

SHARP

ORDINATEUR DE POCHE

MODELE **PC-1500A**

MANUEL D'INSTRUCTION



INTRODUCTION

Permettez-nous d'abord de vous remercier d'avoir choisi l'ordinateur de poche SHARP PC-1500A. Nous sommes persuadés que vous aurez beaucoup de plaisir à vous servir quotidiennement de ce nouvel ami qui, quoique de petite taille, n'en est pas moins puissant. Le PC-1500A constitue l'un des modèles d'ordinateurs de poche les plus élaborés du monde. Bien qu'il ait beaucoup de points communs avec son cousin, l'ordinateur de poche SHARP PC-1211, le PC-1500A vous offre des capacités bien plus avancées, comme, par exemple:

- Un affichage LCD à points programmables de 7 à 156.
- Un générateur de sons destiné à créer des effets spéciaux par de programme.
- Un jeu de caractères ASCII majuscules et minuscules.
- Fonctions scientifiques et mathématiques.
- Touches de fonction définissable par l'utilisateur.
- Une version étendue de BASIC fournissant des tableaux bidimensionnels, chaînes à longueur variable, commandes de graphismes, enchaînement de programmes et nombre d'autres caractéristiques avancées.
- Jusqu'à 16K octets de RAM facultatif.
- Une Interface/Imprimante et Cassette facultative permettant le traçage X-Y en 4 couleurs, la mémorisation de données et de programmes et leur impression avec les caractères d'une taille au choix parmi les 9 disponibles.

Cet appareil est capable d'exécuter les multiples fonctions qui, il n'y a pas si longtemps encore, aurait rempli un entrepôt de tubes, de fils et d'ingénieurs. Mais il n'est pas nécessaire d'être ingénieur pour utiliser un appareil si élaboré. Bien au contraire, le PC-1500A et ce manuel ont été étudiés pour aider le débutant à accéder à cette nouvelle technologie.

Nous avons divisé ce manuel en cinq parties principales de manière à vous rendre compétent en très peu de temps. Les utilisateurs plus avancés peuvent explorer les possibilités du PC-1500A au moyen des parties "Programmation avancée", "Calculations avancées" et "Appendices".

Le langage de ce manuel est celui de la conversation et de nombreux exemples sont fournis. Mais ne vous fiez pas à nos promesses, passez tout de suite au chapitre 0 et vérifiez de par vous-même la facilité avec laquelle on peut débiter. Avant toute chose, assurez-vous que les piles ont été placées dans l'appareil. Si elles ne l'ont pas été, l'appendice B vous fournira les instructions nécessaires pour le faire.

Avant toute chose, amusez-vous bien et n'hésitez pas à expérimenter!

LES INSTRUCTIONS POUR L'IMPRIMANTE ET LE MAGNETO PHONE
LES INSTRUCTIONS POUR L'IMPRIMANTE ET LE MAGNETOPHONE DECRITES
CI-APRES NE SONT DISPONIBLES QUE SUR L'IMPRIMANTE OPTIONNELLE
CE-150 (AVEC INTERFACE A CASSETTE INCORPOREE). L'ORDINATEUR N'ETANT
PAS EQUIPE DE CES INSTRUCTIONS, LEUR UTILISATION N'EST POSSIBLE
QUE LORSQUE L'ORDINATEUR EST CONNECTE A LA CE-150. PAR CON-
SEQUENT, VEILLEZ A CONNECTER L'ORDINATEUR SUR LA CE-150 POUR
PROGRAMMER EN UTILISANT CES INSTRUCTIONS.

REMARQUES SUR LE FONCTIONNEMENT

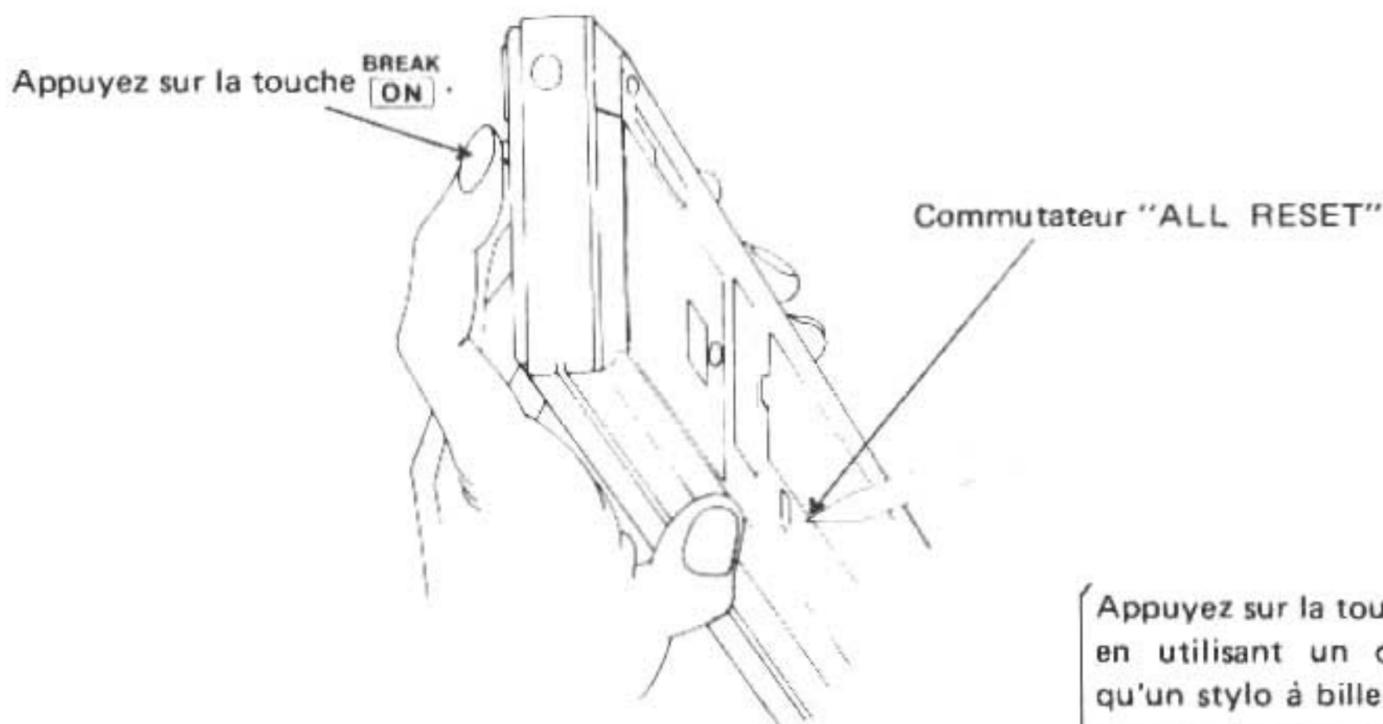
L'affichage LCD de l'ordinateur étant en verre, il est nécessaire de le manipuler avec quelques précautions. **NE PLACEZ PAS** le PC-1500A dans la poche arrière de vos pantalons, car vous risqueriez de l'abîmer quand vous vous asseyez.

Pour assurer une utilisation sans problèmes de votre ordinateur de poche SHARP, nous vous recommandons de :

1. Protéger l'ordinateur des changements extrêmes de températures, de l'humidité et de la poussière.
2. Nettoyer l'ordinateur avec un linge doux et sec. Evitez d'utiliser des solvants, de l'eau ou des linges mouillés.
3. Oter les piles, si vous savez que vous n'allez pas utiliser votre ordinateur pendant longtemps, afin d'éviter des dégâts provoqués par des fuites de liquide des piles.
4. Retourner l'ordinateur uniquement à un centre de réparations autorisé SHARP, quand cela s'avère nécessaire.
5. Garder ce manuel pour le cas où vous auriez besoin de vous y reporter dans le futur.

Localisation des Pannes

Des bruits extérieurs forts ou des chocs durant le fonctionnement de l'appareil peuvent rendre inopérables toutes les touches, y compris la touche **BREAK ON**. Au cas échéant, appuyez sur le commutateur ALL RESET situé sur le dos de l'appareil pendant environ 15 secondes tout en maintenant la touche **BREAK ON** en position basse.



Appuyez sur la touche ALL RESET en utilisant un objet pointu tel qu'un stylo à bille. Ne vous servez pas d'un objet dont la pointe est cassable ou aigüe.

Puis, après vous être assuré que l'affichage indique **NEW0? : CHECK**, (Si NEW0? : CHECK n'est pas affiché, répétez les opérations précédentes.) appuyez sur les touches **CL** **N** **E** **W** **0** **ENTER**. (Note: 0 = zéro, une touche numérique).

Puis, appuyez sur les touches **SHIFT** **MODE** **N** **E** **W** **ENTER**.

Evitez d'utiliser le commutateur ALL RESET hors des cas où les pannes décrites ci-dessus se produisent, car il effectue la destruction des programmes, et du contenu de la réserve.

SPECIFICATIONS DU PC-1500A

Modèle:	Ordinateur de poche PC-1500A																								
Nombres de chiffres de calcul:	10 chiffres (mantisse) + 2 chiffres (exposant)																								
Système de calcul:	Selon la formule mathématique (avec fonction déterminant les priorités)																								
Langage de programme:	BASIC																								
Capacité:	<table> <tr> <td>CPU:</td> <td>CMOS 8 bit</td> </tr> <tr> <td>Système ROM:</td> <td>16 K octets</td> </tr> <tr> <td>Capacité de mémoire adressable (RAM):</td> <td>8,5 K octets</td> </tr> <tr> <td> Système:</td> <td>1,9 K octets</td> </tr> <tr> <td> Tampon d'entrée:</td> <td>80 octets</td> </tr> <tr> <td> Pile:</td> <td>196 octets</td> </tr> <tr> <td> Zone libre du langage machine:</td> <td>1023 octets (7C01H ~ 7FFFH)</td> </tr> <tr> <td> Autres:</td> <td></td> </tr> <tr> <td> Aire de l'utilisateur:</td> <td>6,6 K octets</td> </tr> <tr> <td> Aire de mémoire fixe: (A~Z, A\$~Z\$)</td> <td>624 octets</td> </tr> <tr> <td> Aire de données de programme de base:</td> <td>5946 octets</td> </tr> <tr> <td> Aire de réserve:</td> <td>188 octets</td> </tr> </table>	CPU:	CMOS 8 bit	Système ROM:	16 K octets	Capacité de mémoire adressable (RAM):	8,5 K octets	Système:	1,9 K octets	Tampon d'entrée:	80 octets	Pile:	196 octets	Zone libre du langage machine:	1023 octets (7C01H ~ 7FFFH)	Autres:		Aire de l'utilisateur:	6,6 K octets	Aire de mémoire fixe: (A~Z, A\$~Z\$)	624 octets	Aire de données de programme de base:	5946 octets	Aire de réserve:	188 octets
CPU:	CMOS 8 bit																								
Système ROM:	16 K octets																								
Capacité de mémoire adressable (RAM):	8,5 K octets																								
Système:	1,9 K octets																								
Tampon d'entrée:	80 octets																								
Pile:	196 octets																								
Zone libre du langage machine:	1023 octets (7C01H ~ 7FFFH)																								
Autres:																									
Aire de l'utilisateur:	6,6 K octets																								
Aire de mémoire fixe: (A~Z, A\$~Z\$)	624 octets																								
Aire de données de programme de base:	5946 octets																								
Aire de réserve:	188 octets																								
Calculs:	Quatre calculs arithmétiques, calcul de puissance, fonctions trigonométriques et trigonométriques inverses, fonctions logarithmiques et exponentielles, conversion d'angles, extraction de la racine carrée, fonction de signe, absolus, entiers et calculs logiques.																								
Fonction de mise au point:	Décalage de curseur (▶ ◀) Insertion (INS) Elimination (DEL) Revue de ligne (↓ , ↑)																								
Protection de la mémoire:	Unité CMOS de soutien de piles (protège les mémoires de programme, de données et de réserve)																								
Affichage:	Crystal liquide Largeur de 26 caractères Graphismes à points 7 x 156																								
Touches:	65 Touches, y compris: Alphabétiques Numériques de fonction définissable par l'utilisateur Pré-programmées																								
Source d'alimentation:	6,0V, c.c. 4 piles sèches de type UM-3, AA ou R6																								
Consommation de courant:	6,0V, c.c: 0,13W																								
Durée de fonctionnement:	Approx. 50 heures lorsque 555555555555. est en affichage continu à une température d'utilisation de 20°C. Cette durée peut varier légèrement selon le mode d'utilisation, etc. <ul style="list-style-type: none"> • A raison d'une heure par jour, il est possible d'utiliser la batterie pendant 40 jours environ. Ceci est vrai pour une heure d'utilisation comprenant 10 minutes de calcul ou d'exécutions de programmes et 50 minutes d'affichage. 																								
Température de fonctionnement:	0° à ~ 40°C																								
Dimensions:	Longueur: 195 mm, largeur: 86 mm, épaisseur: 25,5 mm																								
Poids:	Environ 375 g (avec piles)																								
Accessoires:	Housse, quatre piles sèches, deux gabarits de clavier, étiquette pour le nom du propriétaire et manuel d'instruction																								
En option:	Interface imprimante/cassette (CE-150) Module d'expansion de mémoire (CE-151 du type à prise et RAM à 4K octets, CE-155/159 du type à prise et RAM à 8K octets, CE-161 du type à prise et RAM à 16K octets)																								

TABLE DES MATIERES

	Page
Introduction	1
Remarques sur le Fonctionnement	2
Spécifications du PC-1500A.	3
Table des matières	4
0. Programmation instantanée	8
A. Exemple 1	8
B. Exemple 2	10
I. Faisons connaissance	12
A. Touches ON et OFF (marche et arrêt)	12
B. Touches alphabétiques	12
C. Touches numériques	12
D. Touche SHIFT	13
E. Les minuscules et la touche SMALL	13
F. L'affichage.	13
G. Le curseur et le symbole de guidage	14
H. La touche de remise à zéro	14
I. Touche ENTER	14
J. Les messages d'erreur	14
K. L'indicateur de charge des piles	15
L. Utilisation de la touche RCL (rappel)	15
II. Plonger dans l'eau froide	16
A. La touche MODE	16
B. Calculs simples	16
C. Calculs en séries	17
D. Calculs avec des nombres négatifs	17
E. Calculs composés.	18
F. L'utilisation des parenthèses	18
G. Comparaisons logiques et fonctions logiques.	19
H. Les touches et les caractéristiques de mise en page	20
H.1. Touche flèche vers la gauche/DEL (effacement)	20
H.2. Touche flèche vers la droite/INSérer	21
H.3. Fonction de rappel	22
I. Les variables.	22
J. Une pause	25
RESUME.	26
III. L'art mystérieux de la programmation.	27
A. Qu'est-ce qu'un programme?	27
B. Comment bâtir un programme?	28
C. COMMANDES et INSTRUCTIONS	29
D. Numéros de ligne.	29
E. Touches de revue des lignes de programme	30
F. Un examen plus détaillé de certains vieux amis	30
F.1.A La commande NEW	30
F.1.B L'instruction POKE et POKE #	31
F.1.C L'instruction PEEK et PEEK #	31
F.1.D L'instruction CALL	31

F.2.	L'instruction LET	32
F.3.	L'instruction PRINT	32
G.	L'instruction PAUSE	37
H.	L'instruction INPUT	38
I.	Raccourcis et conseils utiles.	43
I.1.	Abréviations.	43
I.2.	Instructions multiples utilisant le double-point	44
J.	La correction d'erreurs sur le mode PROgramme	45
K.	La commande LIST	46
L.	Plus on est de fous, plus on rit	47
L.1.	L'instruction END.	47
L.2.	Numéro des lignes RUN	47
M.	Instructions de contrôle	48
N.	IF THEN	48
O.	GOTO.	49
P.	FOR NEXT.	55
Q.	WAIT	58
R.	READ, DATA, RESTORE	59
S.	REM.	62
T.	GOSUB et RETURN	62
	Résumé des caractéristiques de mise en page sur le mode PROgramme.	64
IV.	Calculs avancés	64
A.	Notation Scientifique.	64
B.	Gamme des calculs: débordement supérieur et inférieur	66
C.	Les racines, les puissances et pi.	67
D.	Les modes angulaires	69
E.	Les fonctions trigonométriques	69
	SIN, COS, TAN, ASN, ACS, ATN	
F.	Les fonctions logarithmiques	70
	LN, LOG, EXP	
G.	Conversion d'angle.	70
	DEG, DMS	
H.	Fonctions diverses	71
	ABS, INT, SGN	
V.	Programmation avancée	72
A.	Tableaux et variables indicées	72
	DIM	
B.	Plus de détails sur les chaînes des caractères	75
B.1.	Chaînes bidimensionnelles.	75
B.2.	Enchaînement	76
B.3.	Comparaison de chaînes	77
C.	Fonctions	78
C.1.	ASC	78
C.2.	CHR\$.	79
C.3.	INKEY\$.	80
C.4.	LEN	80
C.5.	LEFT\$.	81
C.6.	MID\$.	81
C.7.	RIGHT\$.	82
C.8.	RND.	83
C.9.	RANDOM	83
C.10.	STR\$.	84

C.11.	STATUS	84
C.12.	TIME	85
C.13.	VAL	86
D.	PRINT USING	86
	Tableau des caracteres de mise en forme	87
E.	Transfert de contrôle calculé	89
	ON GOTO, ON GOSUB, ON ERROR GOTO	89, 90
F.	Programmation de l'affichage.	91
F.1.	BEEP	91
F.2.	CURSOR	92
F.3.	CLS	94
F.4.	GCURSOR	95
F.5.	GPRINT	97
F.6.	POINT	100
G.	Détection d'erreurs	102
	TRON, TROFF, touches à flèches	102
H.	Nombres hexadécimaux et fonctions de BOOLE	104
H.1.	Spécification hexadécimale	104
H.2.	La Fonction AND (et)	104
H.3.	La Fonction OR (ou)	105
H.4.	La Fonction NOT	105
I.	Pour arrêter l'exécution d'un programme	106
	STOP, CONT	106
J.	Contrôle du mode	106
	LOCK, UNLOCK (Bloquer, débloquer)	106
VI.	Elargissement du PC-1500A.	107
A.	L'INTERFACE IMPRIMANTE/CASSETTE	107
A.1.	Connexion de l'ordinateur sur l'interface	107
A.2.	Alimentation	109
A.3.	Raccordement d'un magnétophone avec l'interface	109
A.4.	Chargement du papier	111
A.5.	Remplacement des stylos	112
B.	Utilisation du magnétophone à cassette	114
B.1.	Opération du magnétophone	114
B.2.	Enregistrement des programmes sur bande magnétique (CSAVE)	114
B.3.	Rechargement de programmes enregistrés sur bande magnétique (CLOAD, CLOAD?).	115
B.4.	Enregistrement et rechargement de données sur et à partir de bandes magnétiques (PRINT#, INPUT#)	116
B.5.	La commande MERGE	118
B.6.	Enchaînement de programmes (CHAIN)	120
B.7.	Utilisation de deux magnétophones.	121
C.	Utilisation de l'imprimante	123
C.1.	Spécifications de l'imprimante CE-150	123
C.2.	La commande TEST	123
C.3.	L'impression de calculs.	124
C.4.	Modes de l'imprimante	125
C.5.	Listage du programme	125
C.6.	Contrôle programmable de l'imprimante	127
C.6.1.	Csize	127

C.6.2.	ROTATE	127
C.6.3.	COLOR	128
C.6.4.	LF	128
C.6.5.	LPRINT	129
C.6.6.	LCURSOR	130
C.6.7.	TAB	130
C.6.8.	SORGN	130
C.6.9.	GLCURSOR	132
C.6.10.	LINE	132
C.6.11.	RLINE	134
VII.	Le mode RESERVE	137
A.	Definition et sélection des touches de réserve	137
B.	Identification des touches de réserve	138
C.	Effacement des programmes de réserve	139
VIII.	Démarrage de l'exécution du programme	140
A.	La touche DEF	140
A.1.	Passages des programmes DEFInissables	140
A.2.	Mots-clés pré-affectés	141
A.3.	L'instruction AREAD	141
B.	Démarrage automatique de programme ARUN	142
C.	Commparaison des méthodes de démarrage	143
C.1.	L'aire de mémoire fixe	143
C.2.	Tableau de comparaison des méthodes de démarrage de programme	144
IX.	Appendices	147
A.	Abréviations	147
B.	Remplacement des piles du PC-1500A	151
C.	Séquence de collation	153
E.	Messages d'erreurs	154
F.	Lectures supplémentaires	160
O.	Ordre d'évaluation d'expressions	160
X.	Différences entre les commandes du PC-1211 et celles du PC-1500A	162
Commandes particulieres AU PC-1500A		163
Z.	Table de référence des commandes	164

Etiquette

Ecrivez votre nom sur l'étiquette ci-jointe et collez-la sur le dos de l'ordinateur.

PROGRAMMATION INSTANTANEE

(Pas besoin d'ajouter de l'eau)

Cette partie est destinée uniquement à une minorité (les auteurs inclus) dont la curiosité l'emporte sur la patience (et peut-être même le bon sens). A ceux parmi vous qui veulent absolument FAIRE quelque chose avec ce produit miraculeux de l'électronique moderne, nous allons présenter un exercice de programmation simple. (AVERTISSEMENT aux timides et aux craintifs: passez au Chapitre 1; "Faisons connaissance", pour une introduction plus tranquille et minutieuse au SHARP PC-1500A.)

Avant de continuer, nous jugeons nécessaire de vous avertir qu'il est important de **passer les étapes dans l'ordre donné**. Contrairement à ce que beaucoup de gens pensent, les ordinateurs ne sont pas des "super-cerveaux." Ils ne peuvent pas deviner ce que vous désirez comme le peut le cerveau humain moyen. Le PC-1500A attend simplement vos ordres et les exécute.

Etes-vous prêts? Bon! On commence!

Exemple 1

Trouvez d'abord la touche "ON" située dans le coin supérieur droit du clavier. En appuyant sur cette touche, vous réveillez le génie électronique qui sommeille (ne vous attendez pas quand même à voir jaillir une colonne de fumée!). La partie affichage de l'ordinateur devrait correspondre à l'illustration ci-dessous:



Appuyez sur la touche **MODE** située à l'extrême droite jusqu'à ce que l'abréviation **PRO** apparaisse dans le coin supérieur droit de l'affichage. (Si vous avez pressé la touche trop souvent, pressez-la de nouveau jusqu'à ce que vous obteniez le résultat désiré). Le SHARP PC-1500A est maintenant prêt à accepter les séries de directives qui constituent un programme d'ordinateur.

Appuyez sur les touches suivantes dans l'ordre donné:

1 **0** **A** **=** **1** **ENTER**

Remarquez que, lorsque vous appuyez sur la touche **ENTER**, l'ordinateur modifie ce que vous avez entré. L'affichage devrait maintenant être comme ci-dessous:



Note: Tout au long de ce manuel nous utiliserons le symbole **0** pour le zéro de façon à le distinguer plus aisément de la lettre **O**.

Pour continuer, appuyez maintenant sur les touches indiquées ci-dessous. Ne vous effrayez pas de voir chaque ligne disparaître au fur et à mesure que vous écrivez la suivante.

2 **0** **P** **A** **U** **S** **E** **A** **ENTER**

3 **0** **A** **=** **A** **+** **2** **ENTER**

4 **0** **G** **O** **T** **O** **2** **0** **ENTER**

A ce point-ci, vous aurez terminé votre premier programme. Maintenant il s'agit de commander à l'ordinateur "d'exécuter" les ordres qu'il a emmagasinés. On appelle cette opération l'"exécution" et on l'accomplit sur le mode RUN. Appuyez de nouveau sur la touche **MODE** et les lettres RUN remplaceront les lettres PRO au sommet de l'affichage.

Une dernière étape: écrivez les lettres **R** **U** **N** et appuyez sur **ENTER**.

Félicitations! Votre premier programme BASIC est maintenant en cours d'exécution. L'ordinateur suit vos directives et dresse la liste de tous les nombres impairs positifs, dans l'ordre.

"Ah! Je suis génial!" Mais... "quand est-ce que ça va s'arrêter?"

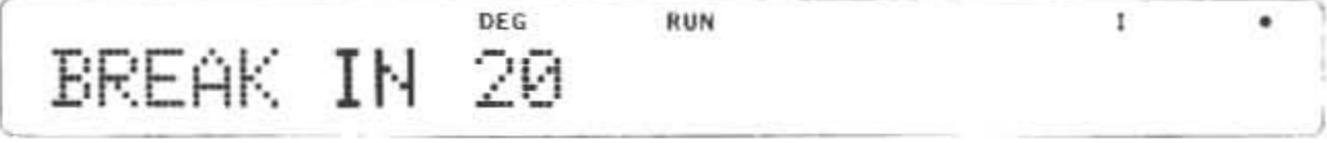
Hé bien . . . malheureusement, ce programme ne se terminera pas, à moins que vous n'interveniez ou que les piles ne se déchargent. Revoyons le programme pour en comprendre la raison:

```
10 A = 1
20 PAUSE A
30 A = A + 2
40 GOTO 20
```

L'action de la ligne numéro 40 est de faire reproduire par l'ordinateur toutes les lignes qui suivent la ligne 20. Cela inclut, bien sûr, la ligne 40 qui, comme nous le savons déjà, ordonne à l'ordinateur de re-reproduire les lignes 20, 30 et 40 qui . . . et cela à perpétuité. Cette répétition sans fin s'appelle "une boucle" dans le jargon des ordinateurs.

Notre programme est bloqué dans une "boucle infinie" et vous, vous seulement, avez le pouvoir de mettre fin à ce tragique gaspillage de courant. Pour ce faire, appuyez sur la touche **ON**. Comme le PC-1500A est déjà en état de marche, vous avez en fait mis en branle la fonction BREAK (briser). Inutile de vous mettre à couvert! Contrairement à ce que ce mot semble indiquer, ce n'est pas une touche d'auto-destruction! Pour mettre les choses au clair, vous ne pouvez en aucun cas endommager l'ordinateur en appuyant simplement sur des touches. N'hésitez donc pas à expérimenter!

Après avoir appuyé sur la touche BREAK, un message semblable à celui ci-dessous apparaîtra sur l'affichage:



The screenshot shows a rectangular display area with a dark background. At the top, the words 'DEG' and 'RUN' are visible, with a small 'I' and a dot to the right. The main text 'BREAK IN 20' is displayed in a large, pixelated font.

Ceci vous apprend quelle instruction était en cours d'exécution quand vous avez interrompu l'ordinateur. Appuyez à nouveau sur la touche BREAK et l'ordinateur sera prêt à accepter d'autres instructions:



The screenshot shows a rectangular display area with a dark background. At the top, the words 'DEG' and 'RUN' are visible, with a small 'I' and a dot to the right. A large, pixelated right-pointing arrow is displayed on the left side of the screen.

Pour ceux d'entre vous qui viennent de se souvenir qu'il est presque l'heure de rentrer pour dîner, le moment est bon pour s'arrêter. (Avant de partir et pour économiser le courant des piles, appuyez sur la touche **OFF** s'il vous plaît). Les autres qui sont déjà en train de devenir des mordus de la programmation veulent probablement continuer leur éducation à l'aide du deuxième exemple. (Nous nous refusons cependant de prendre la responsabilité des conséquences d'un dîner manqué ou d'un rendez-vous oublié.)

Exemple 2

Pour débiter notre second programme, il est nécessaire d'entrer sur le mode de programmation. Pour ce faire, appuyez sur la touche MODE jusqu'à ce que les lettres PRO (abréviation de programme) remplacent les lettres RUN au sommet de l'affichage. Le PC-1500A va nous permettre maintenant d'entrer un nouveau programme ou de modifier l'ancien. Puisque notre nouveau programme ne va pas être bâti sur les fondations de l'ancien, nous devons d'abord effacer les instructions de ce dernier de la mémoire de l'ordinateur. Pour accomplir cette tâche, écrivez d'abord le mot NEW sur la machine et appuyez sur la touche **ENTER**. Après un court moment le symbole de guidage > va réapparaître.

Ecrivez les caractères suivants pour entrer la première ligne du programme:

```
1 0 I N P U T  SHIFT  "  L  O  N  G  U  E  U  R  SPACE
L  I  S  T  E  SHIFT  /  SHIFT  "  SHIFT  +  A  ENTER
```

Remarquez que lorsque vous appuyez sur la touche **SHIFT** avant d'appuyer sur la touche-caractère au haut de laquelle est inscrit un autre caractère, le caractère du haut sera entré. La touche **SHIFT** (de décalage) permet d'utiliser la même touche pour exprimer deux fonctions différentes et c'est pour cela qu'on la nomme parfois touche de "seconde fonction". Ainsi, par exemple, dans la première ligne de notre programme (ligne 10 ci-dessus), quand on a appuyé sur la touche **SHIFT** avant d'appuyer sur la touche marquée **+**, le **:** (point-virgule) sera entré. La totalité de la ligne sera entrée dans l'ordinateur de la façon suivante:

```
10 INPUT DEG "LONGUEUR PRO LISTE?" I*
```

Dans ce manuel-ci, nous illustrerons la sélection du caractère de seconde fonction à l'aide de la touche de décalage (SHIFT) et du caractère désiré. Par exemple, la ligne 10 ci-dessus va être affichée de la manière suivante:

```
1 0 I N P U T  SHIFT  "  L  O  N  G  U  E  U  R  SPACE
L  I  S  T  E  SHIFT  ?  SHIFT  "  SHIFT  ;  A  ENTER
```

Complétez l'entrée de notre second programme en appuyant sur les touches indiquées ci-dessous:

```
2 0 I F A  SHIFT  <  =  0  G  O  T  O  9  9  ENTER
3 0 F O R I = 1 T O A  ENTER
4 0 P A U S E I  SHIFT  ,  I  *  I  ENTER
5 0 N E X T I  ENTER
9 9 B E E P 2  SHIFT  :  E  N  D  ENTER
```

Le PC-1500A a maintenant enregistré notre deuxième programme. Vous rappelez-vous encore quelle est l'opération suivante? Si vous répondez: "Exécuter le programme", vous êtes en voie de devenir un as de la programmation.

Retournez sur le mode RUN (Psit! utilisez la touche MODE!) et écrivez les lettres RUN. Appuyez sur **ENTER** pour commencer l'exécution de notre second programme.

Est-ce que l'ordinateur vous interroge à l'aide des caractères suivants? (s'il n'en est pas ainsi, retournez sur le mode PROgramme et re-vérifiez ce que vous avez écrit.)

```
DEG          RUN          |  |
LONGUEUR LISTE?_
```

Très bien! Notre programme demande à l'utilisateur (c'est vous) les renseignements nécessaires à l'exécution de la tâche que nous, en tant que programmeurs, lui avons ordonné d'exécuter. Rappelez-vous la première ligne de notre programme BASIC que vous avez eu la bonté d'entrer:

```
DEG          PRO          |  |
10 INPUT "LONGUEUR LISTE?"
```

L'ordinateur est actuellement en train de suivre les instructions emmagasinées dans cette ligne. Il attend maintenant que vous entriez (écriviez) certaines données.

Comme ce programme va imprimer une liste de nombres et de leurs carrés (le nombre multiplié par lui-même), l'ordinateur vous demande donc d'abord de lui indiquer combien de nombres et de carrés il doit imprimer (LONGUEUR LISTE?) L'utilisateur (C'est-à-dire vous) répond en écrivant un certain nombre et en appuyant sur (vous l'avez deviné!) la touche **ENTER**.

Ecrivez **8** et appuyez sur **ENTER**.

Regardez attentivement le défilement des nombres sur l'affichage. Le deuxième nombre de chaque paire (celui de droite) constitue le carré du premier. **En tout, 8 paires vont apparaître sur l'affichage, car c'est ce nombre que vous avez demandé en répondant à l'ordinateur.**

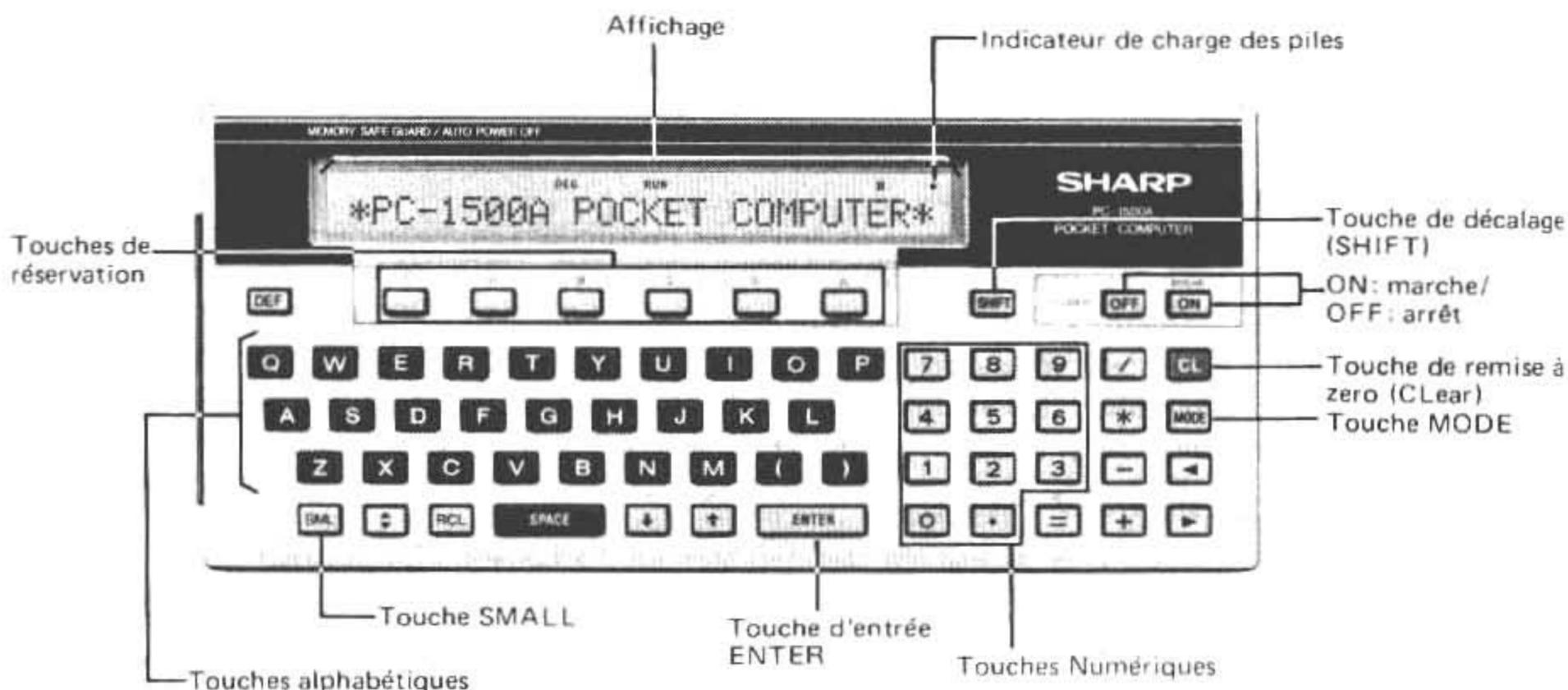
Quand le symbole de guidage réapparaît, faites passer à nouveau le programme (en écrivant RUN et en appuyant ensuite sur **ENTER**) et répondez différemment à la question "LONGUEUR LISTE?". Effectuez l'exécution du programme plusieurs fois, en changeant chaque fois la réponse. Vous devez vous rendre compte maintenant d'un des grands avantages des ordinateurs: ils peuvent répéter la même ennuyeuse opération en la variant légèrement à chaque fois en fonction de l'entrée.

Répétez une fois encore l'opération, mais entrez un zéro cette fois en réponse à la question "LONGUEUR LISTE?". Que se passe-t-il? Le programme s'arrête sans fournir de liste. Bien que cela puisse sembler bizarre, l'ordinateur ne fait que suivre vos directives.

Cette expérience nous montre l'utilité de cet outil qu'est l'ordinateur. On peut le programmer de manière à ce qu'il suive différentes séries d'instructions et traite les différentes données qu'on lui soumet. Du fait des instructions contenues dans la ligne 20, si l'utilisateur entre zéro (ou moins), l'ordinateur omettra de traiter la liste et sautera directement à la fin du programme. Actuellement, il s'est basé sur les demandes de l'utilisateur pour prendre cette décision. En tant que programmeur vous pouvez décréter quelles décisions sont possibles et quand elles sont possibles. Cela signifie que toutes les facultés de l'ordinateur sont à portée de votre main pour vous aider à résoudre vos problèmes particuliers de la manière qui vous convient le mieux.

I. FAISONS CONNAISSANCE

Après avoir déballé votre ordinateur de poche SHARP (nous l'appellerons le SHARP à partir de maintenant) et l'avoir admiré pendant un moment, vous allez probablement vous demander: "Qu'est-ce que c'est que ça? et ceci?" Regardons-le ensemble.



Nous allons décrire l'affichage dans un moment. D'abord, avant de mettre le SHARP en marche, veuillez considérer les plus importants éléments du clavier:

A. Touches ON et OFF (Marche et arrêt)

La fonction de ces touches est évidente: elles servent à mettre l'appareil en marche ou à l'arrêt. Si vous oubliez d'appuyer sur la touche OFF et que vous ne touchez pas l'ordinateur pendant plus de sept minutes, il s'éteindra automatiquement à moins qu'un programme ne soit en cours d'exécution. Remarquez le mot BREAK inscrit juste au-dessus de la touche **ON**. Cela veut dire que la touche **ON** peut servir à interrompre, briser (BREAK) l'exécution d'un programme. L'usage de cette fonction sera expliqué plus loin de manière détaillée.

B. Touches alphabétiques

Ces touches permettent à l'utilisateur (vous) de donner des instructions à l'ordinateur ou d'y entrer des données. De plus, ces touches servent aussi à désigner des "zones de stockage" à l'intérieur de la mémoire de l'ordinateur, zones dans lesquelles vous pouvez stocker des données et les en extraire à nouveau. Nous traiterons de cette utilisation dans la partie du manuel consacrée aux variables. Les touches **SHIFT** et **SML** (SMALL), qui vont être décrites plus loin, permettent d'écrire en minuscules.

C. Touches numériques et touches d'opérations arithmétiques

Elles servent à entrer les données numériques pour le calcul. Les touches **+**, **-**, *****, et **/** commandent au SHARP, respectivement, d'additionner, de soustraire, de multiplier et de diviser. La touche **E** permet l'entrée de nombres en "notation scientifique". L'utilisation de cette notation et d'autres fonctions plus avancées fait le sujet du chapitre "Calcul avancé".

D. SHIFT (Décalage)

On utilise cette touche pour la mise en action des fonctions inscrites au-dessus de nombreuses touches non-alphabétiques. Par exemple, pour écrire le double-point, appuyez d'abord sur la touche **SHIFT**, puis sur la touche avec astérisque. Quand on appuie sur une touche alphabétique après avoir pressé sur **SHIFT**, une minuscule apparaît sur l'affichage. (NOTE: Inversement, sur le mode SMALL, l'utilisation de **SHIFT** avant la touche alphabétique produira une majuscule.)

Quand on appuie sur la touche **SHIFT**, le mot **SHIFT** apparaît dans l'angle supérieur gauche de l'affichage. Ce mode n'est efficace qu'une seule fois pour la touche que l'on enfonce immédiatement après **SHIFT**.

On appelle "TOUCHES DE RESERVE" les six touches situées en haut du clavier, juste en dessous de l'affichage. Grâce à un usage spécial (décrit plus loin) de **SHIFT**, on peut affecter certaines opérations ou commandes fréquentes à ces touches.

NOTE: Si vous appuyez par mégarde sur la touche **SHIFT**, appuyez à nouveau pour éliminer son effet.

