

OBJET Sortir un nombre spécifié de blancs.

EXEMPLE **PRINT SPC (10) ; "A"**

REMARQUES SPC sort des blancs en nombre spécifié par <nombre >. Cette fonction ne peut être utilisée que dans les instructions de sortie comme PRINT. La valeur du <nombre >doit être comprise entre 0 et 255.

(Voir SPACE$ et TAB)

SQR

FORMAT **SQR (<expression numérique>)**

OBJET Renvoyer la racine carrée d'une expression numérique.

EXEMPLE **A = SQR (2)**

REMARQUES SQR renvoie la racine carrée de l'< expression numérque >. La valeur de l'<expression numérique >doit être supérieure à 0.

STR$

FORMAT **STR$ (<expression numérique>)**

OBJET Renvoyer une représentation chaîne de la valeur d'une<expression numérique>

EXEMPLE **A$ = STR$ (123)**

REMARQUES STR$ convertit la valeur spécifiée par l'< expression numérique> en une chaîne. Pour l'<expression numérique> vous pouvez utiliser n'importe quel type de constantes numériques.

(Voir STRING$ et VAL)

STRING$

FORMAT **STRING$ (<expression entière>, | <expression chaîne> / <expression numérique> |)**

OBJET Renvoyer une chaîne de caractères spécifiés.

EXEMPLE **PRINT STRING$ (10, 65)**

REMARQUES STRING$ renvoie une chaîne de caractères spécifiés par l'<expression chaîne> ou l'< expression numérique > de la longueur spécifiée par l'< expression entière >. Si l'<expression chaîne >est spécifiée comme étant une chaîne de caractères à renvoyer, tous les caractères ayant le premier caractère de la chaîne sont renvoyés.

Si l'<expression numérique >est spécifiée, tous les caractères ayant un code ASCII spécifié par l'< expression numérique > sont renvoyés. La valeur de l'< expression numérique > doit être comprise dans la gamme de 0 à 255.

(Voir STR$)

TAB

FORMAT **TAB (<expression numérique >)**

OBJET Se déplacer d'un certain nombre d'espaces sur la ligne où se trouve le curseur.

EXEMPLE **PRINT TAB (10) ; "ABC"**

REMARQUES TAB s'emploie uniquement dans des instructions de sortie comme PRINT et LPRINT. TAB forme des espaces à partir de la position du curseur jusqu'à la position spécifiée par l'< expression numérique > comptés à partir de l'extrémité gauche de l'écran virtuel.

La valeur de l'< expression numérique > peut se situer entre 0 et 255, 0 désignant la position la plus à gauche. Autrement dit, la valeur de l'< expression numérique > correspond à ce qui reste lorsqu'elle est divisée par le nombre de caractères à afficher horizontalement. Si la position spécifiée par l'< expression numérique > se trouve à gauche de la position du curseur, TAB va jusqu'à cette position sur la ligne suivante.

Veuillez noter la différence entre la fonction SPC et la fonction TAB.

(Voir SPC)

TAN

FORMAT **TAN (<expression numérique >)**

OBJET Renvoyer la tangente d'une expression numérique.

EXEMPLE **A = TAN (3.1416/4)**

REMARQUES TAN renvoie la tangente de la valeur de l'< expression numérique > exprimée en radians.

(Voir COS et SIN)

TAPCNT

FORMAT TAPCNT

OBJET Renvoyer la valeur du compteur d'entraînement de la microcassette.

EXEMPLE PRINT TAPCNT
A = TAPCNT

REMARQUES La fonction TAPCNT s'emploie pour lire la valeur du compteur de la microcassette. La valeur retournée est comprise entre –32768 et 32767. Cette valeur du compteur est toujours retournée comme une valeur positive lorsque le compteur a été remis à zéro par une commande WIND. En affectant une valeur à TAPCNT, vous pouvez fixer la valeur du compteur.

(Voir WIND)

EXEMPLE DE PROGRAMME

```
LIST

100 TAPCNT=0
110 OPEN "O",#1,"CAS0:TEST"
120 FOR I=1 TO 10
130 PRINT#1,I;I*I
140 NEXT
150 CLOSE
160 WIND 0
170 OPEN "I",#1,"CAS0:TEST"
180 IF EOF(1) THEN 220
190 INPUT#1,A,B
200 PRINT A,B
210 GOTO 180
220 CLOSE
230 END
RUN
 1              1
 2              4
 3              9
 4              16
 5              25
 6              36
 7              49
 8              64
 9              81
 10             100
>
```

TIME$

FORMAT TIME$ [= "HH:MM:SS"]

OBJET Lire l'heure gardée par l'horloge du calendrier interne.

EXEMPLE PRINT TIME$

REMARQUES TIME$ affiche ou fixe l'heure gardée par l'horloge interne du HX-20. L'heure est fixée ou affichée au format "HH:MM:SS" où HH varie de 00 à 23 et MM et SS de 00 à 59. Lorsque l'heure correcte a été fixée, vous n'avez plus besoin de l'introduire de nouveau, étant donné que l'horloge du HX-20 est toujours à l'heure et la date.

(Voir DATE$ et DAY)

USR

FORMAT USR [< chiffre >] (< argument >]

OBJET Appeler un sous-programme en langage machine défini par l'instruction DEFUSR

EXEMPLE A = USR 1 (B)

REMARQUES USR appelle votre sous-programme en langage machine (fonction définie par l'utilisateur) avec l'< argument >. Avant d'appeler la fonction définie par l'utilisateur, il faut que cette fonction ait été inscrite dans la mémoire et que son adresse de début d'exécution ait été définie par l'instruction DEFUSR. Dix fonctions définies par l'utilisateur au maximum peuvent être désignées par le < chiffre > dont la valeur est comprise entre 0 et 9 et qui correspond au chiffre indiqué dans l'instruction DEFUSR pour ce sous-programme. Si le < chiffre > est omis, on suppose qu'il s'agit de USR0. Avec < argument >, vous pouvez tranférer une valeur de l'EPSON BASIC dans votre sous-programme en langage machine.

(Voir DEFUSR et Chapitre 5 "Sous-Programmes en Langage Machine")

VAL

FORMAT VAL (<**expression chaîne**>)

OBJET Retourner la valeur numérique d'une expression chaîne.

EXEMPLE A = VAL ("–123")

REMARQUES Si le premier caractère de l'<expression chaîne> n'est ni +, ni –, ni &, ni un chiffre, VAL (<expression chaîne>) = 0.
Si un caractère qui n'est pas un chiffre (0 à F pour les nombres hexadécimaux et 0 à 7 pour les nombres en octal) apparaît, les caractères suivants sont ignorés. Les blancs de l'<expression chaîne> sont également ignorés.

(Voir CHRS et STRS)

VARPTR

FORMAT VARPTR (<**nom variable**>)

OBJET Renvoyer l'adresse d'une variable ou d'un réseau.

EXEMPLE PRINT HEXS (VARPTR (A))

REMARQUES Tout type de <nom de variable> (numérique, chaîne ou réseau) peut être spécifié. Une valeur doit être affectée à la variable spécifiée par le <nom de variable> avant l'exécution de VARPTR.
L'adresse renvoyée est un entier de la gamme –32768 à 32767. Si une adresse négative est renvoyée, on obtient la valeur réelle en l'ajoutant à 65536.
Chaque fois qu'une valeur est affectée à une nouvelle variable qui n'est pas un tableau les adresses des tableaux varient. Donc, toutes les variables simples doivent être affectées avant d'appeler VARPTR pour un tableau.

(Voir Chapitre 5 "Programmes en Langage Machine")

Chapitre 5

Informations complémentaires

5.1 Fichiers RAM

Les fichiers RAM (aléatoires) constituent une des nombreuses caractéristiques de l'EPSON BASIC.

Grâce aux fichiers RAM:

- On peut accéder aux données, au hasard.
- On peut modifier les données librement.
- Une grande vitesse d'accès est garantie comparativement à la vitesse de transfert E/S en entrée et en sortie des périphériques.

Grâce aux avantages ci-dessus, les fichiers RAM peuvent être considérés comme identiques à des variables tableaux. Les fichiers RAM présentent également les propriétés suivantes:

- Les données sont gardées même si le commutateur général est tourné sur OFF.
- Des données peuvent être partagées par plusieurs programmes.

Etant donné les caractéristiques ci-dessus, les fichiers RAM représentent l'idéal pour des données qui doivent être traitées plus fréquemment que celles des fichiers séquentiels et sont donc utiles pour:

- Le stockage des tables de variables, des tables de conversion, etc., qui sont à utiliser en permanence. (Calculs scientifiques).
- Le stockage des données nécessaires au traitement des transactions quotidiennes (gestion d'une entreprise).

En EPSON BASIC, cinq programmes distincts peuvent être stockés simultanément. Mais les fichiers RAM doivent être partagés par ces cinq programmes. En utilisant des instructions PUT%, il faut veiller à ce que les fichiers de données ne soient pas détruits accidentellement par d'autres programmes.

En fait, les fichiers RAM occupent, dans leur ensemble, une seule zone, mais la zone du fichier RAM peut être utilisée par des fichiers apparemment distincts utilisant une instruction DEFFIL. L'instruction DEFFIL spécifie l'emplacement du premier enregistrement dans la zone du fichier RAM, ainsi que les emplacements des enregistrements à lire ou à écrire au moyen d'une instruction GET% ou PUT% après que l'exécution de DEFFIL ait été décalée relativement.

Le type de variable des données à lire au moyen de l'instruction GET% doit correspondre aux données écrites par une instruction PUT%. Le nombre d'octets occupés dans l'espace mémoire par chaque type de variable est différent:

Variable entière	2 octets
Variable simple précision	4 octets
Variable double précision	8 octets
Variable chaîne	non définie

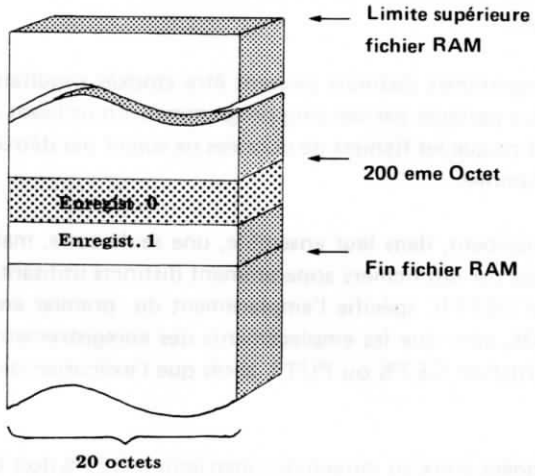
Pour cette raison, le résultat sera totalement différent suivant que les données écrites sous forme d'une valeur entière seront lues comme une valeur double précision ou simple précision.

La longueur de chaque enregistrement est déterminée par l'instruction DEFFIL et, à l'intérieur de cet enregistrement, les variables numériques occupent la longueur spécifiée pour chaque type de variable. La longueur restant de l'enregistrement est allouée aux variables chaînes. Ceci parce que la longueur d'une variable chaîne n'est pas définie et que dans les instructions GET% et PUT%, les variables chaînes peuvent être uniquement utilisées après des variables numériques.

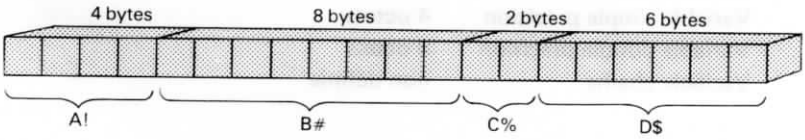
Exemple:

```
100  DEFFIL 20, 200
      :
200  PUT% 0, A!, B#, C%, D$
      :
300  GET% 0, E!, F#, G%, H$
      :
400  END
```

Dans ce programme, l'emplacement d'un fichier est présenté ci-dessous:



La longueur d'enregistrement, par exemple de l'enregistrement 0, est allouée de la façon suivante:



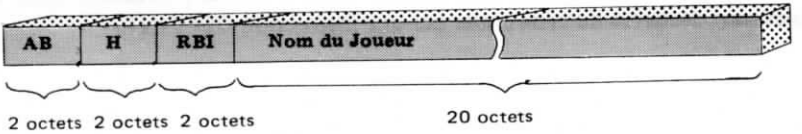
L'exécution du programme aboutit aux résultats suivants:

```
E!=A!, F#=B#
G%=C%, H$=D$
```

A titre d'exemple de l'emploi d'un fichier RAM, préparons un fichier d'enregistrement des statistiques individuelles de chacun des joueurs d'une équipe de baseball en utilisant les numéros d'uniforme des joueurs comme numéros de fichiers pour désigner les fichiers.

La première chose que vous devez considérer est la longueur de l'enregistrement de chaque joueur. Dans cet exemple, nous allons traiter, pour chaque joueur, les At-Bats, Hits et RBI (Runs Batted In). Vous pouvez traiter les trois valeurs comme des entiers. Vous aurez besoin d'enregistrer les noms des joueurs en caractères alphabétiques. Dans ce but, nous allons avoir une variable chaîne de 20 octets au maximum.

AB (At-bats)	2 octets
H (Hits)	2 octets
RBI (Runs Batted In)	2 octets
Nom du joueur	20 octets
Total	26 octets



Ensuite, vous devez déterminer la longueur totale du fichier nécessaire pour les statistiques de l'équipe. Vous connaissez déjà la longueur de l'enregistrement de chaque joueur. Il vous suffit maintenant de multiplier cette longueur de l'enregistrement individuel par le nombre de joueurs de l'équipe. En supposant que l'équipe compte 25 joueurs, la longueur totale du fichier sera:

26 x 25 = 650 octets

Vous pouvez maintenant écrire des programmes pour enrgistrer les statistiques individuelles des joueurs selon les fonctions nécessaires.

5.1.1 Création d'un fichier RAM

Bien que cela ne soit pas tout à fait habituel, nous allons limiter les numéros des uniformes des joueurs à la gamme 0-24. Dans ce programme, il vous suffira d'écrire les données AB, H et RBI disponibles pour chaque joueur jusqu'à ce moment.

```
100 CLEAR 200,650
110 DEFINT U,A,H,R
120 DEFFIL 26,0
130 INPUT "UNIFORM NO. "
;U
140 IF U>24 THEN 210
150 INPUT "NAME     ";NA$
160 INPUT "AT BATS ";A
170 INPUT "HITS     ";H
180 INPUT "RBI      ";R
190 PUT% U,A,H,R,NA$
200 GOTO 130
210 END
```

Même si vous avez introduit un nom de joueur qui dépasse 20 caractères, aucune erreur ne se produit. Mais, comme vous ne disposez que de 20 octets pour la zone du nom du joueur dans un enregistrement du fichier RAM, seuls les 20 premiers caractères du nom sont stockés. Puisque chacune des trois variables (AB, H et RBI) nécessite deux octets, les variables sont déclarées du type entier par une instruction DEFFINT.

5.1.2 Recherche et mise à jour des données d'un fichier RAM.

Ce programme permet de lire les données enregistrées dans le fichier RAM par le programme de l'exemple précédent. Puisque chaque nom de joueur est déjà introduit dans le fichier, tout ce qu'il vous faut pour retrouver les données relatives à un joueur, c'est d'entrer le numéro de son uniforme. Si vous tapez "Y" après les données affichées, l'enregistrement individuel peut être mis à jour. Les statistiques nouvellement introduites sont alors rajoutées aux statistiques déjà enregistrées.

```
220 CLEAR 200,650
230 DEFINT U,A,H,R
240 DEFFIL 26,0
250 PRINT
260 INPUT "UNIFORM NO.  "
;U
270 IF U>25 THEN END
280 GET% U,A,H,R,NA$
290 AV=H/CSNG(A)
300 PRINT NA$
310 PRINT "AT BATS ";A
320 PRINT TAB(10);"HITS
";H
330 PRINT "RBI ";R
340 PRINT TAB(10);"AVERA
GE ";AV
350 PRINT "UPDATE (Y/N)"
;
360 IF INPUT$(1)<>"Y" GO
TO 250
370 PRINT
380 INPUT "AT BATS ";AX
390 INPUT "HITS    ";HX
400 INPUT "RBI     ";RX
410 PUT% U,A+AX,H+HX,R+R
X
420 GOTO 280
```

En plus des deux exemples de programmes donnés, vous pouvez aussi écrire un programme qui liste les enregistrements des performances des joueurs dans l'ordre de la moyenne des lancements ou un programme de sélection et de listage des enregistrements individuels, disons des 5 meilleurs joueurs. Grâce à ces programmes et aux fichiers de données préparés par le HX-20, vous pouvez gérer pratiquement votre équipe de baseball.

5.2 Fichiers séquentiels

Si vous considérez l'enregistrement de la musique sur une cassette audio, vous comprendrez que les morceaux de musique sont enregistrés à la suite et que, pour écouter un morceau particulier, vous devez avancer la bande jusqu'à l'endroit où ce morceau est enregistré. Un fichier séquentiel fonctionne selon le même principe, mais au lieu de musique, une séquence de données est enregistrée sur la bande.

La vitesse et la souplesse des fichiers RAM sont très intéressantes, mais la capacité de leur mémoire limite la quantité des données que l'on peut traiter. C'est pourquoi beaucoup de données ne peuvent pas être traitées sur chaque fichier RAM.
De même, quand vous utilisez un fichier RAM, vous devez écrire un programme en tenant compte de la structure interne du fichier.

Au contraire, les fichiers séquentiels présentent une grande facilité d'utilisation en plus:

- d'une capacité illimitée des traitements de données,
- d'une structure simple du fichier.

Cependant, malgré ces avantages, les fichiers séquentiels ont aussi des inconvénients:

- Vous ne pouvez accéder aux données que séquentiellement.
- Les données ne peuvent donc pas être retrouvées rapidement.
- Il n'est pas possible d'apporter des changements partiels aux fichiers de données.

Lorsque vous préparez un programme en utilisant un fichier RAM ou séquentiel, les avantages et les inconvénients relatifs des deux types de fichiers doivent être pesés avant que vous puissiez faire votre choix. Pour utiliser un fichier séquentiel, vous devez observer la procédure suivante:

(1) Ouvrir le fichier.
(2) Introduire les données du fichier en mémoire en utilisant une instruction INPUT# ou sortir des données de la mémoire vers le fichier en utilisant une instruction PRINT#
(3) Fermer le fichier.

Un numéro de fichier a été affecté à chaque fichier séquentiel au moment où il a été créé et toutes les opérations E/S ultérieures en entrée ou en sortie de ces fichiers (en utilisant INPUT# , PRINT# , etc...) sont effectuées en désignant un fichier particulier par son numéro de fichier. Par conséquent, il ne peut pas exister, en même temps, deux fichiers sous le même numéro de fichier. Lorsqu'un fichier a été OUVERT sous un certain numéro de fichier, aucun autre fichier ne peut être ouvert en utilisant ce numéro avant que le fichier ouvert le premier n'ait été fermé.

Les données qui doivent être lues avec une instruction INPUT# doivent correspondre aux données écrites avec une instruction PRINT# à la fois en type de variables et en nombre de variables. Dans un fichier séquentiel, les données sont lues successivement depuis le début du fichier.

Par conséquent, s'il y a une différence quelconque dans le nombre des variables, l'instruction INPUT# ultérieure lit toutes les données dans un ordre incorrect. S'il y a une différence dans le type de variable, une erreur TM ("Désaccord de type") apparaît.

Après le transfert E/S utilisant le fichier séquentiel, vous devez toujours exécuter une instruction CLOSE (ou END). En EPSON BASIC, le transfert des données entre le HX-20 et un équipement périphérique est effectué en unités de 256 octets. Par exemple, en écrivant des données dans le fichier, une opération d'écriture ne peut se faire avant que 256 octets de données aient été accumulés dans la mémoire tampon. Si une instruction CLOSE (ou END) n'est pas exécutée à la fin du programme, les données restent dans la mémoire tampon et ne sont pas sorties. Dans ce cas, le fichier est laissé incomplet sans qu'un séparateur ne soit écrit à la fin du fichier.

A titre d'exemple d'emploi d'un fichier séquentiel, écrivons un programme pour un annuaire d'adresses. Dans l'exemple précédent de fichier RAM, rappelez-vous que vous avez dû considérer attentivement la structure de l'enregistrement en particulier les éléments définissant l'enregistrement et la longueur de l'enregistrement. Mais, avec les fichiers séquentiels, comme il n'y a pas de limitations particulières à la quantité des données, il vous suffit de décider des éléments de définition de l'enregistrement et de l'ordre dans lequel vous souhaitez les classer.

5.2.1 Création d'un fichier séquentiel

Ce programme est destiné à enregistrer une liste de noms, adresses, numéros de tétéphone et dates de naissance sur une cassette audio.
Bien que vous puissiez écrire des données comme le nom, l'adresse, etc., ensemble, sous forme d'une chaîne, il vaut mieux enregistrer en délimitant ces articles pour les utiliser ultérieurement.

```
100 OPEN"O",#1,"CAS1:ADR
S"
110 PRINT
120 INPUT "NAME ";NAME$
130 IF NAME$="" THEN 250
140 INPUT "ADDRESS";ADR$
150 INPUT "TELEPHONE ";T
EL$
160 PRINT"DATE OF BIRTH"
170 INPUT "  YEAR";Y
180 INPUT " MONTH";M
190 INPUT "   DAY";D
200 PRINT#1,NAME$
210 PRINT#1,ADRS$
220 PRINT#1,TEL$
230 PRINT#1,Y;M;D
240 GOTO 110
250 CLOSE
260 END
```

Si vous observez le fonctionnement de la cassette audio, vous remarquerez probablement que la bande n'avance pas en permanence. Ceci est dû à l'action de la mémoire tampon définie dans le paragraphe précédent. C'est à dire qu'une opération d'écriture est effectuée uniquement lorsqu'une quantité spécifiée de données a été accumulée dans la mémoire tampon. Lorsque vous avez introduit toutes les données correspondant au nombre désiré de personnes, il vous suffit d'appuyer sur **RETURN** à la ligne 120 pour affecter une chaîne nulle à AS. Ceci va brancher le programme à la ligne 250 pour FERMER le fichier Ce pas ne peut pas être sauté. Si vous appuyez sur la touche **BREAK** pour terminer le programme quand toutes les données ont été introduites, les données restent dans le tampon et le fichier n'est pas fermé.

5.2.2 Recherche de données dans un fichier séquentiel.

Pour démontrer les avantages de l'emploi de cet ordinateur, plutôt que de sortir simplement les données enregistrées comme elles ont été introduites, nous allons écrire un programme plus utile pour sortir des données sélectionnées du fichier; par exemple, les données des personnes qui sont nées une certaine année :

```
100 OPEN "I",#1,"CAS1:AD
RS"
110 PRINT "PEOPLE BORN W
HAT YEAR?"
120 INPUT N
130 LPRINT " *** BORN IN
";N;"***"
140 IF EOF(1) THEN GOTO
230
150 INPUT#1,NAME$,ADR$,T
EL$,Y,M,D
160 IF N<>Y GOTO 140
170 LPRINT
180 LPRINT NAME$
190 LPRINT ADR$
200 LLISTNT TEL$
210 LPRINT "BORN ";M;"/"
;D;"/";Y
220 GOTO 140
230 LPRINT
240 LPRINT " *** END OF
RETRIEVAL *** "
250 CLOSE
260 END
```

Dans le précédent programme de création de fichier, vous avez indiqué la fin du fichier en appuyant sur la touche **RETURN** . Mais, lorsque vous lisez un fichier, cette détermination est faite par la fonction EOF. Si vous omettez la ligne 140 dans le programme ci-dessus, une erreur IE ("Introduction après la fin") apparaît.

5.2.3 Correction des données dans les fichiers séquentiels

Dans les fichiers séquentiels, vous ne pouvez pas changer une partie seulement des données enregistrées dans un fichier. Pour corriger le fichier, vous devez en préparer un autre et écrire les données correctes dans le nouveau fichier tout en lisant l'ancien. Comme le HX-20 vous permet d'utiliser à la fois un dérouleur à microcassette ("CAS0:") et un dérouleur à cassette audio ("CAS1:"), vous pouvez effectuer la mise à jour du fichier en utilisant ces deux unités.

```
100 CNT=TAPCNT
110 OPEN "I",#1,"CASI:AD
RS"
120 OPEN "O",#2,"CASO:WO
RK"
130 IF EOF(1) GOTO 320
140 INPUT #1,NAME$,ADR$,
TEL$,Y,M,D
150 PRINT NAME$
160 PRINT ADR$
170 PRINT TEL$
180 PRINT "BORN ";M;"/";
D;"/";Y
190 PRINT "DO YOU WISH T
O MAKE CORRECTIONS (Y/N)
 ?"
200 IF INPUT$(1)<>"Y" GO
TO 290
210 PRINT
220 INPUT "NAME      ";NA
ME$
230 INPUT "ADDRESS   ";AD
R$
240 INPUT "TELEPHØNE";TE
L$
250 PRINT "DATE OF BIRTH
 "
260 INPUT "   YEAR";Y
270 INPUT " MONTH";M
280 INPUT "    DAY";D
290 PRINT#2,NAME$,ADR$,T
EL$
300 PRINT#2,Y;M;D
310 GOTO 130
320 CLOSE
330 WIND CNT
340 PRINT "PLEASE REWIND
 THE CASSETTE"
350 PRINT
360 PRINT "IF OK, PLEASE
 INPUT Y"
370 IF INPUT$(1)<>"Y" GO
TO 360
380 OPEN "O",#1,"CAS1:AD
RS"
390 OPEN "I",#2,"CASØ:WO
RK"
400 IF EOF(2) GOTO 440
410 INPUT#1,NAME$,ADR$,T
EL$,Y,M,D
420 PRINT#2,NAME$,ADR$,T
EL$,Y,M,D
430 GOTO 400
440 CLOSE
450 END
```

Le transfert réel des données de l'ancien fichier au nouveau se termine à la ligne 320. Dans cet état, le nouveau fichier se trouve sur le dérouleur à microcassette. Pour l'utilisation ultérieure du fichier de données, vous devez retransférer les données du nouveau fichier dans le fichier de la cassette audio. La bande du dérouleur à microcassette peut être réenroulée au moyen d'une instruction WIND dans un programme. Mais le dérouleur à cassette audio doit être commandé manuellement. Dans cet exemple de programme, deux fichiers sont utilisés de sorte que vous pouvez stocker une quantité plus importante de données. Si la quantité des données à traiter est faible et que toutes les données peuvent être stockées à la fois dans la mémoire, les corrections du fichier des données peuvent être effectuées exactement comme pour le fichier RAM.

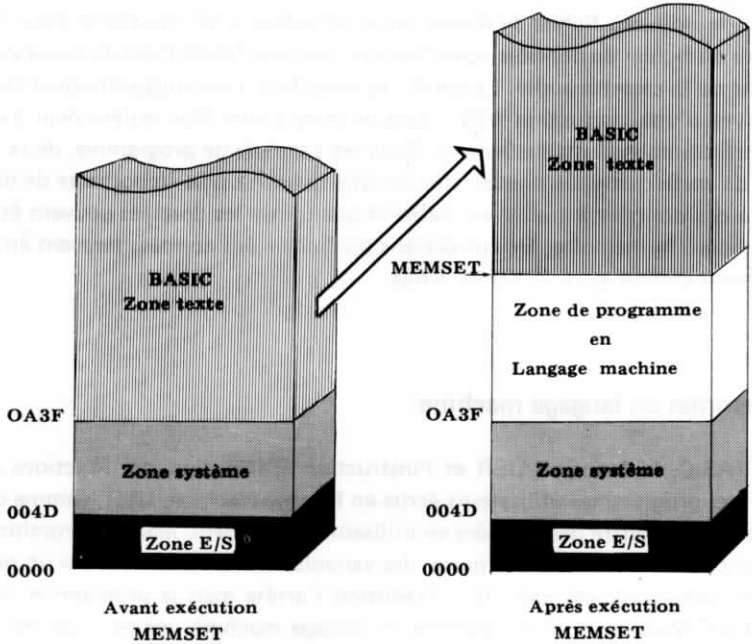
5.3 Programmes en langage machine

En EPSON BASIC, la fonction USR et l'instruction EXEC sont des fonctions prévues pour appeler des programmes utilisateurs écrits en langage machine. USR, comme d'autres fonctions, peut transmettre des données en utilisant un argument. Mais l'instruction EXEC n'a pas de fonction qui permette de passer des variables. En BASIC, même si un programme contenant une erreur est exécuté, l'exécution s'arrête mais le programme lui-même n'est pas détruit. Mais, avec un programme en langage machine, même si un seul bit est incorrect, tous les programmes (y compris les programmes BASIC) peuvent être détruits.

5.3.1. Répartition de la mémoire

Lorsque vous utilisez des programmes utilisateurs écrits en langage machine, l'espace de la mémoire doit être organisé pour que les programmes des données BASIC soient protégés et que les programmes en langage machine ne soient pas détruits par des programmes BASIC. En EPSON BASIC, les programmes en langage machine sont placés avant la zone des programmes BASIC et l'espace mémoire réservé au programme en langage machine est protégé par une commande MEMSET. Par conséquent, vous devez exécuter cette commnde MEMSET avant de charger un quelconque programme en langage machine.

L'exécution de la commande MEMSET protège la zone des programmes en langage machine et en même temps déplace les programmes BASIC. Par conséquent, les programmes BASIC déjà stockés dans la mémoire sont protégés de la destruction.



5.3.2 Ecriture et chargement des programmes

Un programme en langage machine est chargé dans la mémoire en utilisant la fonction MONITOR du HX-20. Des programmes courts peuvent être écrits dans la mémoire en utilisant une instruction POKE. Un programme en langage machine que vous avez rédigé peut aussi être stocké sous forme d'un fichier programme en langage machine de la même manière que des programmes BASIC en utilisant une commande SAVEM ou LOADM. Le HX-20 dispose d'un emplacement libre pour une prise de ROM supplémentaire qui vous permet de stocker vos programmes complètement déterminés dans la ROM supplémentaire, de sorte que vous pouvez toujours les utiliser comme logiciel utilitaire du HX-20. Vous pouvez aussi stocker des programmes qui sont moins souvent utilisés dans la cassette ROM comme fichier de programme.
De cette façon, le temps de chargement des programmes est nettement réduit et les possibilités d'exploitation du HX-20 peuvent être perfectionnées.

5.3.3 Fonction USR

Le format de la fonction USR pour appeler un programme en langage machine est le suivant:

USR[(<chiffre>)](<argument>)

Le < chiffre > peut être un entier de 0 à 9 et doit correspondre au chiffre inclus dans l'instruction DEFUSR. L'< argument > peut être une expression numérique ou chaîne, quelconque.
La fonction USR a un seul argument. L'accumulateur A contient une valeur qui spécifie le type d'argument.