E. Les minuscules et la touche SMALL (SML)

A l'aide de la touche **SML**, vous pouvez spécifier que vous voulez écrire en minuscules certains ou tous les caractères alphabétiques. Si vous ne spécifiez pas le mode SMALL, le SHARP choisira des majuscules chaque fois que vous appuierez sur une touche alphabétique. (On appelle cela le mode "par défaut", ce qui veut dire que si vous n'intervenez pas avec une directive différente, l'appareil décidera "par défaut"). Vous pouvez écrire individuellement des lettres en minuscules en appuyant sur la touche **SHIFT** avant d'appuyer sur la touche alphabétique, comme nous l'avons vu plus haut. On passe sur le mode SMALL en appuyant sur la touche **SML**. Dans ce mode, il suffit d'appuyer sur les touches alphabétiques pour écrire des minuscules tandis que pour les majuscules, il faut d'abord presser la touche **SHIFT**. Les lettres SMALL apparaissent sur la partie supérieure de l'affichage quand l'ordinateur fonctionne sur le mode SMALL. Une fois sur le mode SMALL, l'ordinateur y restera jusqu'à ce que vous appuyiez de nouveau sur la touche **SML**.

NOTE: Nous vous recommandons de limiter votre utilisation des minuscules pour le moment, car le SHARP ne reconnaît que les commandes en majuscules. Quand vous saurez programmer vous trouverez les minuscules très utiles.

F. L'AFFICHAGE

Appuyez sur ON. La partie "vitrée" de l'ordinateur s'appelle l'affichage. Il correspond à l'illustration suivante:



Vous devriez voir le symbole de guidage \triangleright , plusieurs mots ou abréviations et un point (qui indique que les piles sont chargées). Si les abréviations apparaissant sur l'affichage ne correspondent pas à celles de notre dessin ci-dessus, ne vous faites pas de soucis: elles changent au cours de votre utilisation du SHARP.

G. LE CURSEUR et le SYMBOLE de GUIDAGE

Le symbole de guidage ">" se trouve à l'extrême droite de l'affichage. Il est là pour vous inciter à parler au SHARP. Quand il est affiché, il vous indique que le SHARP n'a pas de projets pour l'immédiat et qu'il attend vos ordres. Tapez une lettre de votre choix. Elle remplace > à la gauche de l'affichage tandis qu'à la droite de cette lettre apparaît un – (trait de soulignage) que l'on appelle "curseur". A mesure que vous appuyez sur des touches ce trait avance à travers l'affichage, indiquant chaque fois où le caractère suivant va apparaître. Tapez votre nom et suivez le mouvement du curseur.

Si vous écrivez plus de 25 caractères (la limite de ce qui peut apparaître en une fois) la ligne entière sautera vers la gauche (Faites l'essai!). Les caractères "poussés dehors" ne sont pas perdus: ils restent à l'intérieur du SHARP comme partie intégrante de la ligne écrite et cela jusqu'à un maximum de 80 caractères sur une seule ligne. Nous verrons plus tard comment faire pour "rappeler" ces caractères.

H. REMISE A ZERO

(La touche **CL** rouge dans l'angle droit supérieur du clavier)

Cette touche sert à effacer le contenu de l'affichage. Utilisez-la pour éliminer les caractères que vous venez juste d'écrire. Remarquez que le symbole de guidage vient de réapparaître pour vous avertir que l'ordinateur attend à nouveau vos ordres. La touche **CL** de remise à zéro sert aussi à éliminer une commande erronée. (Voyez plus loin la section concernant les messages d'erreur).

I. Touche ENTER

Les lettres et les nombres apparaissent sur l'affichage au fur et à mesure que vous les écrivez. Cependant le SHARP n'entrera en action que si vous lui indiquez que vous avez fini d'écrire. (C'est normal; après tout il n'est pas censé lire vos pensées!) Vous pouvez faire cela en appuyant sur la touche **ENTER**. A ce moment l'ordinateur examinera les caractères écrits pour voir si la forme est correcte. Certaines erreurs, mais pas toutes, causeront le refus de votre entrée.

N'OUBLIEZ PAS: d'appuyer sur la touche **ENTER** après chaque commande ou donnée élémentaire que vous désirez entrer dans l'appareil.

J. MESSAGES D'ERREUR

Appuyez sur les touches suivantes:

1 + 1 =

Appuyez maintenant sur **ENTER**.

La réponse devrait être affichée. Trois, n'est-ce pas? Non!? L'ordinateur vous répond: "ERROR 1"? Est-ce qu'il a un défaut de fabrication? Jamais de la vie! Vous avez fait une erreur de forme dans votre commande. "ERROR 1" est un code d'erreur qui vous révèle que vous avez effectué votre calcul de façon erronée. (Pour ceux que cela intéresse, il y a une liste complète des messages d'erreurs en appendice à la fin de ce manuel). Nous prenons la responsabilité pour la présente erreur. Plus loin nous vous montrerons quelles touches il faut utiliser pour corriger une commande erronée. Pour l'instant, contentez-vous d'éliminer le message d'erreur à l'aide de la touche **CL**.

K. INDICATEUR DE CHARGE DES PILES

Quand ce point disparaît, il est temps de remplacer les piles du SHARP. Reportez-vous à l'appendice pour les instructions nécessaires.



L. Touche RCL (rappel)

Cette touche sert à rappeler une instruction ou une phrase mise en mémoire antérieurement dans le Mode Réserve.

Pour effectuer le rappel d'une entrée en Mode de Réserve, suivre les étapes ci-après:

Entrer d'abord le Mode de Réserve en appuyant sur la touche de sélection de réserve () pour sélectionner le groupe I, II ou III (la touche de sélection de réserve est située immédiatement à gauche de la touche RCL (rappel)).

Ensuite, appuyer sur la touche RCL (rappel) et la touche de réserve assignée antérieurement. Si l'on a oublié les attributions antérieures, il suffit de suivre les instructions du paragraphe B, page 137.

QUESTIONNAIRE – CHAPITRE 1

Trouvez dans la colonne B un élément correspondant à chaque élément de la colonne A. (Réponses en bas de la page).

Avertissement: La colonne B contient des réponses stupides.

A

- a) Touche SHIFT
- b) L'affichage
- c) Le Curseur
- d) Touche ENTER
- e) Touche BREAK
- f) Touche CL
- g) Le symbole de guidage
- h) La touche EXOTIQUE

B

- 1) Un messenger du temps du deuxième Triumvirat.
- 2) Une touche qui nettoie la "vitre" de l'affichage et efface les résultats des calculs antérieurs.
- 3) Une touche qui choisit l'un des caractères qui ont une touche en commun.
- 4) On la cherche toujours encore.
- 5) Un caractère qui, lorsqu'il apparaît sur l'affichage, indique à l'utilisateur (ou l'utilisatrice) que l'ordinateur attend ses ordres.
- 6) Une touche qui indique à l'ordinateur que l'utilisateur (ou l'utilisatrice) a fini d'écrire.
- 7) Une touche qui interrompt un programme en cours d'exécution.
- 8) Le nom d'un poisson très rare des mers tropicales.
- 9) Le caractère qui, sur l'affichage indique où le prochain caractère va apparaître.
- 10) La "vitre" dans laquelle les données apparaissent.

Réponses: A-3, B-10, C-9, D-6, E-7, F-2, G-5, H-4

II. PLONGER DANS L'EAU FROIDE

En fait, ce chapitre est probablement mal nommé. Apprendre à se servir du PC-1500A est loin d'être aussi traumatisant qu'un saut dans l'eau froide d'une piscine. Cependant il est nécessaire de surmonter ses craintes avant d'aborder l'ordinateur. Comme nous l'avons vu avant, il est impossible d'abîmer l'ordinateur en appuyant sur la mauvaise touche. De plus, les ordinateurs n'ont jamais mordu personne.

Dans ce chapitre, nous allons explorer les caractéristiques fondamentales du SHARP, caractéristiques sur lesquelles sont basés les programmes et les calculs avancés. Prenez votre temps avec les exemples de chaque section. Une bonne compréhension des opérations de base est essentielle si vous vous proposez d'utiliser à fond les capacités de cet appareil.

Bien que nous ne le recommandions pas, si vous estimez que vous êtes assez fort déjà, vous pouvez sauter tout de suite au résumé à la fin de ce chapitre.

A. MODE

Commençons avec une touche que nous avons passé sous silence jusqu'à présent. Tout à fait à la droite du clavier se trouve une touche très importante, marquée "MODE". Appuyez plusieurs fois de suite sur elle et remarquez à chaque fois le changement apporté aux abréviations en haut, à droite de l'affichage. On peut comparer cette opération avec le changement de vitesses dans une voiture. Chaque fois que l'on change de vitesse ou de mode, le même élément du moteur ou de l'ordinateur fonctionne différemment, bien qu'il ne semble pas avoir changé extérieurement. Le SHARP, comme la voiture, doit être sur le mode correct, si vous voulez qu'il fonctionne comme prévu. Et tout comme sur la voiture, si vous passez la mauvaise "vitesse" sur le SHARP, il vous le fera savoir très rapidement (mais sans grincements, bien sûr).

Appuyez à plusieurs reprises sur la touche MODE et vous ferez connaissance avec deux des plus importants modes du SHARP: RUN (passage, exécution) et PROgramme. Un troisième, le mode RESERVE, peut être mis en action en appuyant sur la touche SHIFT avant d'appuyer sur MODE. Nous verrons plus loin, dans d'autres chapitres de ce manuel comment chacun de ces modes apporte sa contribution au bon fonctionnement du SHARP. Pour l'instant, il vous suffit de savoir que si vous voulez vous servir du SHARP en tant que calculateur il faut que vous soyez en mode RUN. Plus tard, vous écrirez et modifierez des programmes en mode PROgramme. A l'aide du mode RESERVE vous pourrez affecter des commandes dont vous vous servez fréquemment à une seule touche de réserve. Tout cela fait l'objet d'explications en détail dans le chapitre 7.

Vous remarquerez que chaque fois que vous remettrez l'ordinateur en marche après avoir changé les piles, il se mettra de lui-même sur le mode RUN. Les autres fois, quand vous remettrez le SHARP en marche, il se rétablira sur le mode engagé au moment où il a été mis à l'arrêt.

B. CALCULS SIMPLES

Plaçons le SHARP sur le mode RUN pour faire l'essai des opérations mathématiques les plus élémentaires. **Il est nécessaire d'appuyer sur la touche CL avant chaque calcul, dans le but d'effacer toute donnée antérieure qui pourrait s'ingérer dans les nouvelles opérations.** Trouvez la réponse aux opérations élémentaires suivantes:

Entrée	Affichage
5 + 2 ENTER	7
5 - 2 ENTER	3
5 / 2 ENTER	2.5
5 * 2 ENTER	10

NOTE: N'écrivez pas le signe d'égalité =. Rappelez-vous que c'est au moyen de la touche ENTER que vous signalez au SHARP que vous avez fini d'écrire et que vous désirez qu'il exécute vos ordres, ou effectue un calcul.

C. CALCULS EN SERIES

Dans l'exemple suivant, on utilise directement la réponse à la première opération en passant à la deuxième opération sans appuyer sur la touche CL entre les deux.

Entrée	Affichage
161.16 - 47.50 ENTER	113.66
- 12.33	113.66 - 12.33
ENTER	101.33

Comme vous voyez, le résultat de la première opération saute à la gauche de l'affichage quand vous débutez la seconde.

NOTE: N'écrivez pas de virgules ou de \$ quand vous faites des calculs. Ces signes possèdent des sens particuliers dans le langage BASIC qui est celui du SHARP.

D'autres opérations peuvent être effectuées de la même façon. Essayez les suivantes:

Entrée	Affichage
5 + 3 ENTER	8
8 + 3 - 1 ENTER	10
10 * 3 - 1 ENTER	29
29 / 3 - 1 ENTER	8.666666667

D. CALCULS AVEC DES NOMBRES NEGATIFS

Prétendez que vous avez offert deux pommes à M. Laflèche, votre professeur d'informatique. Il vous reste cinq pommes en stock et vous vous demandez: "Combien aurais-je maintenant de pommes si je n'avais pas été aussi généreux envers M. Laflèche?" Pour trouver la réponse, imaginez que vous annulez la soustraction que vous avez faite à votre stock de pommes, c'est-à-dire que vous soustrayez la soustraction. Tapez 5 -- 2 ENTER sur le SHARP. Votre stock hypothétique serait sept pommes. Essayez d'effectuer les opérations suivantes qui sont basées sur le même problème des nombres négatifs:

$$5 * - 2$$

$$5 + - 2$$

$$5 / - 2$$

$$- 5 - 2 . 3$$

$$- 5 + - 2$$

$$- 5 / - 2$$

N'OUBLIEZ PAS d'appuyer sur la touche **CL** entre chaque opération.

E. CALCULS COMPOSES

Il est possible d'enchaîner plusieurs opérations avant de demander une réponse au SHARP. Supposons que vous et deux amis vous vouliez partager 5 pommes deux fois par jour pendant une semaine. Combien de pommes vous faut-il pour une semaine?

5 pommes **/** 3 amis ***** 2 fois par jour ***** 7 hours

<u>Touches utilisées</u>	<u>Affichage</u>
$5 \ / \ 3 \ * \ 2 \ * \ 7 \ \text{ENTER}$	23. 33333333

(Achetez 24 pommes et 1 lapin à qui vous donnerez le tiers de pomme restant). Faites les opérations suivantes (c'est à vous d'inventer une histoire, cette fois-ci):

Entrée	Affichage
$5 * 2 - 3.675 \ \text{ENTER}$	6. 325
$5 / 3 * 6.2 + 7 - 47 \ \text{ENTER}$	-29. 66666667

F. L'UTILISATION DES PARENTHESES

Un des problèmes que nous découvrons au cours des calculs complexes est celui de la priorité. Par exemple, l'expression $5 - 3/4$ peut être interprété comme étant: (5 moins 3) divisé par 4 et aura pour résultat 0,5; ou bien comme étant: 5 moins (3 divisé par 4) ce qui fait 4,25.

Les parenthèses, qui sont situées dans la première rangée de touches, permettent de clarifier ce genre d'ambiguïté. Faites les opérations suivantes:

Entrée	Affichage
$5 - 3 / 4 \ \text{ENTER}$	4. 25
$5 - (3 / 4) \ \text{ENTER}$	4. 25
$(5 - 3) / 4 \ \text{ENTER}$	0. 5

Le SHARP est prédisposé (a des priorités "par défaut") à exécuter certaines opérations avant d'autres (pour une liste complète de l'ordre dans lequel le SHARP exécute des opérations, reportez-vous à l'appendice 0). Les divisions et les multiplications sont effectuées avant les additions et les soustractions, à moins que l'on se serve des parenthèses pour modifier cet ordre. Il en résulte que le SHARP interprétera la première équation (sans parenthèses) du diagramme ci-dessus de la même façon que la seconde plutôt que de la troisième. Donc, pour être sûr que la réponse de l'ordinateur va être celle dont vous avez besoin, utilisez les parenthèses pour lui indiquer l'ordre correct des opérations.

Le SHARP est capable d'interpréter un emboîtement de parenthèse comme dans le cas ci-dessous:

Entrée	Affichage
$((6 - 4) / 2) * ((3 - 1) / 4) * 6$ ENTER	3

Les équations situées dans le noyau le plus au centre des parenthèses sont traitées avant les autres.

NOTE: Si vous avez des doutes, utilisez les parenthèses pour clarifier l'ordre de vos opérations arithmétiques.

G. COMPARAISONS LOGIQUES ET FONCTIONS LOGIQUES

Le SHARP vous permet de comparer deux valeurs ou deux équations et vous fournit le résultat de cette comparaison. Cette capacité est fondamentale pour la conception de programmes destinés à prendre des décisions. Les étudiants en mathématiques nouvelles reconnaîtront certainement cette façon de faire cela comme étant une fonction logique (ou encore "inégalité"). (Ne vous désespérez pas si vous ne vous connaissez pas en mathématiques nouvelles: c'est une nouveauté aussi pour l'auteur de ce manuel).

On peut considérer une fonction logique comme étant une comparaison qui est soit vraie soit fausse. Par exemple, il se trouve que la formulation "six divisé par trois est égal à deux" est vraie. Par contre, "six divisé par trois est plus grand que cinq" est une proposition fausse.

Les ordinateurs et les mathématiciens se servent des symboles suivants pour exprimer les différentes sortes de comparaisons:

<	moins grand que
>	plus grand que
=	égal à
<=	moins grand que OU égal à
>=	plus grand que OU égal à
<>	inégal à

Nous pouvons donc récrire les fonctions logiques du texte ci-dessus de la façon suivante: $6 / 3 = 2$ et $6 / 3 > 5$ respectivement.

Quand on présente une fonction logique au SHARP, il détermine si elle est vraie ou fausse. La pratique courante dans le domaine des ordinateurs consiste à représenter une formulation vraie par un 1 et une formulation fausse par un 0. Cette pratique vaut pour le SHARP. Par exemple si vous entrez:

$$6 / 3 = 2 \text{ [ENTER]}$$

Le Sharp répondra avec un 1 (vrai) et pour :

$$6 / 3 > 5 \text{ [ENTER]}$$

la réponse sera 0 (faux).

Mettez le bon jugement du SHARP à l'épreuve à l'aide des exercices suivants (assurez-vous d'être sur le mode RUN).

Entrée	Résultats
[4] [SHIFT] [>] [5] [ENTER]	0
[5] [SHIFT] [>] [4] [ENTER]	1
[2] [*] [2] [SHIFT] [<] [2] [*] [3] [ENTER]	1
[1] [8] [SHIFT] [<] [SHIFT] [>] [1] [8]	0
[2] [=] [2] [ENTER]	1
[5] [SHIFT] [<] [=] [5] [ENTER]	1
[2] [5] [*] [2] [SHIFT] [>] [=] [1] [0] [0] [/] [3] [+] [4] [ENTER]	1

Peut-être que vous avez remarqué les équations peuvent être aussi complexes que nécessaires. Une seule limitation: 80 symboles sur une ligne.

Le problème suivant illustre l'utilisation pratique des inégalités (ou fonctions logiques):

Vous entrez dans une épicerie pour acheter du sucre. On le vend en sacs de 4, 8 et 12 kilos. L'argent que vous avez sur vous vous permet d'acheter soit deux sacs à 12 kilos soit trois sacs à 4 kilos et un de 8 kilos. Vous vous demandez dans quel cas vous obtiendrez le plus de sucre. Vous demandez à votre SHARP:

$$2 * 12 > (3 * 4) + 8$$

Il répond: 1 et vous achetez les 2 sacs à 12 kilos.

Essayez d'appliquer ce système à l'un des problèmes que vous rencontrez dans votre vie quotidienne.

H. LES TOUCHES ET LES CARACTERISTIQUES DE MISE EN PAGE

La plupart des êtres humains (sauf nous autres les génies) ont tendance à commettre des erreurs. Reconnaissant cette faillibilité de l'homme, les techniciens qui conçurent le PC-1500A y ont intégré plusieurs caractéristiques qui facilitent les modifications et les corrections.

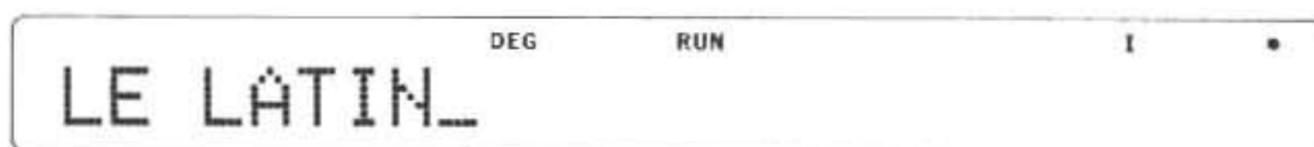
H.1. Flèche à gauche et touche DELete (d'effacement).

Vous avez certainement dû voir déjà la touche à flèche dirigée vers la gauche, en base sur le côté droit du clavier. L'action de cette touche ressemble à celle de la touche de rappel arrière sur les machines à écrire modernes. Elle permet de retourner sur des caractères écrits antérieurement.

Ecrivez les caractères suivants, après avoir engagé le mode RUN et mis l'affichage à zéro (c-à-d que le symbole de guidage doit être visible).

[L] [E] [SPACE] [L] [A] [T] [I] [N]

L'affichage devrait être comme suit:

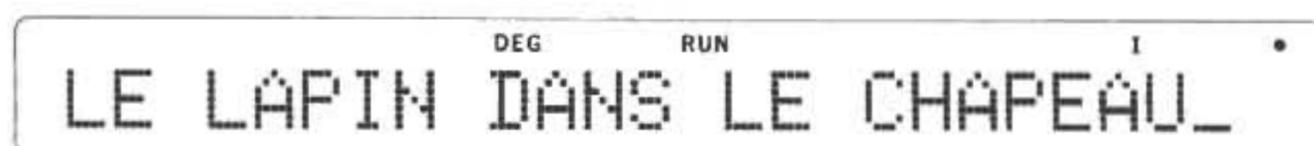


Appuyez une fois sur la touche à flèche à gauche et regardez de quelle manière le curseur change. Cette forme en "grille clignotante" du curseur vous permet de voir le caractère situé à la position présente du curseur. Appuyez plusieurs fois (ou de manière continue) sur la touche à flèche à gauche jusqu'à ce que le curseur soit situé au-dessus du T. (Si vous dépassez le T par mégarde, ramenez le curseur en avant de nouveau à l'aide de la touche à flèche à droite. Ecrivez un P. Le P remplace le T et le curseur avance. Cela ne devrait pas vous surprendre si vous vous souvenez que le curseur indique l'endroit où le prochain caractère va apparaître.

Ecrivez les caractères qui suivent:

I N SPACE D A N S SPACE L E SPACE C H A
P E A U

L'affichage montrera:



Comme vous voyez les caractères sur lesquels on écrit à nouveau ont disparu pour toujours.

En plus de cette fonction de rappel arrière, la touche à flèche à gauche sert aussi à mettre en action la fonction d'effacement dont le symbole DEL inscrites au-dessus de la touche. Pour effacer un caractère, placez le curseur au-dessus du caractère condamné, puis, appuyez sur **SHIFT** et **DEL**.

Faisons l'essai. Ramenez le curseur en arrière sur le D et appuyez sur **SHIFT** et **DEL** cinq fois de suite.

L'affichage indique maintenant:



H.2. Flèche à droite et Touche **INS**ert (d'insertion)

Comme nous l'avons déjà vu, la touche  à flèche à droite fait avancer le curseur sans effacer des caractères. Tout comme la touche  à flèche vers la gauche, celle-ci fera avancer plusieurs fois le curseur si vous la maintenez en position enfoncée.

Amenez le curseur au bout de la ligne. La seconde fonction de cette touche nous permet d'**INS**érer des caractères dans une ligne écrite. Cette caractéristique est éminemment pratique pour ceux d'entre nous qui ont tendance à oublier certaines petites choses (comme des lettres ou des mots, par exemple).

Ramenez le curseur sur le P de LAPIN. Appuyez cinq fois sur la séquence **SHIFT** **INS** et notez que les caractères en existence sautent vers la droite et sont remplacés par des caractères en forme de carrés que l'on pourrait appeler "garde-place". On peut les remplir avec de nouvelles données.

Ecrivez:

L **E** **SPACE**

Et voilà. Vous venez juste d'insérer un mot. (Je voudrais bien que ma machine à écrire puisse accomplir ce genre de prouesse!)

H.3. FONCTION DE RAPPEL

Nous nous sommes occupés jusqu'à présent uniquement des moyens de modifier des messages qui n'avaient pas encore été entrés (c'est-à-dire, après lesquels vous n'avez pas encore appuyé sur la touche **ENTER**). Une fois que vous avez appuyé sur **ENTER**, le SHARP va essayer immédiatement d'exécuter votre calcul. S'il réussit, le résultat remplacera votre équation sur l'affichage. Cette équation n'en est pas perdue pour autant. Elle peut être rappelée (ré-affichée) en appuyant soit sur la touche DEL soit sur la touche INS.

Remettez l'affichage à zéro et écrivez l'équation de votre choix. Appuyez sur ENTER pour obtenir le résultat. Rappelez votre équation. Remarquez que si vous voulez voir le résultat de nouveau, il faudra appuyer à nouveau sur la touche **ENTER**.

Heureusement, cette fonction de rappel agit aussi lorsque le SHARP détecte une erreur quand il évalue (essaie de comprendre) votre entrée. Elle vous permet de rappeler et de corriger l'équation erronée à l'aide des différentes méthodes que vous venez juste d'apprendre. Faites-en l'essai en entrant l'équation incorrectement énoncée qui suit:

45 * 63 / * 2 **ENTER**
↑

Quand le message d'erreur (ERROR 1) apparaît, appuyez sur l'une des touches à flèche pour rappeler l'expression. Vous devriez voir apparaître le curseur en "grille clignotante" au-dessus du deuxième astérisque (symbole de multiplication). Voilà la méthode que le SHARP utilise pour vous indiquer à quel point il a été rendu perplexe (et avec raison dans ce cas-ci). A partir de maintenant vous pouvez effectuer toutes les corrections que vous jugerez nécessaires.

I. LES VARIABLES

La capacité d'agir abstraitement par l'intermédiaire de variables constitue l'une des caractéristiques les plus formidables du SHARP. On peut se représenter les variables comme une série de petites boîtes que l'on peut remplir avec un élément unique de données tel qu'un nombre ou un nom.

Peut-être que vous vous souvenez des variables apprises en algèbre au lycée? Vous avez dû apprendre (ou devons nous dire: on a essayé de vous apprendre) que si $5A = 30$, A doit être égal à 6 et que si $5A = 35$, A est égal à 7. Dans ce cas, le A est une variable qui contient un seul nombre (qui n'est pas toujours le même et donc "variable") que l'on appelle la "valeur" de la variable. Il est très pratique de pouvoir utiliser une lettre (comme A dans le cas ci-dessus) dans une équation à la place d'un nombre spécifique, comme l'exemple suivant nous le montrera:

Monsieur Dupont s'est décidé à acheter un jeu de clubs de golf en plastique de marque Cassevite. Le jeu comprend 5 clubs à 12F chaque, un sac (imitation cuir vert) à 21,99F et trois balles (en papier mâché avec deux couches de peinture protectrice) à 1,56F chaque. Ce jeu est en vente au magasin de sport "Le Mont-Chauve" où l'on fait une remise de 10%, mais où les frais de livraison sont de 8%. Par contre, au supermarché Multiprix, on peut se procurer le jeu avec une remise de 5% seulement, mais la livraison y est gratuite.

Pour trouver où il fera la meilleure affaire, M. Dupont décide de s'adresser à son fidèle PC-1500A. Il calcule le prix de base du jeu et en fait la variable G:

$G = (5 * 12) + 21.99 + (3 * 1.56)$ **ENTER**

Le résultat est affiché et pourra être rappelé de la mémoire en écrivant simplement la lettre **G** , puis **ENTER** .

DEG RUN

I
86.67°

Maintenant M. Dupont calcule le prix réel du jeu au Mont-Chauve:

$G - (G * .10) + 8$ **ENTER**

DEG RUN

I
86.003°

puis chez Multiprix:

$G - (G * .05)$ **ENTER**

DEG RUN

I
82.3365°

Il est évident que c'est chez Multiprix que M. Dupont va faire son achat. (Les lecteurs astucieux auront probablement remarqué que M. Dupont aurait aussi pu résoudre ce problème en se servant de la caractéristique de rappel unique au SHARP (voir Chapitre 2), bien que ce soit un peu plus compliqué)

Nous pouvons tirer plusieurs leçons de l'exemple précédent . . .

Notez que le premier calcul de M. Dupont se présentait sous la forme suivante;

Nom de la variable = expression

On appelle instruction d'affectation les instructions se présentant sous cette forme. Il est très important de ne pas confondre une instruction d'affectation avec une fonction logique. A l'encontre de l'instruction d'affectation, on n'utilise pas les fonctions logiques séparément, mais en tant qu'élément d'autres commandes de programmation.

L'instruction d'affectation ordonne au SHARP de mémoriser le résultat obtenu en calculant l'expression dans le "casier" correspondant au nom de la variable. Par la suite, utiliser le nom de la variable (G dans notre cas) équivaudra à utiliser le résultat lui-même. Prenez note aussi du fait que vous pouvez vous servir de la variable aussi souvent que nécessaire dans le même calcul.

On peut aussi se servir des variables pour d'autres tours de passe-passe. On peut, par exemple, affecter la valeur d'une variable à une autre:

$H = G$

Cette formulation passe les résultats précédents de G à H. G n'a pas changé, mais son résultat est stocké maintenant sous deux variables.

On peut aussi augmenter ou diminuer la valeur d'une variable contenant des nombres, comme l'indique l'exemple:

$$G = G + 5$$

Cette commande rappelle la valeur de G, y ajoute 5 et restocke la nouvelle valeur sous G. Cette capacité s'avère utile dans de nombreux calculs:

Le coût d'un machin est mémorisé dans la variable X. Supposons que la T.V.A. soit de 6,5%, quel est le prix d'achat du machin?

$$X = X * 1.065$$

Si nous avons voulu laisser X inchangé, nous aurions pu écrire $P = X * 1.065$. Et si nous avons stocké le cours de la taxe sous la variable T, nous pourrions aussi écrire $X = X * T$ ou encore $P = X * T$.

Jusqu'à présent nous avons fait usage d'une lettre seulement pour une variable, ce qui met un total de 26 variables à notre disposition (A à Z). En fait, le SHARP met 950 noms de variable à notre disposition pour des nombres simples. Et comme si cela ne suffisait pas encore, l'utilisateur du SHARP peut, aux moyens de méthodes avancées, créer des variables contenant autant de nombres ou de caractères que l'on veut, avec pour seule limitation la quantité de mémoire disponible dans l'ordinateur.

Il n'est pas difficile d'apprendre le système de spécification des variables. on peut choisir les variables numériques (destinées à contenir des nombres) d'après les règles suivantes:

- Le nom peut être constitué d'une lettre de A à Z
- Le nom peut être constitué d'une lettre suivie d'un chiffre unique (de 0 à 9) ou d'une autre lettre.

Ainsi, par exemple, les noms suivants peuvent être adoptés pour des variables:

S, Q1, TX, MM, Z9, R0, E.

NOTE: Les combinaisons suivantes ne doivent pas être utilisées en tant que noms de variables dû au fait qu'elles ont une signification particulière dans le langage BASIC: LF, LF, LN, PI, TO.

Les noms de variables à caractères (contenant des caractères) suivent les mêmes règles que celles énoncées ci-dessus sauf qu'elles sont suivies du symbole \$. Ce symbole avertit le SHARP que cette variable contient des données en caractères. L'exemple suivant vous montre des noms possibles pour des variables à caractères:

T\$, P2\$, T7\$, AA\$, YR\$, X\$, ZH\$, B5\$.

NOTE: Les combinaisons suivantes ne doivent pas être utilisées en tant que noms de variables à caractères parce qu'elles font partie du vocabulaire BASIC: LF\$, IF\$, LN\$, PI\$, TO\$.

Il est important que vous compreniez que la variable A et la variable A\$ sont totalement différentes: la première représente uniquement un nombre et la seconde uniquement un caractère. Comme nous le verrons plus tard, le langage BASIC du SHARP comprend aussi des instructions concernant la conversion de caractères en nombres et de nombres en caractères. Pour mémoriser des caractères sous des variables à caractères, on se sert de notre chère amie, l'instruction d'affectation:

nom de variable à caractères = "caractères"

Voyons l'exemple suivant:

D\$ = "JEAN BART"

Rappelez maintenant le contenu de D\$ en écrivant **D** **SHIFT** **\$** **ENTER**

JEAN BART

Remarquez que quoique l'espace entre les deux mots ait été mémorisé, les "guillemets" ne l'ont pas été. Les guillemets servent à "délimiter", à indiquer que l'on va mémoriser d'autres caractères. On peut mémoriser n'importe quel caractère (y compris les espaces) sauf les guillemets. On appelle ce genre de séquence de caractères encadrée par des guillemets une "chaîne" et, souvent, l'on applique aux variables à caractères le nom de "variables en chaîne". Chaque variable à caractère peut contenir jusqu'à 16 caractères. Il faut garder un oeil sur cette limite pour éviter de perdre des données. L'exercice suivant illustre cela, en régularisant involontairement la boisson de Jean. Ecrivez:

F\$ = "JEAN BOIT DU VINAIGRE"

Rappelez maintenant: **F** **SHIFT** **\$** **ENTER** . Quel changement!

Une note finale à propos des variables: elles ont une mémoire d'éléphant. Des données resteront mémorisées sous une variable jusqu'à ce que:

- 1) Une autre instruction d'affectation soit exécutée pour la même variable
- 2) Une autre commande NEW ou CLEAR soit donnée
- 3) On passe un programme sur le mode RUN
- 4) On remplace les piles de l'ordinateur

La valeur de la variable reste mémorisée même si l'on met l'ordinateur hors tension. Faites l'essai: appuyez sur OFF puis ON de nouveau. Ensuite sur le mode RUN, rappelez la valeur de G en écrivant **G** **ENTER** . Impressionnant, n'est-ce pas? A partir du chapitre prochain vous découvrirez que les variables sont indispensables.

J. UNE PAUSE

Félicitations! Si vous avez persévéré jusqu'ici, vous avez dû assimiler les principes de base du SHARP PC-1500A suffisamment bien pour pouvoir effectuer un grand nombre d'opérations. Grâce à la souplesse du PC-1500A, le lecteur pourra lui trouver une multiplicité d'usages dans son domaine d'activité propre. Quelle que soit votre application, l'heure viendra où vous voudrez apprendre à programmer pour utiliser au maximum les capacités de cet instrument étonnant. C'est à vous de choisir. A ceux dont le coeur se met à battre très vite et dont la respiration s'accélère à la pensée de programmer aussi bien qu'à ceux qui ne se sont pas encore endormis, nous allons demander de revenir en arrière et de relire le Chapitre 0 et combiner vos nouvelles connaissances avec les données qui s'y trouvent. Ensuite passez au Chapitre 3.

Les autres lecteurs qui ont besoin immédiatement de caractéristiques telles que la numération scientifique et les fonctions trigonométriques pourront passer tout de suite au chapitre "Calculs avancés".

- 13) 14)
- 1) **Les modes** – Le SHARP PC-1500A possède trois différents modes d'opérations: RUN (passage), PROgramme, et RESERVE. Chaque changement de mode produit un changement légèrement différent dans le fonctionnement interne, analogue au changement de vitesses sur une voiture. Le mode RUN est utilisé pour les calculs et c'est le mode sur lequel vous devez exécuter les programmes. Le mode PROgramme sert à écrire et mettre en forme (à faire des ajouts et à corriger) les programmes.
 - 2) **Les calculs** – Le SHARP effectue les opérations arithmétiques d'ordre général sur le mode RUN. Appuyez sur la touche CL (de remise à zéro) avant chaque opération pour éliminer le résultat des opérations précédentes. NE PAS appuyer sur cette touche entre les opérations permet d'effectuer des calculs en série utilisant chaque fois le résultat de l'opération précédente.
 - 3) **Caractères spéciaux** – Le symbole \$ et la virgule (,) ont une signification particulière dans le langage BASIC et ne doivent pas être utilisés comme partie d'un nombre dans une opération.
 - 4) **Les nombres négatifs** – sont précédés du signe négatif – (moins), comme par exemple: $-5 + 2 = > -3$
 - 5) **Les calculs complexes** – suivent les lois algébriques générales. Les parenthèses servent à préciser le sens d'une expression donnée. L'appendice D fournit des renseignements complets au sujet de l'ordre d'évaluation de l'expression.
 - 6) **Les fonctions logiques** – Les fonctions logiques (exprimées par les symboles suivants: < moins grand que, > plus grand que, <= moins grand que ou égal à, >= plus grand que ou égal à, <> inégal à) sont jugées ainsi: 0 si elles sont fausses et 1 si elles sont vraies.
 - 7) **La touche à flèche à gauche** – Cette touche sert de rappel arrière "non-destructif" du curseur. Maintenir cette touche en position enfoncée cause une répétition automatique. Le curseur endessous de ligne se métamorphose en "grille clignotante" quand on le place au-dessus d'un caractère écrit. Si l'on appuie sur la touche SHIFT avant la touche <, la fonction DEL (d'effacement) entre en action et efface le caractère au-dessus duquel s'est placé le curseur.
 - 8) **La touche à flèche à droite** – Cette touche avance le curseur de façon "non-destructive". Le maintien de cette touche en position enfoncée cause une répétition automatique. Si l'on appuie sur la touche SHIFT avant la touche >, la fonction INSérer entre en action et insère un caractère "garde-place" à l'endroit où se trouve le curseur. L'on peut ensuite écrire au-dessus de ce caractère la donnée que l'on veut insérer.
 - 9) **La Fonction de rappel** – Il est possible de rappeler l'équation originale, après avoir appuyé sur la touche ENTER et faire afficher le résultat du calcul, en appuyant soit sur la touche < soit sur la touche >. Il est possible alors d'apporter des modifications à l'équation puis de l'entrer à nouveau. La fonction de rappel agira aussi dans le cas où une expression non-programmée résulte en une erreur. Dans ce cas, le curseur sera placé au point où l'erreur a été découverte.
 - 10) **L'utilisation des variables** sur le mode RUN augmente considérablement la puissance ordinatrice du PC-1500A et permet d'abrégier des expressions complexes.
 - 11) **Les instructions d'affectation** qui utilisent les formes suivantes:
 nom de variable = expression
 nom de variable à caractères = "caractères"
 permettent le stockage d'un nombre unique ou d'une chaîne de 16 caractères au maximum, respectivement. On peut se servir de n'importe quel caractère, sauf des guillemets.
 - 12) **Les noms de variables** pour les variables numériques sont constitués par:
 1. Une lettre (A à Z)
 2. Une lettre suivie d'un chiffre (0 à 9) ou d'une autre lettre.
 Les noms de variables à caractères suivent les mêmes règles, sauf qu'on leur ajoute à tous le suffixe \$.

- 13) **Exceptions** — Les symboles suivants constituent des exceptions et ne peuvent être utilisés comme noms de variable: **IF, LF, LN, PI, TO, IF\$, LF\$, LN\$, PI\$, TO\$**.
- 14) **Durée de vie des données** — les données contenues dans des variables sont retenues jusqu'à ce:
- 1) Qu'une commande CL ou NEW soit donnée.
 - 2) Qu'un programme soit exécuté grâce à la commande RUN.
 - 3) Qu'une autre instruction d'affectation soit effectuée pour la même variable.
 - 4) Que l'on remplace les piles de l'ordinateur.

La mise à l'arrêt (OFF) de l'ordinateur ne change en rien les valeurs mémorisées dans les variables.

III. L'ART MYSTÉRIEUX DE LA PROGRAMMATION

L'art de la programmation a été enveloppé dans un voile de mystère pendant si longtemps que la plupart des gens l'associent avec l'art de la magie ou encore les génies mathématiques. Le fait est qu'il n'est pas requis de savoir sortir des lapins d'un chapeau ni de savoir résoudre des équations différentielles partielles. Avant tout il vous faut de la patience, une certaine capacité de raisonnement logique, le goût du détail et surtout la volonté d'apprendre. Et une disposition à relever les défis facilite certainement les choses, car à certains moments la programmation est une véritable gageure. Mais c'en est un des charmes aussi.

En tant qu'art, la programmation demande un peu de talent, un peu de formation et beaucoup de pratique. Nous n'avons pas l'intention de vous transformer en programmeur chevronné avec ce manuel. Nous voulons vous familiariser avec les opérations de base et les concepts de l'art de programmer. Pour devenir un bon conducteur de voiture, il en faut plus que de savoir le maniement du volant et la façon de passer les vitesses. Il en est de même avec la programmation.

Il existe aujourd'hui beaucoup de livres intéressants au sujet de la programmation et nous vous recommandons vivement d'aller faire un tour à la bibliothèque et chez le marchand d'ordinateurs de votre région. Vous trouverez une liste de bons livres à propos de la programmation en général et du langage BASIC en particulier dans l'appendice F.

A. Qu'est-ce qu'un programme?

Vous serez probablement étonné de découvrir qu'un programme est tout simplement une série de directives que l'ordinateur traite séparément l'une après l'autre. Il faut simplement que ces directives soient données dans un langage que l'ordinateur "comprend". Le SHARP PC-1500A parle dans un dialecte BASIC, un langage de programmation très répandu et très populaire. Comme tous les langages, le BASIC a une grammaire et un vocabulaire particuliers qu'il faut respecter en faisant des phrases. L'ordinateur vous fera remarquer votre erreur si vous lui parlez en utilisant des mots inconnus ou dans une mauvaise syntaxe. Mais il n'est pas vraiment difficile de donner les instructions correctes au SHARP. A l'origine, on a développé le langage BASIC pour enseigner les principes de la programmation et nombreuses sont les formulations qui contiennent des mots anglais devenus internationaux et des symboles familiers.