Les valeurs utilisées et leur signification sont les suivantes:

Valeurs dans l'accumulateur	Type d'argument
2	Entier
3	Chaîne
4	Nombre réel simple précision
8	Nombre réel double précision

Si le type d'argument est numérique, la valeur dans le registre index indique l'adresse de "l'accumulateur à point flottant" où l'argument est stocké.
Cet accumulateur à point flottant ne désigne pas l'endroit de la mémoire où une variable elle-même est stockée, mais renvoie à une zone spéciale utilisée lorsque le BASIC effectue une opération arithmétique. Une opération réelle est stockée dans "l'accumulateur à point flottant" sous une forme différente selon le type d'argument.

● Quand l'argument est un entier :

	← Adresse indiquée par le registre index
8 bits supérieurs	
8 bits inférieurs	

● Quand l'argument est un nombre réel simple précision :

Exposant	← Adresse indiquée par le registre index
Bit de signe et 7 bits supérieurs de la mantisse	
8 bits milieu mantisse	
8 bits inférieurs mantisse	

● Quand l'argument est un nombre réel double précision :

Exposant	← Adresse indiquée par le registre index
Bit de signe et 7 bits supérieurs de la mantisse	
8 bits inférieurs mantisse	

● Quand l'argument est une chaîne :

La valeur du registre index indique l'adresse de données à 3 octets appelées "descripteur de chaîne".

longueur de chaîne	← Adresse indiquée par le registre index
Bits supérieurs de l'adresse stockée	
Bits inférieurs de l'adresse stockée	

Le descripteur de chaîne est un pointeur qui indique l'adresse où des données de type de chaîne sont réellement stockées. En se référant à l'adresse indiquée par ce descripteur de chaîne, vous pouvez trouver les données qui ont été introduites comme argument.

Lorsque la fonction USR renvoie une valeur au BASIC, les données doivent être stockées sous le même format à l'emplacement où l'argument a été stocké (c'est à dire à l'emplacement indiqué par l'accumulateur à point flottant ou le descripteur de chaîne). Donc, le type de valeur renvoyé par la fonction USR doit être le même que la valeur spécifiée comme argument.

Pour revenir à un programme BASIC à partir d'une fonction USR, on utilise ce programme RTS (&H39) en langage machine. Pour cela, la valeur du pointeur de pile au moment où le contrôle de programme est revenu au BASIC doit être la même qu'au moment où la fonction USR a été appelée.

5.3.4 Instruction EXEC

Les sous-programmes en langage machine peuvent aussi être exécutés comme une instruction EXEC.

EXEC [<adresse>!]

Lorsqu'un sous-programme en langage machine est chargé dans la mémoire à l'aide d'une commande LOADM, ou lorsque l'adresse de début d'exécution a été spécifiée par EXEC, l'<adresse> peut être omise.
EXEC exécute uniquement le programme à partir de l'adresse spécifiée et n'a aucune fonction qui transmette l'argument. Quand les variables doivent être passées d'un programme BASIC à un programme en langage machine appelé par une instruction EXEC, ceci est effectué par une lecture et une écriture directe des variables stockées en mémoire. Vous pouvez procéder de plusieurs façons :

(1) INSCRIPTION et LECTURE (POKE et PEEK) des adresses spécifiées de la zone du langage machine utilisant le programme BASIC.
(2) Lecture et écriture directe de données dans la zone des variables BASIC en utilisant le programme en langage machine.

Si vous utilisez la méthode (2), vous devez d'abord vérifier les adresses des variables à écrire ou à lire en utilisant la fonction VARPTR.

5.3.5 Fonction VARPTR

Avec la fonction VARPTR, vous pouvez vérifier à quelle adresse de la mémoire est stockée votre variable spécifiée.VARPTR permet de faire passer une variable à un programme en langage machine qui sera appelé par une instruction EXEC, ou de faire passer deux ou plusieurs variables en utilisant la fonction USR.
Si vous utilisez la fonction VARPTR, il faut qu'une valeur ait été attribuée à la variable spécifiée comme argument avant l'exécution de VARPTR.
La valeur renvoyée par la fonction VARPTR est l'adresse la plus élevée des données variables spécifiées. Si la variable est une chaîne, les données ne correspondent pas à la valeur de la variable mais à un <descripteur de chaîne> qui est un pointeur indiquant l'adresse à laquelle les données de type de chaîne sont stockées.

Quand une variable est stockée en mémoire, le type de variable, la longueur du nom de variable et le nom de variable (au maximum 16 caractères), sont stockés immédiatement avant la valeur de la variable.

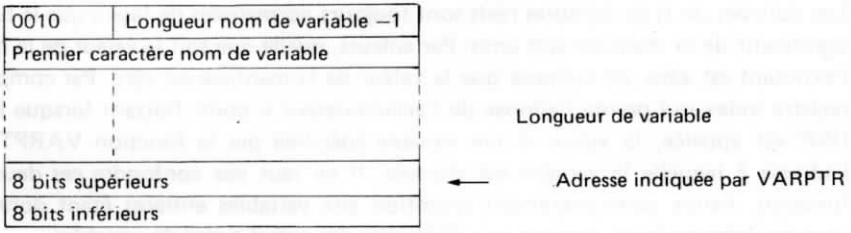
(1) Type de variable
En utilisant les 4 premiers bits du premier octet de données, les variables sont classées selon la longueur occupée par les données, de la façon suivante:

2 : entier
3 : chaine
4 : nombre réel simple précision
8 . nombre réel double précision

(2) Longueur du nom de variable
Dans les 4 derniers bits du premier octet de données, la longueur réelle du nom de variable moins un, est stockée. Donc lorsque le nom de variable comprend un seul caractère, la valeur stockée est zéro.

(3) Nom de la variable
En commençant au second octet de données, le nom de variable de longueur indiquée (2) ci-dessus est stocké en format ASCII. Bien que le nom de la variable soit stocké dans 16 caractères au maximum, on utilise uniquement le nombre d'octets nécessaires.

● Lorsque le type de variable est un entier :



● Lorsque le type de variable est une chaîne

0011	Longueur nom de variable -1
Premier caractère nom de variable	
Longueur de chaîne	
Bits supérieurs adresse stockée	
Bits inférieurs adresse stockée	

Longueur de variable

← Adresse indiquée par VARPTR

● Lorsque le type de variable est un nombre réel simple précision

0100	Longueur nom de variable –1
Premier caractère nom de variable	
Exposant	
Bit de signe et 7 bits supérieurs mantisse	
8 bits milieu mantisse	
8 bits inférieurs mantisse	

Longueur de variable

← Adresse indiquée par VARPTR

● Lorsque le type de variable est un nombre réel double précision

1000	Longueur nom de variable –1
Premier caractère nom de variable	
Exposant	
Bit de signe et 7 bits supérieurs mantisse	
8 bits inférieurs mantisse	

Longueur de variable

← Adresse indiquée par VARPTR

Les données de type nombres réels sont toujours normalisées de façon que le bit le moins significatif de la mantisse soit omis. Par ailleurs, quelle que soit la valeur de la mantisse, si l'exposant est zéro, on suppose que la valeur de la mantisse est zéro. Par comparaison au registre index qui pointe l'adresse de l'accumulateur à point flottant lorsque la fonction USR est appelée, la valeur d'une variable spécifiée par la fonction VARPTR indique l'adresse à laquelle la variable est stockée. Il ne faut pas confondre ces deux types de fonction. Faites particulièrement attention aux variables entières étant donné qu'elles sont stockées de façon quelque peu différente des autres types de variables.

5.4 Comment utiliser la porte RS-232C

Le HX-20 comprend une interface RS-232C qui permet de communiquer avec des dispositifs extérieurs. La porte RS-232C du HX-20 est normalement utilisée pour brancher une imprimante extérieure au HX-20. Le RS-232C est normalisé EIA pour l'interface MODEM et l'équipement terminal de traitement de données associé.

Avec cette interface RS-232C, le HX-20 peut communiquer directement avec d'autres ordinateurs. En connectant un coupleur accoustique à l'interface RS-232C, le HX-20 peut aussi transférer des données en direction ou en provenance d'emplacements éloignés par des lignes d'abonnés.

Le standard RS-232C est largement utilisé par de nombreux types d'équipements de communication. Mais, étant donné que cette norme définit simplement les caractéristiques électriques, l'emploi des lignes de signaux, des procédures de transmission etc., diffère d'un type d'équipement à un autre. Par conséquent, lorsque vous branchez un dispositif extérieur au HX-20, vous devez vérifier soigneusement le dispositif extérieur pour vous assurer que les conditions d'interface sont conformes à celles du HX-20 afin de pouvoir établir des communications dans les mêmes conditions d'interface.

Avec le HX-20, la définition des conditions d'interface peut être effectuée en utilisant le

BASIC. (Voir OPEN"COM0:"). Lorsque des conditions d'interface uniformes ont été établies entre le HX-20 et le dispositif extérieur, toutes les autres procédures sont les mêmes que si vous utilisez des fichiers normaux. Si vous spécifiez <numéro de fichier> dans une instruction de programme, vous pouvez effectuer les opérations E/S en utilisant les instructions PRINT# et INPUT# . Lorsque le HX-20 fonctionne en mode direct, vous pouvez directement transférer les programmes entre le HX-20 et les dispositifs extérieurs en utilisant les commandes LIST"COM0:" et LOAD"COM0:".

NOTE:
Si vous exécutez une commande de sortie en spécifiant "COM0:" comme nom de dispositif alors qu'aucun dispositif extérieur n'est connecté à la porte RS-232C, ou alors que la connection entre le HX-20 et le dispositif extérieur est défectueuse, le fonctionnement du HX-20 risque de s'arrêter. Si cela se produit, appuyez sur la touche **BREAK** *pour ramener le HX-20 au niveau des commandes.*
Si également, une longue ligne de caractères est tranférée sans spécifier la largeur d'impression pour "COM0:" , la valeur implicite est utilisée et le changement de ligne automatique se fait tous les 80 caractères. Lorsque vous envoyez une longue chaîne dépassant 80 caractères, vous devez d'abord exécuter WIDTH "COM0:" , 255 pour fixer la largeur d'impression à l'infini.

5.4.1 Interface avec les dispositifs en option

Tous les dispositifs disponibles en option avec le HX-20 ont des lignes de signaux normalisées Donc, dans la mesure ou vous utilisez les câbles d'interface standard exclusifs spécifiés par EPSON, vous n'avez pas besoin d'avoir des connaissances spéciales sur l'interface RS-232C

(1) Interface entre deux unités HX-20
Vous devez utiliser des câbles d'interface (jeu de câbles en option #715) pour interconnecter votre HX-20 avec un autre HX-20. Avec cette méthode, vous n'avez pas besoin de définir les conditions d'interface et vous pouvez donc supprimer <BLPSC>.
NOTE:
Si vous transférez directement des programmes en utilisant une commande LIST "COM0:", des données risquent d'être perdues pendant le transfert si la vitesse de traitement des données du HX-20 récepteur ne peut pas atteindre la vitesse de transfert des données de l'équipement émetteur. Dans ce cas, pour transférer correctement des programmes, il faut veiller à abaisser la vitesse binaire de l'équipement émetteur.

(2) Interface avec une imprimante comme terminal
Lorsque vous connectez une imprimante extérieure au HX-20, l'imprimante doit être équipée d'une interface RS-232C. (pour toute imprimante de la série EPSON MX, utilisez la carte d'interface #8141 ou #8145). Vous devez utilisez des câbles d'interface spéciaux (jeu de câbles #714).

Il y a deux méthodes d'utilisation d'une imprimante selon que vous souhaitez utiliser les conditions d'interface du HX-20 ou celles de l'imprimante.

- Lorsque vous définissez les conditions d'interface sur l'imprimante:
  Vous n'avez pas besoin de définir les conditions d'interface <BLPSC> pour votre HX-20. En spécifiant simplement le <nom de dispositif> sous la forme "COM0:" soit en mode direct, soit en mode programme, vous pouvez utiliser l'imprimante exactement comme la micro-imprimante "LPTO" incorporée.

Définissez les conditions d'interface de votre imprimante de la façon suivante. (Vous trouverez dans le manuel de l'utilisateur des détails sur la carte d'interface utilisable).

Vitesse binaire	4800 bps
Longueur des mots	8 bits
Bit de parité	Absence de parité
Longueur d'arrêt	2 bits

- Lorsque vous fixez les conditions d'interface sur la HX-20 :
  Si vous spécifiez comme < nom de dispositif > : "COM0:", vous devez définir les conditions d'interface <BLPSC> de la façon suivante:

Vitesse binaire Longueur des mots Bit de parité Longueur d'arrêt	A régler aux mêmes paramètres que l'imprimante
Lignes de commandes actives	Spécifiez "B" (hex) lorsque vous utilisez une carte d'interface série EPSON MX.

Exemple.

Vitesse binaire	300 bps
Longueur des mots	7 bits
Bit de parité	parité paire
Longueur d'arrêt	1 bit
Lignes de contrôle actives	Communication établie par le signal DSR (pour correspondre aux conditions définies en usine pour la carte d'interface # 8145, spécifiez "COM0" (27E1B)")

NOTE:
La vitesse binaire maximale du HX-20 est de 48oo bps. Si la vitesse binaire de l'imprimante a été fixée à 9600 bps, vous devez ramener le réglage de l'imprimante à 4800 bps étant donné que vous ne pouvez pas rendre ce paramètre compatible avec celui de l'imprimante du HX-20.

(3) Interface avec un coupleur accoustique
Pour assurer l'interface du HX-20 avec un coupleur accoustique CX-20, vous devez utiliser des câbles spéciaux d,interface (jeu de câbles en option # 706).
Quand vous utilisez le coupleur accoustique, les conditions d'interface du HX-20 et du CX-20 doivent être fixées en correspondance avec celles de l'équipement émetteur/récepteur.

Longueur des mots Bit de parité Longueur de l'arrêt	A régler aux mêmes paramètres que l'équipement d'émission/réception
Vitesse binaire	Elle doit être identique à celle de l'équipement d'émission/réception, mais elle doit être au maximum de 300 bps par suite des limitations des lignes téléphoniques. (0≤B≤2).
Lignes de contrôle actives	Etant donné que toutes les lignes de contrôle sont normalement utilisées, spécifiez "2" (hex).

5.4.2. Interface avec un autre équipement extérieur

Pour assurer l'interface du HX-20 avec des dispositifs extérieurs différents de ceux qui ont été spécifiés comme options du HX-20, vous devez avoir une connaissance approfondie de l'interface RS-232C. Vous pouvez fixer la vitesse binaire, la longueur des mots, le bit de parité et la longueur d'arrêt en correspondance avec les paramètres du dispositif à connecter au HX-20. Mais les lignes de signaux utilisées, la méthode d'établissement des communications, etc., diffèrent d'un dispositif à l'autre.
En connectant simplement les lignes de signaux de même nom, vous n'êtes pas assuré d'obtenir des communications convenables entre les deux dispositifs. Vous devez avoir une connaissance correcte de la fonction de chaque ligne de signaux.

Le HX-20 utilise les neuf lignes de signaux suivantes:

Broche No	Signal	Direction Signal	Fonction
1	GND	—	Masse signal
2	TXD	Out	Données émises
3	RXD	In	Données reçues
4	RTS	Out	Demande pour émettre
5	CTS	In	Prêt à émettre
6	DSR	In	Terminal prêt
7	DTR	Out	Terminal prêt
8	CD	In	Ponteuse détectée
E	FG	—	Masse de protection

- Broche No 1 — GND (masse)
  Signal de potentiel zéro qui sert de référence à tous les autres signaux.
- Broche No 2 — TXD
  Ligne de signal de sortie pour envoyer des données à partir du HX-20. Les données sont délivrées en logique négative.
- Broche No 3 — RXD
  Ligne des signaux d'entrée permettant au HX-20 de recevoir des données. Les données sont introduites en logique négative.
- Broche No 4 — RTS
  Ligne de signal de sortie demandant l'autorisation de transmettre des données. lorsque ce signal (qui est normalement à un potentiel positif) est actif, cela signifie que le HX-20 est prêt pour la transmission de données. En spécifiant "C" dans le paramètre <BLPSC>, vous pouvez faire en sorte qu'un signal de potentiel négatif ait une signification.

- Broche No 5 – CTS

Ligne de signal d'entrée permettant de recevoir l'autorisation de transmission. Si cette ligne de signal est à un potentiel positif, le HX-20 admet que la demande de transmission qu'il a faite a été autorisée et il commence la transmission des données. Si le potentiel de ce signal change et devient négatif, le HX-20 arrête la transmission et attend que le potentiel du signal redevienne positif pour redémarrer la transmission. En spécifiant "C" dans le paramètre < BLPSC >, vous pouvez ignorer ce signal. Si cela a été fait, le potentiel du signal est toujours supposé positif. Pour assurer l'interface avec un autre équipement que le coupleur acoustique, on connecte souvent directement ce signal au signal RTS, mais dans le HX-20, la transmission et la réception des données peuvent aussi être contrôlées par ce signal.

- Broche No 6 – DSR

Ligne du signal d'entrée utilisée par le HX-20 pour savoir si son correspondant est prêt à recevoir des données. Lorsque le potentiel de signal est positif, le HX-20 admet que son correspondant est prêt à recevoir des données. Par conséquent, la transmission des données peut être terminée en faisant passer ce potentiel de signal pour contrôler la transmission des données. Si vous spécifiez "C" dans le paramètre < BLPSC >, ce signal peut être ignoré. Dans ce cas le potentiel de ce signal est toujours supposé positif. Ce signal est, à l'origine, destiné à servir de signal de réponse au signal DTR pour détecter si oui ou non le circuit de communication est prêt. Mais, dans le cas du HX-20, cette ligne est utilisée comme ligne de signal d'entrée pour détecter l'état OCCUPE du périphérique d'émission/réception. Par conséquent pendant une transmission des données, le HX-20 surveille toujours cette ligne de contrôle.

- Broche No 7 – DTR

Cette ligne de signaux est destinée à faire délivrer un signal par le HX-20 aux dispositifs périphériques pour demander des informations sur l'état disponible ou non des circuits de communication. Cette demande d'information est faite lorsque le niveau de ce signal est positif.
Dans ce cas, cette ligne de signaux a une autre fonction. Lorsque le HX-20 doit fonctionner comme équipement récepteur, ce signal de sortie signale à l'équipement émetteur si oui ou non le HX-20 est prêt à recevoir des données. Lorsque le potentiel de ce signal est à haute impédance, le HX-20 n'est pas prêt à recevoir des données, c'est à dire qu'il est OCCUPE.
NOTE:
Le DTR est connecté à la ligne d'alimentation de RS-232C et à l'étage de commande d'interface série grande vitesse. C'est pourquoi, si l'interface RS-232C ou l'interface grande vitesse est mise sous tension, le signal DTR sera automatiquement activé.

- Broche No 8 – CD

Ligne d'entrée des signaux destinée à détecter si les données sont en cours d'émission par l'équipement émetteur en direction du HX-20. Lorsque le potentiel de ce signal est positif, le HX-20 admet que les données sont en cours d'émission par le périphérique connecté au HX-20. Cette ligne de contrôle est nécessaire lorsque le HX-20 est connecté au coupleur acoustique. Dans tous les autres cas, ce signal peut être ignoré si vous spécifiez "C" dans le réglage <BLPSC>.

Dans le HX-20, la transmission et la réception des données peuvent être contrôlées par des lignes de signaux CTS et DSR. Vous pouvez déterminer l'état de la réception des données par la ligne de signaux DTR. Lorsque vous connectez le HX-20 à un autre dispositif extérieur quelconque vous devez vérifier soigneusement les fonctions de toutes les lignes de signaux prévues dans le dispositif et vous devez faire les connexions nécessaires d'après la fonction et non d'après les désignations de chaque ligne.

Annexes

ANNEXE A Messages d'erreurs

/0 11 Division par zéro.

Une division par zéro est apparue dans une expression.

* Le diviseur d'une expression est zéro.
* Une division par une variable non définie est apparue dans une expression
* L'argument d'une fonction TAN est $\pi/2$.

AO 52 Fichier déjà ouvert.

Un fichier a déjà été ouvert sous le numéro spécifié.

* Un fichier ouvert n'a pas été fermé en mode direct.

BD 58 Données incorrectes dans un fichier.

Format incorrect des données dans un fichier.

* Une tentative a été faite pour lire un fichier programmes de format binaire, comme fichier de données en format ASCII.
* Une tentative a été faite pour lire un fichier programmes en langage machine comme un fichier de données.

BF 21 Mode de fichier incorrect.

Un mode de fichier est incorrect;

* Une tentative a été faite pour ouvrir un fichier dans un mode qui n'est pas admis pour le dispositif correspondant (p.e: ouverture de l'imprimante en mode d'entrée, etc...).

BN 50 Mauvais numéro de fichier.

Un numéro de fichier est incorrect.

* Un numéro de fichier n'appartenant pas à la gamme 1 — 16 a été utilisé dans une instruction OPEN.
* Un numéro de fichier n'appartenant pas à la gamme 1 — 16 a été utilisé dans une instruction E/S.

B0 61 Tampon complet.

Un débordement s'est produit dans le tampon d'entrée.

* Des données d'entrée venant de la porte RS-232C ("COM0:") sont en excès. (Dans le transfert de données sans établissement de communication, la vitesse binaire est trop élevée. Il faut réduire cette vitesse).

BS 9 Indice incorrect.

Un indice sortant de la dimension du réseau, ou un nombre incorrect d'indices a été utilisé.

* La dimension des éléments d'une variable réseau d'une instruction DIM est trop grande.(Cela aboutira normalement à une erreur OM ou OV).
* La valeur d'un indice différent de celui qui a été déclaré dans une instruction DIM, a été utilisé.
* Le nombre des dimensions d'un réseau est incorrect.

* Un indice de valeur supérieure à 11 est utilisé sans déclaration dans une instruction DIM.
* Un indice spécifié à 0 est référencé après une exécution d'une OPTION BASE 1.
* Un numéro d'enregistrement contenu dans une instruction PUT% ou GET% est trop grand.

CN 17 Impossible de poursuivre.

L'exécution du programme ne peut être relancé.

* Un programme s'est arrêté par suite d'une erreur.
* Le programme a été modifié après avois été INTERROMPU.
* Un programme n'est pas exécuté.
* Un abandon s'est produit, au moment ou la touche **BREAK** a été enfonçée pendant une opération E/S.

DD 10 Définition en double.

Un réseau ou une fonction utilisateur est défini en double.

* Des variables réseaux non-déclarées sont utilisées, puis déclarées dans une instruction DIM.
* Des tentatives d'exécution d'une instruction OPTION BASE ont été faites deux fois.
* Il existe une instruction DIM ou DEF FN dans une boucle.

DS 56 Instruction directe dans un fichier.

Au cours d'une opération LOAD ou MERGE, une ligne non-numérotée a été lue.

* Un fichier de données est lu.
* Un programme en langage machine est lu.

DU 60 Dispositif non disponible.

Un dispositif n'est pas disponible.

* Un dispositif non connecté au HX-20 a été spécifié.

FC 5 Appel d'une fonction non valable.

Une fonction ou une instruction a été appelée de façon incorrecte.

* Un paramètre d'une instruction ou d'une fonction est en dehors de la gamme.(Beaucoup de fonctions ne peuvent pas utiliser un paramètre négatif ou égal à zéro).
* La valeur de l'indice d'un réseau est négative.
* Une fonction utilisateur non définie a été employée.
* Le nombre des caractères spécifiés dans une instruction PRINT USING dépasse 25.
* Un réseau ou une variable non déclaré auquel aucune valeur n'a été affectée est utilisé dans une instruction SWAP.
* Un numéro de ligne dépassant 64000 a été trouvé au cours de l'exécution d'une commande RENUM.

* Dans une commande PCOPY, la zone spécifiée est la zone de programme actuellement LOGGED.
* Le décalage d'une instruction DEFFIL est trop grand.
* Un instruction PEEK ou POKE est exécutée dans la zone des programmes EPSON BASIC ou dans la zone E/S jusqu'à l'adresse &H4D.

FD 55 Descripteur de fichier incorrect.

Un descripteur de fichier est incorrect.

* Un élément du descripteur de fichier est mal orthographié.

FN 23 FOR sans NEXT.

Les instructions NEXT sont insuffisantes;

* Un instruction NEXT est partagée par plusieurs instructions FOR.
* Les instructions FOR et NEXT ne se correspondent pas une à une. (Le nom de la variable de contrôle est écrit de façon incorrecte).

ID 12 Mode direct non valable.

Une instruction qui n'est pas valable en mode direct a été introduite.

* Les instructions DEF FN, INPUT et RANDOMIZE etc... ne peuvent pas être exécutées en mode direct.

IE 54 Entrée au delà de la fin.

Toutes les données du fichier ont été lues.

* Le nombre des données et le nombre des variables à lire ne coincident pas.
* Une tentative a été faite pour lire des données en continu sans utiliser la fonction EOF.

IO 53 Erreur de dispositif E/S.

Une erreur s'est produite au cours de la communication avec un dispositif périphérique.

* Une cassette de bande est défectueuse.
* Les réglages de niveau du dérouleur de cassette audio sont inadaptés.
* Les conditions d'interface du RS-232C (Vitesse binaire, lignes de communications, etc...) sont mal adaptées.

IU 59 Dispositif en service.

Le dispositif spécifié est occupé.

* Nom erroné du dispositif.
* La même instruction OPEN a été exécutée deux fois.
* L'exécution de l'instruction CLOSE a été négligée.

LS 15 Chaîne trop longue.

Une chaîne est trop longue.

* Une tentative est faite pour affecter une variable chaîne dépassant 256 caractères.

MO 22 Opérande manquante.

Un paramètre nécessaire manque dans une expression.

* Un point est utilisé à la place d'une virgule entre des nombres.
* Un paramètre essentiel est omis.

NE 63 Fichier inexistant.

Un fichier n'existe pas sous le nom indiqué.

* Un nom de fichier est écrit incorrectement.
* La cartouche ROM utilisée ne contient pas le fichier indiqué.

NF 1 NEXT sans FOR.

Les instructions FOR sont insuffisantes.

* Un bouclage incorrect est exécuté.
* Il existe 2 ou plusieurs instructions NEXT pour une seule instruction FOR.
* Saut accidentel d'une boucle FOR-NEXT à partir d'une autre ligne de programme.

NO 57 Fichier non OUVERT.

Un numéro de fichier a été utilisé pour un fichier qui n'a pas été OUVERT.

* Un numéro de fichier est écrit de façon incorrecte.
* Une instruction OPEN n'a pas été programmée.

NR 19 Pas de relance.

Il n'y a aucune instruction RESUME dans un sous-programme de traitement d'erreur.

* A la fin d'un sous-programme de traitement d'erreur, il doit y avoir une des instructions suivantes: END, RESUME, et ON ERROR GOTO.

OD 4 Données épuisées.

On rencontre une instruction READ alors qu'il n'y a plus de données à lire.

* Le nombre des données est insuffisant.
* Une instruction RESTORE est incorrecte.
* L'emploi des séparateurs est incorrect dans une instruction DATA.

OM 7 Mémoire complète.

La capacité de la mémoire est insuffisante.

* Un programme est trop long.
* Un programme comprend trop de variables.
* Une variable réseau est trop grande.
* Des expressions sont trop complexes.
* Un programme contient trop de boucles FOR...NEXT ou GOSUB... RETURN.
* La dimension de l'espace chaîne du fichier RAM spécifiée par une commande CLEAR est trop importante.
* Le numéro d'adresse indiqué dans une instruction MEMSET est trop grand.

OS 14 Espace chaîne complet.

Un espace chaîne est insuffisant.

* L'espace chaîne, spécifié par une commande CLEAR, est trop petit.

OV 6 Dépassement.

Le résultat d'un calcul est trop grand.

* Le résultat d'une opération sur des constantes entières n'est pas dans la gamme de –32768 à 32767.
* Le résultat d'une opération faite sur des nombres réels n'est pas dans la gamme de –1.70141E38 à 1.70141E38.
* Une valeur contenue dans une commande utilisant l'adresse comme paramètre dépasse la gamme spécifiée.

PP 62 Programme protégé.

Le programme est protégé.

* Une commande NEW ou LOAD ne peut être exécutée pour le programme se trouvant dans la zone qui a été nommée par une instruction TITLE.

RG 3 RETURN sans GOSUB.

Une instruction RETURN est rencontrée avant l'exécution d'une instruction GOSUB.

* L'exécution est branchée à un sous-programme par une instruction GOTO.
* Un sous-programme est exécuté par une commande RUN.
* En l'absence d'instruction END à la dernière ligne du programme principal le sous-programme suivant est exécuté.

RW 20 RESUME sans erreur.

Une instruction RESUME est exécutée alors qu'il n'existe aucune erreur.

* Une routine de traitement d'erreur est introduite par une instruction GOTO ou GOSUB.
* En l'absence d'une instruction END à la dernière ligne du programme principal, le sous-programme suivant le traitement d'erreur est exécuté.

SN 2 Erreur de syntaxe.

Un programme n'a pas été écrit conformément à la syntaxe du langage utilisé.

* Un mot réservé en dehors de ceux qui sont reconnus, a été utilisé.
* Des parenthèses ne correspondent pas.
* Un séparateur a été mal tapé (Virgule à la place de point, deux points à la place de point virgulr, etc...).
* Un nom de variable ne commence pas par un caractère alphabétique.
* Un nom de variable commence par un mot réservé.
* Le nombre de paramètres d'une fonction ou d'une instruction est incorrect.
* Des caractères sans rapport sont écrits dans la dernière partie d'une ligne invisible sur l'écran matériel.
* Une variable chaine est utilisée avant la liste des noms de variables dans une instruction PUT% ou GET%.

ST 16 Formule chaîne trop complexe.

Une expression chaîne est trop complexe.

* Une expression chaîne écrite sur une seule ligne est trop longue ou trop complexe. Il y a trop de parenthèses imbriquées dans une expression chaîne

TM 13 Type inadapté.

Il ya une mauvaise adaptation du type de variable.

* Un nom de variable numérique est affecté à une variable chaîne.
* Un nom de variable chaîne est affecté à une variable numérique.
* Un défaut de type existe dans l'argument d'une fonction.

UF 18 Fonction utilisateur non définie.

Une fonction USR n'a pas été définie.

* Un nom de variable commençant par "FN" est utilisé.
* Le nom de la fonction est incorrect dans une instruction DEF FN.
* L'instruction DEF FN n'est pas exécutée. (L'exécution d'un programme a commencé à partir du milieu du programme par une instruction GOTO ou une instruction du même type).

UL 8 Numéro de ligne non défini.

Erreur dans un numéro de ligne;

* Un numéro de ligne n'est pas indiqué.
* Un numéro de ligne, indiqué dans une instruction GOTO, GOSUB, RESTORE ou RUN n'existe pas.
* La ligne à référencer quand l'instruction RENUM est exécutée n'existe pas.

UP 21 Erreur non imprimable.