B. Comment bâtir un programme?

Quand vous vous servez du SHARP pour programmer, vous suivez une certaine routine. Les directives qui composent un programme sont entrées sur le mode PROgramme. On appelle ces directives des "instructions" dans le langage BASIC. Pour commencer l'exécution de ces instructions, il faut tout d'abord passer sur le mode RUN puis demander au SHARP d'avancer en écrivant le mot RUN. Cela ne sera pas difficile pour vous les "experts" qui ont déjà absorbé les deux programmes du chapitre 0. Pour les autres qui essaient d'avancer à grand-peine, essayons d'entrer et de passer un programme.

Mettez-vous sur le mode PROgramme et lancez la "commande" BASIC (plus de détails plus loin sur la différence entre commandes et instructions).

N **E** **W** **ENTER**

Cette opération va effacer toute instruction précédente qui pourrait être dans la mémoire. Ecrivez la ligne suivante:

Listage du programme:

```
10 PRINT "TRES BIEN!"
```

Touches utilisées

1 **0** **P** **R** **I** **N** **T** **SHIFT** **"** **T** **R** **E** **S** **SPACE**
B **I** **E** **N** **SHIFT** **!** **SHIFT** **"** **ENTER**

Notre programme en une ligne est achevé. Passez au mode RUN et écrivez:

R **U** **N** **ENTER**

Cette commande charge le SHARP de commencer à traiter les instructions (dans ce cas-ci, une seule instruction) de notre programme. Le SHARP obéit et écrit:

```
DEG      RUN      I      •  
TRES BIEN!
```

sur l'affichage. (Quand vous avez fini de lire, avertissez-en le SHARP en appuyant sur ENTER.)

Si l'on désire ajouter, modifier ou effacer quelque chose de notre programme, il faut d'abord retourner sur le mode PROgramme. S'il y avait eu une erreur dans notre programme et qu'il n'ait pas été achevé avec succès, il aurait été nécessaire de revenir sur le mode PROgramme pour corriger les instructions erronées. En somme, quand on travaille sur les programmes on se place sur le mode PROgramme, tandis que l'exécution et l'essai des programmes se fait sur le mode RUN.

C. COMMANDES et INSTRUCTIONS

Vous avez noté peut-être que dans l'exemple précédent nous avons communiqué nos désirs au SHARP avec deux méthodes différentes. Immédiatement après avoir appuyé sur ENTER, les directives NEW et RUN ont été exécutées. Ce type de directive s'appelle "commande". Par contre, la directive PRINT avait été entrée en mode PROgramme. Elle était précédée par un nombre (10), et ne fut pas exécutée immédiatement. Ce type de directive est nommée "instruction". Dans un certain sens, les commandes indiquent au SHARP ce qu'il faut qu'il fasse avec les instructions. Par exemple, la commande NEW efface toutes les instructions mises de côté précédemment. Il est important de se rappeler que l'on ne peut pas utiliser les commandes à l'intérieur d'un programme, mais que les instructions sont presque toujours groupées pour former un programme.

D. Numéros de ligne

Un programme BASIC est constitué par une série de lignes numérotées qui contiennent chacune, une ou plusieurs instructions. Ces numéros de ligne servent à maintenir l'ordre correct pendant l'exécution, mais ne feront pas partie de la sortie lors du passage du programme. On peut entrer les instructions dans n'importe quel ordre, mais leur traitement sera toujours effectué dans l'ordre déterminé par les numéros de ligne, qui, cependant, comme nous le verrons plus tard, est susceptible d'être modifié.

En guise de démonstration, ajoutons une instruction de plus à notre programme en une ligne. Passons en mode PROgramme et écrivons:

Touches utilisées

5 P R I N T SHIFT " A B S O L U M E N
T SPACE
SHIFT " SHIFT ; ENTER

Maintenant passons le programme révisé. Que se passe-t-il? Appuyez sur **ENTER** après l'apparition de "ABSOLUMENT".

Remarquez à la fois ce que le SHARP a fait pour vous et ce que vous avez fait pour vous-même. Le SHARP a arrangé et exécuté les lignes 5 et 10 dans l'ordre correct. Il faut cependant appuyer sur ENTER entre les sorties pour pousser le SHARP de la ligne 5 à la ligne 10 et voir le résultat de cette dernière.

Bien qu'il soit possible de leur assigner n'importe quel numéro de 1 à 65279, nous recommandons vivement que vous numérotiez par intervalles de 10 (c-à-d. 10, 20, 30, . . . etc.) ce qui vous permettra d'insérer jusqu'à 9 instructions entre les instructions existantes (de 11 à 19, par exemple). Certains programmeurs recommandent un intervalle plus grand même, 20 par exemple. Quoique vous en ayez rarement besoin si vous concevez et écrivez votre programme avec soin, il vaut mieux prendre cette précaution car il est bien plus simple d'étaler les numéros de 10 en 10 que d'avoir à renuméroter de nombreuses instructions plus tard.

Rappelez-vous qu'il faut éviter de donner le même numéro à deux lignes différentes, car si l'on faisait ainsi, la plus ancienne serait éliminée. On peut, bien sûr, exploiter cette caractéristique pour effacer une ligne qu'on ne veut plus, rien qu'en écrivant le numéro de celle-ci et en appuyant ensuite sur **ENTER**. De cette manière, une ligne nouvelle, même vide, peut en effacer une autre plus ancienne possédant le même numéro.

Mais si par malheur on assigne involontairement le même numéro à deux lignes différentes, on se crée des problèmes. Pour démontrer cela, entrons la ligne suivante (sur le mode PROgramme, bien sûr).

Touches utilisées

1 0 P R I N T SHIFT " H O R R I B L E
SHIFT " ENTER

Passons maintenant le programme de la même façon qu'avant. Il est "ABSOLUMENT HORRIBLE" de perdre une ligne, n'est-ce pas? Nous espérons que cela vous incitera à faire très attention quand vous écrivez des programmes, car une perte de ligne peut avoir de désagréables conséquences.

E. Touches de revue des lignes de programme

Vous vous demandez peut-être: "Comment puis-je me rappeler les lignes que j'ai entrées?" N'ayez crainte, intrépide programmeur! Cela a été prévu! Il est possible de revoir ce que l'on a entré grâce aux touches à flèche vers le haut  et flèche vers le bas . Considérez-les comme étant des touches de revue de ligne de programme. En appuyant sur la touche appropriée (en mode PROgramme) on peut "monter" ou "descendre" le long des lignes d'un programme (une opération qui se nomme "défilement" vers le haut ou le bas).

Passez au mode PROgramme et appuyez sur la touche à flèche vers le bas pour revoir les lignes de notre programme de haut en bas. Puis appuyez sur la touche à flèche vers le haut pour revenir sur vos pas au début du programme (ligne 10). Remarquez que si vous maintenez l'une de ces deux touches en position basse, les lignes seront affichées en succession automatiquement, (ce qu'il est malheureusement difficile de discerner dans le cas de notre programme à deux lignes).

Quand vous avez atteint la ligne que vous cherchez avec ces touches de revue de ligne de programme, vous pouvez passer à la mise en forme à l'aide des touches à flèche vers la gauche et flèche vers la droite. Vous vous réjouirez probablement de découvrir que le fonctionnement de ces touches en mode PROgramme est identique à celui en mode RUN (voyez le Chapitre 2). Les fonctions DELet (effacement) et INSérer peuvent être utilisées de la même manière qu'avant, pour mettre en forme des instructions.

NOTE: Quelles que soient les modifications que vous apportiez à une ligne de programmes, il faut toujours appuyer sur ENTER pour les faire exécuter par l'ordinateur. N'APPUYER PAS sur l'une des touches de revue de ligne pour passer à la ligne suivante ou précédente avant d'avoir appuyé sur , sinon la mise en forme que vous venez juste d'effectuer disparaîtra.

F. Un examen plus détaillé de certains vieux amis

Maintenant que nous savons comment entrer, exécuter et mettre en forme un programme, nous pouvons élargir notre vocabulaire d'instructions et de commandes. Mais commençons par regarder de plus près nos vieilles amies, la commande NEW et les instructions LET et PRINT.

F.1.A La commande NEW

Comme nous l'avons vu dans nos exemples de programmations précédents, la commande NEW sert à effacer toutes les lignes de programme qui existent en ce moment dans la mémoire. Pour être sûr que les seules directives présentes dans la mémoire du SHARP sont celles de notre programme actuel, nous ferons usage de la commande NEW (en mode PROgramme) avant chaque programme-échantillon. Bien qu'il soit possible et même désirable de garder plusieurs programmes dans la mémoire en même temps, nous allons différer l'utilisation de cette caractéristique afin d'éviter des confusions.

En mode PROgramme, passez la commande NEW. Quel effet ont les touches à flèche vers le haut et vers le bas maintenant?

* Comme il a été vu au début de ce manuel, l'instruction NEW peut aussi être suivie d'une valeur numérique.

Cette valeur déterminera le début de la zone programme utilisateur. Dans une machine en version de base, NEW reviendra à faire NEW & 40C5.

Cette instruction permet de protéger alors une zone pour des programmes en langage machine.

EX: NEW & 4100 ENTER

La zone allant de &40C5 à &40FF est libre pour établir des programmes en langage machine.

NEW tout seul n'effacera que les programmes en BASIC, pas cette zone.

F.1.B. L'instruction POKE et POKE

A l'aide de ces instructions, vous pourrez modifier le contenu d'une case mémoire de votre ordinateur. C'est avec l'aide de l'instruction POKE que vous écrivez vos programmes en langage machine.

Comment l'utilise-t-on ?

POKE & 40C5, & BE, & E6, & 69, &9A ENTER

Ainsi à partir de l'adresse 40C5, on a inscrit le programme suivant: 40C5 SJP E669 saut au sous-programme BEEP 40C8 RTN retour au BASIC

L'instruction POKE # permet d'écrire dans la seconde page mémoire où l'on trouve les portes d'entrée sortie.

F.1.C. L'instruction PEEK et PEEK

A l'inverse des instructions précédentes, ces instructions ne modifient pas le contenu d'une case mémoire mais en permettent la lecture.

Ainsi: PEEK & 40C5 ENTER donne 190, c'est à dire &BE

On retrouve en lisant successivement la mémoire de & 40C5 à & 40C8 le programme écrit précédemment.

PEEK # lit le contenu d'une adresse mémoire dans la seconde page.

F.1.D L'instruction CALL

Cette instruction permet d'exécuter un programme en langage machine. Ainsi, le programme écrit dans le paragraphe F.1.2 s'exécutera en faisant:

CALL & 40C5 ENTER

L'exécution de cette instruction place certaines valeurs dans les registres suivant la syntaxe utilisée:

CALL nn	Appel d'un sous-programme sans passation de paramètre
Accumulateur =	&80
Le registre X =	&C86C
Le registre Y =	&C897
Le registre U =	nn
CALL nn, Z	Appel d'un sous-programme avec passation d'un paramètre numérique
Accumulateur =	&00
Le registre X =	valeur de Z (Z entier $-\&7FFF < Z < \&7FFF$)
Le registre Y =	&C897
Le registre U =	nn

au retour si C = 0 alors le contenu de la variable Z ne change pas.

si C = 1 alors la variable Z prend pour valeur le contenu du registre U.

CALL nn, Z\$	Appel d'un sous-programme avec passation d'un paramètre alpha-numérique
Accumulateur =	Longueur totale de la chaîne ou dimension (A\$ ~ Z\$: A = &10)
Le registre X =	adresse de la variable utilisé: ici Z\$ donc &77F0
Le registre Y =	&C897
Le registre U =	nn

au retour si C = 0 alors Z\$ ne change pas

si C = 1 alors Z\$ prend pour contenu A caractères à parti de l'adresse spécifiée par U.

Touches utilisées:

1 Ø Z SHIFT \$ = SHIFT " O N SPACE

E C R I T SHIFT " ENTER

2 Ø Z Z = Ø ENTER

3 Ø P R I N T SHIFT " O U SPACE

B I E N SHIFT *

SPACE Z E R O

SHIFT " ENTER

4 Ø P R I N T Z Z ENTER

5 Ø P R I N T Z SHIFT \$ ENTER

Après le passage de ce programme, nous devrions obtenir les trois lignes de sortie suivantes (appuyez sur **ENTER** après chaque ligne que vous avez fini de lire.)

DEG RUN I •
OU BIEN, ZERO

DEG RUN I •
Ø

DEG RUN I •
ON ECRIT

Notre première instruction PRINT (ligne 30) affiche une chaîne de caractères. Remarquez que les guillemets ne sont pas imprimés en tant que sortie. Ils servent à délimiter ou à marquer le début ou la fin d'une série de caractères que l'on désire imprimer. On peut incorporer n'importe quel caractère dans cette série, sauf les guillemets.

Les lecteurs reconnaîtront certainement la deuxième et la troisième instruction PRINT (lignes 40 et 50) comme étant des variables. Quand on emploie un nom de variable dans une liste PRINT, la valeur de cette variable sera imprimée. Dans le cas présent, nous savons ce que les valeurs de ZZ et Z\$ seraient, puisque nous les avons spécifiées en lignes 10 et 20. Si l'on imprime une variable "vide", nous obtiendrons un zéro ou un espace vide, selon que la variable est numérique ou à caractère.

Les bons observateurs parmi les lecteurs auront certainement remarqué que l'ordinateur imprime les chaînes de caractères et les valeurs de variables à caractère en partant de la gauche de l'affichage. On dit qu'elles sont "cadrées à gauche". Par opposition, les nombres et les valeurs des variables numériques sont "cadrés à droite".

Il est possible aussi d'imprimer le résultat d'une expression contenue dans l'instruction PRINT. Le programme en une ligne suivant illustre cette possibilité;

Listage du programme:

10 PRINT (1982 - 1956) * 365,25

Touches utilisées:

1 0 P R I N T I 1 9 8 2 - 1 9 5 6)
* 3 6 5 . 2 5 ENTER

Comme prévu, le résultat, un nombre, est cadré à droite.

Pour obtenir le meilleur rendement possible, il vaut mieux éviter de traiter des expressions dans des instructions PRINT à moins que cette instruction (et l'expression qu'elle incorpore) ne soit exécutée qu'une seule fois au cours d'un programme.

Du fait que la plupart des programmes calculent plusieurs résultats à la fois, il arrive souvent que plusieurs éléments soient imprimés en même temps.

Dans le cas le plus simple des instructions PRINT à éléments multiples, l'affichage est divisé en deux parties. Chaque partie contient l'un des deux éléments spécifiés dans la liste d'impression. Une virgule sépare les éléments de la liste. Explorons ce format au moyen du programme suivant:

Listage du programme:

```
10 A = 2 * PI
20 PRINT "2 FOIS PI =", A
```

Touches utilisées:

1 0 A = 2 * P I ENTER
2 0 P R I N T SHIFT " 2 SPACE
F O I S SPACE P I =
SHIFT " SHIFT , A ENTER

Passez le programme. Les nombres sont cadrés à droite et les caractères à gauche, comme dans les instructions PRINT à élément unique. Dans ce cas-ci, le cadrage (l'alignement) s'effectue dans les deux parties de l'affichage.

Modifiez la ligne 20 de la façon suivante en vous servant de vos connaissances en matière de mise en forme:

```
20 PRINT A, "= 2 FOIS PI"
```

Bien qu'il y ait plusieurs façons d'effectuer cette mise en forme, effectuons-la de la manière suivante:

MODE ↓ ↑ ↓ ▶ ▶ SHIFT INS SHIFT INS A
SHIFT ↓ ▶ SHIFT INS =
▶ ▶ ▶ ▶ ▶ ▶ ▶ ▶ ▶ ▶ ▶
SHIFT DEL SHIFT DEL ENTER

La sortie de cette version modifiée devrait vous servir à identifier les deux parties de l'affichage.

Il est possible d'afficher plus de deux éléments sur la même ligne grâce à une variante de l'instruction PRINT qui emploie le point-virgule. Bâissez et vérifiez le programme suivant:

Listage du programme:

```
10 B$ = " BE "  
20 T = 2  
30 PRINT T;B$; "OR NOT"; 12/3 - 2; B$
```

Touches utilisées:

1 [0] [B] [SHIFT] [\$] [=] [SHIFT] ["] [SPACE] [B] [E] [SPACE]
[SHIFT] ["] [ENTER]
2 [0] [T] [=] [2] [ENTER]
3 [0] [P] [R] [I] [N] [T] [T] [SHIFT] [:] [B]
[SHIFT] [\$] [SHIFT] [:]
[SHIFT] ["] [O] [R] [SPACE] [N] [O] [T] [SHIFT] ["]
[SHIFT] [:] [1] [2] [/] [3] [-] [2] [SHIFT] [:]
[B] [SHIFT] [\$] [ENTER]

Le passage de cette composition électronique devrait avoir comme résultat la sortie suivante:

```
DEG RUN I •  
2 BE OR NOT 2 BE
```

C'est du Shakespeare bien connu, et cet exemple montre malgré tout l'effet de l'instruction PRINT quand les éléments à imprimer sont séparés par des points-virgules. Sous ce format, les éléments sont affichés côte-à-côte avec un minimum d'espace entre eux. Cette capacité devient très utile quand on désire créer une sortie qui a l'air "naturel", c-à-d, une sortie qui tient bien ensemble.

NOTE: Si la longueur des données affichées dépasse l'espace disponible sur l'affichage (25 caractères), les éléments situés en fin de liste ne seront pas visibles.

On peut aussi employer le point-virgule complètement à la fin de l'instruction PRINT. Dans ce cas, le point-virgule servira à indiquer que la sortie affichée à ce moment doit être gardée et que, s'il y a lieu, la nouvelle sortie (dérivant de la prochaine instruction PRINT) devra rejoindre l'ancienne sur la même ligne. Comme il est plus facile de programmer que de décrire cette opération, passons tout de suite à l'essai du programme qui suit, sans oublier la commande NEW:

Listage du programme:

```
100 PRINT "MARCEL ";  
110 PRINT "DUPONT"
```

Touches utilisées:

1 0 0 P R I N T SHIFT " M A R C E L
SPACE SHIFT " SHIFT ; ENTER
1 1 0 P R I N T SHIFT " D U P O N T
SHIFT " ENTER

Maintenant exécutez-le. Comme d'habitude, appuyez sur **ENTER** après que la première instruction a fait afficher "MARCEL". A cause du point-virgule, "MARCEL" est retenu et "DUPONT" partage l'affichage avec lui. Comparez ce résultat avec l'exécution du programme suivant dans lequel nous n'employons pas le point-virgule. La différence est évidente:

Listage du programme:

```
10 PRINT "MARCEL "  
20 PRINT "DUPONT"
```

Touches utilisées:

1 0 P R I N T SHIFT " M A R C E L
SPACE SHIFT " ENTER
2 0 P R I N T SHIFT " D U P O N T
SHIFT " ENTER

Le programme-échantillon suivant nous montre que le point-virgule peut relier autant de lignes que l'on peut placer sur l'affichage. Cependant il faut faire attention, car le SHARP ne vous avertira pas si vous commettez le péché d'imprimer trop de données sur l'affichage. C'est à celui ou à celle qui fait le programme d'éviter que cela n'arrive. Essayez le programme culturel suivant:

Listage du programme:

```
20 PRINT "DO ";  
40 PRINT "RE ";  
60 PRINT "MI ";  
80 PRINT "FA ";  
100 PRINT "SOL ";  
120 PRINT "LA ";  
140 PRINT "SI ";  
160 PRINT "DO ";
```

Touches utilisées:

2 Ø P R I N T SHIFT " D O SPACE

SHIFT " SHIFT ; ENTER

4 Ø P R I N T SHIFT " R E SPACE

SHIFT " SHIFT ; ENTER

6 Ø P R I N T SHIFT " M I SPACE

SHIFT " SHIFT ; ENTER

8 Ø P R I N T SHIFT " F A SPACE

SHIFT " SHIFT ; ENTER

1 Ø Ø P R I N T SHIFT " S O L SPACE

SHIFT " SHIFT ; ENTER

1 2 Ø P R I N T SHIFT " L A SPACE

SHIFT " SHIFT ; ENTER

1 4 Ø P R I N T SHIFT " S I SPACE

SHIFT " SHIFT ; ENTER

1 6 Ø P R I N T SHIFT " D O SHIFT "

SHIFT ; ENTER

Maintenant passez le programme, en appuyant à plusieurs reprises sur la touche **ENTER** et chantez joyeusement chaque note de la gamme . . . (ça va, ça va, vous n'avez pas besoin de chanter, mais vous devez appuyer sur **ENTER** quand même).

G. L'instruction PAUSE

L'instruction PAUSE est une forme semi-automatique de l'instruction PRINT. Elle affiche les divers éléments de la liste qui lui est associée pendant une période brève et fixe. Ceci a pour effet de libérer l'utilisateur de la nécessité d'inciter le SHARP à avancer en appuyant sur ENTER. Considérez la PAUSE comme étant une instruction PRINT suivie d'un compte à rebours. Quand ce dernier est terminé, le programme continue.

Les formats de l'instruction PAUSE sont identiques à ceux de l'instruction PRINT. Toutes les techniques discutées à propos de l'instruction PRINT valent pour l'instruction PAUSE, quoique la sortie qui en résulte diffère légèrement, cela va de soi. Pour illustrer l'une des utilisations de l'instruction PAUSE, récrivons notre programme musical:

Listage du programme:

10 PAUSE "DO ";

20 PAUSE "RE ";

30 PAUSE "MI ";

40 PAUSE "FA ";

50 PAUSE "SOL ";

60 PAUSE "LA ";

70 PAUSE "SI ";

80 PAUSE "DO";

Touches utilisées:

1 Ø P A U S E SHIFT " D O SPACE

SHIFT " SHIFT ; ENTER

2 Ø P A U S E SHIFT " R E SPACE

SHIFT " SHIFT ; ENTER

3 Ø P A U S E SHIFT " M I SPACE

SHIFT " SHIFT ; ENTER

4 Ø P A U S E SHIFT " F A SPACE

SHIFT " SHIFT ; ENTER

5 Ø P A U S E SHIFT " S O L SPACE

SHIFT " SHIFT ; ENTER

6 Ø P A U S E SHIFT " L A SPACE

SHIFT " SHIFT ; ENTER

7 Ø P A U S E SHIFT " S I SPACE

SHIFT " SHIFT ; ENTER

8 Ø P A U S E SHIFT " D O

SHIFT " SHIFT ; ENTER

Prenez note de ce qui se passe après l'impression de la dernière "note": le programme s'achève et le symbole de guidage retourne sur l'affichage. La raison en est qu'il n'y a pas d'autres instructions qui suivent la dernière PAUSÉ. Nous pouvons modifier la ligne 80 de la manière suivante pour bloquer l'affichage après la dernière note:

```
80 PRINT "DO"
```

Touches utilisées:

8 Ø P R I N T SHIFT " D O

SHIFT " ENTER

Mais après tout, il n'y a pas de raison pour qu'on ne puisse mêler les instructions PRINT et PAUSE au cours de notre programme. A vous d'en faire l'essai.

H. L'instruction INPUT

A l'aide des différentes versions de l'instruction PRINT, seules ou combinées, il est possible de présenter très clairement les données à l'utilisateur. Cependant, la plupart des éléments qui seront imprimés résultent du traitement de certaines données initiales. L'utilisateur de l'ordinateur est celui qui fournit ces données initiales au programme. L'instruction qui gouverne ce processus constitue l'instruction d'INPUT.

Dans sa forme la plus simple, INPUT nom de variable, l'instruction d'INPUT écrit simplement un ? (point d'interrogation) et attend que l'utilisateur entre les données demandées. Le contenu de la demande dépend de la variable qui apparaît comme partie intégrante de la directive d'INPUT. Par exemple, si la variable est numérique, l'utilisateur devra entrer un nombre qui sera emmagasiné dans la variable.

Est-ce que vous voyez une difficulté dans ce type d'instruction d'INPUT? Bien. Comme certains d'entre vous l'ont probablement déjà réalisé, si l'utilisateur de programme n'est pas en même temps le programmeur (et même dans ce cas, ce n'est pas certain), il ne saura pas quel type de données il devra entrer quand il verra apparaître le point d'interrogation. **C'est le travail du programmeur de garder l'utilisateur au courant à tout instant.**

Comme exemple de programme qui néglige de faire cela, nous proposons le programme suivant:

Listage du programme:

```
10 A = 0
20 INPUT A
30 PRINT A * PI
```

Touches utilisées:

```
1 0 A = 0 ENTER
2 0 I N P U T A ENTER
3 0 P R I N T A * P I ENTER
```

Imaginez maintenant le visage de l'utilisateur quand il passe ce programme . . . La première chose qui apparaît sur l'écran est un point d'interrogation. L'utilisateur tant soit peu au courant comprendra probablement qu'on lui demande d'entrer certaines données, mais il ne saura pas quelles données. Supposons même qu'il entre un nombre, au hasard, et devine juste. D'un seul coup, un numéro très long et très compliqué apparaît sur l'affichage. Qu'est-ce que cela signifie? Il appuie sur ENTER pour continuer, mais le programme s'achève. De son point de vue, toute l'opération n'est qu'une perte de temps. Pourquoi? A cause d'une piètre programmation.

La solution de ce problème est d'employer les instructions PRINT et PAUSE pour aider l'utilisateur en cours de route. Récrivons notre programme en tenant compte de cela:

Listage du programme:

```
10 A = 0
20 PAUSE "ENTREZ UN NOMBRE"
30 INPUT A
40 AP = A * PI
50 PRINT A; " FOIS PI = "; AP
```

Touches utilisées:

```
1 0 A = 0 ENTER
2 0 P A U S E SHIFT " E N T R E Z SPACE
  U N SPACE N O M B R E SHIFT " ENTER
3 0 I N P U T A ENTER
4 0 A P = A * P I ENTER
5 0 P R I N T A SHIFT ; SHIFT " SPACE
  F O I S SPACE P I SPACE =
  SPACE SHIFT " SHIFT ; A P ENTER
```

Cette version est beaucoup plus utile parce qu'elle "pousse" l'utilisateur à faire des entrées et parce qu'elle identifie la sortie.

Cette opération (impression d'une demande d'entrée) est tellement fréquente que l'on a créé des versions plus avancées de l'instruction INPUT qui l'incorpore. La première s'écrit avec le point-virgule:

INPUT "caractères" ; nom de variable

Les lecteurs attentifs ont dû reconnaître déjà notre vieille amie la chaîne de caractères. Ces chaînes apparaîtront sur l'affichage sous forme de symbole de guidage. Le curseur remplaçant le point d'interrogation suivra sur la même ligne et le SHARP attendra l'entrée de l'utilisateur. Cette version de l'instruction INPUT nous permet de récrire l'exemple précédent:

```
Listage du programme:  
10 A = 0  
20 INPUT "ENTREZ UN NOMBRE "; A  
30 AP = A * PI  
40 PRINT A; " FOIS PI = "; AP
```

Touches utilisées:

```
1 0 A = 0 ENTER  
2 0 I N P U T SHIFT " E N T R E Z SPACE  
  U N SPACE N O M B R E SPACE  
  SHIFT " SHIFT ; A ENTER  
3 0 A P = A * P I ENTER  
4 0 P R I N T A SHIFT : SHIFT " SPACE  
  F O I S SPACE P I SPACE =  
  SPACE SHIFT " SHIFT ; A P ENTER
```

Au cours du passage de ce programme, notez la manière dont l'emploi de l'instruction d'INPUT diffère de l'instruction PAUSE.

La seconde version de l'opération d'inciter une entrée est presque la même que la première, sauf que la virgule remplace le point-virgule:

INPUT "caractères", nom de variable

Quand cette version de l'instruction est exécutée, la chaîne de caractères qui lui est associée est affichée et l'ordinateur attend une entrée. Cette fois, cependant, quand l'utilisateur commence à écrire, le message "d'incitation" disparaît et les données de l'utilisateur prennent sa place. Ceci permet l'entrée de données après un très long "guidage" sans déborder de l'affichage.

Remplacez le point-virgule en ligne 20 de notre programme par une virgule et repassez le programme.

Bien sûr, l'entrée de données n'est pas limitée à des nombres comme nos exemples semblaient l'indiquer jusqu'à présent. Il suffit de spécifier une variable à caractère pour entrer des caractères, comme le brillant programme déductif ci-dessous le démontre:

Listage du programme:

```
10 INPUT "ENTREZ VOTRE NOM "; L$
20 INPUT "ENTREZ VOTRE PRENOM "; F$
30 I$ = LEFT$(F$, 1) + " " + LEFT$(L$, 1) + " ."
40 PAUSE "OH,"
50 PRINT "VOS INITIALES SONT "; I$
```

Touches utilisées:

1 Ø I N P U T SHIFT " E N T R E Z SPACE
V O T R E SPACE N O M
SPACE SHIFT " SHIFT :
L SHIFT \$ ENTER

2 Ø I N P U T SHIFT " E N T R E Z SPACE
V O T R E SPACE P R E N O M
SPACE SHIFT "
SHIFT : F SHIFT \$ ENTER

3 Ø I SHIFT \$ = L E F T SHIFT \$ (F
SHIFT \$ SHIFT , 1
) + SHIFT " . SHIFT " +
L E F T SHIFT \$ (L SHIFT \$
SHIFT , 1) + SHIFT " .
SHIFT " ENTER

4 Ø P A U S E SHIFT " O H SHIFT ,
SHIFT " ENTER

5 Ø P R I N T SHIFT " V O S SPACE
I N I T I A L E S SPACE S O N T SPACE
SHIFT " SHIFT : I SHIFT \$ ENTER

P.S: Ne vous souciez pas de la ligne 30: on s'y est servi de techniques avancées que vous apprendrez plus tard.

Il est possible d'utiliser l'instruction INPUT pour rassembler et stocker plusieurs valeurs tout aussi bien qu'une seule donnée élémentaire. Pour programmer une opération de ce type, il suffit de dresser une liste des variables destinées à contenir des données pour l'intégrer dans l'instruction INPUT. Chaque variable de la liste devra être séparée des autres par une virgule.

Listage du programme:

```
10 W = 0 : X = 0 : Y = 0 : Z = 0
20 INPUT "ENTREZ 4 NOMBRES", W, X, Y, Z
30 S = W + X + Y + Z : A = S / 4
40 PAUSE "SOMME = "; S
50 PRINT "MOYENNE "; A
```

Touches utilisées:

```
1 0 W = 0 SHIFT : X = 0 SHIFT : Y = 0
  SHIFT : Z = 0 ENTER
2 0 I N P U T SHIFT " E N T R E Z SPACE
  4 SPACE
  N O M B R E S SHIFT " SHIFT ↵
  W SHIFT ↵ X SHIFT ↵ Y SHIFT ↵
  Z ENTER
3 0 S = W + X + Y + Z SHIFT : A = S / 4
  ENTER
4 0 P A U S E SHIFT " S O M M E SPACE
  = SPACE SHIFT "
  SHIFT : S ENTER
5 0 P R I N T SHIFT " M O Y E N N E SPACE
  SHIFT " SHIFT : A ENTER
```

Lors de son passage, ce programme vous incitera à entrer 4 nombres. Le premier nombre que vous écrivez remplacera le symbole de guidage parce que nous avons utilisé une virgule immédiatement après la chaîne de caractères dans l'instruction INPUT. Vous devez appuyer sur **ENTER** après avoir terminé l'écriture du premier nombre. Le point d'interrogation précédera chaque nombre suivant après lesquels vous devrez appuyer chaque fois sur **ENTER**.

Si nous avons employé un point-virgule dans l'instruction INPUT, le premier nombre entré n'aurait pas remplacé le symbole de guidage, mais aurait partagé la ligne avec lui. Les autres entrées successives seraient les mêmes qu'avec la virgule. Faites-en l'essai en changeant la ligne 20 de la manière suivante:

```
20 INPUT "ENTREZ 4 NOMBRES"; W, X, Y, Z
```

Touches utilisées:

```
2 0 I N P U T SHIFT " E N T R E Z
  SPACE 4 SPACE
  N O M B R E S SHIFT " SHIFT ;
  W SHIFT ↵ X SHIFT ↵ Y SHIFT ↵
  Z ENTER
```

I. Raccourcis et conseils utiles

Cette partie constitue une petite récompense pour votre patience et votre assiduité. Vous préféreriez probablement de l'argent bien sonnante, mais certains des conseils et données ci-dessous valent leur pesant d'or.

I.1. Abréviations

Ceux d'entre vous dont les doigts sont un peu lents à voler d'une touche à l'autre ont dû remarquer que nos programmes-échantillons sont devenus plus longs très rapidement. Lors de la mise au point du SHARP, ses créateurs ont prévu vos difficultés et ont donné au SHARP la capacité de reconnaître des abréviations d'instructions et de commandes fréquemment utilisées.

De manière générale, une abréviation se constitue d'une ou de plusieurs lettres suivies d'un point. Le point est essentiel pour éviter la confusion avec les noms de variables. Dans l'exemple qui suit, les instructions du programme sont en abrégé:

Listage du programme:

```
15 PA. "BONJOUR, HUMAIN."  
25 I. "QUEL EST VOTRE NOM?"; N$  
35 P. "ENCHANTE, "; N$
```

Touches utilisées:

```
1 5 P A . SHIFT " B O N J O U R SHIFT ,  
H U M A I N . SHIFT " ENTER  
2 5 I . SHIFT " Q U E L SPACE E S T SPACE  
V O T R E SPACE N O M SHIFT ? SHIFT "  
SHIFT ; N SHIFT $ ENTER  
3 5 P . SHIFT " E N C H A N T E  
SHIFT → SPACE  
SHIFT " SHIFT ;  
N SHIFT $ ENTER
```

Lorsque vous appuyez sur ENTER à la fin de chaque ligne, vous noterez que le SHARP développe toutes les abréviations situées sur la ligne. La clarté qui en résulte sera énormément utile plus tard lorsque vous vérifierez s'il n'y a pas d'erreur dans le programme. Comme bénéfice supplémentaire caché, vous trouverez que l'espace nécessaire pour mémoriser les abréviations d'instructions n'est ni plus, ni moins que l'espace nécessaire pour mémoriser les instructions dans leur forme complète. La possibilité d'abrégé rend simplement la programmation plus aisée pour l'utilisateur.

Cependant, dans le but d'en faciliter la lecture, nous n'emploierons pas ces abréviations dans le listage de nos programmes-échantillons. Mais cela ne vous empêche pas de vous en servir, bien sûr, quand vous entrez un programme.

Suivent les abréviations acceptables pour des commandes et des instructions que nous connaissons déjà:

PRINT	P. PR. PRI.
PAUSE	PA. PAU.
INPUT	I. IN. INP.
RUN	R.

Une liste complète d'abréviations se trouve dans un appendice à la fin du manuel.

1.2. Les instructions multiples et le double-point

Comme nous l'avons déjà dit auparavant dans la section concernant les numéros de ligne, plusieurs instructions peuvent occuper la même ligne. Le SHARP exécute ces instructions de la gauche vers la droite. Pour permettre au SHARP de distinguer la fin d'une instruction du début de la suivante, un caractère indicateur s'avère nécessaire, et l'on a choisi le double-point (:) pour remplir cette fonction. Sa fonction, dans ce contexte, est semblable à celle du point dans les phrases normales. Il indique un arrêt dans la lecture.

Plusieurs exemples de l'utilisation du double-point ont déjà apparus auparavant dans nos programmes-échantillons. En règle générale, elle prend la forme suivante:

numéro de ligne instruction1 : instruction2 : instruction3 (etc)

"Quand utiliser le double-point?" est une question de style de programmation. L'utilisation à l'aveuglette du double-point dans l'écriture de programmes très denses produit des programmes qui sont très difficiles à lire, à corriger ou à élargir. Et puisque le PC-1500A a comme avantages principaux, la faculté de s'adapter à votre personnalité et la faculté d'agir réciproquement, il est désirable que vous puissiez modifier et élargir des programmes selon vos besoins. Nous vous recommandons donc de faire un usage modéré du double-point.

Il est très profitable de grouper seulement les instructions ayant un rapport conceptuel entre elles quand on se sert du double-point. C'est-à-dire que vous devriez placer sur la même ligne uniquement les instructions destinées à effectuer une tâche particulière parmi la multitude de tâches qui forment le programme.

Comme exercice, examinez les instructions de programme suivantes qui introduisent 3 nombres, trouvez leur moyenne, calculez de combien chacun s'éloigne de cette moyenne et imprimez la somme de leurs différences:

Listage du programme:

```

10 N1 = 0 : N2 = 0 : N3 = 0
20 INPUT "ENTREZ 3 NOMBRES", N1, N2, N3
30 A = (N1 + N2 + N3) / 3
40 D1 = N1 - A : D2 = N2 - A : D3 = N3 - A
50 SD = D1 + D2 + D3
60 PRINT "SOMME DES DIFFERENCES = "; SD

```

Touche utilisées:

```
1  Ø N 1 = Ø SHIFT : N 2 = Ø SHIFT : N 3
   = Ø ENTER
2  Ø I N P U T SHIFT " E N T R E Z SPACE 3
   SPACE N O M B R E S SHIFT " SHIFT ,
   N 1 SHIFT , N 2 SHIFT , N 3 ENTER
3  Ø A = I N 1 + N 2 + N 3 ) / 3 ENTER
4  Ø D 1 = N 1 - A SHIFT : D 2 = N 2 - A
   SHIFT : D 3 = N 3 - A ENTER
5  Ø S D = D 1 + D 2 + D 3 ENTER
6  Ø P R I N T SHIFT " S O M M E SPACE
   D E S SPACE D I F F E R E N C E S
   = SPACE
   SHIFT " SHIFT : S D ENTER
```

Notez que chaque ligne du programme effectue une opération complète et nécessaire. La ligne 10 efface toutes les valeurs que les variables N1, N2 et N3 auraient pu contenir jusqu'à présent (une précaution destinée à prévoir le cas où l'utilisateur oublierait d'entrer les 3 nombres). La ligne 20 recueille les données de l'utilisateur. La ligne 30 fait la moyenne des nombres. La ligne 40 calcule les différences. La ligne 50 fait la somme des différences et la ligne 60 imprime le résultat.

Les instructions de chaque ligne ne correspondent pas accidentellement à la description en langage courant du programme. Bien au contraire, c'est en accordance avec les principes de la bonne programmation, principes que nous essayons de vous inculquer.

A la différence de l'utilisation des abréviations, l'utilisation du double-point a un effet sur la quantité de mémoire utilisée pour stocker le programme. Voilà la raison principale de l'utilisation du double-point pour placer plusieurs instructions sur la même ligne. Chaque numéro de ligne réserve plusieurs "phases" (unité de mémoire) de programme et par conséquent, moins il y a de lignes dans un programme, moins on occupe de mémoire dans ce programme.

En conclusion, nous dirons, au sujet de l'emploi du double-point, que chaque programmeur devra garder un équilibre entre la lisibilité et la flexibilité du programme et la mémoire nécessaire pour son application.

J. La correction d'erreurs sur le mode PROgramme

Quoique les abréviations et les double-points nous facilitent l'entrée des programmes, ils ne peuvent empêcher même les meilleurs d'entre nous de faire des erreurs. Même les programmeurs professionnels ne réussissent pas à découvrir leurs propres erreurs quand ils passent en revue leurs programmes. Nous voulons dire par là que tôt ou tard vous rencontrerez une erreur pendant ou après le passage de votre programme. En général ces erreurs peuvent être corrigées facilement, si vous les considérez comme part d'une énigme à résoudre et si vous localisez le problème soigneusement. Vous serez aidé dans cette tâche par plusieurs caractéristiques du PC-1500A. Quand le SHARP rencontre une instruction incorrecte, il interrompt ses opérations et vous indique, le problème avec un message laconique comme le suivant.

```
                                RUN
ERROR 1 IN 20
```

Dans le but d'illustrer ce cas, passons un programme dans lequel nous avons introduit délibérément une erreur:

Listage du programme:

```
25 PAUSE "TOUT EST BIEN"  
50 PRINT "QUI FINIT BIEN"
```

Touches utilisées:

```
2 5 P A U S E SHIFT " T O U T SPACE  
E S T SPACE B I E N SHIFT " ENTER  
5 0 P R I M T SHIFT " Q U I SPACE F I N I T  
SPACE B I E N SHIFT " ENTER
```

Passons le programme. Quand le message d'erreur apparaît, appuyez sur la  touche. Tant que vous appuiez sur cette touche, la ligne qui a troublé le SHARP apparaîtra sur l'affichage. La "grille clignotante" fournira PEUT-ETRE un indice de la nature du problème.