Signale une erreur ayant un code d'erreur non défini.

* Une instruction ERROR est exécutée en l'absence d'un quelconque sous-programme de traitement d'erreur.
* Les codes d'erreur 26 à 49 et 64 à 255 feront afficher ce message.

WE 24 WHILE sans WEND.

* Ce message est employé avec le BASIC disque.

WH 25 WEND sans WHILE.

* Ce message est employé avec le BASIC disque.

Comment lire la table des messages

SN 2 Erreur de syntaxe

Le programme n'est pas écrit en respectant la syntaxe

* Un mot réservé différent des mots reconnus a été utilisé.

Signification et explication du message d'erreur

Code d'erreur

Message d'erreur

ANNEXE B Noms des dispositifs

Nom de dispositif	Nom de l'équipement	Entrée	Sortie	Remarque
KYBD	Clavier	○	X	
SCRN:	Ecran	X	○	
LPT0:	Micro-imprimante incorporée	X	○	
COM0:	Porte RS-232C	○	○	
CAS0:	Unité à microcassette	○	○	Option
CAS1:	Cassette audio	○	○	
PAC0:	Cartouche ROM	○	X	Option
A:	Unité à disque souple A	○	○	Noms des dispositifs pour le BASIC DISK
B:	Unité à disque souple B	○	○	
C:	Unité à disque souple C	○	○	
D:	Unité à disque souple D	○	○	

○: Application X : Non applicable

ANNEXE C Table de Correspondance entre les Noms des dispositifs et les Commandes BASIC

Command \ Device	KYBD:	SCRN:	LPT0:	COM0:	CAS0:	CAS1:	PAC0:
LOAD	×	×	×	○	○	○	○
LOADM	×	×	×	×	○	○	○
LOAD?	×	×	×	×	○	○	×
RUN "<file descriptor>"	×	×	×	○	○	○	○
MERGE	×	×	×	○	○	○	○
FILES	×	×	×	×	○	○	○
INPUT#	○	×	×	○	○	○	○
INPUT$	○	×	×	○	○	○	○
FOF	–	×	×	○	○	○	○
LOF	–	×	×	○	–	–	○
SAVE	×	○	○	○	○	○	×
SAVEM	×	×	×	×	○	○	×
LIST	×	○	○	○	○	○	×
PRINT# (USING)	×	○	○	○	○	○	×
POS	×	○	○	○	–	–	×
OPEN mode	I	O	O	I/O	I/O	I/O	I

NOTE:

○ou X sont employés dans la table ci-dessus pour indiquer qu'un dispositif spécifié pour une commande ou une instruction

○: peut être utilisé

X: ne peut pas être utilisé. Une erreur FC en résulte

–: n'entraîne pas d'erreur mais la commande n'est pas valable.

ANNEXE D Caractères de Formattage

Chaîne format	Fonction
!	Spécifie la sortie du premier caractère de la chaîne seulement
ç ... ç	Spécifie le nombre de caractères à sortir à partir du début d'une chaîne donnée.
&	Spécifie les positions de sortie des caractères d'une chaîne donnée.
#	Spécifie chaque position de chiffre.
.	Spécifie la position du point décimal.
+	Sort le signe d'un nombre (plus ou moins) avant ou après le nombre.
–	Sort les nombres négatifs avec un signe moins devant.
* *	Fait combler les espaces avant de la chaîne numérique par des astérisques.
S S	Fait sortir le signe dollar immédiatement à gauche du nombre formatté.
**S	Fait combler les espaces du début avec des astérisques et sortir un signe dollar avant le nombre.
,	Fait sortir une virgule au troisième chiffre à gauche du point décimal.
----	Sort une valeur numérique sous forme exponentielle.
–	Sort l'un quelconque des caractères de formattage ci-dessus comme un caractère littéral.

NOTE: *Les caractères de formattage présentés ci-dessus s'appliquent au jeu de caractères Français. Si vous sélectionnez d'autres jeux de caractères, certains des caractères de formattage seront sortis différemment, comme l'indique la table ci-dessous.*

Code Décimal	France	Allemagne	Norvège	Danemark	Suède	ASCII
36	$	$	¤	$	¤	$
92	ç	Ö	ø	ø	Ö	\
94	^	^	Ü	Ü	Ü	^

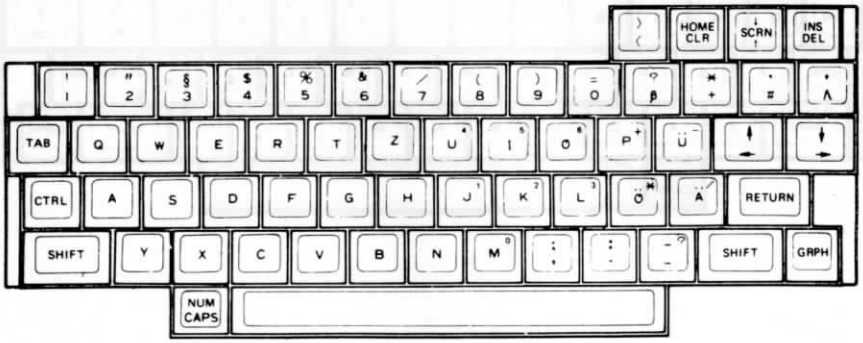
(Voir Chapitre 3 POKE)

ANNEXE E Disposition des claviers et Affectations des touches

1. Disposition des claviers

France

Allemagne



Suède

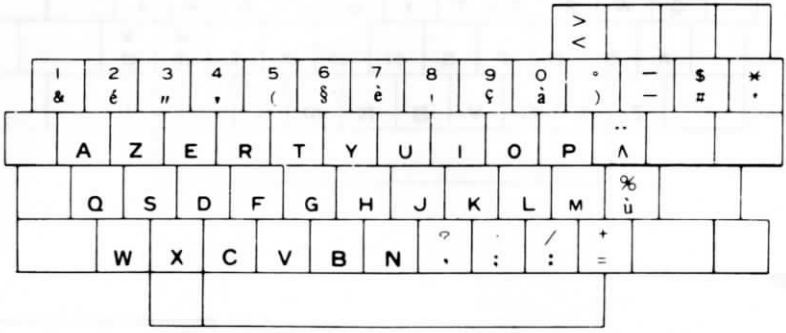
Danemark

Norvège (option)

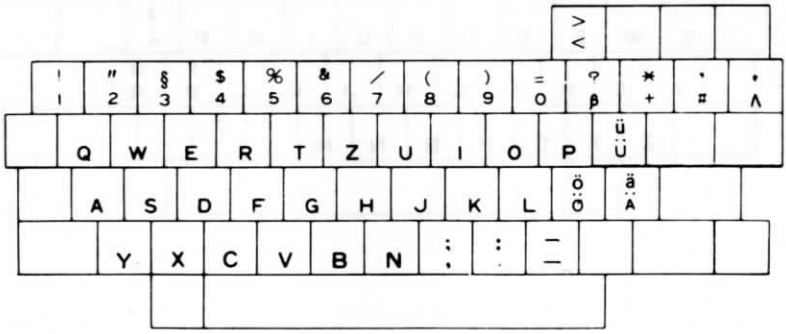
2. Affectation des touches pour chaque mode du clavier

(1) Mode majuscule et mode minuscule

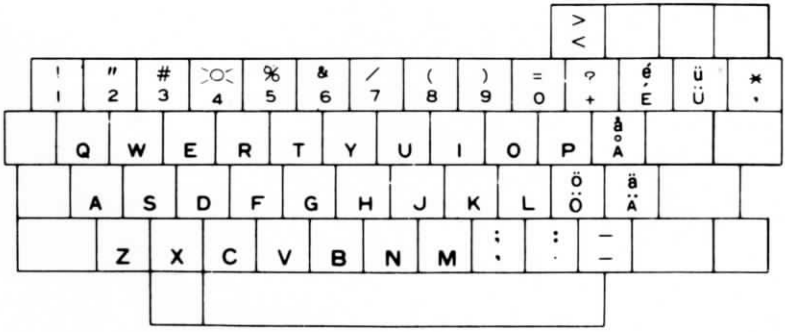
France



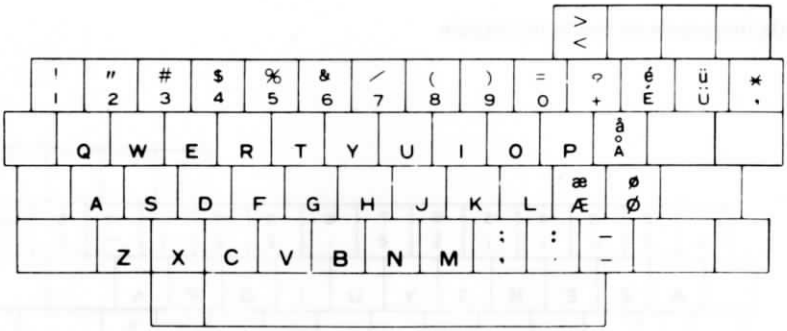
Allemagne



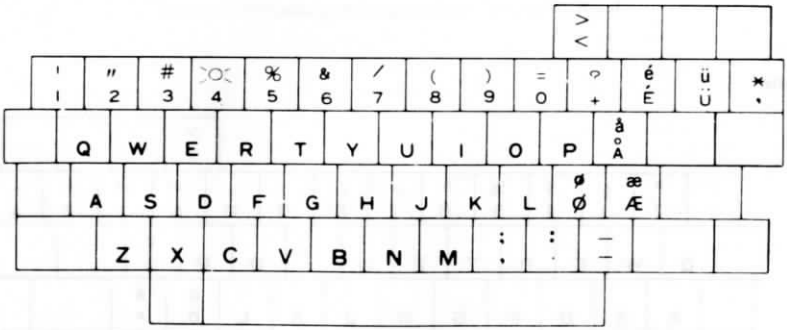
Suède



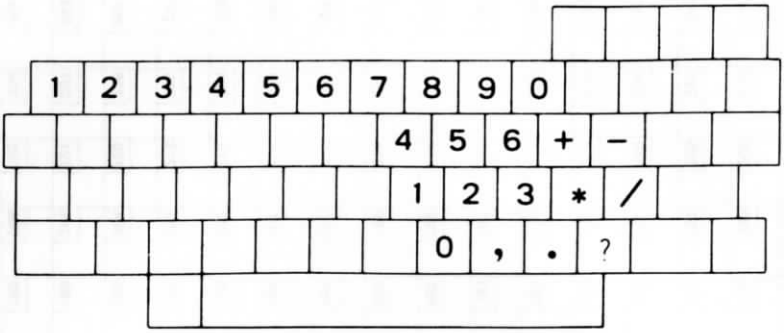
Danemark



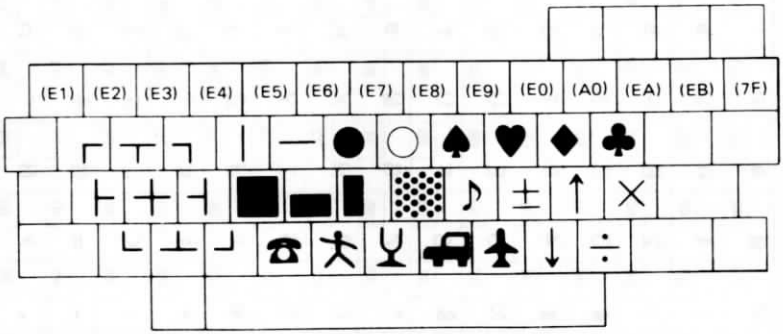
Norvège



(2) Mode des touches numériques



(3) Emplacements des touches graphiques



NOTE: E0 à EB, A0 et 7F, présentés entre parenthèses sur la figure ci-dessus sont des codes de caractères de nombres hexadécimaux qui peuvent être introduits en appuyant sur les touches correspondantes, tout en maintenant la touche GRPH enfoncée. (Voir Chapitre 8, "Définition des configurations Graphiques" dans le manuel Opérateur du HX-20).

ANNEXE F Table des codes de caractères

1. France

	Hex. No.	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
Hex. No.	Binary No.	0000	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100	1101	1110	1111
0	0000	0	16	SP 32	0 48	à 64	P 80	` 96	p 112	+ 128	° 144	160	176	192	208	224	240
1	0001	1	17	! 33	1 49	A 65	Q 81	a 97	q 113	⊥ 129	♠ 145	161	177	193	209	225	241
2	0010	2	18	" 34	2 50	B 66	R 82	b 98	r 114	T 130	♥ 146	162	178	194	210	226	242
3	0011	3	19	# 35	3 51	C 67	S 83	c 99	s 115	┤ 131	♦ 147	163	179	195	211	227	243
4	0100	4	20	$ 36	4 52	D 68	T 84	d 100	t 116	├ 132	♣ 148	164	180	196	212	228	244
5	0101	5	21	% 37	5 53	E 69	U 85	e 101	u 117	─ 133	♪ 149	165	181	197	213	229	245
6	0110	6	22	& 38	6 54	F 70	V 86	f 102	v 118	│ 134	ª 150	166	182	198	214	230	246
7	0111	7	23	' 39	7 55	G 71	W 87	g 103	w 119	┌ 135	± 151	167	183	199	215	231	247
8	0100	8	24	(40	8 56	H 72	X 88	h 104	x 120	┐ 136	♥ 152	168	184	200	216	232	248
9	0101	9	25	) 41	9 57	I 73	Y 89	i 105	y 121	└ 137	¥ 153	169	185	201	217	233	249
A	0110	10	26	* 42	: 58	J 74	Z 90	j 106	z 122	┘ 138	₧ 154	170	186	202	218	234	250
B	0111	11	27	+ 43	; 59	K 75	[91	k 107	é 123	▒ 139	↑ 155	171	187	203	219	235	251
C	1000	12	28	, 44	< 60	L 76	¥ 92	l 108	ù 124	▌ 140	↓ 156	172	188	204	220	236	252
D	1001	13	29	- 45	= 61	M 77	§ 93	m 109	è 125	▬ 141	× 157	173	189	205	221	237	253
E	1010	14	30	. 46	> 62	N 78	^ 94	n 110	¨ 126	▐ 142	÷ 158	174	190	206	222	238	254
F	1011	15	31	/ 47	? 63	O 79	_ 95	o 111	127	● 143	± 159	175	191	207	223	239	255

	Hex. No.	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
Hex. No.	Binary No.	0000	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100	1101	1110	1111
0	0000	0	16	SP 32	0 48	§ 64	P 80	` 96	p 112	┼ 128	○ 144	160	176	192	208	224	240
1	0001	1	17	! 33	1 49	A 65	Q 81	a 97	q 113	┴ 129	♠ 145	161	177	193	209	225	241
2	0010	2	18	" 34	2 50	B 66	R 82	b 98	r 114	┬ 130	♥ 146	162	178	194	210	226	242
3	0011	3	19	# 35	3 51	C 67	S 83	c 99	s 115	┤ 131	♦ 147	163	179	195	211	227	243
4	0100	4	20	$ 36	4 52	D 68	T 84	d 100	t 116	├ 132	♣ 148	164	180	196	212	228	244
5	0101	5	21	% 37	5 53	E 69	U 85	e 101	u 117	─ 133	♪ 149	165	181	197	213	229	245
6	0110	6	22	& 38	6 54	F 70	V 86	f 102	v 118	│ 134	α 150	166	182	198	214	230	246
7	0111	7	23	' 39	7 55	G 71	W 87	g 103	w 119	┌ 135	± 151	167	183	199	215	231	247
8	1000	8	24	(40	8 56	H 72	X 88	h 104	x 120	┐ 136	■ 152	168	184	200	216	232	248
9	1001	9	25	) 41	9 57	I 73	Y 89	i 105	y 121	└ 137	¥ 153	169	185	201	217	233	249
A	1010	10	26	* 42	: 58	J 74	Z 90	j 106	z 122	┘ 138	⅄ 154	170	186	202	218	234	250
B	1011	11	27	+ 43	; 59	K 75	Ä 91	k 107	ä 123	▒ 139	↑ 155	171	187	203	219	235	251
C	1100	12	28	, 44	< 60	L 76	Ö 92	l 108	ö 124	▐ 140	↓ 156	172	188	204	220	236	252
D	1101	13	29	- 45	= 61	M 77	Ü 93	m 109	ü 125	▬ 141	× 157	173	189	205	221	237	253
E	1110	14	30	. 46	> 62	N 78	^ 94	n 110	ß 126	▌ 142	÷ 158	174	190	206	222	238	254
F	1111	15	31	/ 47	? 63	O 79	_ 95	o 111	127	● 143	± 159	175	191	207	223	239	255