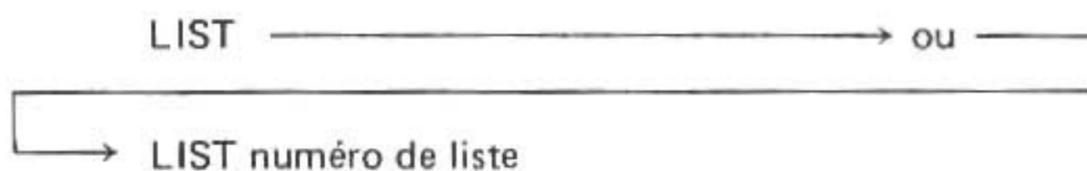
Pour corriger l'instruction erronée, appuyer sur  pour quitter le programme et passez au mode PROgramme. Appuyez sur la touche, mais ne la maintenez pas en position basse. L'affichage va indiquer à nouveau la ligne erronée:

```
50 PRINT "QUI FINIT BIEN"
```

Vous pouvez maintenant passer à la mise en forme de cette ligne en vous servant des touches INSérer, DEL (effacer) et les touches à flèches que vous connaissez déjà. N'oubliez pas d'appuyer sur  quand vous avez fini pour ordonner au SHARP de stocker la ligne corrigée.

K. La commande LIST

Une autre façon d'afficher une ligne définie d'un programme, consiste à employer la commande LIST dont la forme est la suivante:



Si l'on ne donne pas de numéro de ligne, la première ligne du programme sera affichée.

Si l'on spécifie un numéro de ligne, la ligne en question apparaîtra sur l'affichage. Si aucune ligne du programme n'a le numéro donné, la ligne suivante avec un numéro plus grand, apparaîtra sur l'affichage. Par exemple, dans le programme suivant:

```
15 PRINT "POLICHINELLE"  
30 PRINT "FUT UN";  
45 PRINT "CLOWN"
```

la commande LIST 40 amènera la ligne 45 sur l'affichage. La commande LIST amène la ligne 15 et la commande LIST 30, la ligne 30.

NOTE: Si vous spécifiez une ligne dont le numéro est plus grand que les numéros de ligne en existence dans le programme, vous obtiendrez le message d'erreur ERROR 11.

L. Plus on est de fous, plus on rit

Comme nous l'avons déjà signalé auparavant, il est possible de garder plus d'un programme à la fois dans la mémoire. La ruse est de donner à chaque programme sa propre gamme de numéros de ligne. Par exemple, on pourra donner à un programme les numéros de 10 à 200 tandis que l'autre recevra les numéros de ligne de 300 à 500. Méfiez-vous, bien sûr, du mélange accidentel d'instructions relevant de programmes différents (au moyen d'un numérotage incorrect), car vous courrez le risque d'obtenir des résultats bizarres.

L.1. L'instruction END

Un autre problème que l'on rencontre quand on mémorise plusieurs programmes simultanément est celui qui découle du fait que chaque programme n'est en fait qu'un groupe d'instructions numérotées qui sont exécutées dans l'ordre ascendant. Comment est-ce que le SHARP saura quand il a terminé l'exécution d'instructions d'un programme donné? La réponse est qu'il ne peut pas le deviner si on ne le lui indique pas. L'instruction dont on se sert pour indiquer à l'ordinateur qu'il est arrivé au bout d'un programme, est l'instruction END.

Jusqu'à présent nous ne nous sommes pas servis et, nous n'avons pas eu besoin de l'instruction END. Le SHARP a toujours exécuté toutes les lignes de notre programme dans l'ordre ascendant jusqu'à ce qu'il arrive à la ligne finale. A ce moment, il a conclu que le programme était terminé et s'est remis en position d'attente pour notre prochaine commande. Maintenant, par contre nous voulons indiquer au SHARP qu'il doit s'arrêter d'exécuter les instructions avant de passer au programme suivant.

Entrons les lignes suivantes pour illustrer l'usage de programmes multiples:

```
10 PAUSE "QUE DIT"  
20 PAUSE "UN HOMME"  
30 PRINT "QUI SE NOIE?"  
40 END  
  
200 PAUSE "IL"  
210 PRINT "DISPARAIT"  
220 END
```

L.2. Numéro de ligne de RUN

Bon, maintenant que vous avez deux programmes dans la mémoire, comment allez-vous les passer séparément? Ceux qui ont le goût de l'aventure ont peut-être essayé d'utiliser déjà la commande RUN ordinaire et ont découvert qu'elle convient parfaitement pour démarrer le premier programme. Voyons maintenant la procédure à suivre pour le second programme. Il va falloir que nous employions une variante de la commande RUN qui indiquera au SHARP à quelle ligne il doit commencer. C'est la commande de numéro de ligne de RUN. Pour démarrer le second programme, écrivez:

```
R U N 2 0 0
```

et voilà, c'est fait!

Cette commande, comme la plupart des commandes qui nous sont les plus utiles, peut malgré tout nous poser des problèmes. Parce qu'elle indique au SHARP l'endroit à partir duquel il devra exécuter les instructions et parce que SHARP est un serviteur si fidèle, il est possible de démarrer un programme au milieu. Ceci comme vous devez le deviner ne devrait pas être sujet à un choix.

Si on le fait, eh bien . . . les résultats ont tendance à être quelque peu extraordinaires. Faites-en l'essai avec notre premier programme en donnant la commande:

RUN 30

L'erreur est légère dans ce cas, mais imaginez ce qui se passerait dans un programme complexe!

M. Instructions de contrôle

Jusqu'à présent notre programme était constitué d'une séquence d'instructions exécutées une fois par l'ordinateur. Ceux qui ont de l'expérience à donner des instructions se rendent compte peut-être que cela ne constitue pas toujours le meilleur moyen d'accomplir une tâche. Souvent l'on donne à celui qui écoute la possibilité de choisir entre plusieurs plans. Parfois l'on "condense" des instructions avec une commande du type: "Et maintenant retracez les 3 derniers pas jusqu'à ce que vous ayez terminé."

Ces capacités ont été incorporées dans des instructions en BASIC que l'on appelle instructions de contrôle. Elles déterminent si, quand et comment s'effectuera l'exécution des autres instructions. Ces instructions de contrôle permettent de façonner des programmes très puissants et très souples. Dans les sections qui suivent nous allons examiner les IF . . . THEN, les GOTO et les GOSUB qui forment les principales instructions de contrôle du langage BASIC.

N. IF (si) et THEN (alors)

L'instruction IF offre la possibilité de faire des choix durant un programme BASIC. Equipé d'une instruction IF, le programme peut évaluer votre entrée et prendre des décisions:

Listage du programme:

```
10 PAUSE "DORMEZ VOUS?"
20 INPUT "ESCRIVEZ OUI OU NON "; SX$
30 IF SX$ = "OUI" THEN PRINT "OH, EXCUSEZ MOI"
40 END
```

Touches utilisées:

```
1 0 P A U S E SHIFT " D O R M E Z SPACE
  V O U S SHIFT ?
  SHIFT " ENTER
2 0 I N P U T SHIFT " E S C R I V E Z
  SPACE
  O U I SPACE O U SPACE N O N SPACE
  SHIFT " SHIFT ; S X SHIFT $ ENTER
3 0 I F S X SHIFT $ = SHIFT " O U I SHIFT "
  T H E N P R I N T SHIFT " O H
  SHIFT ↵ SPACE E X C U S E Z SPACE M O I
  SHIFT " ENTER
4 0 E N D ENTER
```

Ce programme soporifique ne produira une sortie que si vous répondez affirmativement. La forme générale de l'instruction IF, illustrée par la ligne 30, est la suivante:

IF condition THEN instruction

Pendant le passage, l'épreuve renfermée dans la proposition IF est exécutée. L'exécution ou la non-exécution de l'instruction dépend du résultat de l'épreuve. Ordinairement cette épreuve est une fonction logique et on l'appelle "condition". Souvenez-vous que les fonctions logiques sont des comparaisons qui sont soit vraies soit fausses. Si la fonction logique est vraie l'instruction sera exécutée, si elle est fausse, l'instruction sera sautée.

Dans notre programme-échantillon l'épreuve consiste à déterminer si la variable SX\$ est égale à (contient) la chaîne de caractères "F". Si elle est égale-et seulement dans ce cas-l'ordinateur ignorera l'instruction PRINT. Dans les deux cas, que l'instruction PRINT soit exécutée ou non, le SHARP passera à la ligne suivante (qui dans le cas de notre programme est la ligne 40).

Notez que nous aurions pu inverser notre épreuve en modifiant plusieurs lignes de la manière suivante:

```
30 IF SX$ = "NON" THEN END
40 PRINT "OH, EXCUSEZ MOI"
```

Ce programme n'est pas exactement le même que l'original. Il permet à notre ordinateur de communiquer avec un utilisateur qui fait une faute de frappe ou qui ne répond pas négativement. De plus, ce programme a, en fait, deux fins: l'une déterminée par l'instruction END à la ligne 30 et la fin implicite au-delà de la ligne 40. Ce n'est pas une bonne coutume de programmer plusieurs fins dans un même programme. Cependant, notre renversement de ces instructions démontre qu'un arrangement correct des instructions et une bonne mise à l'épreuve sont nécessaires au fonctionnement correct d'un programme. Dans la prochaine section, nous aurons l'occasion de voir une troisième façon d'exprimer notre programme à l'aide de l'instruction GOTO qui résoudra les problèmes rencontrés durant la seconde.

Bien que la condition d'une instruction IF soit généralement une inégalité, il n'en est pas nécessairement ainsi. Dans le PC-1500A BASIC, toute expression qui s'évalue à non-zéro est considérée comme Vraie et toute expression qui s'évalue à zéro est considérée comme Fausse. Ceci explique pourquoi les inégalités marchent en tant que conditions: il faut se rappeler qu'une inégalité qui est Vraie retourne un 1 et que les inégalité qui est Vraie retourne un 1 et que les inégalités qui sont Fausse retournent un 0.

Un autre rappel est à l'ordre du jour. Si l'instruction qui suit le THEN est une instruction d'affectation, le mot-clé **DOIT** être employé. Si l'on néglige de le faire, il en résultera un message d'erreur. Reportez-vous à la section sur l'instruction LET où nous avons examiné ce type de situation.

Avertissement: Voir page 171.

O. L'instruction GOTO

Dans la section précédente vous avez dû remarquer que nos choix étaient limités après le test dans l'instruction IF. Il nous était permis d'exécuter seulement une instruction si la fonction logique était vraie. Par souci de commodité nous voudrions exécuter plusieurs instructions et c'est l'instruction GOTO qui va nous le permettre.

L'instruction GOTO modifie la "course" de l'exécution de l'instruction. Elle ordonne au SHARP "d'aller à" ("go to" en anglais) une ligne autre que celle qui suivrait normalement et de commencer l'exécution d'instructions dans l'ordre à partir de cette ligne. Par conséquent, on saute certaines instructions complètement. Voyons cela en passant en revue le programme qui suit:

Listage du programme:

```
10 PAUSE "EVITEZ ";
20 GOTO 50
30 PRINT X * 3 / 4 + 2
40 PRINT "TOUT LIVRE AYANT ";
50 PRINT "DES ERREURS!"
60 END
```

Touches utilisées:

```
1 [ ] [ ] [ P ] [ A ] [ U ] [ S ] [ E ] [SHIFT] [ " ] [ E ] [ V ] [ I ] [ T ] [ E ] [ Z ]
   [SPACE] [SHIFT] [ " ] [SHIFT] [ ; ] [ENTER]
2 [ ] [ ] [ G ] [ O ] [ T ] [ O ] [ 5 ] [ ] [ENTER]
3 [ ] [ ] [ P ] [ R ] [ I ] [ N ] [ T ] [ X ] [ * ] [ 3 ] [ / ] [ 4 ] [ + ] [ 2 ] [ENTER]
4 [ ] [ ] [ P ] [ R ] [ I ] [ N ] [ T ] [SHIFT] [ " ] [ T ] [ O ] [ U ] [ T ] [SPACE] [ L ] [ I ] [ V ]
   [ R ] [ E ]
   [SPACE] [ A ] [ Y ] [ A ] [ N ] [ T ]
   [SHIFT] [ " ] [SHIFT] [ ; ] [ENTER]
5 [ ] [ ] [ P ] [ R ] [ I ] [ N ] [ T ] [SHIFT] [ " ] [ D ] [ E ] [ S ] [SPACE]
   [ E ] [ R ] [ R ] [ E ] [ U ] [ R ] [ S ] [SHIFT] [ ! ] [SHIFT] [ " ] [ENTER]
6 [ ] [ ] [ E ] [ N ] [ D ] [ENTER]
```

Ordinairement l'exécution de ce programme s'effectuerait dans l'ordre des numéros de ligne du plus petit au plus grand numéro. Le GOTO entré dans la ligne 20 a pour effet de forcer le SHARP à passer tout de suite à la ligne 50 et de commencer l'exécution à partir de là. Les lignes 30 et 40 ne seront pas exécutées.

La forme générale de l'instruction GOTO est la suivante:

GOTO expression

dans laquelle:

L'expression est évaluée à un nombre qui est un numéro valide de ligne d'un programme (c'est-à-dire de 1 à 65279).

NOTE: Si l'on spécifie un numéro de ligne qui n'existe pas, on obtient un message d'erreur ERROR 11.

Si, en conséquence d'un certain choix, nous décidons de faire exécuter plusieurs instructions, nous devons nous servir d'instructions GOTO en conjonction avec l'instruction IF.

```

10 IF (si) test THEN GOTO (alors allez à) 100
20 { instructions ici sont
   :   exécutées seulement si
   :   le test est faux }
90 GOTO 200
100 { instructions ici sont
    :   exécutées seulement si
    :   le test est vrai }
200 { instructions ici sont
    :   toujours exécutées }
999 END

```

La logique de ce programme est applicable dans de nombreuses situations. Il est possible d'insérer autant d'instructions que l'on veut dans chacune des sections formées par les instructions GOTO.

Comme exemple de cette structure dans la pratique, le segment de programme de compte courant qui suit, détermine si le montant d'une entrée est un dépôt (nombre positif) ou un retrait (nombre négatif). L'utilisateur entre un solde initial:

Listage du programme:

```

10 INPUT "SOLDE INITIAL?", B
20 INPUT "MONTANT?", TA
25 IF TA = 0 THEN GOTO 80
30 B = B + TA
40 IF TA < 0 THEN GOTO 70
50 PRINT "DEPOT DE $"; TA; " INSCRIT"
60 GOTO 80
70 PRINT TA;" DOLLARS RETIRES"
80 PRINT "SOLDE FINAL = "; B
90 END

```

Touches utilisées:

```

1  Ø I N P U T SHIFT " S O L D E
   SPACE I N I T I A L
   SHIFT ? SHIFT " SHIFT , B ENTER
2  Ø I N P U T SHIFT " M O N T A N T
   SHIFT ? SHIFT " SHIFT , T A ENTER
2  5 I F T A = 0 T H E N G O T O
   8 Ø ENTER
3  Ø B = B + T A ENTER

```

```

4  Ø I F T A SHIFT < Ø T H E N
    G O T O 7 Ø ENTER
5  Ø P R I N T SHIFT " D E P O T
    SPACE D E SPACE SHIFT $
    SHIFT " SHIFT ; T A SHIFT ;
    SHIFT " SPACE I N S C R I T SHIFT "
    ENTER
6  Ø G O T O 8 Ø ENTER
7  Ø P R I N T T A SHIFT : SHIFT "
    SPACE D O L L A R S SPACE R E
    T I R E S SHIFT " ENTER
8  Ø P R I N T SHIFT " S O L D E
    SPACE F I N A L = SPACE
    SHIFT " SHIFT ; B ENTER
9  Ø E N D ENTER

```

Comme vous le voyez, ce programme n'a de fin que si l'opération se ramène à zéro. La deuxième instruction IF illustre la structure mentionnée auparavant. Remarquez que, en plus des deux actions différentes associées avec l'instruction IF (ligne 50 et 70), il y a d'autres instructions (ligne 80 et 70) que l'ordinateur exécute sans tenir compte du résultat de la mise à l'épreuve du IF.

Maintenant nous pouvons passer à l'écriture d'une troisième version du programme de la section précédente:

Listage du programme:

```

10 PAUSE "DORMEZ VOUS?"
20 INPUT "ECRIVEZ OUI OU NON "; SX$
30 IF SX$ <> "OUI" THEN GOTO 99
40 PRINT "OH, EXCUSEZ MOI"
99 END

```

Touches utilisées:

```

1  Ø P A U S E SHIFT " D O R M E Z SPACE
    V O U S SHIFT ?
    SHIFT " ENTER
2  Ø I N P U T SHIFT " E C R I V E Z SPACE
    O U I SPACE O U SPACE N O N SPACE
    SHIFT " SHIFT ; S X SHIFT $ ENTER
3  Ø I F S X SHIFT $ SHIFT < SHIFT >
    SHIFT " O U I SHIFT " T H E N
    G O T O 9 9 ENTER

```

4 Ø P R I N T SHIFT " O H SHIFT y
 SPACE E X C U S E Z SPACE M O I
 SHIFT "
 ENTER
 9 9 E N D ENTER

Cette variante inverse l'preuve conditionnelle en spécifiant qu'une certaine opération doit être effectuée si l'entrée faite par l'utilisateur n'est PAS égale au "OUI" ("YES"). Elle élimine de ce fait les problèmes de la deuxième version et l'on pourrait la développer dans la forme élargie de l'instruction IF décrite ci-dessus. Ce type de réarrangement d'instructions s'avère très utile en programmation. Si vous prenez la peine de faire l'expérience par vous-même, vous obtiendrez de meilleurs programmes comme récompense.

Un autre usage très commun de l'instruction GOTO consiste à lui faire répéter l'exécution d'une série d'instructions. Cette opération s'appelle un "bouclage". Le programme ci-dessous illustre un bouclage simple:

Listage du programme:

```
10 WAIT 30
20 PRINT "TEUF TEUF TEUF TEUF"
30 PRINT "    TEUF TEUF TEUF TEUF"
40 GOTO 20
```

Touches utilisées:

1 Ø W A I T 3 Ø ENTER
 2 Ø P R I N T SHIFT " T E U F SPACE
 T E U F SPACE T E U F SPACE T E U F
 SHIFT " ENTER
 3 Ø P R I N T SHIFT " SPACE SPACE
 SPACE SPACE T E U F SPACE T E U F
 SPACE T E U F SPACE T E U F SHIFT "
 ENTER
 4 Ø G O T O 2 Ø ENTER

Malheureusement ce programme continuera à faire teuf-teuf à perpétuité. (Vous pouvez l'arrêter à l'aide de la touche BREAK). C'est à nous de fournir un moyen de faire s'arrêter de tels programmes. Grâce à l'emploi d'un "compteur" avec l'instruction IF, nous pouvons réaliser cela. Le compteur est une variable sous laquelle nous notons combien de fois nous avons fait quelque chose (c'est-à-dire que nous "comptons"). Grâce à cette technique, l'instruction IF vérifie si l'ordinateur a exécuté nos instructions PRINT un nombre de fois déterminé. C'est à nous, bien sûr, de déterminer combien de fois. Choisissons 10 fois pour chaque instruction PRINT en guise de vérification (au delà de 10 cela deviendrait ennuyeux).

Servons-nous du compteur et de l'instruction IF pour écrire:

Listage du programme:

```
10 WAIT 30
15 C = 1
20 PRINT "TEUF TEUF TEUF TEUF"
30 PRINT "    TEUF TEUF TEUR TEUF "
40 C = C + 1
50 IF C <= 10 THEN 20
60 END
```

Touches utilisées:

```
1 0 W A I T 3 0 ENTER
1 5 C = 1 ENTER
2 0 P R I N T SHIFT " T E U F SPACE
    T E U F SPACE T E U F SPACE T E
    U F SHIFT " ENTER
3 0 P R I N T SHIFT " SPACE SPACE
    SPACE SPACE T E U F SPACE T E U F
    SPACE T E U F SPACE T E U F SHIFT "
    ENTER
4 0 C = C + 1 ENTER
5 0 I F C SHIFT < = 1 0 T H E N
    2 0 ENTER
6 0 E N D ENTER
```

Examinez le fonctionnement du compteur lors de l'exécution de chaque boucle et vous remarquerez que nous devons affecter une valeur initiale au compteur à la ligne 15, puis l'augmenter à la ligne 40 pour réfléchir une exécution de plus des instructions 20 et 30.

Il y a beaucoup d'autres possibilités d'emploi des compteurs et des boucles au cours des programmes. Malheureusement l'espace nous manque ici pour les décrire. Il vous faudra vous reporter à l'un des livres listés dans l'Appendice F.

L'instruction GOTO est une commande en même temps. Son usage en tant que commande diffère naturellement de son usage en tant qu'instruction. Lancée comme commande sur le mode RUN, le GOTO débute l'exécution d'un programme d'une manière identique à la commande RUN. Nous trouverons la différence dans certains préparatifs internes qui sont faits avant l'exécution des instructions du programme. (Pour une comparaison des méthodes utilisées pour démarrer un programme, reportez-vous à la section intitulée "Commencer l'exécution d'un programme" dans le Chapitre V). A la différence de la commande RUN, la commande **GOTO n'efface pas les valeurs mémorisées dans les variables avant de commencer l'exécution d'un programme.**

Pour démarrer l'exécution d'un programme à l'aide de la commande GOTO, entrez:

GOTO numéro de ligne

où:

numéro de ligne est constitué par le numéro de la première ligne du programme destiné à être exécuté.

NOTE: Si vous spécifiez un numéro de ligne qui n'existe pas, vous obtiendrez un message d'erreur ERROR 11.

P. FOR ... NEXT

Comme nous l'avons vu dans la section précédente, il est très utile de pouvoir répéter une série d'instructions. En fait, on exploite cette faculté tellement souvent qu'on a incorporé dans le BASIC plusieurs instructions qui rendent cette opération automatique. Ce sont l'instruction FOR et son partenaire, l'instruction NEXT. Ensemble, ces deux instructions renferment une série d'instructions qui sont répétées plusieurs fois. L'instruction FOR est associée avec un compteur et comprend un test conditionnel intégré. Elle permet aussi de spécifier la valeur initiale et l'augmentation de la valeur de la variable-compteur.

La forme de toutes ces données est la suivante:

FOR variable-compteur = valeur initiale TO valeur finale STEP valeur ajoutée

où:

"Variable-compteur" constitue le nom de la variable qui sert à compter les boucles.

la valeur initiale est constituée par la valeur mémorisée sous la variable-compteur avant le premier passage par la boucle. La gamme permise pour cette valeur va de -32768 à 32767.

la valeur finale constitue le nombre employé dans le test. Si la variable-compteur contient une valeur supérieure à la valeur finale, le bouclage est terminé. La gamme permise pour ce nombre va de -32768 à 32767.

STEP valeur ajoutée constitue une proposition facultative. La valeur ajoutée indique de combien on augmente ou diminue la variable-compteur à chaque passage par la boucle. Sa gamme permise va de -32768 à 32767. Si l'on omet entièrement cette proposition, on considère que la valeur ajoutée est égale à un.

Comme cela doit paraître un peu difficile à assimiler, passons à l'observation de programmes-échantillons simples. Le premier ressemble à celui où nous avons employé le compteur. Au lieu de faire TEUF-TEUF, nous imprimons la valeur de la variable-compteur C:

```
15 FOR C = 1 TO 10
30 PAUSE C
50 NEXT C
```

(Pour obliger ce programme à faire "TEUF" comme auparavant, insérez simplement les instructions 10, 20 et 30 de ce programme). Remarquez que cette version est plus propre et plus concise parce qu'elle utilise moins d'instructions pour effectuer les mêmes fonctions de comptage et de bouclage que la version précédente.

Dans le cas où il y aurait toujours encore des malentendus au sujet de l'action des instructions FOR et NEXT, nous vous présentons une comparaison d'une boucle FOR ... NEXT et les instructions équivalentes:

<pre> 10 FOR I = 1 TO 8 20 [quelques instructions à répéter] : 90 NEXT I 100 END </pre>	<pre> 10 I = 1 12 IF I > 8 THEN 100 20 [quelques instructions à répéter] 90 I = I + 1 92 GOTO 12 100 END </pre>
--	---

Notre second programme-échantillon illustre la manière de programmer une "boucle générale" dont les répétitions sont contrôlées par la valeur d'une variable. Nous demandons d'abord à l'utilisateur combien de nombres il ou elle pensent entrer. Nous mémorisons ce nombre sous une variable N et nous passons la boucle par les instructions qui prennent un numéro pour le traitement. La boucle est exécutée N fois, à moins que N soit égal ou inférieur à zéro. Dans ce cas-là, nous passons à END sans traitement aucun.

```

Listage du programme:
10 N = 0 : V = 0 : T = 0 : A = 0
20 WAIT 0
30 INPUT "COMBIEN DE VALEURS? "; N
40 IF N = 0 THEN GOTO 999
50 FOR I = 1 TO N
60 CLS : CURSOR 0
70 INPUT V
80 T = T + V
90 NEXT I
100 WAIT : CLS : CURSOR 0
110 A = T / N
120 PAUSE "TOTAL = "; T
130 PRINT "MOYENNE = "; A
999 END

```

Touches utilisées:

```

1 0 N = 0 SHIFT : V = 0 SHIFT : T = 0
  SHIFT : A = 0 ENTER
2 0 WAIT 0 ENTER
3 0 INPUT SHIFT " COMBIEN SPACE
  DE SPACE VALEURS SHIFT ? SPACE
  SHIFT " SHIFT : N ENTER
4 0 IF N = 0 THEN GOTO
  999 ENTER

```

```

5 0 F O R I = 1 T O N ENTER
6 0 C L S SHIFT : C U R S O R 0 ENTER
7 0 I N P U T V ENTER
8 0 T = T + V ENTER
9 0 N E X T I ENTER
1 0 0 W A I T SHIFT : C L S SHIFT :
    C U R S O R 0 ENTER
1 1 0 A = T / N ENTER
1 2 0 P A U S E SHIFT " T O T A L
    SPACE = SPACE SHIFT " SHIFT :
    T ENTER
1 3 0 P R I N T SHIFT " M O Y E N N E
    SPACE = SPACE SHIFT " SHIFT :
    A ENTER
9 9 9 E N D ENTER

```

Il n'est pas nécessaire d'augmenter la variable-compteur par un, ni d'utiliser 1 comme valeur initiale. A l'aide de la proposition STEP, le programmeur peut spécifier la taille de l'augmentation (ou de la diminution). Notre programme suivant démontre cela sur le ton d'un slogan publicitaire:

Listage du programme:

```

10 WAIT 30
20 FOR HS = 2 TO 8 STEP 2
30 PRINT HS; "! ";
40 NEXT HS
50 WAIT 60 : CLS : CURSOR 0
60 PRINT "QUI ESTIMONS NOUS?"
70 PRINT "LE SHARP PC-1500A!"
80 END

```

Touches utilisées:

```

1 0 W A I T 3 0 ENTER
2 0 F O R H S = 2 T O 8 S T E P
    2 ENTER
3 0 P R I N T H S SHIFT : SHIFT "
    SHIFT ! SPACE SHIFT " SHIFT :
    ENTER
4 0 N E X T H S ENTER
5 0 W A I T 6 0 SHIFT : C L S SHIFT :
    C U R S O R 0 ENTER

```

```

6 0 P R I N T SHIFT " Q U I SPACE
  E S T I M O N S SPACE N O U S
  SHIFT ? SHIFT " ENTER
7 0 P R I N T SHIFT " L E SPACE
  S H A R P SPACE P C - 1 5 0 0 A
  SHIFT ! SHIFT " ENTER
8 0 E N D ENTER

```

En plus du comptage dans l'ordre normal la proposition STEP permet aussi au SHARP de compter à rebours. Pour effectuer cette opération, inversez les valeurs initiales et finales et spécifiez une augmentation négative. Le programme suivant (dédié aux consommateurs du monde entier) illustre cette opération:

Le Bazar du Tapis Roulant liquide son stock à des prix extraordinaires. Toute une série de tapis ronds sont offerts à 0,99\$ le mètre carré. Le rayon des tapis varie entre 1 et de 40 mètres. Entre chaque tapis et celui qui suit en taille (en plus petit) il y a une différence de rayon de 3 mètres. M. Tournanron qui a besoin de tapis neufs, décide de calculer les prix de chaque tapis à l'aide d'un programme de son fidèle SHARP PC-1500A. Il écrit le programme suivant:

```

10 FOR R = 40 TO 1 STEP -3
20 P = .99 * (PI * R ^ 2)
30 PRINT R; " LE RAYON EST $"; P
40 NEXT R
50 END

```

et trouve que les prix varient entre 4976,2\$ et 3,11\$.

Q. WAIT

L'instruction WAIT sert à modifier l'opération de l'instruction PRINT. Elle spécifie la période pendant laquelle des données affichées à l'aide de l'instruction PRINT resteront sur l'affichage.

Le format de l'instruction WAIT est comme suit:

WAIT argument

L'argument est facultatif. Si aucun argument n'est spécifié, la période de défaut est "infinie", ce qui veut dire que les données resteront affichées jusqu'à ce que l'utilisateur appuie sur **ENTER**. C'est le mode sur lequel la plupart de nos programmes opèrent.

Si l'on offre un argument, toutes les instructions PRINT qui suivent tiendront leurs données sur l'affichage pendant une période proportionnelle au numéro spécifié comme argument. Ce type d'instruction PRINT ressemble à une instruction PAUSE, excepté que la période de l'instruction PAUSE est fixée. Notez que l'instruction WAIT n'a aucun effet sur l'opération de l'instruction PAUSE.

Le numéro (ou expression qui a pour résultat un numéro) donné comme argument a une gamme permise allant de 0 à 65535. L'instruction WAIT 0 affiche les données pour une période si courte qu'on ne peut pratiquement pas la lire. Et WAIT 65535 affichera les données chaque instruction PRINT pendant à peu près 17 minutes! Plus concrètement parlant, WAIT 64 équivaut à une période d'environ une seconde et WAIT 3840, à environ une minute. Pour retourner à l'opération ordinaire de l'instruction PRINT (de façon à ce que l'ordinateur attende que l'utilisateur appuie sur ENTER), employez l'instruction WAIT sans argument.

Programme de démonstration

Ce programme sert à illustrer les effets de l'instruction WAIT sur les instructions PRINT qui la suivent. Nous allons varier ici la période de WAIT de 0 à 102 par additions de 2 durant l'impression de périodes en boucle. Le BEEP (avertissement sonore) est là uniquement pour vous aider à comprendre l'intervalle de temps car le passage sur l'affichage est difficile à capturer avec les yeux à cause de la grande vitesse.

Listage de programme:

```
10 FOR W = 0 TO 102 STEP 2
20 WAIT W
30 BEEP 1,5
40 PRINT ". ";
50 NEXT W
60 END
```

Touches utilisées:

```
1 0 F O R W = 0 T O 1 0 2 S T E P
  2 ENTER
2 0 W A I T W ENTER
3 0 B E E P 1 SHIFT 5 ENTER
4 0 P R I N T SHIFT " . " SHIFT "
  SHIFT ; ENTER
5 0 N E X T W ENTER
6 0 E N D ENTER
```

R. READ, DATA, RESTORE

L'utilisateur du programme n'est pas nécessairement celui qui entre toutes les données d'un programme. Souvent, les données le plus utiles sont relativement statiques, comme par exemple les tables d'impôts dans le cas d'une application dans le domaine financier, ou encore les constantes de charge dans le cas d'une application technique. On peut insérer ce type de données dans un programme et s'en servir quand on en a besoin grâce aux instructions DATA, READ et RESTORE. Ces instructions agissent de concert pour spécifier les données employées dans un programme, transférer des données à des variables et répéter l'opération si nécessaire.

L'instruction DATA est constituée du mot-clé DATA suivi d'une liste de données élémentaires comprenant des nombres en notation scientifique ou réelle et des chaînes de caractères. Les éléments de la liste sont séparés par des virgules. Des instructions DATA peuvent être placées n'importe où dans le programme, mais beaucoup de programmeurs préfèrent les grouper au début du programme, ce qui permet de les trouver plus facilement lors de la lecture du programme.

Une instruction DATA typique ressemblera à la ligne suivante:

```
10 DATA "BALZAC", 20000, "ROMANCIER", "M", 112
```

L'instruction READ est constituée du mot-clé READ suivi d'une liste de noms de variables. Ces noms sont soit des noms de variables numériques soit des noms de variables à caractères. Sur la liste, les noms de variables sont séparés par des virgules. L'instruction READ sert à faire "lire" une ou plusieurs données élémentaires d'une instruction DATA et à la/les mémoriser avec les variables associées. Ce qui suit est une instruction READ qui correspond à notre instruction DATA précédente:

```
120 READ N$, WT, C$, SX$, L
```

Chaque fois qu'une instruction READ est exécutée, le SHARP exige qu'il y ait une donnée élémentaire correspondante dans une instruction DATA. Par exemple, le programme suivant produira une erreur en ligne 30 parce que toutes les données élémentaires ont été épuisées par l'instruction READ en ligne 20:

```
10 DATA 1, 2, 3
20 READ A, B, C
30 READ D
```

Pour corriger l'erreur, nous pouvons ajouter une donnée élémentaire à la ligne 10:

```
10 DATA 1, 2, 3, 65
```

ou nous pouvons employer une instruction DATA différente n'importe où dans le programme:

```
10 DATA 1, 2, 3
20 READ A, B, C
30 READ D
40 DATA 65
```

Ceci illustre le fait que le SHARP considère toutes les instructions DATA du programme comme une liste unique de données élémentaires. Au fur et à mesure que l'ordinateur rencontre un nom de variable dans une instruction READ, il affecte la donnée élémentaire suivante de la liste à cette variable. Si le SHARP ne peut pas satisfaire une demande de donnée, il arrête le programme et envoie un message d'erreur. Les éléments en surplus à la fin naturelle du programme sont ignorés.

Si le type (de caractère ou numérique) de l'élément suivant ne correspond pas au type de la variable à remplir, il se produira une erreur. Un bon programmeur groupe les données élémentaires dans des instructions DATA différentes, correspondant chacune à une instruction READ du programme. Ceci est illustré dans le programme suivant qui lit trois données élémentaires quatre fois:

```
10 DATA 1, "A", 1
20 DATA 2, "B", 3
30 DATA 5, "C", 8
40 DATA 13, "D", 21
50 FOR I = 1 TO 3
60 READ A, A$, Z
70 T = T + A * Z
80 NEXT I
```

Nous aurions pu écrire les lignes 10 à 40 de la manière suivante:

```
10 DATA 1, "A", 1, 2, "B", 3, 5, "C", 8, 13, "D", 21
```

ou même ainsi:

```
10 DATA 1
20 DATA "A"
30 DATA 1
40 DATA 2
:
(etc)
```

Chacune de ces formes alternées cache le fait que l'instruction READ lit chaque fois les trois données élémentaires. Les formes alternées rendent aussi plus difficile la vérification des types de données élémentaires puisque le schéma: nombre, caractère, nombre n'est pas visible immédiatement.

Parfois il est désirable de réutiliser quelques unes ou toutes les valeurs des instructions DATA. L'instruction RESTORE nous permet d'effectuer cette opération. L'instruction RESTORE oblige SHARP à réutiliser des données élémentaires en commençant par la première instruction DATA du programme. De cette façon, toute instruction READ effectuée après RESTORE recevra des données élémentaires utilisées auparavant.

L'instruction RESTORE peut aussi spécifier une réutilisation plus sélective de données élémentaires. L'instruction:

RESTORE numéro de ligne

provoquera la ré-affectation de données élémentaires à partir de l'instruction DATA de la ligne spécifiée. Par exemple:

```
10 DATA .8, 4           110 READ M, N, O, P
15 DATA 1, 3, 6, 8E2    120 RESTORE 15
100 READ X, Y           130 READ A, B
```

A la fin de ce programme les valeurs des variables A et B seront de 1 et 3, respectivement.

En plus des numéros de ligne, on peut "étiqueter" les instruction DATA avec un caractère unique. Puis, à l'aide de l'instruction RESTORE, on pourra remettre en circulation des données élémentaires à partir de l'instruction DATA ayant une étiquette donnée. Pour un exemple d'instruction DATA étiquetée ainsi, voyez l'exemple ci-dessous:

```
20 "A" : DATA 1, -12.2, 8
```

Le segment de programme qui suit rend (RESTORE = rendre, minuscule) à l'instruction DATA marquée "X":

```
10 DATA 4.2, 3, 1
20 "X" : DATA -2, 0, 3, 5
100 READ Q, Y, Z
110 READ MA, MB, MC, MD
120 RESTORE "X"
130 READ N, Z
```

A la fin du programme, N contient -2 et Z contient 0.

Lorsque deux programmes ou davantage sont stockés dans l'ordinateur au moyen d'une instruction MERGE, l'instruction RESTORE dans le second et chaque autre programme suivant doit être spécifiée sous la forme d'une **expression RESTORE (ou "label")**

Lorsqu'on se réfère à un label situé dans un autre programme en se servant de "RESTORE label", l'instruction RESTORE label doit être suivie de l'instruction GOTO pour effectuer le branchement sur une ligne disposant d'un numéro de ligne plus élevé.

```
Exemple      10 "A": DATA . . . .
              :
              : } Programme A
```

```

10 "B": RESTORE "A": GOTO 20
20 READ X
30 IF X = 0 THEN 20
:
:
:

```

Programme B

S. REM (REMARQUE)

L'instruction REM fournit la possibilité d'insérer des observations entre les instructions d'un programme. Quoique le SHARP ignore ces observations, elles sont très importantes, car elles aident d'autres utilisateurs à lire votre programme. Plus votre programme sera lisible et compréhensible, plus d'autres programmeurs voudront s'en servir et, peut-être, l'améliorer.

Pour alléger le travail de l'impression, nous avons omis de traiter des observations dans ce manuel. Nous vous recommandons de ne pas suivre notre exemple. Les remarques sont probablement de la plus haute importance quand vous imprimez, enregistrez ou mettez un programme en commun. Ne vous fiez pas à votre mémoire pour vous souvenir de la signification de chaque variable d'un programme, utilisez l'instruction REM. Si vous ne le faites pas, vous l'aurez oubliée dans six mois.

Les observations qui suivent le mot-clé REM, peuvent être insérées dans leur propre ligne ou à la fin d'une autre instruction ou série d'instructions. Il ne faut pas cependant les placer avant ou au milieu d'instructions destinées à être exécutées, parce que quand le SHARP tombe sur le mot-clé REM, il ignore le reste des caractères sur la ligne. Si l'on utilisait REM au début d'une ligne, des instructions de programmes valables seraient ignorées.

T. GOSUB et RETURN

Quand vous commencerez à projeter des programmes, vous découvrirez que vous utilisez certaines fonctions plusieurs fois au cours du même programme. Par exemple, cela pourrait consister à calculer la surface d'un cercle ou de recevoir et vérifier un nombre donné par l'utilisateur. Une telle répétition cause une duplication des instructions qui effectuent la fonction. Pour éviter une duplication inutile le programmeur peut utiliser l'instruction GOSUB.

L'instruction GOSUB permet de mettre de côté un groupe d'autres instructions employées dans plusieurs endroits du programme. On appelle ce groupe une "sous-routine" ("subroutine" en anglais d'où GOSUB). A chaque point du programme où le groupe d'instructions devrait être placé, on insère une instruction GOSUB.

Cette instruction GOSUB signale au SHARP de commencer à exécuter le groupe d'instructions mis de côté. On appelle cela "appeler une sous-routine". L'instruction GOSUB est semblable à l'instruction GOTO en ce qu'elle entraîne un changement dans le cours normal séquentiel de l'exécution. Il y a une différence cependant: avant que le SHARP commence à effectuer les instructions de la sous-routine, il "se souvient" de l'endroit où il s'était arrêté. Cela s'appelle "retourner" d'une sous-routine.

Mais comment est-ce que le SHARP discerne la fin des instructions qui forment la sous-routine? La réponse est qu'on doit l'en informer au moyen de l'instruction RETURN. Le schéma de l'instruction GOSUB est:

GOSUB numéro de ligne

numéro de ligne qui est constitué par le numéro de la première ligne de la sous-routine. Le schéma de l'instruction RETURN est simplement:

RETURN

Comme exemple pratique de la sous-routine, considérez un programme à traiter et comparez la surface de deux rectangles dont la longueur et la largeur sont données:

Listage du programme:

```
10 REM READ IN LONGUEUR ET
20 REM LARGEUR DE DEUX RECTANGLES
30 FOR I = 1 TO 2
40 PAUSE "RECTANGLE "; I; " : "
50 INPUT "ENTREZ LONGUEUR, LARGEUR", L, W
60 GOSUB 200
70 IF I = 1 THEN LET A1 = A
80 IF I = 2 THEN LET A2 = A
90 NEXT I
100 REM PRINT AND COMPAREZ LES
110 REM AIRES DES RECTANGLES.
120 PRINT "AIRE DU RECTANGLE 1 "; A1
130 PRINT "AIRE DU RECTANGLE 2 "; A2
140 IF A1 > A2 THEN 170
150 PRINT "AIRE DE 2 EST > AIRE 1"
160 GOTO 180
170 PRINT "AIRE DE 1 EST > AIRE 2"
180 END
200 REM SOUS-ROUTINE POUR CALCULER AIRE
210 REM DU RECTANGLE, DONNE
220 REM LA LONGUEUR ET LA LARGEUR
230 A = L * W
240 RETURN
```

Notez l'emplacement de la sous-routine. Elles doivent toujours être placées après l'instruction END finale du programme principal. Ceci évite leur usage accidentel lors de l'opération ordinaire d'exécution séquentielle. Chaque sous-routine **DOIT** se terminer par une instruction RETURN.

On peut introduire n'importe quelle instruction permise dans une sous-routine et lui faire accomplir n'importe quel traitement. Les bons programmeurs BASIC conçoivent leurs programmes comme des séries de pièces ou jeux de "modules". En général une sous-routine sert à transcrire un module. Le programme principal sert à contrôler l'ordre dans lequel les sous-routines sont exécutées. Pour plus de renseignements sur ce sujet, reportez-vous à l'un des livres sur la programmation avancée listé dans l'Appendice F.

Exemple 6: Introduire $0.00001234567 \times 10^{24}$:

Touches utilisées

Affichage

• 0 0 0 0 1 2

3 4 5 6 7 E 2 4

ENTER

. 00001234567E24_

1. 234567E 19

Remarquez que dans les exposants seuls les trois chiffres écrits à la fin ont de l'effet.

Exemple 7: Introduire 3×10^{123} :

Touches utilisées

Affichage

3 E 1 2 3

3E123_

ENTER

3E 23

Exemple 8: Introduire 4×10^{-3210} :

Touches utilisées

Affichage

4 E - 3 2 1 0

4E-3210_

ENTER

4E-10

B. Gamme des calculs: débordement supérieur et inférieur

La plupart des appareils opèrent à l'intérieur d'une certaine gamme de nombres. Le PC-1500A a une gamme s'étendant sur tous les nombres entre $9,999999999 \times 10^{99}$ et $-9,999999999 \times 10^{99}$. Si un nombre trop grand déborde des limites de cette gamme, l'ordinateur, étant incapable de le traiter, signalera ce dépassement de capacité à l'aide du message d'erreur ERROR 37. Si un nombre devient trop petit, on dit qu'il se produit un dépassement de capacité inférieur; il ne sera pas signalé par un message d'erreur ou un arrêt dans le fonctionnement. Tout nombre se situant entre -1×10^{-99} sera considéré comme égal à zéro, comme illustré dans le tableau suivant:

	-1×10^{-99}	1×10^{-99}		
	$-9,999999999 \times 10^{99}$	0	$9,999999999 \times 10^{99}$	
ERREUR	GAMME DE CALCUL	CONSIDERE EGAL A ZERO	GAMME DE CALCUL	ERREUR

Exemple 1: Si vous essayez de résoudre l'équation $(5,67 \times 10^{55}) \times (8,90 \times 10^{65})$, vous provoquerez un dépassement de capacité:

Touches utilisées

(5 . 6 7 E 5 5) *

(8 . 9 0 E 6 5)

ENTER

Affichage

(5. 67E55) * (8. 90E65)_

ERROR 37

Le message d'erreur ERROR 37 indique le dépassement de capacité de calcul.

C. Les racines, les puissances et pi

Racine carrée

Exemple 1: Pour trouver la racine carrée de 73:

Touches utilisées

SHIFT √ 7 3

ENTER

Affichage

√ 73 _

8. 544003745

Exemple 2: Pour trouver $\sqrt[4]{256}$:

Touches utilisées

SHIFT √ SHIFT √ 2 5 6

ENTER

Affichage

√√ 256 _

4

Exemple 3: Pour trouver $\sqrt{3^2 + 4^2}$:

Touches utilisées

SHIFT √ (3 * 3 +

4 * 4)

ENTER

Affichage

√ (3*3+4*4)_

5

On peut aussi calculer cette équation de la manière suivante:

Exemple 4:

Touches utilisées

SHIFT √ (3 SHIFT ^ 2 + 4

SHIFT ^ 2)

ENTER

Affichage

√ (3^2+4^2)_

5

Puissance

La puissance ou fonction exponentielle, permet d'élever un nombre à une puissance.

Exemple 1: Calculer $4^3 (= 4 \times 4 \times 4)$:

Touches utilisées

4 **SHIFT** **^** **3**

ENTER

Affichage

4^3_

64

Exemple 2: Calculer $3^{3,2} \times 4^{-2,4}$:

Touches utilisées

3 **SHIFT** **^** **3** **.** **2** *****

4 **SHIFT** **^** **-** **2** **.** **4**

ENTER

Affichage

3^3.2*4^-2.4_

1.207380162

Exemple 3: Calculer $4^{(3^2)}$:

Touches utilisées

4 **SHIFT** **^** **3** **SHIFT**

^ **2**

ENTER

Affichage

4^3^2_

262144

PI π

La valeur de PI (3,141592654) est mémorisée à la fois sous le symbole PI et π comme constante fixe. On peut utiliser l'un ou l'autre de ces symboles dans les calculs où l'on a besoin de la valeur de Pi.

Par exemple, pour calculer la surface d'un tapis d'un diamètre de cinq mètres, sur le mode RUN:

Touches utilisées

P **I** ***** **(** **5** **/** **2** **)**

SHIFT **^** **2**

ENTER

Affichage

PI* (5/2) ^2_

19.63495408

D. Les modes angulaires

Le PC-1500A permet de calculer des fonctions angulaires sur l'un ou l'autre des trois modes angulaires suivants:

Pour placer le PC-1500A sur le mode degré, écrivez:

DEG.

(DEG apparaîtra dans la partie supérieure de l'affichage)

Pour placer le PC-1500A sur le mode radian, écrivez:

RAD.

(RAD apparaîtra dans la partie supérieure de l'affichage)

Pour placer le PC-1500A sur le mode grade, écrivez:

GRA.

(GRAD apparaîtra dans la partie supérieure de l'affichage)

E. Les fonctions trigonométriques

Le PC-1500A fournit six fonctions trigonométriques: SIN, COS, TAN, ASN, ACS, et ATN. Chacune peut être calculée sur le mode GRAD, DEG ou RAD. L'exécution s'effectue de manière suivante:

DEG. <input type="button" value="ENTER"/>	→	Place sur le mode degré.
SIN 30 <input type="button" value="ENTER"/>	<input type="text" value="0.5"/>	SIN 30 en degrés
<input type="button" value="CL"/>	→	Remet l'affichage à zéro.
GRA. <input type="button" value="ENTER"/>	→	Place sur le mode grade.
SIN 30 <input type="button" value="ENTER"/>	<input type="text" value="4.539904997E-01"/>	SIN 30 en grade
<input type="button" value="CL"/>	→	Remet l'affichage à zéro.
RAD. <input type="button" value="ENTER"/>	→	Place sur le mode radian.
SIN 30 <input type="button" value="ENTER"/>	<input type="text" value="-9.880316241E-01"/>	SIN 30 en radians

Dans les exemples ci-dessus, le sinus de 30 est calculé sur chaque mode avec trois résultats différents mais équivalents. Il est possible aussi de calculer des fonctions trigonométriques inverses de la façon suivante:

RAD. <input type="button" value="ENTER"/>	→	Place sur le mode
ASN -0.5 <input type="button" value="ENTER"/>	<input type="text" value="-5.235987756E-01"/>	Arc sinus de -.5
DEG. <input type="button" value="ENTER"/>	→	Place sur le mode
ASN -0.5 <input type="button" value="ENTER"/>	<input type="text" value="-30"/>	Arc sinus de -.5
ACS (-.5 + .1) <input type="button" value="ENTER"/>	<input type="text" value="113.5781785"/>	Arc cosinus
ATN 2.3 <input type="button" value="ENTER"/>	<input type="text" value="66.50143432"/>	Arc tangente

F. Les fonctions logarithmiques

LN, LOG

La fonction LN calcule le logarithme naturel (base e) tandis que la fonction LOG calcule le logarithme décimal (base 10). Leur exécution se déroule de la façon suivante:

LN 7.4

2.00148

LOG 7.4

8.692317197E-01

LN 25

3.218875825

LOG 100

2

EXP

La fonction inverse de LOG est constituée d'un nombre amené à la puissance 10. Par exemple:

LOG 100

2

10 2

100

Une fonction inverse est nécessaire parce que le logarithme naturel (LN) n'est pas basé sur une puissance 10, mais sur une puissance e.

Exemple:

LN 7.4

2.00148

EXP 2.00148

7.399999998

G. Conversion d'angle

Le PC-1500A permet d'effectuer les conversions d'angles de la forme DMS (degré, minute, seconde) à la forme DEG (degré décimal). Quand on effectue la conversion opposée-degrés décimaux en degrés, minutes et secondes- la réponse est constituée d'une partie nombre entier représentant les degrés et d'une partie fractionnaire dont les deux premiers chiffres représentent les minutes et les deux suivants, les secondes. Les chiffres allant du 5e à la fin représentent les degrés décimaux. Pour accomplir la conversion en degrés décimaux, d'un angle donné en degrés, minutes et secondes, il faut l'entrer dans l'ordinateur comme nombre entier et décimales, respectivement.

Exemple 1: Convertir 16.1932 degrés décimaux en DMS:

DMS 16.1932

16.113552

Exemple 2: Convertir 32.2513 DMS en degrés décimaux:

DEG 32.2513

32.42027778

H. Fonctions diverses

ABS (valeur absolue)

La fonction ABS établit la valeur absolue d'une valeur numérique ou d'une variable.

Exemple 1:

ABS (25 - 86)

61

Normalement, $25 - 86 = -61$. Pour arriver à 61, la fonction ABS prend la différence réelle des nombres.

INT (nombre entier)

La fonction INT arrondit une valeur numérique au nombre entier le plus grand qui ne dépasse pas la valeur numérique elle-même.

Exemple 1:

(25/3) + 7

15.33333333

INT (25/3) + 7

15

Exemple 2:

(31.62 + 21.18)

52.8

INT (31.62 + 21.18)

52

Dans l'exemple 2, la réponse n'arrondit pas 52,8 à 53, car 53 est plus grand que la valeur originale 52,8.

NOTE: N'oubliez pas l'ordre d'évaluation. Utilisez les parenthèses si vous désirez obtenir un nombre entier comme expression du résultat. Modifions, en guise d'illustration, l'exemple précédent en omettant toutes les parenthèses:

Exemple 3:

INT 31.62 + 21.18

52.18

L'ordinateur arrondit 31,62 au nombre entier 31 et l'ajoute à 21,18 pour obtenir 52,18 ce qui diffère légèrement du résultat du calcul précédent, n'est-ce pas?

SGN (fonction de signe)

La fonction SGN sert à établir le signe d'un nombre X, c'est à dire qu'elle indique au moyen des valeurs suivantes s'il est positif, égal à zéro ou négatif:

+1 si $X > 0$

0 si $X = 0$

-1 si $X < 0$

Exemples:

5 - 10	ENTER	-5
SGN (5 - 10)	ENTER	-1
12 - 4	ENTER	8
SGN (12 - 4)	ENTER	1
15 - 15	ENTER	0
SGN (15 - 15)	ENTER	0

V. PROGRAMMATION AVANCEE

A. Tableaux et variables indicées

Jusqu'à présent vous vous êtes servi dans nos programmes-échantillons uniquement d'un petit nombre de variables. Au fur et à mesure que vous commencerez à utiliser au maximum la capacité de traitement du PC-1500A, vous découvrirez que des variables, emmagasinant un seul nombre présentent des désavantages sérieux. Si, par exemple, vous considérez un programme qui entre cinquante nombres et les arrange, vous conclurez très vite que, quoique le PC-1500A mette à votre disposition un nombre de variables plus que suffisant, il doit y avoir une façon de faire plus facile. Elle existe et elle s'appelle "variable de tableaux".

"Un tableau" est constitué tout simplement d'un groupe de zones consécutives de stockage ou "emplacements" ayant un nom commun. Chaque zone de stockage peut contenir un seul nombre ou une chaîne de caractères. Dans un tableau donné, toutes les zones de stockage doivent emmagasiner le même type de données.

Le nombre d'emplacements à l'intérieur d'un seul tableau est déterminé par les spécifications de l'utilisateur, avec 256 comme maximum. Si par exemple vous définissez un tableau numérique comme ayant 50 emplacements, cela signifie que vous pouvez stocker jusqu'à 50 nombres sous un seul nom. Si vous définissez un tableau de chaînes (qu'on appelle tableaux de caractères) vous pouvez aussi spécifier la taille des chaînes, jusqu'à un maximum de 80 caractères par chaînes. La façon de créer des chaînes de caractères de tailles différentes est décrite sous Section B. 1.

On se sert de l'instruction DIM (dimension en abrégé) pour définir un tableau. A la différence des variables à valeur unique dont nous nous sommes servi jusqu'à présent, il faut toujours "déclarer" (définir) les tableaux avant de les utiliser. La forme de l'instruction DIMension numérique est la suivante:

DIM nom de variable numérique (taille)

dans laquelle:

le nom de variable numérique est un nom de variable qui se conforme aux règles ordinaires des noms de variables numériques examinées précédemment.

la taille est le nombre d'emplacements de stockage qui doit être situé dans la gamme allant de 0 à 255. Remarquez que lorsque vous spécifiez un nombre pour la taille vous établissez un emplacement de plus que ce que vous avez spécifié.

H. Fonctions diverses

ABS (valeur absolue)

La fonction ABS établit la valeur absolue d'une valeur numérique ou d'une variable.

Exemple 1:

ABS (25 - 86)

61

Normalement, $25 - 86 = -61$. Pour arriver à 61, la fonction ABS prend la différence réelle des nombres.

INT (nombre entier)

La fonction INT arrondit une valeur numérique au nombre entier le plus grand qui ne dépasse pas la valeur numérique elle-même.

Exemple 1:

(25/3) + 7

15.33333333

INT (25/3) + 7

15

Exemple 2:

(31.62 + 21.18)

52.8

INT (31.62 + 21.18)

52

Dans l'exemple 2, la réponse n'arrondit pas 52,8 à 53, car 53 est plus grand que la valeur originale 52,8.

NOTE: N'oubliez pas l'ordre d'évaluation. Utilisez les parenthèses si vous désirez obtenir un nombre entier comme expression du résultat. Modifions, en guise d'illustration, l'exemple précédent en omettant toutes les parenthèses:

Exemple 3:

INT 31.62 + 21.18

52.18

L'ordinateur arrondit 31.62 au nombre entier 31 et l'ajoute à 21,18 pour obtenir 52,18 ce qui diffère légèrement du résultat du calcul précédent, n'est-ce pas?

SGN (fonction de signe)

La fonction SGN sert à établir le signe d'un nombre X, c'est à dire qu'elle indique au moyen des valeurs suivantes s'il est positif, égal à zéro ou négatif:

+1 si $X > 0$

0 si $X = 0$

-1 si $X < 0$

Exemples:

5 - 10	ENTER	-5
SGN (5 - 10)	ENTER	-1
12 - 4	ENTER	8
SGN (12 - 4)	ENTER	1
15 - 15	ENTER	0
SGN (15 - 15)	ENTER	0

V. PROGRAMMATION AVANCEE

A. Tableaux et variables indicées

Jusqu'à présent vous vous êtes servi dans nos programmes-échantillons uniquement d'un petit nombre de variables. Au fur et à mesure que vous commencerez à utiliser au maximum la capacité de traitement du PC-1500A, vous découvrirez que des variables, emmagasinant un seul nombre présentent des désavantages sérieux. Si, par exemple, vous considérez un programme qui entre cinquante nombres et les arrange, vous conclurez très vite que, quoique le PC-1500A mette à votre disposition un nombre de variables plus que suffisant, il doit y avoir une façon de faire plus facile. Elle existe et elle s'appelle "variable de tableaux".

"Un tableau" est constitué tout simplement d'un groupe de zones consécutives de stockage ou "emplacements" ayant un nom commun. Chaque zone de stockage peut contenir un seul nombre ou une chaîne de caractères. Dans un tableau donné, toutes les zones de stockage doivent emmagasiner le même type de données.

Le nombre d'emplacements à l'intérieur d'un seul tableau est déterminé par les spécifications de l'utilisateur, avec 256 comme maximum. Si par exemple vous définissez un tableau numérique comme ayant 50 emplacements, cela signifie que vous pouvez stocker jusqu'à 50 nombres sous un seul nom. Si vous définissez un tableau de chaînes (qu'on appelle tableaux de caractères) vous pouvez aussi spécifier la taille des chaînes, jusqu'à un maximum de 80 caractères par chaînes. La façon de créer des chaînes de caractères de tailles différentes est décrite sous Section B. 1.

On se sert de l'instruction DIM (dimension en abrégé) pour définir un tableau. A la différence des variables à valeur unique dont nous nous sommes servi jusqu'à présent, il faut toujours "déclarer" (définir) les tableaux avant de les utiliser. La forme de l'instruction DIMension numérique est la suivante:

DIM nom de variable numérique (taille)

dans laquelle:

le nom de variable numérique est un nom de variable qui se conforme aux règles ordinaires des noms de variables numériques examinées précédemment.

la taille est le nombre d'emplacements de stockage qui doit être situé dans la gamme allant de 0 à 255. Remarquez que lorsque vous spécifiez un nombre pour la taille vous établissez un emplacement de plus que ce que vous avez spécifié.

Exemples d'instructions DIMension numériques permises:

```
DIM X (5)
DIM AA (24)
DIM Q5 (0)
```

La première instruction établit un tableau X avec 6 emplacements de stockage. La seconde établit un tableau AA avec 25 emplacements. La troisième établit un tableau avec un seul emplacement ce qui est plutôt bête-en ce qui concerne les nombres du moins-puisqu'il équivaut à déclarer une variable numérique à valeur unique.

Il est très important que vous sachiez que le SHARP voit la distinction et la séparation entre une variable de tableau X et une variable X. Le premier X représente une série d'emplacements de stockage numérique et le second une seule location différente.

Maintenant que vous savez comment créer des tableaux vous devez vous demander comment l'on indique l'emplacement de stockage. Puisque le groupe entier n'a qu'un seul nom, on indique un emplacement particulier (qu'on appelle "élément") en faisant suivre le nom du groupe par un numéro entre parenthèses. On appelle ce numéro un "indice". Pour emmagasiner par exemple, le nombre 8 dans le cinquième élément de notre tableau X (déclaré auparavant) nous écrirons:

```
X (4) = 8
```

Si l'usage du 4 vous paraît étonnant, rappelez-vous que la numérotation des éléments commence à zéro et continue jusqu'au numéro de taille déclaré dans l'instruction DIM.

Le grand avantage des tableaux consiste à fournir la possibilité d'utiliser une expression ou un nom de variable en tant qu'indice. Pour établir par exemple, une table des carrés des nombres de 0 à 9, nous pouvons écrire les instructions suivantes:

```
10 DIM SQ (9)
20 FOR I = 0 TO 9
30 SQ (I) = I * I
40 NEXT I
```

Dans cet exemple la variable I sert à choisir l'emplacement de stockage qui contiendra le résultat aussi bien qu'à calculer ce résultat.

Une forme légèrement différente de l'instruction DIM sert à déclarer un tableau de caractères.

DIM nom de variable à caractères (taille) * longueur

formule dans laquelle:

le nom de variable à caractères consiste en un nom de variables qui est conforme aux règles auxquelles sont soumises les variables à caractères ordinaires examinées précédemment.

la taille constitue le nombre d'emplacements de stockage qui doit se situer dans la gamme allant de 0 à 255. Notez que lorsque vous spécifiez un nombre pour la taille, vous établissez un emplacement de plus que ce que vous avez spécifié.

* la longueur est facultative. Quand on l'utilise, elle spécifie la longueur de chacune des chaînes qui forment le tableau. La longueur est un nombre dans une gamme allant de 1 à 80. Si l'on ne se sert pas de cette proposition, les chaînes auront par défaut une longueur de 16 caractères.

Exemples de déclarations de tableau de caractères permises:

```
DIM X$ (4)
DIM NM$ (10) * 10
DIM IN$ (1) * 80
DIM R$ (0) * 26
```

Le premier exemple crée un tableau de cinq chaînes capables chacune de mémoriser 16 caractères. La seconde instruction DIM déclare un tableau NM de onze chaînes à 10 caractères chacune. Définir explicitement la longueur de chaînes plus petites que la longueur par défaut permet d'économiser l'espace de mémorisation. Le troisième exemple déclare un tableau à deux éléments de chaînes de 80 caractères et le dernier, une seule chaîne de vingt-six caractères (Voir section B.1).

En plus des tableaux simples que nous venons juste de voir, des tableaux à deux dimensions sont possibles sur le PC-1500A. Par analogie, on pourrait considérer un tableau à une dimension comme étant une liste de données arrangées en une seule colonne, tandis qu'un tableau à deux dimensions consisterait en un tableau de données en rangées et en colonnes. Le tableau à deux dimensions se déclare à l'aide de l'instruction suivante:

DIM nom de variable numérique (rangées, colonnes)

ou

DIM nom de variable à caractères (rangées, colonnes) * longueur

dans laquelle:

les rangées spécifient le nombre de rangées dans le tableau, nombre qui doit provenir de la gamme entre 0 et 255. Remarquez que lorsque vous spécifiez le nombre de rangées, vous établissez une rangée de plus que celui que vous avez spécifié.

les colonnes spécifient le nombre de colonnes dans le tableau, nombre qui doit provenir de la gamme entre 0 et 255. Remarquez que lorsque vous spécifiez le nombre de colonnes, vous établissez une colonne de plus que celui que vous avez spécifié.

Le schéma suivant illustre les emplacements de stockage qui résultent de la déclaration DIM T (2, 3) et les indices (composés de deux nombres maintenant) qui correspondent à chaque emplacement de stockage.

	colonne 1	colonne 2	colonne 3	colonne 4
Rangée 0	T (0, 0)	T (0, 1)	T (0, 2)	T (0, 3)
Rangée 1	T (1, 0)	T (1, 1)	T (1, 2)	T (1, 3)
Rangée 2	T (2, 0)	T (2, 1)	T (2, 2)	T (2, 3)

NOTES: ● Des tableaux à deux dimensions peuvent occuper très rapidement l'espace de stockage. Par exemple, un tableau à 25 rangées et 35 colonnes occupe 875 emplacements!

Les tableaux sont des outils très puissants. Nous vous recommandons de lire d'autres publications pour obtenir une vue plus complète de leurs capacités et utilisations.

- La table ci-dessous montre le nombre d'octets utilisé pour définir chaque type de variables ainsi que le nombre d'octets pris par les instructions mises en mémoire:

Variable	Caractéristiques de la variable	Données	
Tableau de variables numériques	7 octets	8 octets	
Tableau de variables alphanumériques	7 octets	Tableau de variable	Nombre d'octets spécifié*
		Variable à 2 caractères	16 octets

- * Par exemple: si DIMZ\$ (2, 3) * 10 est spécifié, 12 variables, chacune capable de stocker 10 caractères sont réservées en mémoire. Ceci demande donc 7 octets (de déclaration du tableau) + 10 octets (nombre de caractères) x 12 (nombre de variables: (2 + 1) * (3 + 1)) = 127 octets.

Élément	N° de ligne	Longueur de la ligne	Commande ou Instruction	Autres	ENTER
Place utilisée	2 octets	1 octet	2 octets	1 octet	1 octet

B. Plus de détails sur les chaînes des caractères

B.1. Etablir la DIMension des chaînes

La longueur des chaînes à caractères est limitée par défaut à seize caractères. Il est possible cependant de créer une chaîne d'une longueur de 80 caractères en établissant la dimension d'une chaîne à caractères. Il est possible aussi d'économiser l'aire de la mémorisation en diminuant la longueur de la chaîne.

L'instruction DIMension ci-dessous spécifie la longueur d'une chaîne.

DIM nom de variable (limite) * longueur

dans laquelle:

nom de variable constitue le nom du tableau de chaînes de caractères.

limite constitue l'indice maximum du tableau.

la longueur constitue la longueur de chaque chaîne du tableau.

Si l'on n'a besoin que d'une seule chaîne, il est possible de spécifier un tableau à un élément (ayant zéro pour indice) pour économiser l'aire de mémorisation. La déclaration suivante d'une chaîne à 26 caractères illustre ce cas:

DIM AS (0) * 26

B.2. Enchaînement

Il est possible de combiner plusieurs chaînes de caractères (ou caractères à l'intérieur de variables à caractères) pour former une seule chaîne. On appelle "enchaînement" cette action de combiner des chaînes de caractères. On exprime l'enchaînement sous la forme générale suivante:

$$\text{variable} = \frac{\text{chaîne de caractères}}{\text{variable à caractères}} + \frac{\text{chaîne de caractères}}{\text{variable à caractères}}$$

Exemple 1:

```
10 S$ = "SUPER"  
20 T$ = S$ + "MAN"  
30 PRINT T$
```

Sortie:

SUPERMAN

Dans la ligne 20, le contenu de la variable S\$ ("SUPER") a été ajouté à la chaîne "MAN". Notez que l'on n'insère pas d'espace vide pendant l'enchaînement. On peut enchaîner plusieurs chaînes pour former une seule expression, comme dans l'exemple qui suit:

Exemple 2:

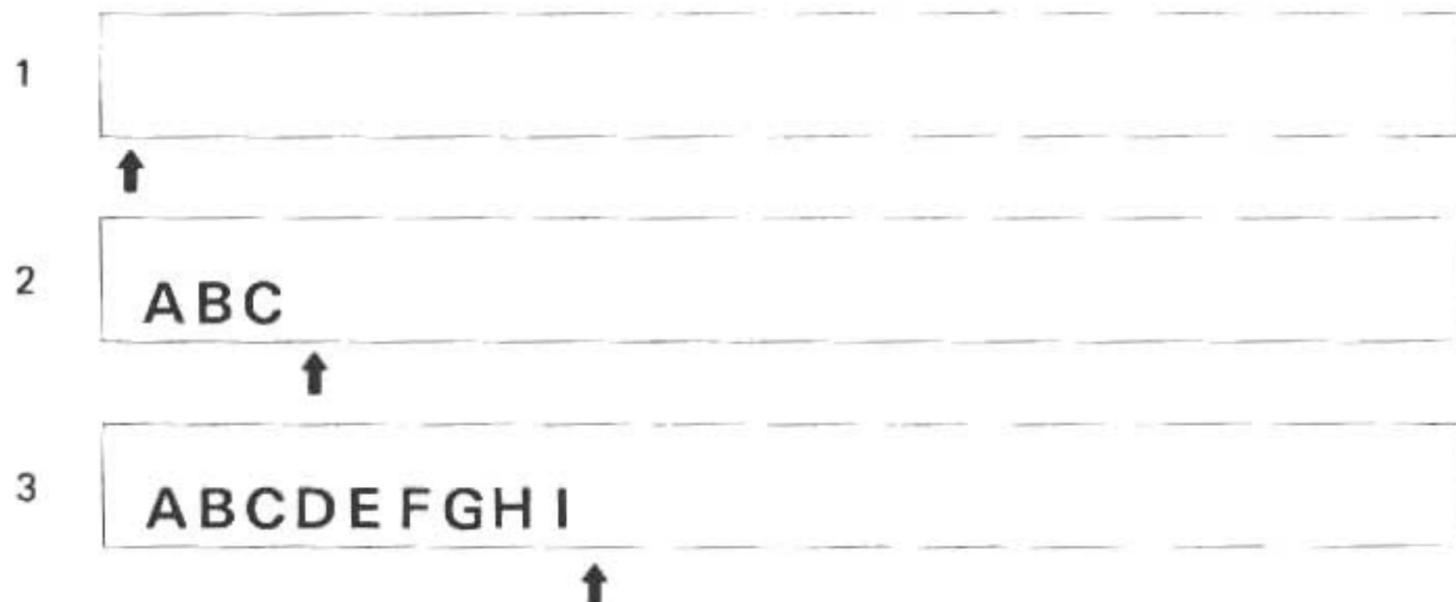
```
5 WAIT 100  
10 A$ = "ANCE"  
20 B$ = "DE"  
30 C$ = "NAISS"  
40 D$ = "DATE"  
50 PRINT "ECRIVEZ VOTRE" + D$  
60 PRINT B$ + C$ + A$  
70 GOTO 50
```

Sortie:

ECRIVEZ VOTRE DATE
DE NAISSANCE

Quand on effectue des enchaînements, on se sert, pour forger les nouvelles chaînes, d'une aire temporaire de stockage de caractères. Cette aire a une capacité de stockage de 80 caractères. Si la nouvelle chaîne dépasse cette limite on obtiendra un message d'erreur ERROR 15. Ci-dessous vous trouverez une illustration de cette aire pendant un enchaînement.

Exemple 3:





NOTE: Le symbole \uparrow représente un index de caractères interne qui contrôle la quantité de mémoire utilisée.

- 1) Au début de l'exécution, l'aire de stockage sera remise à zéro et l'index de caractères reviendra à sa position de départ.
- 2) On entre "ABC" qui occupera les trois premières positions de l'aire.
- 3) On ajoute, après "ABC", "DEFGHI" à l'aire de stockage de caractères.
- 4) La fonction LEFT\$ agit sur la chaîne "DEFGHI" pour extraire "DE" qui remplacera "DEFGHI" dans l'aire.
- 5) L'affectation est accomplie et l'aire de stockage est remise à zéro de nouveau.

B.3. Comparaison de chaînes

On peut comparer des chaînes de caractères pour déterminer quelle chaîne est "plus grande" ou moins grande que l'autre. Cette opération se base sur la séquence de collation donnée dans l'appendice C qui consiste dans l'ordre dans lequel l'ordinateur reconnaît les caractères.

Si les chaînes contiennent un nombre inégal de caractères, on "rembourre" la chaîne la plus courte avec des caractères NULS (ASCII 0). Les opérateurs valides pour la comparaison de chaîne sont:

- = Vrai si les deux chaînes sont d'une longueur égale et contiennent les mêmes caractères dans le même ordre.
- <> Vrai si les deux chaînes diffèrent en longueur, en caractères et par l'ordre de leurs caractères.
- > Vrai si les caractères de la première chaîne sont "plus grands" (arrivent plus tard dans l'ordre de leur liste) que les caractères de la seconde chaîne.
- < Vrai si les caractères de la première chaîne sont "plus petits" (arrivent plus tôt dans l'ordre de leur liste) que les caractères de la seconde chaîne.
- \geq Vrai si les caractères de la première chaîne sont égaux ou "supérieurs" aux caractères dans la deuxième chaîne.
- \leq Vrai si les caractères de la première chaîne sont égaux ou "inférieurs" aux caractères dans la deuxième chaîne.

Le format de la comparaison de chaînes est le suivant:

$$\frac{\text{chaîne de caractères}}{\text{variable de caractères}} \quad \text{OP} \quad \frac{\text{chaîne de caractères}}{\text{variable de caractères}}$$

dans lequel OP constitue l'un des opérateurs listés plus haut.

Exemples:

"MARY" > "MARI"	est vrai
"MARY" = "MARY "	est faux
"abc" <> "ABC"	est vrai
"DATA 1" < "DATA 2"	est vrai
"?" < "#"	est faux

C. Fonctions

C.1. ASC

Il y a deux fonctions qui servent à convertir des caractères en code ASCII et à les dé-convertir de ce code. La fonction ASC convertit un caractère unique dans son code décimal ASCII. La fonction inverse CHR\$ convertit le code décimal ASCII en une chaîne d'un caractère unique.

ASC { "caractère"
nom de variable à caract.

Dans cette fonction, l'argument consiste en une chaîne quelconque de caractères ou un nom de variable de chaîne de caractères. La valeur rendue par cette fonction constitue le code ASCII correspondant au premier caractère de la chaîne spécifiée.

Exemple 1:

```
10 LET X$ = "PATTI"  
20 LET A = ASC X$  
30 PRINT A
```

Sortie:

RUN	80
-----	----

Dans l'exemple ci-dessus, on affecte à X\$ la valeur de la chaîne de caractères "PATTI". La fonction ASC prend le premier caractère (P) et le convertit en son numéro de code ASCII (80).

Exemple 2:

```
10 PRINT ASC "K"
```

Sortie:

RUN	75
-----	----

ASC "K" rapporte 75 qui constitue le code ASCII de "K".

C.2. CHR\$

CHR\$ constitue la fonction complémentaire de ASC. La fonction CHR\$ prend un code décimal ASCII, de 0 à 127 et retourne la chaîne de caractères équivalente. (Note: Certains codes représentent des caractères spéciaux qu'on ne peut imprimer.)

CHR\$		code décimal ASCII
		variable numérique

Exemple 1:

```
10 PRINT (CHR$ 67) + "OP"
```

Sortie:

```
COP                                RUN •
```

Exemple 2:

```
10 Z = 65  
20 PRINT CHR$ Z
```

Sortie:

```
A                                RUN •
```

Le premier exemple illustre l'utilisation d'un code décimal ASCII comme argument. Le résultat est enchaîné à la chaîne "OP" pour produire "COP". Le second exemple affecte une valeur numérique à la variable Z. Ensuite on se sert du nom de variable comme argument, avec le caractère "A" comme résultat.

Notre programme-échantillon suivant convertit les majuscules d'une chaîne texte en minuscules à l'aide des deux fonctions ASC et STR\$ à la fois:

Exemple 3:

```
5 WAIT 0  
10 INPUT "ENTREZ MESSAGE", M$  
20 FOR I = 1 TO LEN (M$)  
30 T$ = MID$ (M$, I, 1)  
40 L = ASC (T$)  
45 IF (L < 65) OR (L > 90) THEN 60  
50 T$ = CHR$ (L + 32)  
60 PRINT T$;  
70 NEXT I  
80 WAIT : PRINT
```

C.3. La Fonction INKEY\$

Cette fonction recueille n'importe quel caractère du clavier et le stocke dans la variable spécifiée. Il n'est pas nécessaire d'appuyer sur **ENTER** car le caractère sera accept automatiquement.

variable = INKEY\$

A moins qu'on ne se serve d'une instruction PRINT antérieure, le symbole de guidage ne sera pas affiché durant l'exécution de cette instruction. Le caractère entré n'est pas renvoyé sur l'affichage qui ne subit donc pas d'altération.

Exemple:

```
10 WAIT 0
20 A$ = INKEY$
30 IF A$ = " " THEN PRINT "AUCUNE TOUCHE": GOTO 20
40 PRINT A$
50 GOTO 20
```

Cette fonction accepte seulement un caractère. Si l'on entre plus d'un caractère au clavier, seul le premier passera; tous les autres seront ignorés.

C.4. La fonction LEN

Quand on manipule des caractères, il est souhaitable de connaître le nombre de caractères d'une chaîne. Pour obtenir ce renseignement on se sert de la fonction LEN qui rapportera ce nombre de caractères dans une expression spécifiée ou une variable de caractères.

LEN { "chaîne de caractères"
nom de variable de caractères

Exemple 1:

```
10 A$ = "CATON"
20 C = LEN A$
30 PRINT C
```

Sortie:

RUN	I •
	5

Exemple 2:

```
10 C = LEN "CAT"
20 PRINT C
```

Sortie:

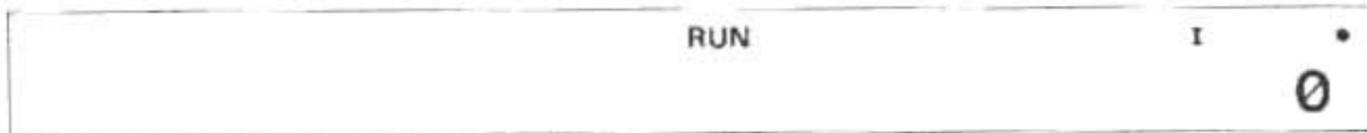
RUN	I •
	3

Un zéro sera rapporté si l'on sert de LEN sur une chaîne vide (c'est-à-dire qu'il n'y a rien entre les guillemets).

Exemple 3:

```
10 A = LEN " "  
20 PRINT A
```

Sortie:



C.5. La fonction LEFT\$

Il y a trois fonctions dont on se sert pour sélectionner ou extraire des sections spécifiées d'une chaîne de caractères. La fonction LEFT\$ extrait des caractères de la gauche, la fonction RIGHT\$ de la droite et MID\$ du milieu.

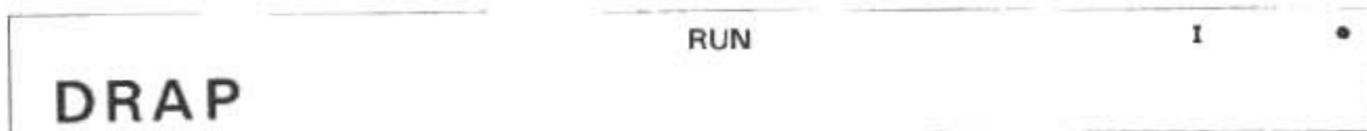
```
LEFT$ { ("chaîne de caractères", nombre)  
      { (nom de variable de caractères, nombre)
```

L'argument "nombre" spécifie combien de caractères doivent être extraits en partant de la gauche.

Exemple 1:

```
10 A$ = LEFT$ ("DRAPEAU", 4)  
20 PRINT A$
```

Sortie:



Exemple 2:

```
10 B$ = "PARACHUTE"  
20 A$ = LEFT$ (B$, 4)  
30 PRINT A$
```

Sortie:



Dans les deux exemples, on extrait des caractères de la chaîne en partant de la gauche pour les stocker sous A\$. Si l'on écrit A\$ sur le clavier on obtient "DRAP" et "PARA" respectivement.

C.6. La fonction MID\$

On sert de la fonction MID\$ pour extraire la portion du milieu d'une chaîne de caractères.

```
MID$ { ("chaîne de caractères", #1, #2)  
     { (variable de chaîne de caractères, #1, #2)
```

Exemple 1:

```
10 A$ = "FAITES VOS JEUX"  
20 B$ = MID$ (A$, 8,3)  
30 PRINT B$
```

Sortie:

```
VOS                                RUN                                I                                •
```

Exemple 2:

```
10 T$ = MID$ ("(415) 743 - 1602", 6, 3)  
20 PRINT T$
```

Sortie:

```
743                                RUN                                I                                •
```

Le premier argument de cette fonction consiste en une chaîne de caractères ou une variable de chaîne de caractères. Le second argument est un nombre qui représente le premier caractère à extraire. Le troisième argument constitue le nombre total de caractères à extraire, y compris le premier.

Dans le premier exemple, "FAITES VOS JEUX" est stocké sous A\$. MID\$ extrait trois caractères en commençant par le huitième et les loge dans B\$. Si l'on imprime B\$, on trouvera qu'il contient "VOS".

Dans le deuxième exemple, une chaîne contenant un numéro de téléphone constitue le premier argument. On repère le sixième caractère ("7") et on le stocke avec les deux caractères suivants sous la variable T\$. Le résultat imprimé sera "743".