	Hex. No.	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
Hex. No.	Binary No.	0000	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100	1101	1110	1111
0	0000	0	16	SP 32	0 48	É 64	P 80	é 96	p 112	+ 128	○ 144	160	176	192	208	224	240
1	0001	1	17	! 33	1 49	A 65	Q 81	a 97	q 113	⊥ 129	♠ 145	161	177	193	209	225	241
2	0010	2	18	" 34	2 50	B 66	R 82	b 98	r 114	T 130	♥ 146	162	178	194	210	226	242
3	0011	3	19	# 35	3 51	C 67	S 83	c 99	s 115	┤ 131	♦ 147	163	179	195	211	227	243
4	0100	4	20	¤ 36	4 52	D 68	T 84	d 100	t 116	├ 132	♣ 148	164	180	196	212	228	244
5	0101	5	21	% 37	5 53	E 69	U 85	e 101	u 117	─ 133	♪ 149	165	181	197	213	229	245
6	0110	6	22	& 38	6 54	F 70	V 86	f 102	v 118	│ 134	♫ 150	166	182	198	214	230	246
7	0111	7	23	' 39	7 55	G 71	W 87	g 103	w 119	┌ 135	↑ 151	167	183	199	215	231	247
8	1000	8	24	(40	8 56	H 72	X 88	h 104	x 120	┐ 136	↓ 152	168	184	200	216	232	248
9	1001	9	25	) 41	9 57	I 73	Y 89	i 105	y 121	└ 137	→ 153	169	185	201	217	233	249
A	1010	10	26	* 42	: 58	J 74	Z 90	j 106	z 122	┘ 138	← 154	170	186	202	218	234	250
B	1011	11	27	+ 43	; 59	K 75	Ä 91	k 107	ä 123	▒ 139	↑ 155	171	187	203	219	235	251
C	1100	12	28	, 44	< 60	L 76	Ö 92	l 108	ö 124	▌ 140	↓ 156	172	188	204	220	236	252
D	1101	13	29	− 45	= 61	M 77	Å 93	m 109	å 125	▬ 141	× 157	173	189	205	221	237	253
E	1110	14	30	. 46	> 62	N 78	Ü 94	n 110	ü 126	▐ 142	÷ 158	174	190	206	222	238	254
F	1111	15	31	/ 47	? 63	O 79	_ 95	o 111	127	● 143	± 159	175	191	207	223	239	255

	Hex. No.	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
Hex. No.	Binary No.	0000	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100	1101	1110	1111
0	0000	0	16	SP 32	Ø 48	É 64	P 80	é 96	p 112	┼ 128	○ 144	160	176	192	208	224	240
1	0001	1	17	! 33	1 49	A 65	Q 81	a 97	q 113	┴ 129	♠ 145	161	177	193	209	225	241
2	0010	2	18	" 34	2 50	B 66	R 82	b 98	r 114	┬ 130	♥ 146	162	178	194	210	226	242
3	0011	3	19	# 35	3 51	C 67	S 83	c 99	s 115	┤ 131	♦ 147	163	179	195	211	227	243
4	0100	4	20	$ 36	4 52	D 68	T 84	d 100	t 116	├ 132	♣ 148	164	180	196	212	228	244
5	0101	5	21	% 37	5 53	E 69	U 85	e 101	u 117	─ 133	♪ 149	165	181	197	213	229	245
6	0110	6	22	& 38	6 54	F 70	V 86	f 102	v 118	│ 134	ª 150	166	182	198	214	230	246
7	0111	7	23	' 39	7 55	G 71	W 87	g 103	w 119	┌ 135	± 151	167	183	199	215	231	247
8	1000	8	24	(40	8 56	H 72	X 88	h 104	x 120	┐ 136	■ 152	168	184	200	216	232	248
9	1001	9	25	) 41	9 57	I 73	Y 89	i 105	y 121	└ 137	² 153	169	185	201	217	233	249
A	1010	10	26	* 42	: 58	J 74	Z 90	j 106	z 122	┘ 138	³ 154	170	186	202	218	234	250
B	1011	11	27	+ 43	; 59	K 75	Æ 91	k 107	æ 123	▒ 139	↑ 155	171	187	203	219	235	251
C	1100	12	28	, 44	< 60	L 76	ø 92	l 108	ø 124	▌ 140	↓ 156	172	188	204	220	236	252
D	1101	13	29	- 45	= 61	M 77	Å 93	m 109	å 125	▬ 141	× 157	173	189	205	221	237	253
E	1110	14	30	. 46	> 62	N 78	Ü 94	n 110	ü 126	▐ 142	÷ 158	174	190	206	222	238	254
F	1111	15	31	/ 47	? 63	O 79	_ 95	o 111	127	● 143	± 159	175	191	207	223	239	255

5. Norvège(option)

	Hex. No.	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
Hex. No.	Binary No.	0000	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100	1101	1110	1111
0	0000	0	16	SP 32	0 48	É 64	P 80	é 96	p 112	┼ 128	○ 144	160	176	192	208	224	240
1	0001	1	17	! 33	1 49	A 65	Q 81	a 97	q 113	┴ 129	♠ 145	161	177	193	209	225	241
2	0010	2	18	" 34	2 50	B 66	R 82	b 98	r 114	┬ 130	♥ 146	162	178	194	210	226	242
3	0011	3	19	# 35	3 51	C 67	S 83	c 99	s 115	┤ 131	♦ 147	163	179	195	211	227	243
4	0100	4	20	¤ 36	4 52	D 68	T 84	d 100	t 116	├ 132	♣ 148	164	180	196	212	228	244
5	0101	5	21	% 37	5 53	E 69	U 85	e 101	u 117	─ 133	♪ 149	165	181	197	213	229	245
6	0110	6	22	& 38	6 54	F 70	V 86	f 102	v 118	│ 134	ª 150	166	182	198	214	230	246
7	0111	7	23	' 39	7 55	G 71	W 87	g 103	w 119	┌ 135	± 151	167	183	199	215	231	247
8	1000	8	24	(40	8 56	H 72	X 88	h 104	x 120	┐ 136	● 152	168	184	200	216	232	248
9	1001	9	25	) 41	9 57	I 73	Y 89	i 105	y 121	└ 137	¥ 153	169	185	201	217	233	249
A	1010	10	26	* 42	: 58	J 74	Z 90	j 106	z 122	┘ 138	₧ 154	170	186	202	218	234	250
B	1011	11	27	+ 43	; 59	K 75	Æ 91	k 107	æ 123	▒ 139	↑ 155	171	187	203	219	235	251
C	1100	12	28	, 44	< 60	L 76	Ø 92	l 108	ø 124	▐ 140	↓ 156	172	188	204	220	236	252
D	1101	13	29	- 45	= 61	M 77	Å 93	m 109	å 125	▬ 141	× 157	173	189	205	221	237	253
E	1110	14	30	. 46	> 62	N 78	Ü 94	n 110	ü 126	▌ 142	÷ 158	174	190	206	222	238	254
F	1111	15	31	/ 47	? 63	O 79	_ 95	o 111	127	● 143	± 159	175	191	207	223	239	255

6. ASCII

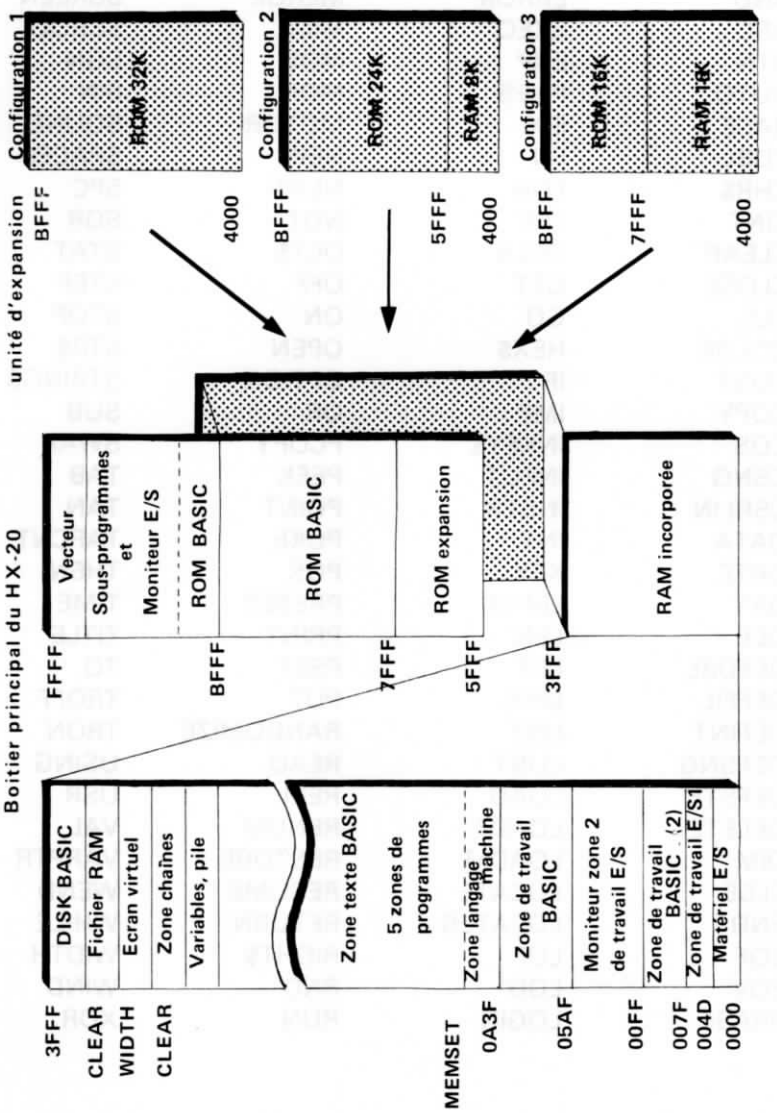
	Hex. No.	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
Hex. No.	Binary No.	0000	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100	1101	1110	1111
0	0000	0	16	SP 32	0 48	@ 64	P 80	` 96	p 112	┼ 128	○ 144	160	176	192	208	224	240
1	0001	1	17	! 33	1 49	A 65	Q 81	a 97	q 113	┴ 129	♠ 145	161	177	193	209	225	241
2	0010	2	18	" 34	2 50	B 66	R 82	b 98	r 114	┬ 130	♥ 146	162	178	194	210	226	242
3	0011	3	19	# 35	3 51	C 67	S 83	c 99	s 115	┤ 131	♦ 147	163	179	195	211	227	243
4	0100	4	20	$ 36	4 52	D 68	T 84	d 100	t 116	├ 132	♣ 148	164	180	196	212	228	244
5	0101	5	21	% 37	5 53	E 69	U 85	e 101	u 117	─ 133	♪ 149	165	181	197	213	229	245
6	0110	6	22	& 38	6 54	F 70	V 86	f 102	v 118	│ 134	ª 150	166	182	198	214	230	246
7	0111	7	23	' 39	7 55	G 71	W 87	g 103	w 119	┌ 135	± 151	167	183	199	215	231	247
8	1000	8	24	(40	8 56	H 72	X 88	h 104	x 120	┐ 136	º 152	168	184	200	216	232	248
9	1001	9	25	) 41	9 57	I 73	Y 89	i 105	y 121	└ 137	¥ 153	169	185	201	217	233	249
A	1010	10	26	* 42	: 58	J 74	Z 90	j 106	z 122	┘ 138	⅄ 154	170	186	202	218	234	250
B	1011	11	27	+ 43	; 59	K 75	[91	k 107	{ 123	▒ 139	↑ 155	171	187	203	219	235	251
C	1100	12	28	, 44	< 60	L 76	\ 92	l 108	¦ 124	▄ 140	↓ 156	172	188	204	220	236	252
D	1101	13	29	- 45	= 61	M 77	] 93	m 109	} 125	▬ 141	× 157	173	189	205	221	237	253
E	1110	14	30	. 46	> 62	N 78	^ 94	n 110	~ 126	▌ 142	÷ 158	174	190	206	222	238	254
F	1111	15	31	/ 47	? 63	O 79	_ 95	o 111	127	● 143	± 159	175	191	207	223	239	255

ANNEXE G Codes de contrôle

Décimal	Hexadécimal	Fonction	Touches
1	01	Déplace l'écran matériel dans l'angle gauche de l'écran virtuel	CTRL + A
3	03	Sort du mode AUTO	CTRL + C
4	04	Déplace l'écran matériel vers la droite	CTRL + D, CTRL + ↓
5	05	Supprime tous les caractères jusqu'à la fin de la ligne	CTRL + E
6	06	Déplace l'écran matériel dans l'angle droit de l'écran virtuel	CTRL + F
8	08	Espace arrière	CTRL + H, INS DEL
9	09	TAB (espacés de 8 colonnes)	CTRL + I, TAB
10	0A	Changement de ligne	CTRL + J
11	0B	Amène le curseur à sa position de repos	CTRL + K, SHIFT + HOME CLR
12	0C	Efface l'écran virtuel	CTRL + L, HOME CLR
13	0D	Retour de chariot	CTRL + M, RETURN
16	10	Fait monter l'écran virtuel	CTRL + P SC ⇕ RN
17	11	Fait descendre l'écran virtuel	CTRL + Q, SHIFT + SC ⇕ RN
18	12	Mode d'insertion	CTRL + R, SHIFT + INS DEL
19	13	Déplace l'écran matériel vers la gauche	CTRL + S, CTRL + ↑
22	16	Met le curseur en service (ON)	CTRL + V
23	17	Met le curseur hors service	CTRL + W
26	1A	Supprime tous les cractères depuis la position du curseur jusqu'à la ligne inférieure de l'écran virtuel	CTRL + Z
28	1C	Déplace le curseur vers la droite	→
29	1D	Déplace le curseur vers la gauche	←
30	1E	Fait monter le curseur	SHIFT + ↑
31	1F	Fait descendre le curseur	SHIFT + ↓

Les codes de caractères 0 à 31 ne sont pas affichés comme des caractères même s'ils sont dans une instruction de sortie. Mais, s'ils font partie d'une chaîne, chaque code de caractère sera compté comme pour une longueur d'un caractère.

ANNEXE H Configuration de la mémoire



ANNEXE I Table des mots réservés

ABS	ERL	LPRINT	SAVE
ALL	ERR	MEMSET	SAVEM
AND	ERROR	MERGE	SCREEN
ASC	EXEC	MID$	SCROLL
ATN	EXP	MOD	SGN
AUTO	FILES	MON	SIN
BASE	FIX	MOTOR	SOUND
CDBL	FN	NEW	SPACE$
CHR$	FOR	NEXT	SPC
CINT	FRE	NOT	SQR
CLEAR	GCLS	OCT$	STAT
CLOSE	GET	OFF	STEP
CLS	GO	ON	STOP
COLOR	HEX$	OPEN	STR$
CONT	IF	OPTION	STRING$
COPY	IMP	OR	SUB
COS	INKEY$	PCOPY	SWAP
CSNG	INPUT	PEEK	TAB
CSRLIN	INSTR	POINT	TAN
DATA	INT	POKE	TAPCNT
DATE	KEY	POS	THEN
DAY	LEFT$	PRESET	TIME
DEF	LEN	PRINT	TITLE
DEFDBL	LET	PSET	TO
DEFFIL	LINE	PUT	TROFF
DEFINT	LIST	RANDOMIZE	TRON
DEFSNG	LLIST	READ	USING
DEFSTR	LOAD	REM	USR
DELETE	LOAD?	RENUM	VAL
DIM	LOADM	RESTORE	VARPTR
ELSE	LOCATE	RESUME	WEND
END	LOCATES	RETURN	WHILE
EOF	LOF	RIGHT$	WIDTH
EQV	LOG	RND	WIND
ERASE	LOGIN	RUN	XOR

ANNEXE J Liste des commandes et instructions

AUTO

FORMAT AUTO [<numéro de ligne>][, [<incrément>]]
OBJET créer un numéro de ligne automatiquement.
EXEMPLE
AUTO 100, 10
AUTO 200,
AUTO 300
AUTO

CLEAR

FORMAT CLEAR [<mémoire pour chaînes de caractères>] , <mémoire pour fichier RAM>]]
OBJET Initialiser des variables et fixer la dimension de la mémoire des chaînes de caractères et des fichiers RAM.
EXEMPLE CLEAR 200, 256

CLOSE

FORMAT CLOSE [[#]<numéro de fichier>, [#]< numéro de fichier>...]]
OBJET Fermer des fichiers
EXEMPLE CLOSE # 3

*CLS

FORMAT CLS
OBJET Effacer un écran texte
EXEMPLE CLS

COLOR

FORMAT COLOR [< couleur texte >][, [< couleur fond >] [, < numéro de l'ensemble >]]
OBJET Spécifier les couleurs de l'écran de l'affichage extérieur.
EXEMPLE COLOR 0, 3, 0

CONT

FORMAT CONT
OBJET Reprendre l'exécution d'un programme qui a été arrêté.
EXEMPLE CONT

COPY

FORMAT COPY

OBJET Sortir les caractères et les graphiques affichés par le LCD sur la micro-imprimante incorporée.

EXEMPLE COPY

DATA

FORMAT DATA <constante>[, <constante>...]

OBJET Stocker les constantes numériques et les chaînes auxquelles on accède par la ou les instructions(s) READ.

EXEMPLE DATA HX, 20, EPSON

DEFFIL

FORMAT DEFFIL <longueur d'enregistrement>, <adresse relative>

OBJET Définir l'adresse relative de l'enregistrement 0 d'un fichier RAM et la longueur d'enregistrement.

EXEMPLE DEFFIL 20, 200

DEF FN

FORMAT DEF FN <nom>[(<paramètre>[, <paramètre>...])]= <définition de fonction>

OBJET Définir une fonction créée par l'utilisateur

EXEMPLE DEF FNZ (X,Y) X 2 Y 3 A

DEFINT/SNG/DBL/STR

FORMAT DEF INT <gamme(s) de lettres>
SNG
DBL
STR

OBJET Déclarer que les variables sont du type entier, simple précision, double précision et chaînes.

EXEMPLE DEFSTR A, X – Z

DEF USR

FORMAT DEF USR [<chiffre>] = <adresse de départ>

OBJET Spécifier l'adresse de départ d'un sous-programme en langue machine

EXEMPLE DEF USR6 = &H0C00

DELETE

FORMAT DELETE [<numéro de ligne de départ>][—[<numéro de ligne de fin>]]

OBJET Supprimer des lignes spécifiées d'un programme.

EXEMPLE
DELETE 100–200
DELETE 100–
DELETE –200
DELETE 100

DIM

FORMAT DIM < variable > (< variable maximale d'indice >[, < valeur maximale d'indice > ...])[,...]

OBJET Déclarer la dimension des éléments d'une variable réseau

EXEMPLE DIM A(40, 10), BS (50)

END

FORMAT END

OBJET Fermer tous les fichiers et achever l'exécution d'un programme.

EXEMPLE END

ERASE

FORMAT ERASE <variable tableau>[, <variable tableau> ...]

OBJET Eliminer les tableaux d'un programme.

EXEMPLE ERASE A,B

ERROR

FORMAT ERREUR <expression entière>

OBJET Simuler l'apparition d'une erreur ; ou permettre à l'utilisateur de définir des codes d'erreur.

EXEMPLE ERREUR 225

EXEC

FORMAT EXEC [<adresse de départ>]

OBJET Démarrer l'exécution d'un sous-programme en langage machine

EXEMPLE EXEC &H0C00

FILES

FORMAT FILES [« <nom de dispositif> »]

OBJET Afficher les noms de tous les fichiers résidents dans une mémoire spécifiée.

EXEMPLE FILES «CAS1:»

FOR...TO...STEP-NEXT

FORMAT FOR <variable> = <valeur initiale> TO <valeur finale>
[STEP <incrément>]
•
•
•
NEXT [<variable> [, <variable>...]]

OBJET Répéter un certain nombre de fois les instructions placées entre les instructions FOR et NEXT.

EXEMPLE FOR I=0 TO 100 STEP 5
•
•
•
NEXT I

GCLS

FORMAT GCLS

OBJET Effacer un écran graphique

EXEMPLE GCLS

GET%

FORMAT GET% < numéro d'enregistrement >, < nom de variable >[, < nom de variable > ...]

OBJET Lire des données d'un fichier RAM pour les utiliser comme variables.

EXEMPLE GET% 0, A!, B # , C$

GOSUB...RETURN

FORMAT GOSUB <numéro de ligne >
•
•
•
RETURN

OBJET Branchement à un sous-programme et retour de sous-programme.

EXEMPLE GOSUB 500

GO TO/GOTO

FORMAT (1) GO TO <numéro de ligne >, ou
(2) GOTO <numéro de ligne >

OBJET Brancher l'exécution d'un programme à un numéro de ligne particulier.

EXEMPLE GOTO 300

IF...THEN...ELSE/IF...GOTO...ELSE

FORMAT IF <expression logique>
THEN <instruction> [ELSE < instruction >]
<numéro de ligne > <numéro de ligne >
GOTO <numéro de ligne >

OBJET Choisir une voie particulière pour l'exécution d'un programme d'après des conditions établies dans une expression logique.

EXEMPLE IF A >10 THEN A =0 ELSE 200

INPUT

FORMAT INPUT ["<chaîne d'interrogation >"] , <variable> [, <variable>...]

OBJET Permettre l'entrée au clavier d'une variable spécifiée pendant l'exécution d'un programme.

EXEMPLE INPUT «NAME» ; A$

INPUT#

FORMAT INPUT# <numéro de fichier >, <variable >[, <variable >...]

OBJET Lire les éléments de données dans un fichier spécifié et les affecter aux valeurs d'un programme.

EXEMPLE INPUT# 1, A, B, C$

KEY

FORMAT KEY <numéro de touche >, <chaîne >

OBJET Définir les touches de fonctions programmables

EXEMPLE KEY 1 , "LIST"

KEY/LIST/KEY LLIST

FORMAT (1) KEY LIST
(2) KEY LLIST

OBJET Afficher les chaînes affectées aux touches de fonctions programmables sur l'écran et la micro-imprimante respectivement.

EXEMPLE KEY LLIST

LET

FORMAT [LET]<variable >=<expression >

OBJET Affecter la valeur d'une expression à une variable.

EXEMPLE LET A =3.141592

LINE

FORMAT LINE [(< coordonnée horizontale 1 >, < coordonnée verticale 1 >)]–(<coordonnée horizontale 2 >, <coordonnée verticale 2 >),
| PSET | [,<couleur >]
| PRESET |

OBJET tracer une ligne droite entre deux points définis.

EXEMPLE LINE (0,0) – (119,31), PSET

LINE INPUT

FORMAT LINE INPUT [« <chaîne d'interrogation >»;] <variable chaîne >

OBJET Introduire une ligne entière comme variable chaîne.

EXEMPLE LINE INPUT « WHAT?» ; A$

LINE INPUT#

FORMAT LINE INPUT #< numéro de fichier >, <variable chaîne >

OBJET Lire une ligne entière d'un fichier séquentiel de données et l'introduire dans une variable chaîne.

EXEMPLE LINE INPUT # 1, A$

LIST/LLIST

FORMAT (1) LIST [< numéro de ligne de départ >][–[< numéro de la dernière ligne >]]
(2) LLIST [< numéro de ligne de départ >][– [< numéro de la dernière ligne >]] .

OBJET Sortir un listage de programme (1) sur l'écran LCD ou sur l'écran extérieur ou (2) sur la micro-imprimante.

EXEMPLE
LIST 100 – 200
LIST – 200
LIST 100 –
LIST 200
LIST.
LIST

LIST<descripteur de fichier>

FORMAT LIST <descripteur de fichier >[, [<numéro de ligne >]
[–[<numéro de ligne >]]]

OBJET Sortir un listage de programme sur un fichier spécifié.

EXEMPLE LIST "COM0:"

***LIST "COM0:"**

FORMAT LIST" COM0: [(< BLPSC >)] " [, [< numéro de ligne >][—[< numéro de ligne >]]]

OBJET Spécifier les conditions d'interfaces de la porte RS — 232C et exécuter LIST.

EXEMPLE LIST" COM0: (270 1B)".

LOAD

FORMAT LOAD [< descripteur de fichier >[,R]]

OBJET Charger un fichier programme dans la mémoire.

EXEMPLE LOAD " CAS 1: PROG1.ASC"

***LOAD "COM0:"**

FORMAT LOAD "COM0: [(< BLPSC >)] "

OBJET Spécifier les conditions d'interface de la porte RS — 232C et l'exécution de LOAD.

EXEMPLE LOAD" COM0: (68N2B)" [,R]

LOADM

FORMAT LOADM < descripteur de fichier >][, [valeur de décalage] [, R]]

OBJET Charger un fichier programme en langage machine dans la mémoire.

EXEMPLE LOADM "CAS 1: ABC"

***LOAD?**

FORMAT LOAD? [< descripteur de fichier >]

OBJET Vérifier les fichiers

EXEMPLE LOAD? "CAS 1: PROG 1.ASC"

***LOCATE**

FORMAT LOCATE < coordonnée horizontale >, < coordonnée verticale > [, < commutateur du curseur >]

OBJET Spécifier la position du curseur sur l'écran.

EXEMPLE LOCATE 10, 10, 0

***LOCATES**

FORMAT LOCATES < coordonnée horizontale > , < coordonnée verticale >

OBJET Spécifier la position de l'écran matériel

EXEMPLE LOCATES 0, 0

***LOGIN**

FORMAT LOGIN <expression>[, R]
OBJET Commuter les zones de programmes
EXEMPLE LOGIN 3

***MEMSET**

FORMAT MEMSET [<adresse la plus faible de la mémoire>]
OBJET Spécifier la limite inférieure de la mémoire.
EXEMPLE MEMSET &H0D00

MERGE

FORMAT MERGE [<descripteur de fichier>[, R]
OBJET Fusionner un fichier programme spécifié avec le programme se trouvant à ce moment dans la mémoire.
EXEMPLE MERGE "CAS 1: PROG 3.ASC"

***MERGE "COM0:"**

FORMAT MERGE "COM0: [(<BLPSC>)" [, R]]
OBJET Spécifier les conditions d'interface de la porte RS – 232C et exécuter la commande MERGE.
EXEMPLE MERGE "COM0": (68N2B)" ,R

MIDS

FORMAT MID$ (<expression chaîne 1>, <n>[, <m>]) = <expression chaîne 2>
où n et m sont des expressions entières et <expression chaîne 1> et <expression chaîne 2> sont des expressions chaînes.
OBJET Remplacer une portion de chaîne par une autre.
EXEMPLE MID$ (AS,2) = "BASIC"

MON

FORMAT MON
OBJET Transférer la commande du programme au moniteur en langage machine.
EXEMPLE MON

MOTOR

FORMAT MOTOR [<commutateur>]
OBJET Mettre en marche ou arrêter le moteur du dérouleur à cassette audio extérieur
EXEMPLE MOTOR ON

***NEW**

FORMAT NEW
OBJET Supprimer le programme se trouvant en mémoire et effacer toutes les variables
EXEMPLE NEW

ON ERROR GOTO

FORMAT ON ERROR GOTO <numéro de ligne>
OBJET Permettre un branchement sur une erreur et spécifier la première ligne du sous-programme de traitement des erreurs.
EXEMPLE ON ERROR GOTO 100

ON...GOSUB/ON...GOTO

FORMAT ON <expression> |GOSUB| <numéro de ligne>[, <numéro de ligne>...]
|GOTO|
OBJET Brancher l'exécution à l'un des numéros de ligne spécifiés.
EXEMPLE ON A GOSUB 100, 200, 300, 400
ON B GOTO 100, 200, 300, 400

OPEN

FORMAT OPEN "mode" , [#] <numéro de fichier>, <descripteur de fichier>
OBJET Ouvrir un fichier spécifié pour E/S.
EXEMPLE OPEN "O" , #1 "CAS0:TEST.BAS"

***OPEN "COM0:"**

FORMAT OPEN "<mode>", [#] <numéro de fichier>, "COM0:[(<BLPSC>)]"
OBJET Spécifier les conditions d'interface d'une porte RS – 232C et exécuter OPEN
EXEMPLE OPEN "0" , #1 , "COM0: (68N2B)".

OPTION BASE

FORMAT OPTION BASE |0|
|1|
OBJET Déclarer la valeur minimale des indices des variables tableaux.
EXEMPLE OPTION BASE 1

***PCOPY**

FORMAT PCOPY <expression>
OBJET Copier un programme BASIC dans une autre zone de programme.
EXEMPLE PCOPY 3

***POKE**

FORMAT POKE <adresse>, <expression numérique>
OBJET Ecrire un octet dans un emplacement spécifié de la mémoire.
EXEMPLE POKE &H0C00, &H39

PRESET

FORMAT PRESET (<coordonnée horizontale>, <coordonnée verticale>)
OBJET Effacer un point sur un écran graphique.
EXEMPLE PRESET (40, 25)

PRINT/LPRINT

FORMAT | PRINT / LPRINT | [<expression>[| , / ; | <expression>...]]
OBJET Sortir des données sur l'écran ou sur la micro-imprimante incorporée.
EXEMPLE PRINT "EPSON"

PRINT USING/LPRINT USING

FORMAT | PRINT / LPRINT | USING <"format chaîne">; [<expression>[| ; / , | <expression>....]]
OBJET Sortir des chaînes ou des valeurs numériques en utilisant un format déterminé
EXEMPLE PRINT USING "####" ; A,B

***PRINT#**

FORMAT PRINT #<numéro de fichier>, [<expression>...]
OBJET Ecrire des données dans un fichier séquentiel.
EXEMPLE PRINT #1, 1, B

PRINT#USING

FORMAT PRINT #<numéro de fichier>, USING <"chaîne de format">;
[<expression>[| ; / , | <expression>...]]
OBJET Ecrire des chaînes et des nombres dans un fichier séquentiel en utilisant un format particulier.
EXEMPLE PRINT #1, USING "###" ; A

PSET

FORMAT PSET (<coordonnée horizontale>, <coordonnée verticale>)[, <couleur>]

OBJET Tracer des points sur un écran graphique spécifié.