C.7. La fonction RIGHTS

RIGHTS fonctionne comme LEFT\$ la différence étant que RIGHTS commence à exécuter à partir de l'extrémité inverse de la chaîne (la droite). Les arguments sont les mêmes que pour LEFT\$:

$$\text{RIGHTS} \quad \left\{ \begin{array}{l} (\text{"chaîne de caractères, nombre"}) \\ (\text{variable de caractères, nombre}) \end{array} \right.$$

L'argument "nombre" de cette fonction spécifie combien de caractères sont à extraire de la chaîne, en partant de la droite.

Exemple 1:

```
10 X$ = "READ ONLY MEMORY"  
20 Y$ = RIGHTS (X$, 6)  
30 PRINT Y$
```

Dans ce programme, la fonction RIGHTS prend six caractères de l'extrémité droite de la chaîne et les stocke sous la variable Y\$. Y\$ contient maintenant "MEMORY".

C.8. La fonction RND

Il vous faudra parfois fournir des nombres aléatoires à votre programme. La fonction RND permet à l'ordinateur d'engendrer des nombres aléatoires dans une gamme de un à un nombre spécifié. (Note: la gamme commence toujours par un.)

Exemple 1:

```
10 A = RND 5
```

Dans cet exemple A pourrait avoir n'importe quelle valeur entre un et cinq, inclusivement. Si vous désirez des nombres aléatoires dans une gamme commençant par un autre nombre que 1 (par exemple, de 40 à 50) il vous faudra simuler cela en engendrant des nombres aléatoires entre 1 et 10 et en leur ajoutant une constante (39 dans notre exemple).

Exemple 2:

```
10 FOR I = 1 TO 5
20 B = 40 + RND 10
40 PRINT B;
50 NEXT I
```

Sortie:

```
42 44 47 48 42
```

RUN

C.9. La fonction RANDOM

Les nombres aléatoires sont engendrés par l'intermédiaire d'une formule mathématique et nous sont accessibles au moyen de la fonction RND. Chaque fois que l'ordinateur est mis en marche, il engendre une série de nombres aléatoires. Cette liste de nombres aléatoires ne change pas, à moins que l'on utilise la fonction RANDOM. Ceci signifie qu'un programme utilisera la même série de nombres aléatoires chaque fois que l'ordinateur est mis en marche. Pour éviter cela, la fonction RANDOM recompose la "semence" utilisée par la formule pour engendrer ses nombres aléatoires.

Exemple 1:

```
10 FOR I = 1 TO 5
15 A = RND 3
20 PRINT A;
30 NEXT I
```

Sortie: 1 2 2 3 1

Exemple 2:

```
10 FOR I = 1 TO 5
15 A = RND 3
20 PRINT A;
30 NEXT I
```

Sortie: 1 2 2 3 1

Pour obtenir de vrais nombres aléatoires dans ce cas, la fonction RANDOM doit apparaître avant l'instruction RND. Cette fonction introduit une nouvelle semence dans le processus de génération de nombres aléatoires et cause ainsi la différence qui existe entre eux. Conséquemment, deux programmes que l'on passe dans des conditions identiques produisent des données de sortie différentes.

Exemple 1:

```
10 RANDOM
15 FOR I = 1 TO 5
20 A = RND 3
30 PRINT A;
40 NEXT I
```

Sortie: **3 1 2 2 3**

Exemple 2:

```
10 RANDOM
15 FOR I = 1 TO 5
20 A = RND 3
30 PRINT A;
40 NEXT I
```

Sortie: **2 3 1 3 2**

C.10. La fonction STR\$

STR\$ fonctionne de façon contraire à VAL. STR\$ reconvertit une variable numérique interne en sa chaîne de caractères ou expression originale.

Exemple:

```
10 INPUT "ENTREZ UN NOMBRE "; I
20 S$ = STR$ (I)
30 PAUSE "CE NOMBRE EST"
40 PRINT "LA CHAINE"; S$
```

Le programme ci-dessus accepte un nombre, forme la représentation en caractère de ce nombre et le stocke dans la variable de caractère S\$. Du fait que la représentation numérique interne ne peut pas être affichée, elle est reconvertie automatiquement en une chaîne de caractères par l'instruction PRINT.

C.11. La fonction STATUS

Pour afficher la quantité de mémoire de programme encore disponible ou pour connaître la quantité de mémoire utilisée par un programme, il suffit d'opérer la fonction STATUS.

STATUS 0

affichera le nombre "d'étapes" du programme qui sont disponibles.

STATUS 1

affichera le nombres des étapes déjà utilisées.

STATUS 2

affichera l'adresse de la mémoire ou le programme en cours s'achève. Il est à noter que cette adresse est actuellement un degré plus grande que celle où le programme s'arrête effectivement.

STATUS 3

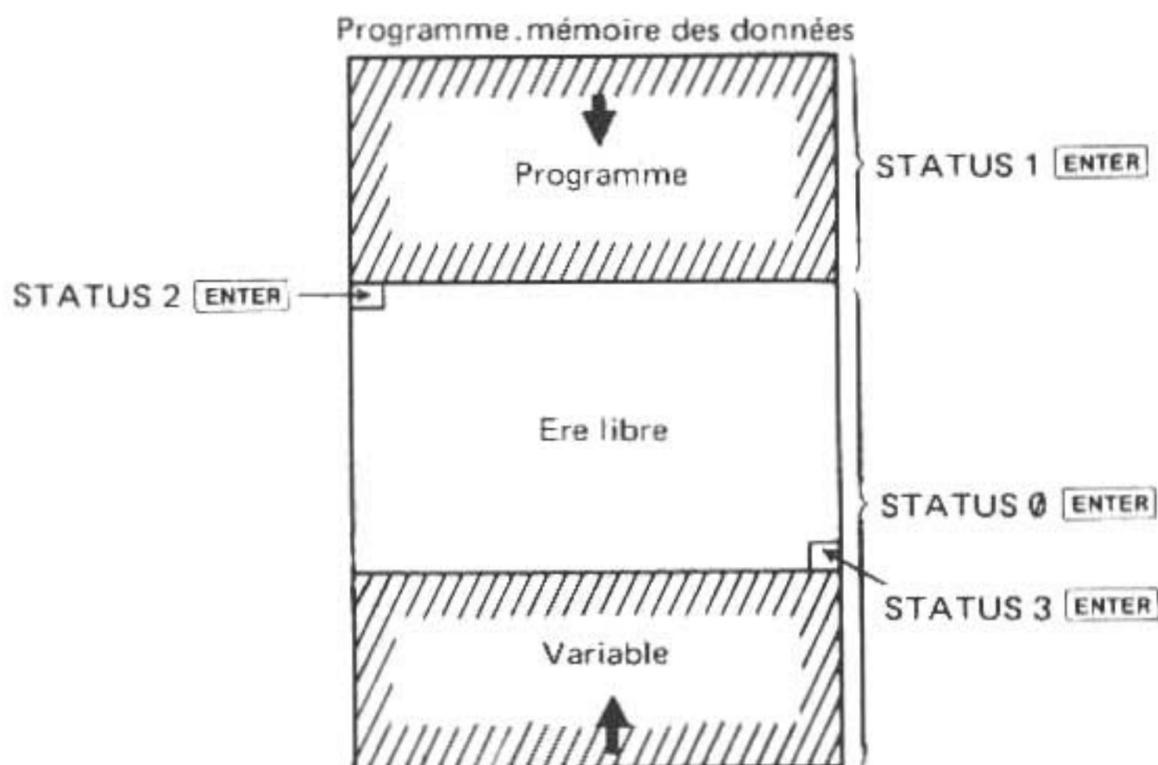
affichera l'adresse de la mémoire où les variables sont mémorisées.

STATUS 4 ~ STATUS 100

affichera le numéro de la ligne du programme qui était en cours d'exécution lorsque cette dernière fut arrêtée.

MEM

La commande MEM est équivalente à la directive STATUS 0



Ere libre: est obtenue par le "STATUS 3 – STATUS 2 + 1"

C.12. La fonction TIME

Pour afficher ou régler le mois, le jour et l'heure, on se sert de la fonction TIME de la manière suivante:

Réglage: TIME = MMDDHH.MMSS ENTER

Affichage: TIME ENTER

formules dans lesquelles MM représente deux chiffres pour le mois, DD, deux chiffres pour le jour et HH, deux chiffres pour l'heure. La portion fractionnaire définit les minutes (MM) et les secondes (SS).

Quand on affiche TIME, la sortie est dans le même format que celui donné ci-dessus. On peut se servir du résultat de la fonction de la même manière qu'un nombre et l'on peut aussi s'en servir librement dans des expressions.

Le programme suivant simule une horloge. Ne manquez pas de régler l'heure avant de passer le programme.

Listage du programme:

```

10 WAIT 0
20 A$ = STR$ TIME
30 IF TIME > 99999 THEN 50
40 A$ = "0" + A$
50 M$ = LEFT$ (A$, 2)
60 D$ = MID$ (A$, 3, 2)
70 H$ = MID$ (A$, 5, 2)
80 DS$ = M$ + " / " + D$ + " / 82"
90 DS = VAL (M$ + D$ + "00")
100 PRINT DS$;
110 T = TIME - DS
120 IF T >= 1 THEN 140
130 T = T + 12
140 IF T > 23.5959 GOTO 20
150 CURSOR 18 : PRINT USING "###.####" ; T
160 GOTO 110

```

C.13. La fonction VAL

VAL and STR\$ se complètent en tant que fonctions de conversion de chaînes de caractères en variables numériques et inversement de variables en chaînes. La fonction VAL convertit une chaîne contenant la représentation en caractères d'un nombre en un nombre qu'elle stocke ensuite dans une variable.

NOTE: Si l'on utilise autre chose qu'un chiffre de 0 à 9, un point décimal., un signe positif (+) ou négatif (-) ou la notation scientifique (E) au cours d'une expression, l'ordinateur l'ignorera.

Exemple 1:

```
10 ZS = VAL "-37"
```

Résultat:

ZS contient le nombre -37

Exemple 2:

```
10 A = VAL "237.6"
```

Résultat:

A contient le nombre 237,6

D. PRINT USING

L'instruction USING permet à un programmeur de contrôler strictement le format des données sur l'affichage. Ceci permet de standardiser les affichages et d'éviter les pertes de données.

La proposition USING, seule ou combinée avec une instruction PRINT ou PAUSE, définit le format de toutes les instructions PRINT ou PAUSE qui suivent jusqu'au moment la prochaine proposition USING apparaît au cours du programme.

Il se peut que plusieurs propositions USING apparaissent dans une seule instruction PRINT ou PAUSE. Dans ce cas, chacune d'entre elles définit le format à utiliser pour imprimer les variables listées jusqu'à utiliser pour imprimer les variables listées jusqu'à ce que la proposition USING suivante se présente.

On appelle "chaîne de mise en forme" la chaîne de caractères spéciaux qui sert à spécifier un format. Les caractères de la chaîne de mise en forme définissent les zones disponibles pour l'affichage de données et limitent le type de données pouvant être imprimées dans ces zones. Ce procédé est le même que celui employé par les autres langages tel que le COBOL et le PL/I.

Il est possible de stocker une chaîne de mise en forme dans une variable à chaînes. Dans ce cas le nom de la variable devra remplacer la chaîne de mise en forme dans la proposition USING. Ceci rend possible des formats multiples qui sont sélectionnés sous contrôle de programme.

Les caractères que l'on est autorisé à employer dans une chaîne de mise en forme sont examinés sommairement ci-dessous:

TABLEAU DES CARACTERES DE MISE EN FORME

Caractère

Utilisation

#	Spécifie un champ numérique. Les nombres sont cadrés à droite dans le champ. Si le champ n'est pas assez large pour contenir le nombre, un message d'erreur ERROR 36 se produit. Des zéros en tête de nombre sont convertis en espaces vides.
*	Spécifie "Asterisk Fill" (remplissage par des astérisques) des positions spécifiées d'un champ numériques, positions qui ne contiennent pas de données.
.	Provoque l'affichage d'un point décimal dans un champ numérique.
,	Utilisé au début d'un champ numérique pour spécifier l'insertion de virgules après chaque trois chiffres.
^	Utilisé dans un champ numérique pour provoquer l'affichage en notation scientifique d'un nombre.
+	Utilisé dans un champ numérique pour forcer l'impression du <u>signe</u> des données.
&	Spécifie un champ de caractères. Les caractères sont cadrés à gauche dans ce champ. Si le champ n'est pas assez large pour contenir la chaîne de données, la chaîne est coupée.

NOTE: Il faut toujours que la largeur du champ numérique soit plus grande d'un espace que la largeur des données pour faire de la place pour le signe des données. Ceci est nécessaire que vous employiez le caractère de mise en forme + ou non.

NOTE: L'utilisation de la virgule rend nécessaire l'insertion d'un # de plus pour chaque virgule dans la chaîne de mise en forme.

Exemples

X = PI Y = 1234 A\$ = "ABCDEF"

PRINT USING "###"; X

3

PRINT USING "+###.###"; X

+3.141

PRINT USING "###.##^"; X

3.14E 00

PRINT USING "###.^"; X

3.E 00

PRINT USING "*#####"; Y

**1234

PRINT USING "***##"; Y

1234

PRINT USING "&&&&&#####"; AS; Y

ABCDEF 1234

PRINT USING "&&&"; AS

ABC

10 US = "*#####.##"
20 USING US
30 PRINT Y; "\$"

**1234.00\$

PRINT X; "\$"

*****3.14\$

PRINT USING; AS; X

ABCDEF 3.141592654

PRINT USING "###, ###, ###"; 246813

246, 813

Note: Utilisez le nombre de repères # (* compris) pour la désignation de variable (entier) dans la gamme suivante:

Avec une ponctuation à 3 chiffres (,) non utilisée: Maximum 11 (y compris le signe)

Avec une ponctuation à 3 chiffres utilisée: Maximum 14 (y compris le signe)

Cet ordinateur est doté de 10 chiffres significatifs pour les nombres.

Lorsqu'un format dépassant 10 entiers est désigné par l'instruction USING et qu'un nombre dépassant 10 entiers est affiché (imprimé) par l'instruction PRINT (LPRINT), le nombre affiche (imprimé) risque d'être incorrect.

Exemple:	Mode RUN		Affichage
	USING "#####"	ENTER	→ >
	PRINT 888888888888	ENTER	→ 888888888800
	(LPRINT 888888888888	ENTER	→ 88888888880)
		12 chiffres	

E. Transfert de contrôle calculé

En plus des instructions de contrôle décrites dans le Chapitre III, le PC-1500A est doté de deux autres instructions de contrôle de grande utilité. Les instructions ON GOSUB et ON GOTO. Comme leurs noms vous l'auront fait deviner, ces instructions agissent de la même façon que les instructions GOSUB et GOTO examinées auparavant. Leur différence réside dans le fait que ON GOSUB et ON GOTO peuvent transférer le contrôle (c'est-à-dire exécuter des instructions à un emplacement différent) automatiquement. GOTO et GOSUB vont vers une instruction (ou sous-routine) parmi plusieurs autres selon la valeur d'une certaine variable numérique. C'est cette dépendance d'une variable pour le guidage qui a valu aux instructions ON le surnom d'"instructions de contrôle calculé".

Les instructions ON prennent la forme:

<u>ON</u> expression	{	GOTO	ligne #1,	ligne #2,	ligne #3, ... (etc)
		GOSUB	"	"	"

L'expression qui suit le mot-clé ON doit se résoudre en un nombre entier positif, plus grand que zéro et moins grand que le nombre de numéros de ligne listés après les mots-clés GOTO et GOSUB. Quand l'ordinateur rencontre l'instruction ON durant l'exécution, il transfère le cours de l'exécution au numéro de ligne qui correspond à la valeur de l'expression.

Une instruction ON typique pourrait être comme suit:

ON TX GOSUB 100, 200, 250, 300

Dans ce cas la variable TX DOIT contenir un nombre entre 1 et 4 parce qu'il n'y a que quatre numéros de ligne listés. Un nombre autre que ces quatre produirait une erreur puisqu'il n'y a pas de numéro de ligne correspondant. Pour cette raison, il s'avère nécessaire d'inclure plusieurs épreuves (instructions IF) pour être sûr que nos expressions résultent en un nombre valide.

Les instructions ON s'avèrent très utiles pour automatiser une série de choix. Considérons, par exemple, le morceau de programme suivant qui permet à l'utilisateur de choisir une table de taux de taxation parmi plusieurs. On pourrait écrire cela comme ci-dessus si l'on n'utilisait pas l'instruction ON:

```
10 PAUSE "TAUX DES IMPOTS:"
20 PAUSE " (1) CELIBATAIRE,"
30 PAUSE " (2) MARIE,"
40 PAUSE " (3) PROFESSION?"; TT
50 IF (TT < 1) OR (TT > 3) GOTO 10
60 REM CHOISISSEZ TABLEAU APPROPRIE
70 IF TT = 1 THEN GOTO 220
80 IF TT = 2 THEN GOTO 300
90 IF TT = 3 THEN GOTO 450
(etc)
```

Grâce à l'instruction ON, nous pouvons combiner les lignes 70, 80 et 90 en une seule instruction:

```
70 ON TT GOTO 220, 300, 450
```

ON ERROR GOTO

On permet à un programme de détecter quand une erreur se produit grâce à l'utilisation d'une variante de transfert contrôle. Après la détection, le programme peut exécuter des instructions qui essaient de se remettre de l'erreur. De telles instructions peuvent renseigner ou donner des instructions à l'utilisateur ou garder des données importantes.

L'instruction ON ERROR GOTO indique au SHARP où il doit se porter lorsqu'il découvre qu'une erreur s'est produite. Cette instruction prend la forme suivante:

ON ERROR GOTO numéro de ligne

dans laquelle le numéro de ligne constitue le numéro d'une ligne de programme qui contient les directives à suivre en cas d'erreur.

F. Programmation de l'affichage

L'affichage incorporé dans le PC-1500A se prête d'une manière remarquablement souple à la sortie de données. Pour permettre aux programmeurs d'utiliser à fond la puissance de cet affichage, de nouvelles instructions ont été ajoutées au langage BASIC utilisé par le PC-1500A.

Pour afficher jusqu'à 26 caractères à la fois, on s'est servi de la technologie du cristal liquide. Chaque caractère du jeu dont l'ordinateur est doté occupe une matrice à points de 5 x 7. Au moyen de la commande GPRINT, les programmeurs peuvent développer et afficher leurs propres caractères.

Il est possible d'utiliser le champ entier de l'affichage comme une matrice à points de 7 x 156 pour réaliser des graphiques. Pour créer des graphiques, des figures ou des symboles spéciaux, on peut stimuler des points individuels dans l'une quelconque des 156 colonnes. La commande POINT permet de "toucher" une colonne quelconque pour découvrir quels points sont actuellement stimulés.

Un haut-parleur et un générateur de son permettent au programmeur d'ajouter une dimension sonore à l'interaction entre l'homme et la machine. Il y a 256 fréquences (de 230 Hz à environ 7 KHz) sur lesquelles on peut produire des sons. La répétition automatique d'un son et le contrôle de la durée du son sont également possible sous contrôle de programme.

F.1. BEEP

L'instruction BEEP permet au programmeur d'utiliser le générateur de sons pour créer des effets sonores durant les jeux sur ordinateur, pour avertir d'une erreur par son et pour nombre d'autres applications dans le domaine de la communication entre l'ordinateur et l'utilisateur. Le format de l'instruction BEEP pour créer des sons est le suivant:

BEEP expression 1 , expression 2 , expression 3

dans lequel:

expression 1 constitue le seul paramètre requis et spécifie combien de fois le son sera répété.

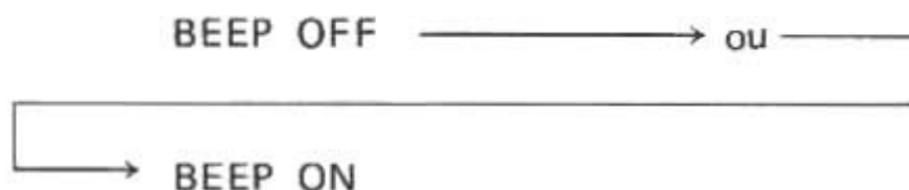
La gamme permise des répétitions va de 0 à 65535.

expression 2 est facultative et spécifie la fréquence du ou des sons, fréquence représentée par un nombre entre 0 et 255.

expression 3 est aussi facultative et spécifie la durée de chaque son, durée spécifiée par un nombre d'une gamme allant de 0 à 65279.

On peut aussi se servir de l'instruction BEEP pour mettre sous et hors tension le haut-parleur interne du PC-1500A. On élimine ainsi le bruit produit par les opérations SAVE (conservation) et LOAD (chargement) de cassette.

Le format de l'instruction BEEP qui contrôle le haut-parleur interne est le suivant:



- NOTE:**
- Si l'on met le PC-1500A à l'arrêt (OFF) puis de nouveau en marche (ON), on remet le haut-parleur sous tension.
 - Les sons produits par l'instruction BEEP peuvent varier selon la combinaison des expressions.

Programme de démonstration

```
10 D = 60
20 DATA 14
30 DATA 245, 1, 245, 1, 160, 1, 160, 1
40 DATA 143, 1, 143, 1, 160, 2
50 DATA 180, 1, 180, 1, 195, 1, 195, 1
60 DATA 220, 1, 220, 1, 245, 2
100 READ X
110 FOR I = 1 TO X
120 READ N, S
130 BEEP 1, N, (D * S)
140 NEXT I
150 END
```

F.2. L'instruction CURSOR

L'instruction CURSOR place le curseur sur l'un des 26 emplacements de caractères disponibles sur l'affichage. Cette instruction prend la forme suivante:

CURSOR expression d'emplacement

dans laquelle:

expression d'emplacement se réduit à un nombre de la gamme allant de 0 à 25, nombre qui spécifie l'emplacement où le curseur se rendra.

Ordinairement, on utilise la commande CURSOR pour placer le curseur avant l'impression de données. Utilisée de cette manière, elle permet au programmeur de définir sa propre séparation entre données élémentaires comme dans les instructions suivantes, par exemple:

Listage du programme:

```
10: WAIT 20
20: X = 8
30: PRINT "A"
40: CURSOR 4
50: PRINT "B"
60: CURSOR X
70: PRINT "C"
80: CURSOR ((X ^ 2) / 4) - 4
90: PRINT "D"
100: END
```

Touches utilisées:

```
1 0 W A I T 2 0 ENTER
2 0 X = 8 ENTER
3 0 P R I N T SHIFT " A SHIFT " ENTER
4 0 C U R S O R 4 ENTER
5 0 P R I N T SHIFT " B SHIFT " ENTER
6 0 C U R S O R X ENTER
7 0 P R I N T SHIFT " C SHIFT " ENTER
8 0 C U R S O R ( ( X SHIFT ^ 2 )
  / 4 ) - 4 ENTER
9 0 P R I N T SHIFT " D SHIFT " ENTER
1 0 0 E N D ENTER
```

Ce programme provoquera l'apparition des lettres A, B, C et D respectivement dans les emplacements 0, 4, 8 et 12 de l'affichage:

```

A      B      C      D      RUN

```

NOTE: Si l'on spécifie un numéro d'emplacement plus grand que 25 ou moins grand que 0, on obtiendra un message d'erreur ERROR 19.

Programme de démonstration

```
10 WAIT 0
20 DIM A$(0) * 13
30 INPUT "ENTREZ VOTRE NOM", A$(0)
40 C = 0 : CLS
50 FOR I = 1 TO (LEN A$(0) - 1)
60 CURSOR C
70 PRINT MID$(A$(0), I, 1)
80 C = C + 2
90 NEXT I
100 WAIT
110 CURSOR C
120 PRINT RIGHT$(A$(0), 1)
130 END
```

Touches utilisées:

1 0 W A I T 0 ENTER
2 0 D I M A SHIFT \$ (0) * 1 3 ENTER
3 0 I N P U T SHIFT " E N T R E Z SPACE
V O T R E SPACE N O M SHIFT "
SHIFT , A SHIFT \$ (0) ENTER
4 0 C = 0 SHIFT : C L S ENTER
5 0 F O R I = 1 T O (L E N A SHIFT \$ (0))
- 1) ENTER
6 0 C U R S O R C ENTER
7 0 P R I N T M I D SHIFT \$ (A SHIFT \$ (0))
SHIFT , I SHIFT , 1) ENTER
8 0 C = C + 2 ENTER
9 0 N E X T I ENTER
1 0 0 W A I T ENTER
1 1 0 C U R S O R C ENTER
1 2 0 P R I N T R I G H T SHIFT \$
(A SHIFT \$ (0)) SHIFT , 1)
ENTER
1 3 0 E N D ENTER

F.3. L'instruction CLS (remise à zéro de l'affichage)

L'instruction CLS efface l'affichage en mettant hors tension tous les points qui le forment. On s'en sert avant d'écrire d'autres instructions pour effacer toutes les données restantes de l'affichage. La commande CLS ramène le curseur à la gauche de l'affichage. Le format est simplement CLS.

Programme de démonstration

```
10 GPRING "7 F 7 F 7 F 7 F 7 F"  
20 CLS  
30 PRINT "NOUVELLE DONNEE"
```

Touches utilisées:

1 0 G P R I N T SHIFT " 7 F 7 F 7 F
7 F 7 F SHIFT " ENTER
2 0 C L S ENTER
3 0 P R I N T SHIFT " N O U V E L L E SPACE
D O N N E E SHIFT " ENTER

F.4. L'instruction G_CURSOR

L'instruction G_CURSOR spécifie l'une des 156 colonnes de points disponibles de l'affichage comme colonne de départ pour tout affichage ultérieur de données. Le format de l'instruction G_CURSOR est le suivant:

G_CURSOR expression d'emplacement

dans lequel:

expression d'emplacement se réduit à un nombre de la gamme allant de 0 à 155. Ce nombre spécifie l'une des 156 colonnes à 7 points de l'affichage.

NOTE: Si une expression d'emplacement résulte en un nombre moins grand que 0 et plus grand que 155, un message d'erreur ERROR 19 apparaîtra.

On utilise généralement l'instruction G_CURSOR conjointement avec l'instruction G_PRINT pour réaliser un affichage de graphiques. (Nous illustrerons cette fonction avec plus de minutie dans la section suivante). On peut faire suivre l'instruction G_CURSOR par d'autres types de directives puisque cette instruction n'écrit pas de données. G_CURSOR indique seulement dans quelle colonne des données ultérieures seront écrites. Les lignes qui suivent illustrent cette fonction:

Listage de programme:

```
10 G_CURSOR 50
20 PRINT "A"
30 G_CURSOR 80
40 PRINT 26/3
```

Touches utilisées:

```
1 [ ] [ ] [G] [ ] [C] [ ] [U] [ ] [R] [ ] [S] [ ] [O] [ ] [R] [ ] [5] [ ] [ ] [ENTER]
2 [ ] [ ] [P] [ ] [R] [ ] [I] [ ] [N] [ ] [T] [SHIFT] [ ] ["] [ ] [A] [SHIFT] [ ] ["] [ENTER]
3 [ ] [ ] [G] [ ] [C] [ ] [U] [ ] [R] [ ] [S] [ ] [O] [ ] [R] [ ] [8] [ ] [ ] [ENTER]
4 [ ] [ ] [P] [ ] [R] [ ] [I] [ ] [N] [ ] [T] [ ] [2] [ ] [6] [ ] [ / ] [ ] [3] [ENTER]
```

ce qui produit une sortie ayant un air normal aux emplacements 50 et 80:

```

                                     RUN
                                     .
A                                     8. 666666667
```

Que se passerait-il si nous changions la ligne 30 de façon à commencer à imprimer à l'emplacement G_CURSOR 93? Faisons l'essai en lui substituant la ligne suivante:

```
30 G_CURSOR 93
```

O surprise! La deuxième sortie a été "tronquée", coupée là où elle débordait de l'extrémité de l'affichage.

Comme exemple final d'instruction G_CURSOR nous vous présentons un programme plus évolué qui crée un effet insolite au moyen d'un chevauchement de caractères:

Listage du programme:

```
10 WAIT 0
20 DIM A$(0) * 10
30 A$(0) = " " : C = 0
40 INPUT "ENTREZ MESSAGE (< 10 CARA)", A$(0)
50 CLS
60 FOR I = 1 TO (LEN A$(0))
70 GCURSOR C
80 FOR J = 1 TO 3
90 GCURSOR C
100 PRINT MID$(A$(0), I, 1)
105 C = C + 3
110 NEXT J
120 C = C + 5
130 NEXT I
140 WAIT
150 GCURSOR 155
160 GPRINT "00"
170 END
```

Touches utilisées:

1 0 W A I T 0 ENTER

2 0 D I M A SHIFT \$ (0) * 1 0 ENTER

3 0 A SHIFT \$ (0) = SHIFT " SHIFT "

SHIFT : C = 0 ENTER

4 0 I N P U T SHIFT " E N T R E Z SPACE

M E S S A G E SPACE (SHIFT < 1 0

SPACE C A R A) SHIFT " SHIFT ↵

A SHIFT \$ (0) ENTER

5 0 C L S ENTER

6 0 F O R I = 1 T O (L E N A SHIFT \$

(0)) ENTER

7 0 G C U R S O R C ENTER

8 0 F O R J = 1 T O 3 ENTER

9 0 G C U R S O R C ENTER

1 0 0 P R I N T M I D SHIFT \$ (A

SHIFT \$ (0) SHIFT ↵ I SHIFT ↵

1)) ENTER

1 0 5 C = C + 3 ENTER

1 1 0 N E X T J ENTER

1 2 0 C = C + 5 ENTER

```

1 3 0 N E X T I ENTER
1 4 0 W A I T ENTER
1 5 0 G C U R S O R 1 5 5 ENTER
1 6 0 G P R I N T SHIFT " 0 0 SHIFT " ENTER
1 7 0 E N D ENTER

```

F.5. GPRINT

L'instruction GPRINT fournit un moyen de contrôle direct et programmable sur les points de l'affichage. L'instruction GPRINT est, comme nous l'avons vu plus haut, utilisée conjointement avec l'instruction GCURSOR puisqu'elle règle les points de toutes les colonnes à 7 points. L'instruction GCURSOR choisit la colonne appropriée à la modification tandis que l'instruction GPRINT manipule les points de la colonne en question. L'instruction GPRINT est aussi capable d'imprimer plusieurs colonnes attenantes de données en une seule instruction.

Pour comprendre le format de l'instruction GPRINT il est nécessaire de comprendre d'abord comment l'on contrôle les points d'une colonne. On peut spécifier le motif de points alimentés (par un courant) dans une colonne soit par un nombre décimal soit par une chaîne de caractères hexadécimale. Si l'on utilise le système décimal, on peut imaginer que chaque rangée est numérotée du haut en bas, par une puissance deux. Ceci est illustré ci-dessous:

```

1  - - - - -
2  - - - - -
4  - - - - -
8  - - - - -
16 - - - - -
32 - - - - -
64 - - - - -

```

Considérant qu'il y a 7 points par colonne et que l'on peut activer ou désactiver chacun d'entre eux, nous obtenons 128 motifs possibles. Pour spécifier un motif particulier, on emploie le format:

GPRINT expression de motif; expression de motif 2 . . (etc) . .

dans lequel:

l'expression de motif se réduit à un nombre de la gamme allant de 0 à 127 et spécifie le motif de points sous tension. On peut, facultativement, spécifier plusieurs expressions de motif que l'on doit séparer par un point-virgule ou une virgule. Si l'on utilise une virgule, une colonne vide existera entre chaque colonne imprimée.

Nous allons illustrer l'utilité de l'instruction GPRINT en créant un nouveau caractère: une flèche dirigée vers le haut. Dessinons d'abord ce caractère à l'intérieur d'une grille représentant les rangées et les colonnes:

```

1  - - - - * - - - -
2  - - * - * - * - -
4  * - - - * - - - *
8  - - - - * - - - -
16 - - - - * - - - -
32 - - - - * - - - -
64 - - - - * - - - -
    1  2  3  4  5

```

Attention

Observez ce qui suit lors d'une correction de l'instruction PRINT dans un programme pour une instruction GPRINT:

Exemple:

Ré-écrivez complètement "PRINT" sous la forme "GPRINT".

20 PRINT A\$

Le fait d'insérer seulement "G" ne permet pas à l'ordinateur de le juger en tant que "GPRINT". (C'est considéré comme étant "G" et "PRINT".)

La même chose s'applique lorsque l'instruction CURSOR est corrigée par l'instruction GCURSOR, ou par les instructions de l'imprimante. (Voir page 174)

Il nous faudra 5 numéros séparés dans la liste expression de motif de l'instruction GPRINT parce que notre caractère s'étend sur 5 colonnes. Les numéros représentant les colonnes 1 et 5 doivent chacun spécifier un seul point dans la rangée ayant 4 comme étiquette. De même, les numéros représentant les colonnes 2 et 4 doivent chacun spécifier un seul point dans la rangée 2. Voici l'instruction finale:

GPRINT 4; 2; 127; 2; 4

La spécification de la troisième colonne (127) constitue la seule valeur numérique dont la dérivation n'est pas évidente au premier coup d'oeil. 127 constitue la somme d'un point dans la première rangée (1) d'un point dans la deuxième rangée (2) d'un point dans la troisième rangée (4) et ainsi de suite. 127 consiste donc en $1 + 2 + 4 + 8 + 16 + 32 + 64$ et spécifie ainsi tous les 7 points de la colonne. On peut créer toutes sortes de motifs en spécifiant une rangée ou une somme de plusieurs rangées.

Si l'on utilise le système hexadécimal d'adressage, on divise les 7 rangées de l'affichage en un groupe inférieur de 3 rangées et un groupe supérieur de 4 rangées. Chaque groupe est numéroté de haut en bas par des puissances deux, comme indiqué ci-dessous:

1	-----
2	-----
4	-----
8	-----
1	-----
2	-----
4	-----

Il est donc possible de représenter tous les motifs d'un groupe par un seul chiffre hexadécimal. La gamme de chiffres permise pour le groupe inférieur ira de 0 à 7 puisque ce groupe n'a que trois rangées. Des deux chiffres hexadécimaux requis pour spécifier toute une colonne, le premier représente le groupe inférieur et le second le groupe supérieur.

Voici la forme de l'instruction GPRINT hexadécimale:

GPRINT "chaîne hexadécimale"

dans laquelle:

la chaîne hexadécimale consiste en une chaîne de chiffres hexadécimaux, dont chaque paire spécifie le motif des points d'une colonne.

Utilisons ce format pour créer le caractère à flèche dirigée vers le haut de l'exemple précédent au moyen de l'instruction:

GPRINT "04027F0204"

Tableau de caractères hexadécimaux

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

Programme de démonstration

Ce programme imprime tous les motifs de points possibles, dans l'ordre, de 0 à 127.

Listage du programme:

```

10 WAIT 0 : CLS
20 FOR I = 0 TO 127
30 GCURSOR I
40 GPRINT I
50 NEXT I
60 WAIT
70 GCURSOR 155 : GPRINT "00"
80 END

```

Touches utilisées:

```
1 [ ] [ ] W A I T [ ] [ ] SHIFT [ ] [ ] : [ ] [ ] C L S [ ] [ ] ENTER
2 [ ] [ ] F O R I = [ ] [ ] T O 1 2 7 [ ] [ ] ENTER
3 [ ] [ ] G C U R S O R I [ ] [ ] ENTER
4 [ ] [ ] G P R I N T I [ ] [ ] ENTER
5 [ ] [ ] N E X T I [ ] [ ] ENTER
6 [ ] [ ] W A I T [ ] [ ] ENTER
7 [ ] [ ] G C U R S O R 1 5 5 [ ] [ ] SHIFT [ ] [ ] :
   [ ] [ ] G P R I N T [ ] [ ] SHIFT [ ] [ ] " [ ] [ ] [ ] [ ] SHIFT [ ] [ ] " [ ] [ ] ENTER
8 [ ] [ ] E N D [ ] [ ] ENTER
```

F.6. La fonction POINT

La fonction POINT rapporte un nombre qui représente le motif de points activés dans une colonne donnée. La fonction POINT permet donc de "tâter" une colonne quelconque de l'affichage sous contrôle de programme.

Ci-dessous, le format de la fonction POINT:

POINT expression d'emplacement

dans lequel:

l'expression d'emplacement se réduit à un nombre de la gamme allant de 0 à 155 et représente la colonne à sonder.

La valeur rapportée par la fonction POINT consiste en un nombre de la gamme allant de 0 à 127. L'interprétation de ce nombre est une somme de puissances 2 comme nous l'avons vu dans la section concernant GPRINT.

A titre d'illustration, supposez qu'il y ait un 1 majuscule sur l'affichage, s'étendant sur les colonnes 40 à 44:

```
1  -- * - * - * --
2  ---- * ----
4  ---- * ----
8  ---- * ----
16 ---- * ----
32 ---- * ----
64 -- * - * - * --
   4  4  4  4  4
   0  1  2  3  4
```

L'expression:

POINT 40 rapporterait 0,
POINT 41 rapporterait 65,
POINT 42 rapporterait 127.

Programme de démonstration

Le programme listé ci-dessous remplit l'affichage avec le caractère stocké dans A\$ et crée des motifs bizarres en inversant ce caractère. Ce programme utilise plusieurs instructions examinées dans le présent chapitre.

Listage du programme:

```
10 A$ = "X"  
20 WAIT 0  
30 Y = 5 : X = 155/Y : C = 0  
40 FOR I = 1 TO X  
50 GCURSOR C  
60 PRINT A$  
70 C = C + Y  
80 NEXT I  
90 FOR I = 0 TO 155  
100 GCURSOR I  
110 A = 127 - POINT I  
120 GPRINT A  
130 NEXT I  
140 GOTO 90
```

Touches utilisées:

```
1 0 A SHIFT $ = SHIFT " X SHIFT " ENTER  
2 0 W A I T 0 ENTER  
3 0 Y = 5 SHIFT : X = 1 5 5 / Y  
  SHIFT : C = 0 ENTER  
4 0 F O R I = 1 T O X ENTER  
5 0 G C U R S O R C ENTER  
6 0 P R I N T A SHIFT $ ENTER  
7 0 C = C + Y ENTER  
8 0 N E X T I ENTER  
9 0 F O R I = 0 T O 1 5 5 ENTER  
1 0 0 G C U R S O R I ENTER  
1 1 0 A = 1 2 7 - P O I N T I ENTER  
1 2 0 G P R I N T A ENTER  
1 3 0 N E X T I ENTER  
1 4 0 G O T O 9 0 ENTER
```

NOTE: Si l'on inclut la dernière ligne, le programme se trouvera dans une boucle infinie lors de son passage (la touche BREAK sert à y mettre fin). On peut changer aussi le caractère affiché en insérant un nouveau caractère dans l'affectation en ligne 10.

G. Détection d'erreurs

Aussi consciencieux que vous soyez, il vous arrivera tôt ou tard de créer un programme qui n'accomplit pas exactement ce que vous désiriez qu'il accomplisse. Les créateurs du SHARP ont prévu une méthode spéciale d'exécution de programme appelée "mode Trace" pour vous aider à identifier la source du problème. Quand on le place sur le mode Trace le PC-1500A affiche le numéro de ligne de chaque ligne de programme et s'arrête après l'exécution de cette ligne. Ceci permet de suivre la trace des instructions au fur et à mesure de leur exécution. Durant la pause de programme à la fin de l'exécution d'une ligne vous pourrez examiner ou altérer les valeurs des variables.

La forme de l'instruction servant à débiter le mode Trace est simplement: TRON en tant que commande (sur le mode RUN); on peut aussi l'insérer en tant qu'instruction dans un programme. En tant que commande, TRON indique au SHARP qu'il faut "pister" pendant l'exécution de tous les programmes qui suivent. Après cela, on commence les programmes de la manière ordinaire, à l'aide de la commande GOTO ou RUN.

Utilisé comme instruction, TRON amorce le mode Trace seulement quand la ligne qui la contient est exécutée. Si pour une raison quelconque, on n'atteint pas cette ligne, le mode Trace restera inactif.

Une fois amorcé, le mode d'opération Trace reste efficace jusqu'à qu'il soit annulé par une directive TROFF. On peut lancer cette directive tout aussi bien en tant que commande qu'en tant qu'instruction. On peut aussi annuler le mode Trace avec la séquence-clé suivante:

SHIFT **CL**

En guise d'illustration de l'utilisation du mode Trace, entrez le programme suivant pour calculer la longueur de l'hypoténuse d'un triangle dont les côtés sont donnés:

Listage du programme:

```
10 INPUT A, B
20 A = A * A : B = B * B
30 H =  $\sqrt{A + B}$ 
40 PRINT "HYPOTENUSE = "; H
```

Sur le mode RUN, lancez la commande TRON suivie par la commande RUN. Remarquez que la commande INPUT fonctionne de manière ordinaire et affiche un point d'interrogation pour chaque valeur requise. Dès que vous avez entré deux valeurs, le numéro de ligne de l'instruction INPUT apparaît:

```
                                RUN                                | •
10 :
```

Vous pouvez réexaminer la ligne tout entière en appuyant sur la touche **↑** (flèche vers le haut) et en la maintenant en position basse:

```
                                RUN                                | •
10 : INPUT A, B_
```

Pour continuer le programme, appuyez une fois sur la touche **↓** (flèche vers le bas). Cette opération provoque l'exécution de la ligne suivante et l'affichage de son numéro. Ensuite vous pouvez à nouveau revoir la ligne à l'aide de la touche **↑** (flèche vers le haut). Vous pouvez aussi vérifier le contenu de n'importe quelle variable en imprimant son nom et en appuyant sur **ENTER** :

A **ENTER**

(dans ce cas A est une variable de programme)

RUN 1 •
4

Il est nécessaire d'appuyer une fois pour chaque ligne sur la touche  (flèche vers le bas) pour permettre l'exécution de chaque ligne, et cela jusqu'à la fin du programme. Si vous ne voulez pas continuer, l'exécution ligne par ligne ordinaire, appuyez sur ENTER pour suspendre l'exécution du programme. Si vous changez d'avis à nouveau, la commande CONT vous permettra de continuer les programmes suspendus.

Utilisons notre programme à hypoténuse pour les opérations suivantes:

Touches utilisées	Affichage
	>
T R O N	TRON_
ENTER	>
R U N	RUN_
ENTER	?
3	3_
ENTER	?
4	4_
ENTER	10 :
	10 : INPUT A, B_
	20 :
	20 : A=A*A : B=B*B_
A	A_
ENTER	9
B	B_
ENTER	16
	30 :
H	H_
ENTER	5
	HYPOTENUSE= 5
	40 : PRINT "HYPOTENUSE=" ; H_
	40 :
	>

H. Nombres hexadécimaux et fonctions de BOOLE

H.1. Nombres hexadécimaux

Le PC-1500A vous offre la possibilité d'utiliser un nombre hexadécimal (base 16) à l'intérieur d'une expression dans laquelle on peut utiliser un nombre décimal. Les nombres hexadécimaux se distinguent des nombres décimaux par l'esperluète (&) qui les précède. Les nombres de l'exemple ci-dessous constituent des formes valides de nombres hexadécimaux:

&16 &F &7ECA &08 &99A -&5B

On peut utiliser les nombres hexadécimaux dans les calculs:

10 + &A

```
                                RUN                                ' 20'
```

Ou encore dans les programmes:

Programmes:

```
35 GPRINT &F, 54, &3E
40 DATA 67, &7F, &2B, 12, 305
```

H.2. La fonction AND (et)

La fonction AND fournit un AND (et) booléen de la représentation interne de deux valeurs. Les valeurs doivent se situer dans la gamme allant de -32768 à 32767. Des nombres hors de cette gamme provoquent une erreur, ERROR 19.

<u>Exemple:</u>	<u>Résultat:</u>
10 AND &F	10
1 AND 0	0
-1 AND 1	1
55 AND 64	0
16 AND 63	16

H.3. La Fonction OR (ou)

La fonction OR effectue ou OR (ou) booléen sur les représentations internes de deux valeurs, les valeurs doivent se situer dans la gamme allant de -32768 à 32767. Des nombres hors de cette gamme provoquent une erreur, ERROR 19.

<u>Exemple:</u>	<u>Résultat:</u>
10 OR &F	15
1 OR 0	1
-1 OR 1	-1
55 OR 64	119
16 OR 63	63

H.4. La Fonction NOT

La fonction NOT rapporte le NOT (non) booléen, ou complément, de la représentation interne d'une valeur unique. Cette valeur doit se situer dans la gamme allant de -32768 à 32767. Si la valeur est hors gamme, un message d'erreur (ERROR 19) apparaîtra.

<u>Exemple:</u>	<u>Résultat:</u>
NOT Ø	-1
NOT &F	-16
NOT 55	-56
NOT 1	-2
NOT -2	1

VI. ELARGISSEMENT DU PC-1500A

A. L'INTERFACE IMPRIMANTE/CASSETTE

L'interface imprimante/cassette constitue une option pour l'ordinateur de poche SHARP PC-1500A. Cette unité permet d'effectuer le raccordement avec un ou deux magnétophones à cassette qui serviront à enregistrer des programmes et des données sur des cassettes audio standard. Ces programmes et données peuvent être "rechargés" par la suite dans le PC-1500A, évitant ainsi à l'utilisateur l'ennui d'avoir à les ré-entrer au clavier.

L'utilisation de ces caractéristiques constitue le sujet des sections suivantes.

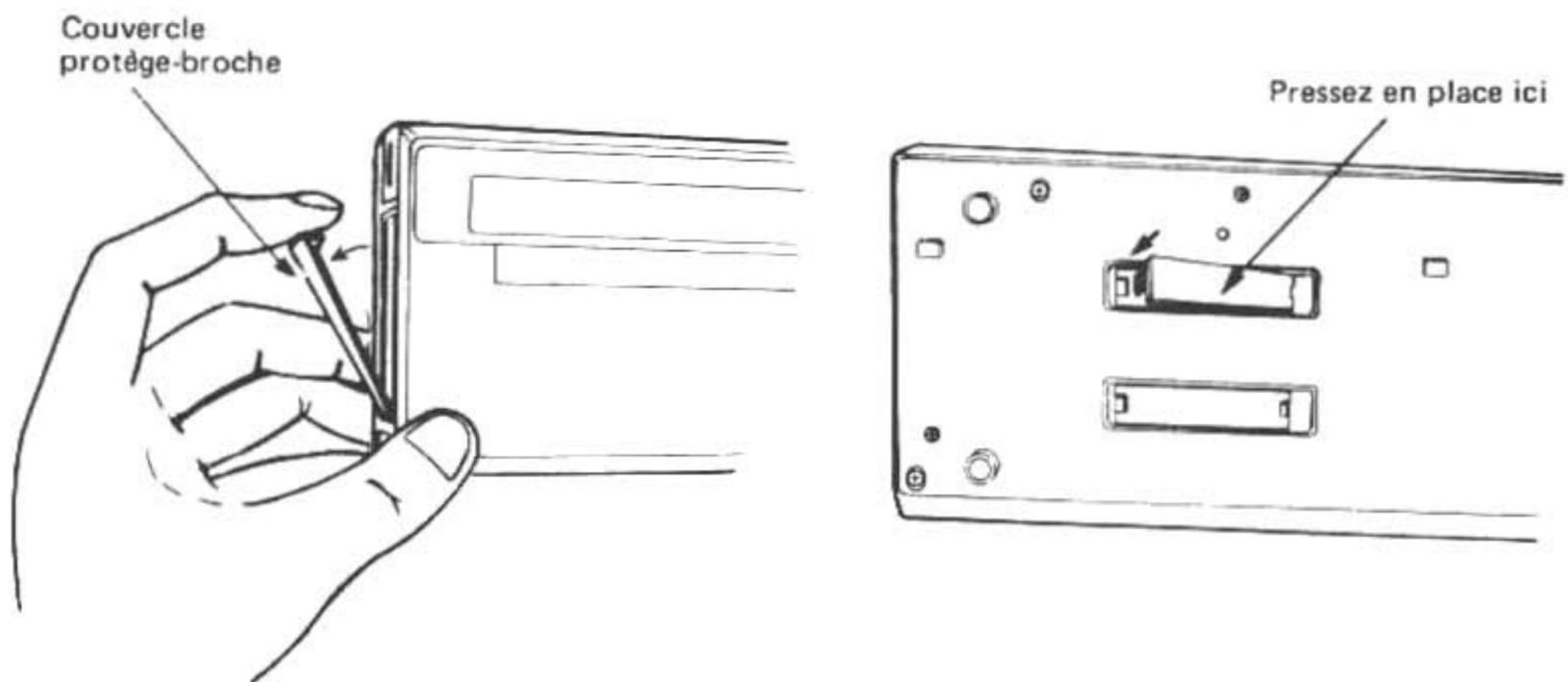
1. Connexion de l'ordinateur sur l'interface

Effectuez la connexion entre l'interface (CE-150) et l'ordinateur (PC-1500A) de façon suivante:

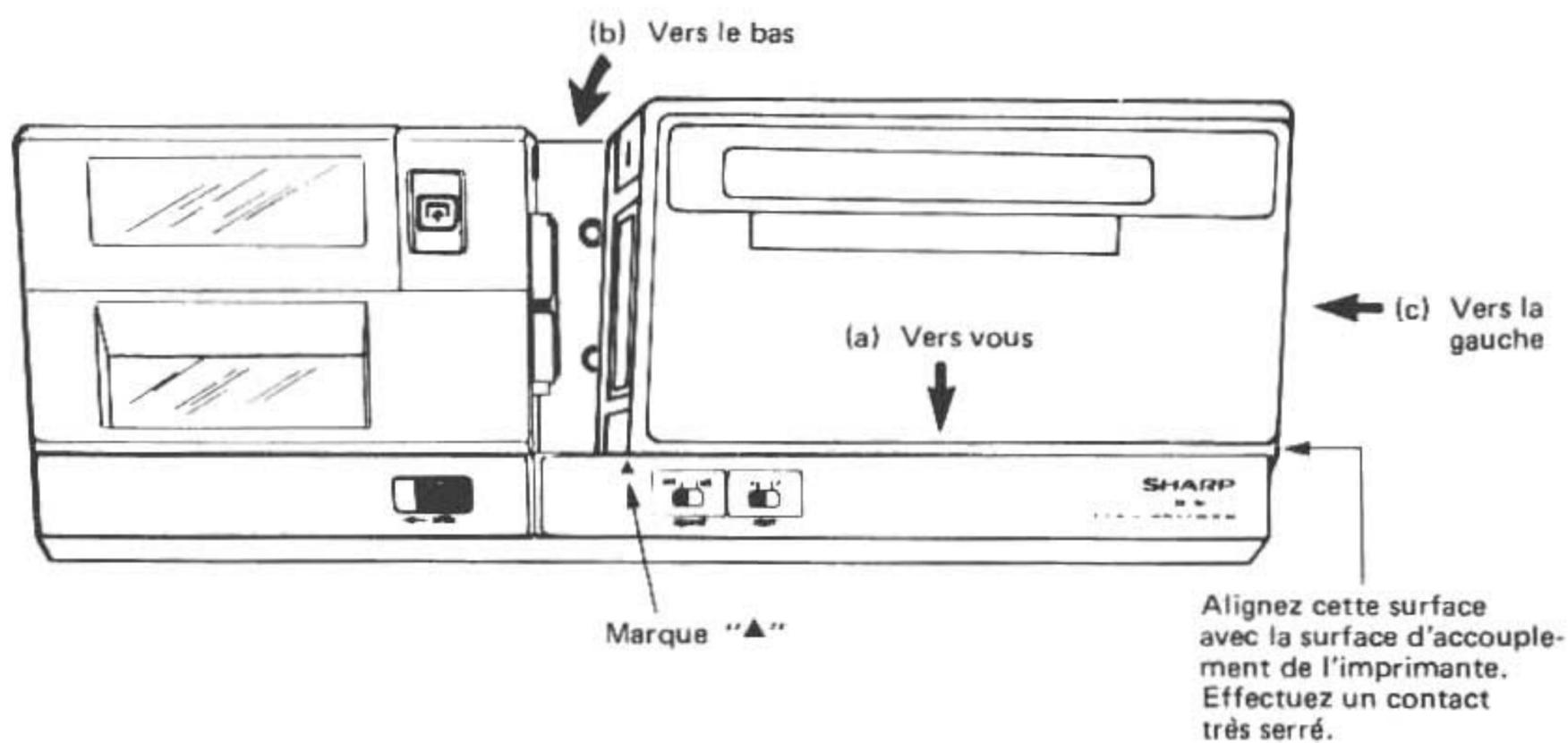
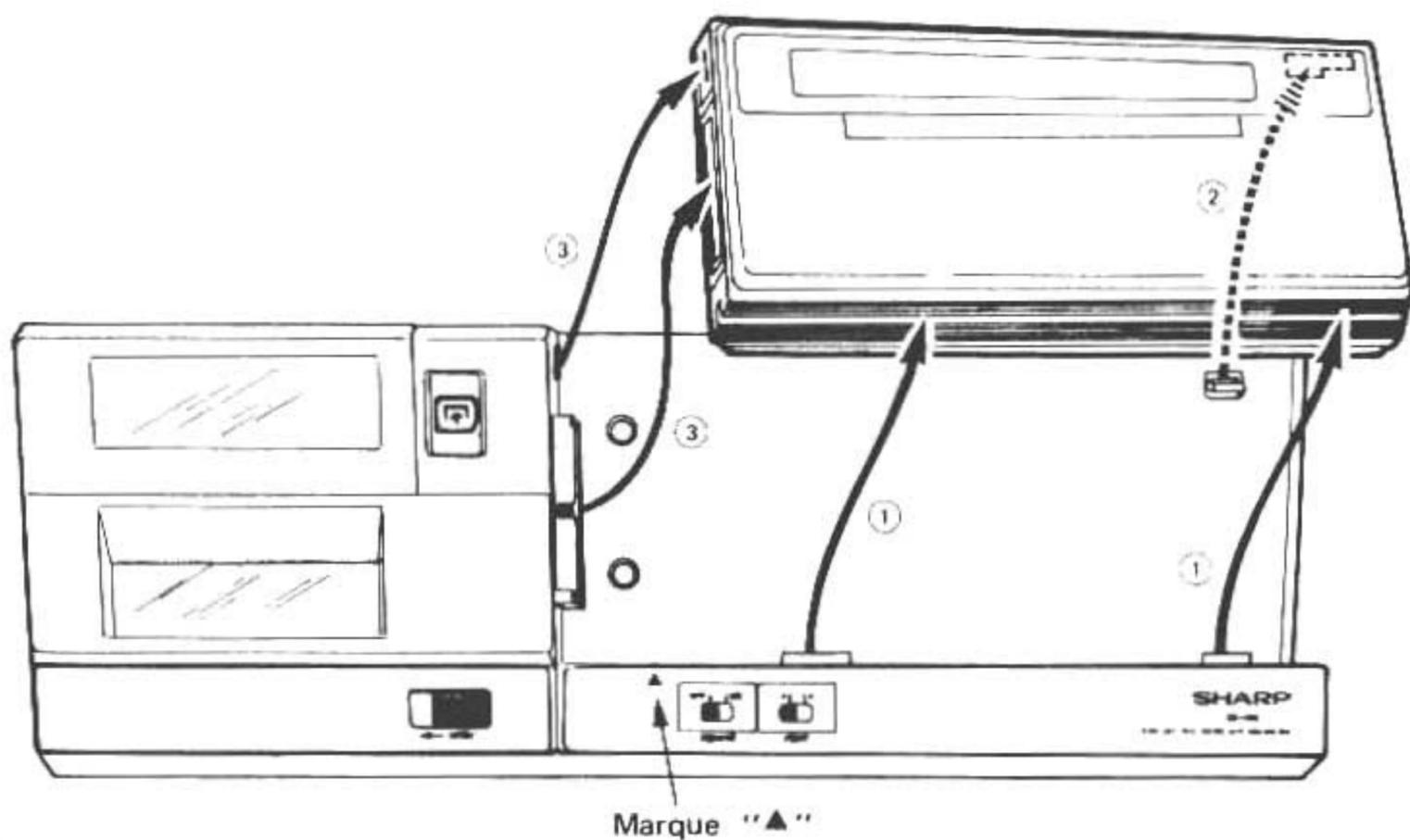
- (1) Mettez l'ordinateur hors-tension (OFF).

Note important! Ce point est essentiel. Si l'on laisse l'ordinateur sous alimentation (ON), il se peut que toutes les touches soient mises hors fonction. Si cela se produit, appuyez sur le commutateur ALL RESET situé au dos de l'ordinateur tout en maintenant la touche **ON** en position basse.

- (2) Détachez le couvercle protège-broche située sur le côté gauche de l'ordinateur et pressez-le en place sur le dos de l'ordinateur (voir l'illustration).



- (3) Placez la tranche inférieure de l'ordinateur dans le berceau de façon à aligner les languettes-guides de l'imprimante avec les fentes correspondantes de l'ordinateur.
- (4) Déposez l'ordinateur à plat.
- (5) Faites-le glisser doucement vers la gauche de façon à ce que les broches de l'imprimante entrent dans l'ordinateur. (Voir l'illustration).



Évitez de monter en force l'ordinateur sur l'imprimante. S'il s'avère difficile d'effectuer la mise en place, déplacez l'ordinateur avec précaution, légèrement, de droite à gauche pour obtenir le contact correct des surfaces.

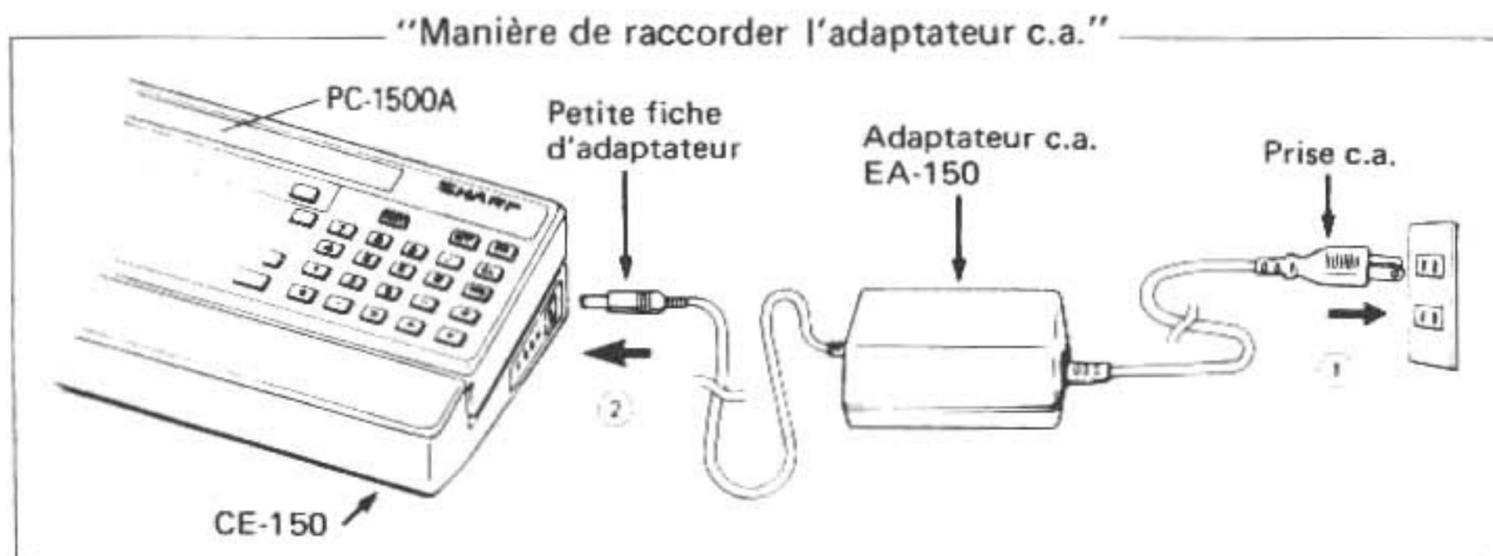
2. Alimentation

L'alimentation ou CE-150 est fournie par les piles Ni-CAD rechargeables. Il s'avère donc nécessaire de recharger les piles immédiatement après le déballage de l'appareil et, plus tard, chaque fois que l'affichage indique le message suivant. (Dans ce cas, l'imprimante est bloquée. Pour la débloquent, appuyez, dans l'ordre, sur les touches **OFF** et **ON** de l'ordinateur après avoir rechargé les piles).

(1) **ERROR 80** ou **ERROR 78**

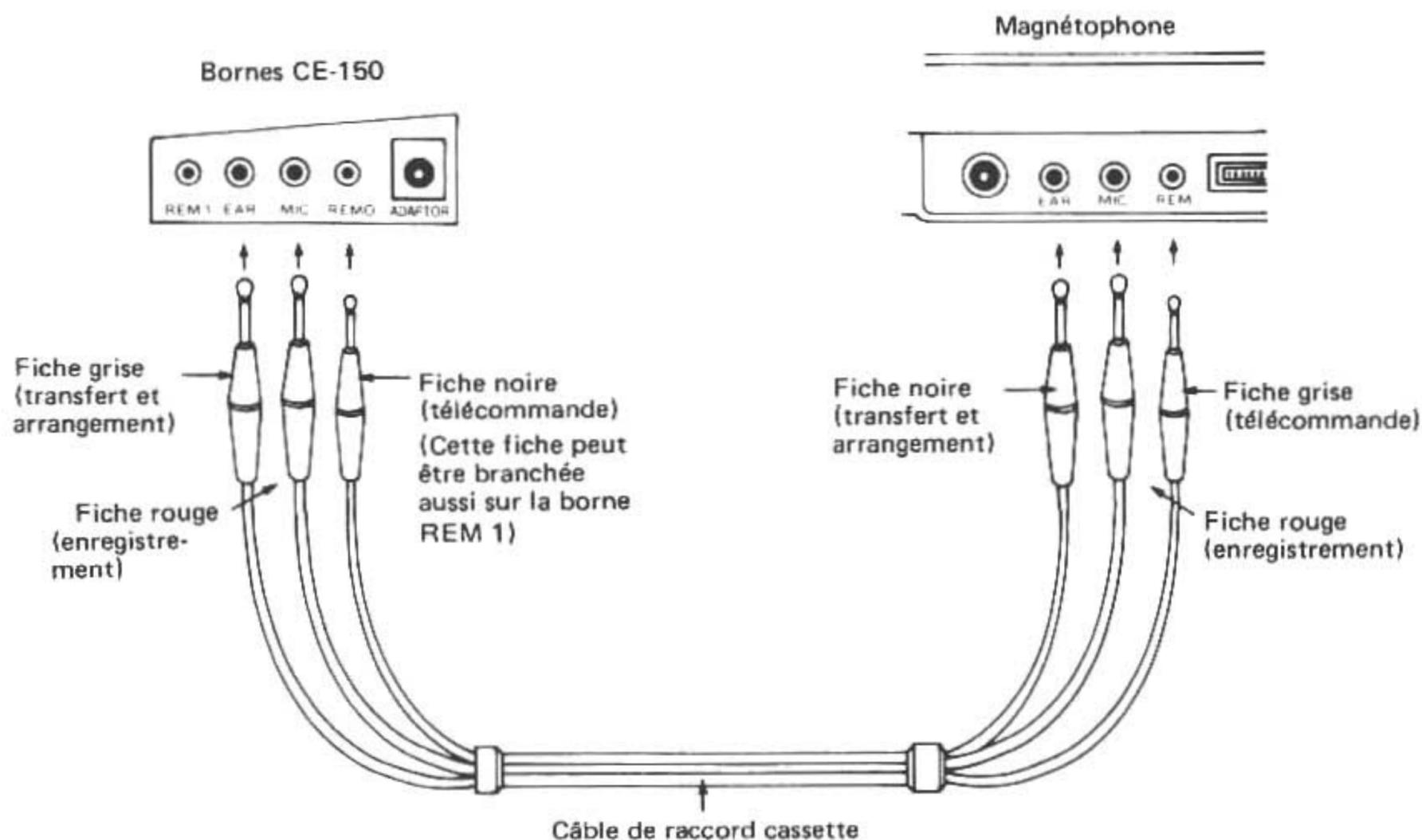
Note: Il se peut que le message d'erreur ERROR soit affiché lors du remplacement des stylos de l'imprimante.

(2) **:CHECK 6** ou **NEW 0? :CHECK 6**



3. Raccordement d'un magnétophone avec l'interface

Après avoir monté l'ordinateur sur le CE-150, raccordez le magnétophone avec le CE-150 de la façon indiquée par le schéma ci-dessous:



Ci-dessous vous trouverez une description des conditions minimum qu'un magnétophone doit remplir pour pouvoir être branché sur le CE-150:

Article	Conditions à remplir
1. Type de magnétophone	Il est possible d'utiliser n'importe quel magnétophone à cassette, micro-cassette ou à bande ouverte tant qu'il satisfait aux conditions énoncées ci-dessous.
2. Prise d'entrée	Le magnétophone doit posséder une mini-prise d'entrée "MIC". Ne jamais utiliser la prise "AUX".
3. Impédance d'entrée	L'impédance à la prise d'entrée doit être basse (200–1000 ohm).
4. Niveau d'entrée minimum	Inférieur à 3mV ou –50dB.
5. Prise de sortie	Une mini-prise "EXT". "MONITOR", "EAR" ou un équivalent.
6. Impédance de sortie	Doit être inférieure à 10 ohm.
7. Niveau de sortie	Supérieur à 1V (sortie pratique maximum supérieure à 100mW)
8. Distorsion	Doit se situer à 15% dans une gamme allant de 2kHz à 4kHz.
9. Taux de pleurage et scintillation	0,3% au maximum
10. Autres	La vitesse du moteur du magnétophone doit rester constante.

* Au cas où la minifiche fournie avec le CE-150 ne serait pas compatible avec les prises d'entrée ou de sortie de votre magnétophone, vous devrez vous procurer un raccord d'adaptation.

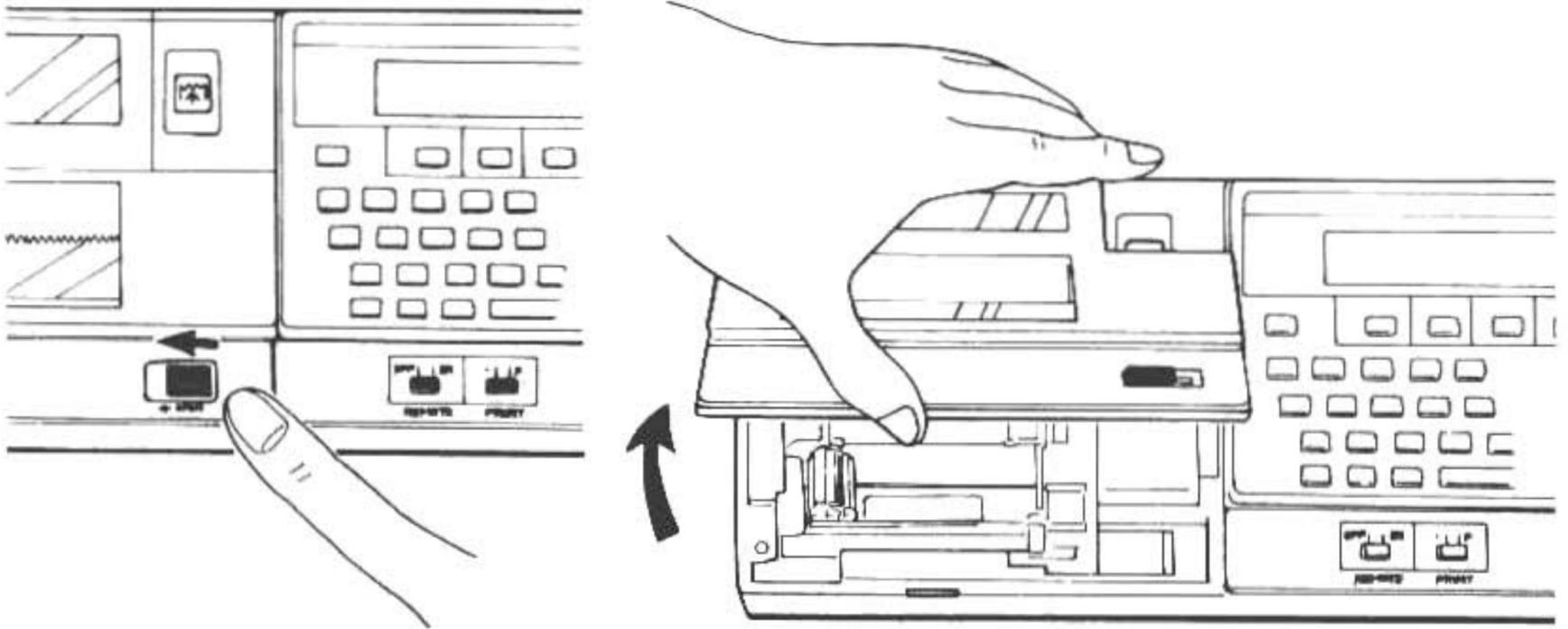
NOTE:

- Certains magnétophones ne peuvent pas être raccordés dû à l'incompatibilité des spécifications. D'autres ne sont pas satisfaisants parce que de longues années d'usage ont diminué leur puissance et augmenté leurs parasites et leur distorsion.
- Précautions à prendre lors de l'utilisation d'un magnétophone
 - (1) Lors d'un transfert ou d'un arrangement, utilisez toujours le magnétophone sur lequel l'enregistrement a été effectué. Il peut arriver que le transfert ou l'arrangement soit impossible si l'on néglige de prendre cette précaution.
 - (2) Assurez-vous de la propreté des têtes du magnétophone, afin d'éviter une augmentation du niveau de distorsion ou une diminution du niveau d'enregistrement.
 - (3) Evitez d'utiliser des bandes à réponse en fréquences très basse ou encore des bandes froissées ou égratignées.

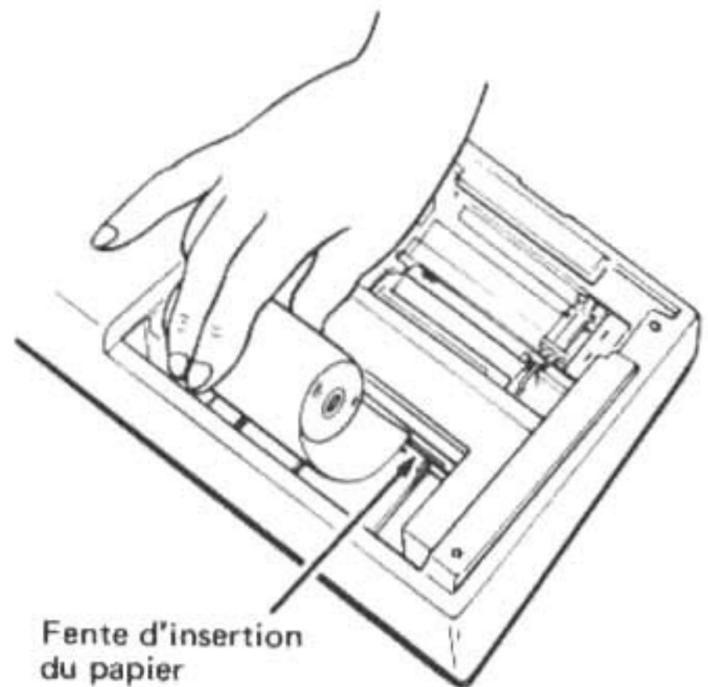
4. Chargement du papier

Pour plus de détails, reportez-vous au manuel CE-150.

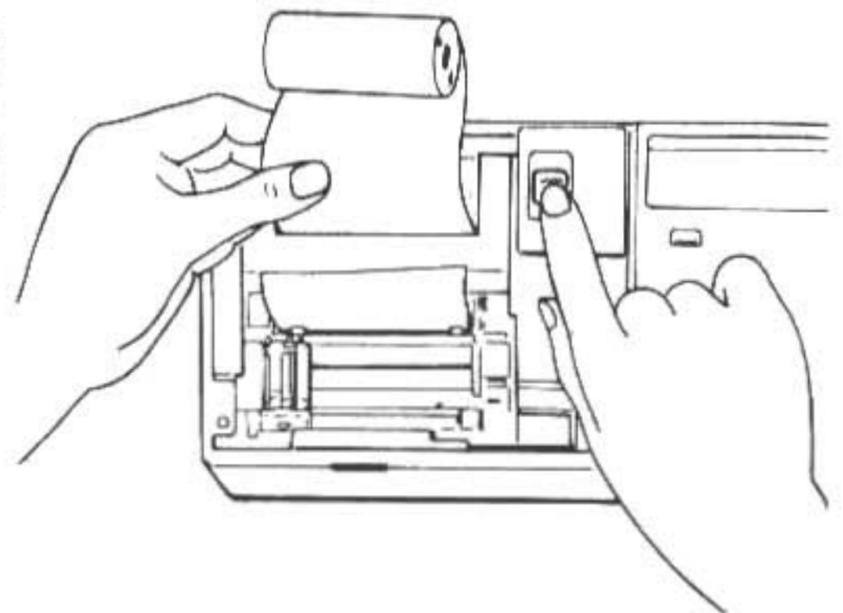
- (1) Poussez le levier de fermeture dans la direction de la flèche pour ôter le couvercle de l'imprimante.



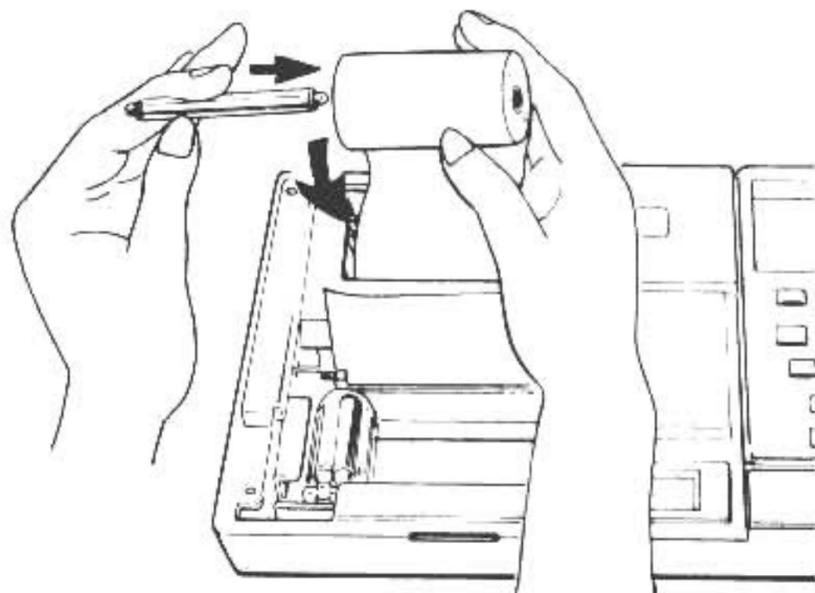
- (2) Coupez proprement l'extrémité du rouleau de papier, puis introduisez-la dans la fente d'insertion du papier. (Un pli ou un gondolage quelconque à l'extrémité du papier risque d'empêcher son insertion).



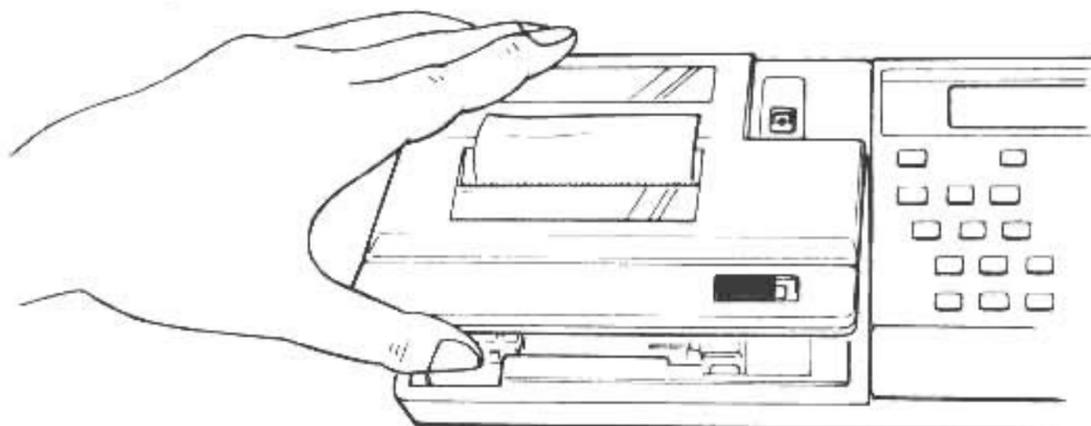
- (3) Mettez l'ordinateur en marche en appuyant sur la touche **ON**, puis sur la touche  pour effectuer l'alimentation en papier. A ce moment, laissez le papier sortir de 3 à 5 cm de l'imprimante.



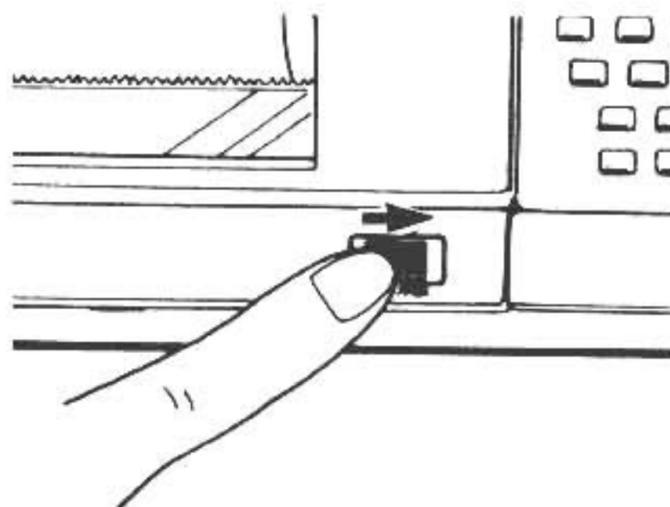
- (4) Introduisez l'arbre dans le rouleau de papier et placez l'ensemble dans le casier à papier.



- (5) Remettez le couvercle en place en faisant glisser à ce point l'extrémité du papier par la fente du coupe-papier.

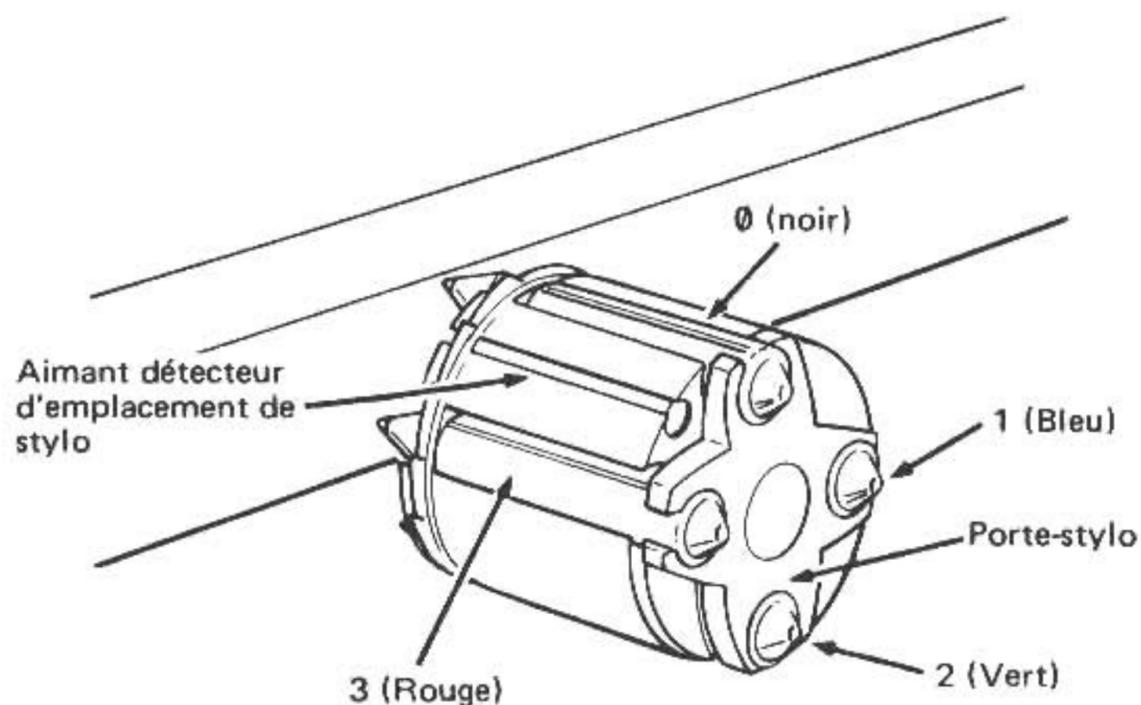


- (6) Fermez le couvercle à l'aide du levier.