EXEMPLE PSET (30,20)

***PUT%**

FORMAT PUT% <numéro d'enregistrement>, <variable>[, <variable>...]

OBJET Ecrire les valeurs des variables dans un fichier RAM

EXEMPLE PUT%0 , A!, B #, C$

RANDOMIZE

FORMAT RANDOMIZE [<expression>]

OBJET Réensemencer le générateur de nombres aléatoires.

EXEMPLE RANDOMIZE

READ

FORMAT READ <variable>[, <variable>...]

OBJET Lire les valeurs d'une instruction DATA et les affecter aux variables.

EXEMPLE READ A, I, C$

REM

FORMAT REM [<remarque>]

OBJET Faire des remarques explicatives qui sont insérées dans un programme.

EXEMPLE REM COMMENT MESSAGE

RENUM

FORMAT RENUM [<Nouveau numéro de ligne>][, [<ancien numéro de ligne>][, [<incrément>]]

OBJET Renuméroter les lignes de programmes.

EXEMPLE RENUM

RESTORE

FORMAT RESTORE [<numéro de ligne>]

OBJET Permettre de relire des instructions DATA à partir d'un point spécifié.

EXEMPLE RESTORE 1000

RESUME

FORMAT RESUME [| NEXT | <numéro de ligne> |]

OBJET Continuer l'exécution d'un programme après exécution d'une procédure de traitement d'erreur.

EXEMPLE RESUME 100

RUN

FORMAT (1) RUN [<numéro de ligne>] ou
(2) RUN <descripteur de fichier> [, R]

OBJET Démarrer l'exécution du programme.

EXEMPLE (1) RUN 300
(2) RUN "CAS0: PROG4.ASC"

RUN "COM0:"

FORMAT RUN "COM0: [(<BLPSC>)]" ['#]

OBJET Spécifier les conditions d'interface de la porte RS-232C, et exécuter RUN.

EXEMPLE RUN"COM0: (68N2B)"

SAVE

FORMAT SAVE <description de fichier> [, A]
V

OBJET Sauvegarder un programme EPSON BASIC sur un fichier spécifié.

EXEMPLE SAVE "CAS0:ABC"

*SAVE "COM0:"

FORMAT SAVE"COM0: [(<BLPSC>)]" , A

OBJET Spécifier les conditions d'interface de la porte RS-232C et exécuter la commande SAVE.

EXEMPLE SAVE"COM0: (68E13)" , A

*SAVEM

FORMAT SAVEM <descripteur de fichier>, <adresse supérieure>, <adresse inférieure>, <adresse de début d'exécution> [, V]

OBJET Sauvegarder le contenu dans la mémoire sur un fichier spécifié.

EXEMPLE SAVEM"CAS1:ABC" , &H0C00, &H0B00, &H0B00

***SCREEN**

FORMAT SCREEN <texte>, <mode graphique>
OBJET Spécifier le mode de l'écran : texte ou graphique.
EXEMPLE SCREEN 0,2

***SCROLL**

FORMAT SCROLL [<vitesse>][, [<mode>][, <pas de défilement>, <pas de défilement Y>]]
OBJET Spécifier le défilement de l'écran matériel.
EXEMPLE SCROLL 9,0,10,4

***SOUND**

FORMAT SOUND <tonalité>, <durée>
OBJET Faire retentir une tonalité spécifiée.
EXEMPLE SOUND 10, 10

***STAT**

FORMAT STAT [| ALL | <expression> |]
OBJET Afficher l'état de chaque zone de programme
EXEMPLE STAT 3

STOP

FORMAT STOP
OBJET Terminer l'exécution d'un programme et retourner au niveau des commandes.
EXEMPLE STOP

SWAP

FORMAT SWAP <variable 1>, <variable 2>
OBJET Echanger les valeurs de deux variables
EXEMPLE SWAP A$, B$

***TITLE**

FORMAT TITLE <nom de programme>
OBJET Nommer les programmes
EXEMPLE TITLE "TEST1"

TRON/TROFF

FORMAT TRON
TROFF

OBJET Suivre l'exécution des instructions du programme.

EXEMPLE TRON
TROFF

*WIDTH

FORMAT WIDTH <caractères par ligne>, <nombre de lignes>[, <marge de défilement>]

OBJET Fixer la dimension de l'écran virtuel.

EXEMPLE WIDTH 20, 25, 5

WIDTH <nom de dispositif>

FORMAT WIDTH | "LPT0:" / "COM0:" | ,<nombre de chiffres>

OBJET Régler la largeur d'impression de l'imprimante.

EXEMPLE WIDTH "LPT0:", 20

WIND

FORMAT WIND [<valeur du compteur>]

OBJET Commander le moteur de la microcassette pour le déroulement rapide et le réenroulement.

EXEMPLE WIND 0

FONCTIONS

ABS

FORMAT ABS (< expression numérique >)
OBJET Renvoyer la valeur absolue d'une expression numérique.
EXEMPLE A = ABS (– 1.6)

ASC

FORMAT ASC (<chaîne >)
OBJET Renvoyer le code de caractère d'un caractère
EXEMPLE A = ASC ("A")

ATN

FORMAT ATN (<expression numérique >)
OBJET Renvoyer l'arc tangente d'une expression numérique.
EXEMPLE A = ATN (0.5)

CDBL

FORMAT CDBL (<expression numérique >)
OBJET Convertir les entiers et les nombres simple précision en nombres double précision.
EXEMPLE A# = CDBL (B! /2)

CHR$

FORMAT CHR$ (<expression numérique >)
OBJET Renvoyer le caractère correspondant à un code de caractères spécifiés.
EXEMPLE A$ = CHR$ (&H41)

CINT

FORMAT CINT (<expression numérique >)
OBJET Convertir des nombres simple et double précisions en entiers.
EXEMPLE A% = CINT (B# /2)

COS

FORMAT COS (<expression numérique >)
OBJET Renvoyer le cosinus d'une expression numérique
EXEMPLE A = COS (3.1415926/2)

CSNG

FORMAT CSNG (<expression numérique >)

OBJET Convertir des entiers et des nombres double précision en des nombres simple précision.

EXEMPLE A! = CSNG (B#)

CSRLIN

FORMAT CRSLIN

OBJET Renvoyer la position verticale du curseur sur l'écran virtuel.

EXEMPLE Y– CSRLIN

DATE$

FORMAT DATE$ [= MM/DD/YY]

OBJET Introduire la date en cours et renvoyer la date gardée par l'horloge du calendrier interne.

EXEMPLE PRINT DATE$

DAY

FORMAT DAY [= <numéro du jour >]

OBJET Introduire le jour de la semaine et afficher le jour de la semaine gardé par l'horloge du calendrier interne.

EXEMPLE PRINT DAY

EOF

FORMAT EOF (<numéro de fichier >)

OBJET Renvoyer le code de fin de fichier

EXEMPLE IF EOF (3) THEN CLOSE #1 ELSE GOTO 100

ERL/ERR

FORMAT ERL
ERR

OBJET Renvoyer le code d'erreur d'une erreur qui s'est produite et le numéro de la ligne sur laquelle l'erreur s'est produite.

EXEMPLE A = ERL
B = ERR

EXP

FORMAT EXP (<expression numérique>)
OBJET Renvoyer la valeur d'une fonction exponentielle de base
EXEMPLE A = EXP(1)

FIX

FORMAT FIX (<expression numérique>)
OBJET Renvoyer la partie entière tronquée d'une expression numérique.
EXEMPLE A = FIX (–B/3)

FRE

FORMAT FRE (<expression>)
OBJET Renvoyer la dimension d'une zone de mémoire inutilisée.
EXEMPLE PRINT FRE (0)
PRINT FRE ("AS")

HEX$

FORMAT HEX$ (<expression numérique>)
OBJET Renvoyer une chaîne qui représente la valeur hexadécimale de l'argument décimal.
EXEMPLE A$ = HEX$ (65535)

INKEY$

FORMAT INKEY$
OBJET Renvoyer une chaîne à un caractère de la touche de caractères sur laquelle vous avez appuyé ou une chaîne nulle si vous n'avez appuyé sur aucune touche de caractère.
EXEMPLE A$ = INKEY$

INPUT$

FORMAT INPUT$ (<nombre de caractères> [, [# [<numéro de fichier>)
OBJET Renvoyer une chaîne de caractères lue dans un fichier spécifié.
EXEMPLE A$ = INPUT$ (5, #3)

INSTR

FORMAT INSTR ([< expression numérique > ,] < chaîne 1 > , < chaîne 2 >)
OBJET Rechercher la première apparition d'une chaîne dans une autre chaîne et renvoyer la position de la chaîne recherchée.
EXEMPLE B = INSTR (A$, "XYZ")

INT

FORMAT INT (<expression numérique >)
OBJET Renvoyer la valeur entière la plus grande (tronquée).
EXEMPLE PRINT INT (–B/3)

LEFT$

FORMAT LEFT$ (<chaîne >, <expression numérique >)
OBJET Renvoyer une longueur arbitraire de chaîne à partir des caractères les plus à gauche d'une chaîne.
EXEMPLE B$ = LEFT$ (A$, 4)

LEN

FORMAT LEN (<chaîne >)
OBJET Renvoyer le nombre total des caractères contenus dans une chaîne.
EXEMPLE A = LEN (A$)

LOF

FORMAT LOF (<numéro de fichier >)
OBJET Renvoyer la dimension d'un fichier spécifié
EXEMPLE A = LOF (3)

LOG

FORMAT LOG (<expression numérique >)
OBJET Renvoyer le logarithme népérien d'une expression numérique.
EXEMPLE PRINT LOG (2.7812818)

MID$

FORMAT MID$ (<chaîne >, <expression 1 > [, <expression 2 >])
OBJET Renvoyer une longueur arbitraire de chaîne à partir d'une chaîne
EXEMPLE B$ = MID$ (A$, 2, 3)

OCT$

FORMAT OCT$ (<expression numérique>)
OBJET Renvoyer une chaîne qui représente la valeur octale de l'argument décimal.
EXEMPLE PRINT OCT$ (123 +456)

*PEEK

FORMAT PEEK (<adresse>)
OBJET Renvoyer l'octet lu dans un emplacement spécifié de la mémoire.
EXEMPLE A = PEEK (&H0C00)

POINT

FORMAT POINT (<coordonnée horizontale>, <coordonnée verticale>)
OBJET Renvoyer l'état d'un point situé à un emplacement précisé de l'écran graphique
EXEMPLE PRINT POINT (100, 10)

POS

FORMAT POS (<nombre>)
OBJET Renvoyer la position horizontale du curseur sur l'écran virtuel ou la position horizontale de la tête d'imprimante.
EXEMPLE X = POS (0)

RIGHT$

FORMAT RIGHT$ (<chaîne>, <expression numérique>)
OBJET Renvoyer une longueur de chaîne arbitraire à partir des caractères les plus à droite d'une chaîne.
EXEMPLE PRINT RIGHT$ ("ABC", 3)

RND

FORMAT RND (<expression numérique>)
OBJET Renvoyer un nombre aléatoire
EXEMPLE A = RND (1)

SGN

FORMAT SGN (<expression numérique>)
OBJET Renvoyer le signe de la valeur d'une expression numérique.
EXEMPLE B = SGN (A)

SIN

FORMAT SIN (<expression numérique>)
OBJET Renvoyer le sinus d'une expression numérique.
EXEMPLE PRINT SIN (3.1415926/2)

SPACE$

FORMAT SPACE$ (<expression numérique>)
OBJET Renvoyer une chaîne d'espaces de longueur spécifiée.
EXEMPLE A$ = "A" + SPACE$ (10) + "C"

SPC

FORMAT SPC (<nombre>)
OBJET Sortir un nombre spécifié de blancs.
EXEMPLE PRINT SPC (10) ; "A"

SQR

FORMAT SQR (<expression numérique>)
OBJET Renvoyer la racine carrée d'une expression numérique.
EXEMPLE A = SQR (2)

STR$

FORMAT STR$ (<expression numérique>)
OBJET Renvoyer une représentation chaîne de la valeur d'une <expression numérique>
EXEMPLE A$ = STR$ (123)

STRING$

FORMAT STRING$ (<expression entière>, | <expression chaîne> | <expression numérique> |)
OBJET Renvoyer une chaîne de caractères spécifiés.
EXEMPLE PRINT STRING$ (10, 65)

***TAB**

FORMAT TAB (<expression numérique>)
OBJET Se déplacer d'un certain nombre d'espaces sur la ligne où se trouve le curseur.
EXEMPLE PRINT TAB (10) ; "ABC"

TAN

FORMAT TAN (<expression numérique>)
OBJET Renvoyer la tangente d'une expression numérique.
EXEMPLE A = TAN (3.1416/4)

*TAPCNT

FORMAT TAPCNT
OBJET Renvoyer la valeur du compteur d'entraînement de la microcassette.
EXEMPLE PRINT TAPCNT
A = TAPCNT

TIMES

FORMAT TIMES [= "HH:MM:SS"]
OBJET Lire l'heure gardée par l'horloge du calendrier interne.
EXEMPLE PRINT TIMES

USR

FORMAT USR [<chiffre>] (<argument>]
OBJET Appeler un sous-programme en langage machine défini par l'instruction DEFUSR
EXEMPLE A = USR 1 (B)

VAL

FORMAT VAL (<expression chaîne>)
OBJET Retourner la valeur numérique d'une expression chaîne.
EXEMPLE A = VAL ("–123")

VARPTR

FORMAT VARPTR (<nom variable>)
OBJET Renvoyer l'adresse d'une variable ou d'un réseau.
EXEMPLE PRINT HEX$ (VARPTR (A))

EPSON OVERSEAS MARKETING LOCATIONS

EPSON AMERICA, INC.
2780 Lomita Blvd., Torrance, Calif. 90505, U.S.A.
Phone: (213)539-9140
Telex: 182412

EPSON DEUTSCHLAND GmbH
Zulpicher Strasse 6, 4000 Düsseldorf 11, F.R. Germany
Phone: (0211)56030
Telex: 8584786

EPSON (UK) LTD.
Dorland House, 388 High Road, Wembley, Middlesex, HA9 6UH, U.K.
Phone: (01)902-8892
Telex: 8814169

EPSON FRANCE S.A.
55, rue Deguingand, 92300, Levallois-Perret, France
Phone: (1)739-6770
Telex: 614202

EPSON AUSTRALIA PTY. LTD.
Unit 3, 17 Rodborough Road, Frenchs Forest, NSW 2086, Australia
Phone: (02)452-5222
Telex: 75052

EPSON ELECTRONICS (SINGAPORE) PTE. LTD.
No. 1 Maritime Square, #02-19, World Trade Centre Singapore 0409
Phone: 2786071/2
Telex: 39536

EPSON ELECTRONICS TRADING LTD.
30/F, Far East Finance Centre
Harcourt Road, Central, Hong Kong
Phone: 5-282555
Telex: 65542

EPSON ELECTRONICS TRADING LTD. (TAIWAN BRANCH)
1, 8F K.Y. Wealthy Bldg. 206, Nanking, E. Road, Sec. 2, Taipei, Taiwan R.O.C.
Phone: 536-4339; 536-3567
Telex: 24444

EPSON CORPORATION
80 Hirooka, Shiojiri-shi, Nagano 399-07
Japan
Phone: (0263)52-2552
Telex: 3342-214

EPSON

Printed in Japan 85.07-3