5. Remplacement des stylos

Quatre stylos différents peuvent être montés sur ce dispositif. Le schéma ci-dessous indique les différents emplacements de montage de stylos. Consultez le manuel d'instruction de l'interface pour plus de détails.



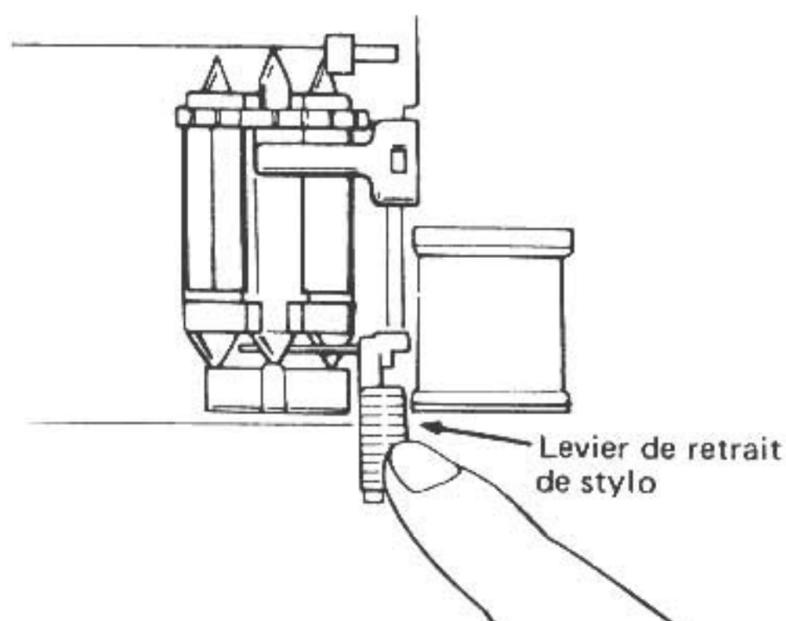
L'instruction COLOR désigne les emplacements de stylos, dans le sens des aiguilles d'une montre, à partir de l'aimant détecteur avec les numéros 0, 1, 2, et 3 comme indiqué sur la figure ci-contre. (Le porte-stylo tourne dans le sens inverse des aiguilles d'une montre pour amener le stylo choisi en haut.)

Pour monter ou remplacer les stylos, suivez les instructions ci-dessous:

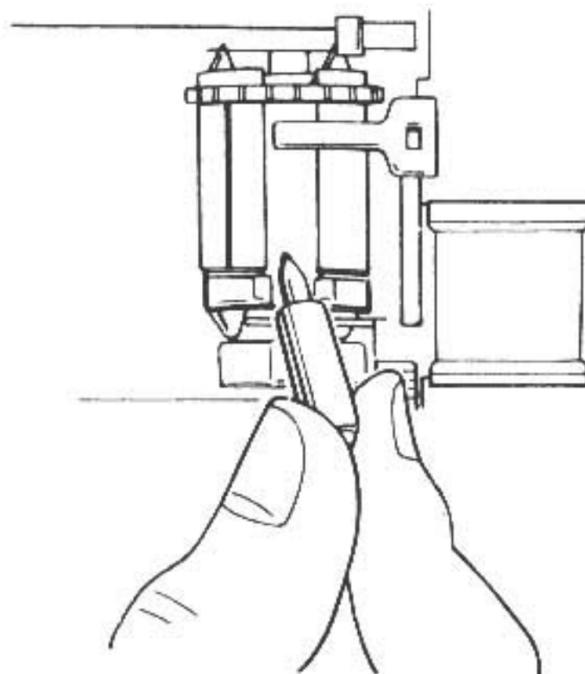
(1) Maintenez la touche **[O]** de l'ordinateur en position basse et appuyez sur la touche **[F]** de l'imprimante. Cette opération amène l'imprimante dans la position de remplacement de stylo: le porte-stylo se déplace vers la gauche et tourne. Quand le stylo situé en position supérieure a été remplacé, le porte-stylo se déplace vers la droite. (Relâchez la touche quand il commence à se déplacer).

(2) Pour déposer le stylo, appuyez sur le levier de retrait de stylo. Cette opération fait sauter le stylo situé en haut du porte-stylo.

(Note): Pour empêcher le stylo de retomber dans le boîtier de l'imprimante, retenez-le légèrement.



(3) Mettez en place un nouveau stylo.



(4) Appuyez à nouveau sur la touche **[F]** pour monter ou démonter le stylo suivant. Le porte-stylo revient sur la gauche et tourne jusqu'à ce que le stylo suivant arrive à la position supérieure. Vous pouvez maintenant procéder à son extraction et remplacement comme décrit sous (1) et (2).

(5) Quand le montage ou remplacement de stylo est terminé, appuyez sur la touche **[F]** de l'imprimante en maintenant la touche **[CL]** de l'ordinateur en position basse. Ceci libère l'imprimante de sa position de remplacement de stylo et le stylo revient sur la gauche.

(Note): Montez toujours quatre stylos sur le porte-stylo. Même si l'on omet simplement de monter un seul stylo, le mécanisme de changement de couleur risque de fonctionner mal.

Entretien des stylos:

Montez les stylos sur l'imprimante quand vous vous en servez et démontez-les quand vous ne vous en servez pas. Capuchonnez-les avant de les ranger dans leur étui.

Si vous les laissez en place sur l'imprimante et que vous ne vous en servez pas pendant longtemps, ou si vous les laissez à l'air libre, l'encre risque de se dessécher.

- | | |
|--|---------------------|
| 3. Ordinateur en marche? | oui _____ non _____ |
| 4. Programme entré? | oui _____ non _____ |
| 5. Raccordement effectué correctement? | oui _____ non _____ |
| 6. Numéro du compteur de bande noté? | oui _____ non _____ |
| 7. Etes-vous habillé? | aucun rapport _____ |

Si vous avez répondu "NON" à l'une de ces questions, relisez la section précédente pour résoudre le problème.

Bon, maintenant une chose encore et vous serez prêt à démarrer:

En même temps que la commande d'enregistrement vous devez donner au programme un "nom de fichier" pour des raisons de référence. La longueur de ce nom de fichier ne doit pas dépasser 16 caractères. Pour enregistrer le programme avec un nom de fichier, écrivez:

CSAVE SHIFT " PROG.-1 SHIFT "

Votre programme sera enregistré sous le nom de "PROG.-1". Vous êtes libres de le nommer comme vous voulez pourvu qu'il soit facile de s'en souvenir et que sa longueur ne dépasse pas la limite de 16 caractères. La partie dépassant cette limite sera simplement ignorée. Nous vous recommandons de commencer dès maintenant un carnet de bord dans lequel vous noterez le nom des programmes, le numéro de début et de fin de l'enregistrement sur bande (si votre magnétophone est doté d'un compteur) et une brève description de la fonction du programme.

Appuyez sur la touche ENTER . Vous devriez entendre maintenant le son aigu d'un avertisseur et la bande devrait défiler. De plus l'indicateur "busy" (occupé) devrait s'allumer pour vous indiquer que l'ordinateur est occupé à transférer votre programme de sa mémoire à la bande. Si cela ne se produit pas, recommencez de nouveau au début de cette section. Il m'a fallu seulement 10 tentatives avant d'arriver à le faire correctement! Ne vous découragez donc pas. Si moi j'ai réussi à le faire, vous le pouvez aussi.

Dès qu'il arrive au bout du programme l'ordinateur éteint l'indicateur "busy", le magnétophone s'arrête et le symbole de guidage réapparaît sur l'affichage. Félicitations! Vous venez d'enregistrer un programme pour la première fois! Pour nous assurer cependant que tel a bien été le cas nous allons le retransférer dans la mémoire de la manière indiquée dans la section suivante.

* On peut aussi à l'aide de cette instruction sauver des programmes en langage machine.

CSAVE M SHIFT "L.M SHIFT "; nn, pp, qq

Ceci sauvera le programme ou la zone mémoire allant de l'adresse nn à l'adresse pp.

Le troisième paramètre peut être omis, il représente l'adresse d'entrée du programme (le retour sour BASIC se fait alors par àée et non par &9A).

nn adresse début zone

pp adresse fin zone

qq adresse d'entrée de la zone

3. La commande CLOAD

Maintenant que votre programme a été enregistré sur bande, vous voudrez certainement vérifier si le programme y est réellement. Ce qui est relativement facile à faire à l'aide de la commande CLOAD, bien sûr.

Vous allez vous demander: "Mais quel est son effet?". Ecrire simplement CLOAD?. L'ordinateur compare le programme enregistré avec celui dans sa mémoire. Si tout s'est bien passé, il affichera calmement votre nom de fichier et finira sa vérification. Si tout ne s'est pas bien passé, il affichera un message d'erreur, généralement ERROR 43. Ce message vous indique que le

programme sur bande diffère de celui de la mémoire du SHARP. Effacez la portion de bande où l'erreur s'est produite et recommencez après avoir vérifié tous les raccords et augmenté légèrement le volume et la tonalité.

Pour utiliser la commande CLOAD? il vous faut amener la bande jusqu'à la tonalité porteuse qui précède les données enregistrées. La tonalité porteuse est un signal fixe, non modulé d'environ 2,5kHz. Si la bande n'est pas amenée au point correct, aucune indication d'erreur n'est indiquée et aucune donnée n'est présentée sur l'affichage.

Ci-dessous vous trouverez les instructions concernant le rechargement du programme dans l'ordinateur.

1. Placez le commutateur "remote" de l'interface sur OFF.
2. Rebobinez la bande jusqu'au point de départ en vous servant du compteur. (Très pratique, ce compteur, n'est-ce pas?)
3. Arrêtez le rebobinage.
4. Placez le commutateur "remote" sur ON.
5. Appuyez sur la touche PLAY.
6. Ecrivez:

CLOAD **SHIFT** "PROG.-1 **SHIFT** "
et appuyez sur la touche **ENTER** .

(Rappelez-vous que "PROG.-1" constitue le nom de fichier que vous avez donné à votre programme. Si vous l'avez enregistré sous un autre nom, il vous faudra utiliser ce dernier au lieu de "PROG.-1".)

7. L'indicateur "busy" s'allume maintenant et le programme est ramené dans la mémoire de l'ordinateur pour être utilisé.
 8. Ecrivez RUN et à ce moment le programme que vous aviez enregistré auparavant réapparaît. Comme la copie sur bande est restée intacte, vous pouvez recharger le même programme aussi souvent que vous voudrez!
- * De même que pour CSAVE, à l'aide du paramètre M, cette instruction peut permettre de charger un programme en langage machine.

CLOAD M **SHIFT** "L.M **SHIFT** " ; nn

nn peut être omis, nn représente l'adresse à partir de laquelle le programme en langage machine devra être chargé. En cas d'émission, il se chargera à l'adresse où il était au moment de la sauvegarde.

4. Les Commandes PRINT # et INPUT

PRINT # :

Maintenant que nous avons expliqué l'utilisation de CSAVE et CLOAD, nous allons introduire deux commandes qui leur ressemblent. La commande PRINT # enregistre la valeur d'une variable ou d'une série de variables sur bande, alors que CSAVE enregistre un programme. La raison pour laquelle nous enregistrons des données, c'est que cela nous permettra de nous servir des mêmes données dans un autre programme. Par exemple, on se sert de la variable T\$ dans le programme suivant:

```
10: PRINT "QUEL EST VOTRE NOM?"  
20: INPUT T$  
30: PRINT T$
```

Si vous voulez enregistrer la valeur de TS pour l'utiliser dans un autre programme, vous devez lancer la commande PRINT# pour l'enregistrer sur bande. Il y a deux manières d'effectuer cette opération:

1. Manuellement
2. Par l'intermédiaire d'un programme

Note: comme le magnétophone est nécessaire pour cette opération, préparez-le en vue de recevoir des données. Si vous n'êtes pas sûr de savoir comment faire, reportez-vous à la section précédente.

1. La méthode manuelle

Plusieurs formats aux choix sont possibles:

CHOIX 1:

Après le passage d'un programme, passez sur le mode RUN et écrivez:

`P R I N T` `SHIFT` `#` A, B, C (puis appuyez sur `ENTER`)

Le magnétophone se met alors en marche et enregistre les valeurs de toutes les variables sur la bande.

CHOIX 2:

Ecrivez selon le format suivant si vous voulez identifier seulement certaines variables à enregistrer:

`P R I N T` `SHIFT` `#` "nom de fichier"; A, B, C

Dans cet exemple-ci, vous avez spécifié les variables A, B et C comme étant celles qui sont à enregistrer sur bande sous le nom de fichier donné.

CHOIX 3:

Vous pouvez aussi spécifier toutes les valeurs de variables apparentées en écrivant:

`P R I N T` `SHIFT` `#` "nom de fichier" ; B (*)

Le symbole * causera l'enregistrement de toutes les variantes de "B", y compris B (1) et B lui-même.

2. Par l'intermédiaire d'un programme

Dans ce cas-ci, il suffit d'affecter un numéro de ligne à la commande PRINT# de votre programme. Vous pouvez choisir un format quelconque parmi ceux décrits ci-dessus. Quand le SHARP rencontre ce numéro de ligne il met automatiquement le magnétophone en marche pour amorcer le transfert de données sur la bande. Une fois de plus, N'HESITEZ PAS à expérimenter. Si vous rencontrez des difficultés, reportez vous aux sections précédentes. Parfois l'on oublie les choses les plus simples.

NOTE: Le nom de fichier représente le contenu d'une chaîne de caractères ou une variable de caractère libellée avec". Par conséquent, si la première variable qui doit être enregistrée ou stockée comme spécifié sans nom de fichier est une variable de caractère, cette variable de caractère est considérée comme un nom de fichier. Dans ce cas, veiller à placer un nom de fichier.

Exemple 1: 10 A\$ = "TAPE"

20 PRINT# A\$; X\$

Ceci est identique à PRINT# "TAPE"; X\$

Exemple 2: PRINT# X

Ceci n'enregistre que le contenu de la variable numérique X sans nom de fichier.

Exemple 3: PRINT# X\$

Ceci n'enregistre pas seulement le contenu de la chaîne de caractères X\$ sans nom de fichier, mais le contenu de X\$ est considéré comme nom de fichier. Aucun point-virgule (;) n'est placé après le nom de fichier, ce qui provoque une erreur de syntaxe (ERROR 1).

Par conséquent, si la première variable est une variable de caractère, veiller à placer un nom de fichier tel que:

```
PRINT# "AAA": X$
```

INPUT# :

Cette commande permet les mêmes formats que PRINT#. A la différence de PRINT# qui transfère des données de l'ordinateur sur la bande, INPUT# transfère des données de la bande à l'ordinateur.

Vous pouvez utiliser la commande INPUT# manuellement ou comme part d'un programme. N'oubliez pas de préparer le magnétophone avant de vous servir de cette commande.

- NOTE:**
- Si comme partie de votre ligne de command INPUT#, vous spécifiez plus de variables qu'il n'en existe sur la bande, l'ordinateur affectera aux variables en excès la valeur 0. Si vous spécifiez moins de variables qu'il n'en existe sur la bande, l'ordinateur ignorera celles qui ne sont pas spécifiées.
 - Si le nombre de données enregistrées est inférieur à celui des mémoires de données devant être garnies avec elles, l'erreur 43 se produit après que toutes les données ont été transférées dans les mémoires de données dans l'instruction INPUT#.
 - Si le nombre des données enregistrées est supérieur à celui des mémoires de données devant être garnies avec elles, les données sont transférées jusqu'à ce que les mémoires de données soient remplies, et les variables restantes sont ignorées.

5. La commande MERGE

La commande MERGE vous permet de stocker en même temps de nombreux programmes dans la mémoire de l'ordinateur. Supposons, par exemple, que la mémoire de l'ordinateur contienne le programme suivant:

```
10: PRINT "MOINS VALEUR DEDUCTIBLE"  
20: INPUT "Entrez votre méthode: " ; A
```

A ce moment, vous vous souvenez tout d'un coup que vous possédez, enregistré sur bande, un morceau de programme semblable sous le nom de fichier "DEP1". Vous voulez, bien sûr, vérifier si ce programme ne contient pas des morceaux qui pourraient vous servir dans le programme que vous êtes en train de fabriquer. Le premier pas consistera à trouver la bande contenant "DEP1". Amener la bande à l'endroit où "DEP1" commence et écrivez: MERGE "DEP1" et appuyez sur **ENTER**. L'ordinateur va maintenant charger "DEP1" dans sa mémoire EN PLUS du programme ci-dessus. Quand "DEP1" est chargé, vous trouverez dans la mémoire quelque chose du type suivant:

```
10: PRINT "MOINS VALEUR DEDUCTIBLE"  
20: INPUT "Entrez votre méthode: " ; A  
10: "DEP1" : REM >> Second Module <<  
20: PRINT "MONTANT DES INTERETS"  
30: INPUT "Somme empruntée: " ; B  
:  
(etc.)
```

Notez que, à la différence de la commande CLOAD, le nouveau programme n'a PAS remplacé celui en existence, et que vous avez des numéros de lignes en double à présent. Vous remarquerez aussi qu'une "indication" a été utilisée sur la première ligne du module mis en forme. Ceci permet l' "ENCHAINEMENT" des modules (Voir ENCHAINEMENT DE MODULES MIS EN FORME-ci-dessous).

Une révision des informations suivantes s'impose avant de poursuivre l'édition ou la programmation:
NOTES IMPORTANTES:

Après qu'une commande MERGE a été effectuée, aucune INSERTION, SUPPRESSION ou MODIFICATION n'est possible dans les lignes de programme existant précédemment.

Exemple:

```
10 "A" REM Ceci est un programme existant
20 FOR T = 1 TO 100
30 LPRINT T
40 NEXT T
   (etc.)
```

AVANT de procéder à la mise en forme du programme suivant, apportez les modifications nécessaires à ce programme.

Ensuite, mettez le programme suivant en forme: MERGE "PROG2"

(exemple)

```
10 "B" REM Ceci est un programme mis en forme
20 INPUT "Entrez la dépréciation: "; D
30 INPUT "Nombre d'années: "; Y
40 . Etc.
```

Vous pouvez maintenant apporter des modifications au programme ci-dessus puisqu'il s'agit de la dernière partie mise en forme. Si vous devez modifier le premier programme, suivez alors cette procédure:

1. Enregistrer sur bande (CSAVE) ce que vous avez réalisé jusqu'à présent; lors de l'enregistrement, utilisez un nouveau nom, c'est à dire "PROG3" (exemple).
2. Ramener le programme enregistré à l'étape 1 dans la mémoire (CLOAD).
3. Apportez maintenant toute modification ou changement désiré. Rappelez-vous cependant que: les changements ne sont possibles QUE dans le premier programme ou dans la PREMIERE APPARITION de tout numéro de ligne en double. C'est à dire que si la ligne 10 apparaît dans le premier de même que dans le second programme, les changements n'affecteront que la première apparition de la ligne 10. Si vous essayez donc d'éditer la seconde apparition de la ligne 10, la modification apparaîtra de façon erronée dans la première partie du programme uniquement.

ADJONCTION DE LIGNES DE PROGRAMMATION SUPPLEMENTAIRES:

Des lignes supplémentaires peuvent être ajoutées à la fin des programmes existants SEULEMENT SI ELLES POSSEDENT DES NUMEROS DE LIGNE PLUS GRANDS QUE CEUX UTILISES PRECEDEMMENT.

Remarquez que les lignes de programmation supplémentaires ne doivent pas apparaître nécessairement "tout en bas" du listage. Comme les modules doivent être enchaînés au moyen d'indications (voir ci-dessous) ne vous faites donc pas de souci à ce sujet.

ENCHAINEMENT DE MODULES (programmes) MIS EN FORME

Comme l'unité de traitement exécute vos lignes de programmation dans l'ordre logique, il s'arrêtera s'il rencontre une rupture dans l'ordre de numérotage des lignes. En d'autres termes, si les numéros de ligne 10, 20, 30 sont suivis par des numéros de ligne en double dans le second module, l'unité de traitement s'arrête après la ligne 30 du premier module.

Pour "ENCHAINER" vos différents modules, utilisez les techniques suivantes: GOTO "B", GOSUB "B", IF . . . THEN "B" (B est utilisé uniquement à titre d'exemple; vous pouvez utiliser n'importe quelle indication, sauf les mots ou lettres réservés qui apparaissent dans la 2ème rangée du clavier, c'est à dire de Q à P).

NOTE: Lorsqu'un autre programme est appelé par un sous-programme avec l'instruction GOSUB dans une pluralité de programmes interclassés, programmer une instruction GOTO (un numéro de ligne supérieur à celui de la ligne devant être exécutée) après l'instruction GOSUB.

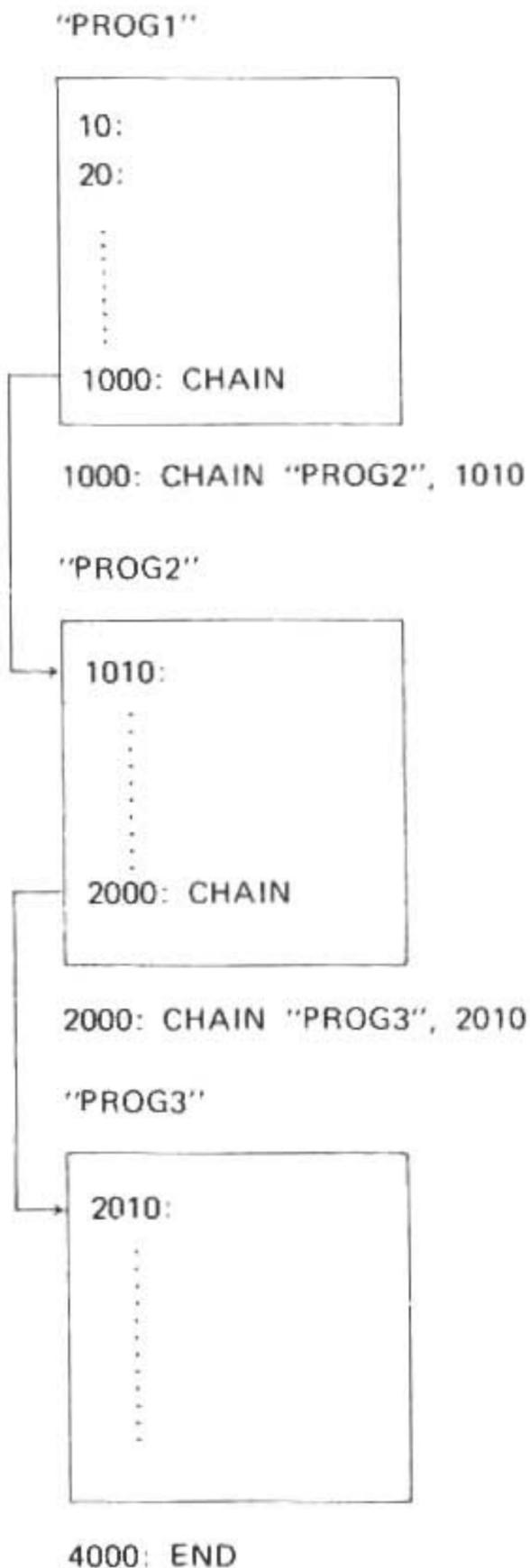
Exemple:

```
10 "A" : INPUT A
20 GOSUB "B" : GOTO 30
30 GOTO 10
10 "B" : PRINT A
20 RETURN
```

6. L'instruction CHAIN

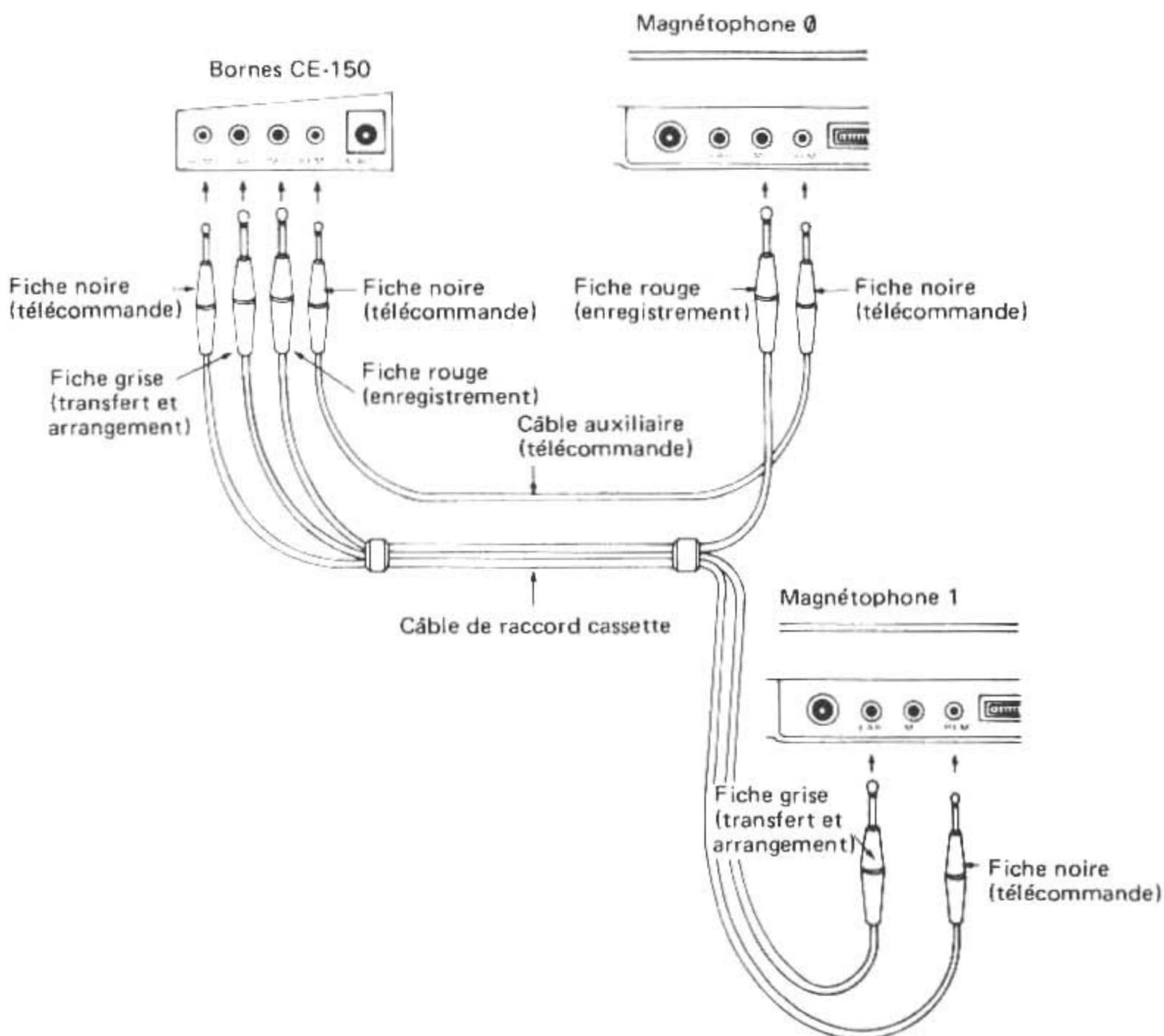
CHAIN constitue une instruction de programme. On peut l'utiliser uniquement à l'intérieur d'un programme. On ne peut pas l'utiliser manuellement comme CSAVE, CLOAD et MERGE. L'instruction CHAIN vous permet de passer un programme trop grand pour être logé tout entier dans la mémoire en une fois. Ce type de programme est coupé en morceaux à la fin de chacun desquels nous placerons une instruction CHAIN. Nous pourrons ensuite enregistrer ces morceaux sur bande.

Supposons, comme exemple, que nous ayons trois sections de programme nommées PROG1, PROG2, PROG3. Chacune se termine par une instruction CHAIN.



Quand l'ordinateur rencontre l'instruction CHAIN au cours de l'exécution, la nouvelle section est rappelée dans la mémoire et exécutée. De cette façon, toutes les sections finiront par passer.

7. Utilisation de deux magnétophones



Lorsqu'on utilise deux magnétophones, l'un sert à enregistrer et l'autre réécouter. Comme l'indique le schéma, l'interface imprimante/cassette et les deux magnétophones sont raccordés au moyen d'un câble de raccord cassette et d'un câble auxiliaire pour la télécommande.

- Le CE-150 est équipé de deux bornes de télécommande, REM 0 et REM 1; on peut se servir de l'une ou de l'autre. Durant l'opération programmée (non manuelle) cependant, le programme indique si le magnétophone est connecté à la borne REM 0 ou la borne REM 1. Leur connexion est donc à effectuer en accordance avec le programme.

La manière d'utiliser le second magnétophone est expliqué dans ce chapitre.

1. Sauvegarde

Procédures:

- (1) Ecrivez RMT OFF, puis appuyez sur **ENTER** pour régler à nouveau la deuxième fonction de télécommande (pour le contrôle de magnétophone 1 de la figure précédente).
- (2) Placez une bande dans le magnétophone.
- (3) Ecrivez RMT ON, puis appuyez sur **ENTER** pour régler la deuxième fonction de télécommande.
- (4) Réglez le volume et la tonalité de la même manière que dans le cas d'un seul magnétophone. (Voir plus haut).
- (5) Appuyez simultanément sur RECORD et PLAY.

- (6) Exécutez l'instruction ENREGISTREMENT.
Programme: **CSAVE-1 "nom de fichier"**
Données: **PRINT# -1 "nom de fichier"; variable, variable, . . .**
(Exemple) Placez sur le mode PRO ou RUN.
CSAVE-1 "PR-1"

Après la sauvegarde, le symbole de guidage réapparaît sur l'affichage et la bande s'arrête. Rebobinez-la pour l'arrangement.

2. Arrangement du contenu de l'ordinateur et de la bande

Procédures:

- (1) Ecrivez **RMT OFF** pour remettre les fonctions de télécommande à zéro.
- (2) Robobinez la bande jusqu'au point de départ à l'aide du compteur.
- (3) Entrez **RMT ON**, puis appuyez sur la touche pour régler les fonctions de télécommande.
- (4) Réglez le volume et la tonalité de la même manière que dans le cas d'un seul magnétophone.
- (5) Appuyez sur la commande de réécoute.

CLOAD?-1" nom de fichier"

(Exemple) Placez sur le mode PRO ou RUN. **CLOAD?-1 "PR-1"**

L'exécution est achevée quand les deux versions concordent et que le symbole de guidage réapparaît.

3. Transfert à partir de la bande

Procédures:

- (1) Entrez **RMT OFF**, puis appuyez sur la touche pour remettre les fonctions de télécommande à zéro.
- (2) Placez une bande avec enregistrement dans le magnétophone.
- (3) Ecrivez **RMT ON**, puis appuyez sur pour régler les fonctions de télécommande.
- (4) Réglez le volume et la tonalité de la même manière que dans le cas d'un seul magnétophone.
- (5) Appuyez sur la commande de réécoute.
- (6) Exécutez l'instruction TRANSFERT

Programme: **CLOAD-1 "nom de fichier"**

Données: **INPUT# -1, "nom de fichier"; variable, variable,**

(Exemple) Placez sur le mode PRO ou RUN.

CLOAD-1 "PR-1"

Après le transfert, le symbole de guidage réapparaît sur l'affichage.

C. UTILISATION DE L'IMPRIMANTE

NOTE: Toutes les explications et les exemples de cette section se basent sur la supposition que vous avez déjà:

- 1) Effectué correctement la connexion entre le PC-1500A et l'interface CE-150.
- 2) Fourni une source d'alimentation au CE-150 sous la forme de piles ou de l'adaptateur EA-150.
- 3) Chargé le papier dans l'imprimante.
- 4) Monté les stylos sur l'imprimante.

Si vous n'avez pas effectué ces opérations préliminaires, retournez à la section 1 de ce chapitre pour les instructions.

C.1. Spécifications de l'imprimante CE-150

Caractères par ligne:	4, 5, 6, 7, 9, 12, 18, ou 36 selon la taille de caractère choisie.
Tailles des caractères:	1,2 x 0,8 mm à 10,8 x 7,2 mm selon la taille de caractère choisie.
Vitesse d'impression:	Maximum: 11 caractères par seconde quand on imprime avec les caractères les plus petits.
Rotation:	L'impression peut se faire dans les deux sens de chacun des axes des X et des Y
Couleurs:	4 – Rouge, bleu, vert et noir
Système graphique:	Traçage de l'axe X-Y
Alimentation en papier:	Manuelle ou programmable.

C.2. La commande TEST

La première chose à faire est de tester le fonctionnement de l'imprimante CE-150. L'ordinateur et l'interface étant sur ON, écrivez:

TEST (et appuyez sur **ENTER**)

L'imprimante va maintenant dessiner 4 carrés, chacun d'une couleur différente. La couleur des carrés, de droite à gauche, correspond à celle que vous avez choisi comme étant couleur 0, 1, 2 et 3 quand vous avez monté les stylos.

C.4. Modes de l'imprimante

L'imprimante fonctionne sur l'un de deux modes: TEXT ou GRAPH. Ils correspondent grosso modo à l'écriture à la machine et au dessin. Comme la plupart des commandes ne sont efficaces que sur un seul mode, il est important de choisir le mode correct avant de commencer à donner des instructions.

On se sert du mode TEXT pour imprimer des nombres et des caractères. Le papier à imprimer est divisé en colonnes dont le numéro est en relation avec la taille de caractère spécifiée. Des commandes de contrôle vertical et horizontal permettent de créer le format du texte.

Sur le mode GRAPH, on peut procéder à la création de différents types de figures, de courbes et de tables. Des commandes pour le dessin de lignes continues ou discontinues, à l'aide d'un système de coordonnées relatives ou directes sont à la disposition de l'utilisateur. Tous les dessins sont basés sur le schéma ordinaire de coordonnées X-Y.

Pour spécifier le mode TEXT, il suffit d'écrire l'instruction:

TEXT

Certaines commandes (dont nous parlerons plus loin) provoquent un passage automatique sur le mode TEXT.

La spécification du mode GRAPH est tout aussi simple. L'instruction:

GRAPH

amorcera ce mode et fera passer le stylo à l'extrémité gauche du papier.

C.5. Listage du programme

La commande LLIST provoque l'impression du programme courant ou de morceaux de ce programme. La commande LLIST s'avère extrêmement utile durant le développement du programme parce qu'elle permet l'impression au choix de certaines parties du programme.

La forme de la commande LLIST est semblable à celle de la commande LIST. Les subtiles différences sont dues à la plus grande souplesse de la commande LLIST. Les formes de LLIST sont esquissées ci-dessous:

LLIST

- Imprime toutes les lignes de programme en existence à ce moment dans la mémoire.

LLIST expression

- Imprime seulement la ligne de programme dont le numéro est donné par l'expression.

LLIST , expression

- Imprime toutes les lignes de programme jusqu'à celle dont le numéro est donné par l'expression, y compris cette dernière:

LLIST expression,

- Imprime toutes les lignes de programme suivant celle dont le numéro est donné par l'expression, y compris celle-ci.

LLIST expression1, expression2

- Imprime les lignes de programme commençant par la ligne dont le numéro est donné par expression1 et se terminant par la ligne dont le numéro est donné par expression2. Donc, si la commande est:

LLIST 100, 150

toutes les lignes, entre 100 et 150 (s'il y en a) seront listées.

LLIST "étiquette"

- Imprime la ligne de programme contenant l'étiquette donnée.

LLIST "étiquette",

- Imprime les lignes de programme qui suivent la ligne contenant l'étiquette en commençant par celle-ci.

NOTE: Un message d'erreur ERROR 11 signalera la spécification d'une étiquette qui n'existe pas.

Lors de l'impression d'un programme, la couleur utilisée est celle qui a été spécifiée précédemment. Si vous venez juste de mettre l'appareil en marche, ce sera la couleur de stylo que vous avez choisi comme correspondant à Couleur 0. Pour changer la couleur vous devrez lancer la commande COLOR (voir la section appropriée).

La taille des caractères utilisés pour lister un programme correspond à celle spécifiée auparavant. Si vous avez spécifié précédemment des caractères de taille 1 ou 2, cette taille sera utilisée; si la taille spécifiée antérieurement était plus grande que 2, taille 2 sera utilisée.

La commande LLIST établit l'imprimante sur le mode TEXT. Si vous étiez sur le mode GRAPH et que vous souhaitiez retourner sur ce mode vous devez lancer la commande GRAPH.

Pendant le listage du programme, l'ordinateur PC-1500A essaie de cadrer les lignes de programme avec comme objectif la lisibilité. Il accomplit cela en laissant des espaces vides entre les numéros de ligne. Des numéros de ligne d'une largeur de 1 à 3 chiffres seront cadrés à droite dans un champ de 3 caractères. Des numéros imprimés dans un champ de 5 caractères:

```
10: REM LARGEUR 3
20: REM      "
300: REM      "
2001: REM LARGEUR 5
2010: REM      "
```

C.6. Contrôle programmable de l'imprimante

C.6.1. CSIZE

La commande CSIZE spécifie la taille des caractères pour toute impression ultérieure. Les neuf tailles disponibles permettent d'aller de 36 caractères par ligne imprimée à 4 caractères par ligne imprimée. La forme de la commande CSIZE est la suivante:

CSIZE expression
(sur l'un ou l'autre mode)

L'expression doit consister en un nombre allant de 1 à 9. La largeur et la hauteur des caractères pour chaque taille sont indiquées dans le tableau suivant:

Table 1:

CSIZE	1	2	3	4	5	6	7	8	9
Caractères par ligne imprimée	36	18	12	9	7	6	5	4	4
Hauteur de chaque caractère (mm)	1,2	2,4	3,6	4,8	6,0	7,2	8,4	9,6	10,8
Largeur de chaque caractère (mm)	0,8	1,6	2,4	3,6	4,0	4,8	5,6	6,4	7,2

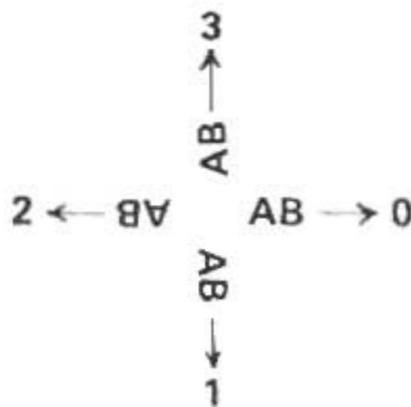
C.6.2. La commande ROTATE

Cette commande ROTATE (faire pivoter) est utilisée sur le mode GRAPH uniquement pour spécifier le sens dans lequel l'impression s'accomplit. Quatre sens sont possibles: vers le haut, vers le bas, de la gauche à la droite et de la droite à la gauche (avec les lettres à l'envers). La figure 1 illustre ces mouvements. La forme de la commande ROTATE est la suivante:

ROTATE expression
(sur le mode GRAPH uniquement)

L'expression doit consister en un nombre de 0 à 3. ROTATE 0 spécifie la façon ordinaire d'imprimer des caractères, de la droite à la gauche.

Figure 1:



C.6.3. La commande COLOR

Cette commande COLOR (Couleur) sert à spécifier le stylo que l'on va utiliser dans les impressions et les dessins ultérieurs. Si chaque emplacement de stylo contient un stylo de couleur différente, la commande COLOR peut servir à changer de couleur de plume. La forme de la commande COLOR est:

COLOR expression
(sur les deux modes)

L'expression doit consister en un nombre entier allant de 0 à 3. Chaque nombre entier représentée par le nombre varie selon l'ordre dans lequel on a monté les stylos sur les porte-stylos. On peut vérifier quel nombre correspond à quelle couleur à l'aide de la commande TEST (décrite plus haut).

Des nombres qui ne sont pas des nombres entiers mais toujours encore de 0 à 3 seront ramenés à des entiers. Tous les autres nombres résulteront en un message d'erreur ERROR 19.

Si l'on arrête puis remet en marche le PC-1500A, le stylo correspondant au zéro sera sélectionné.

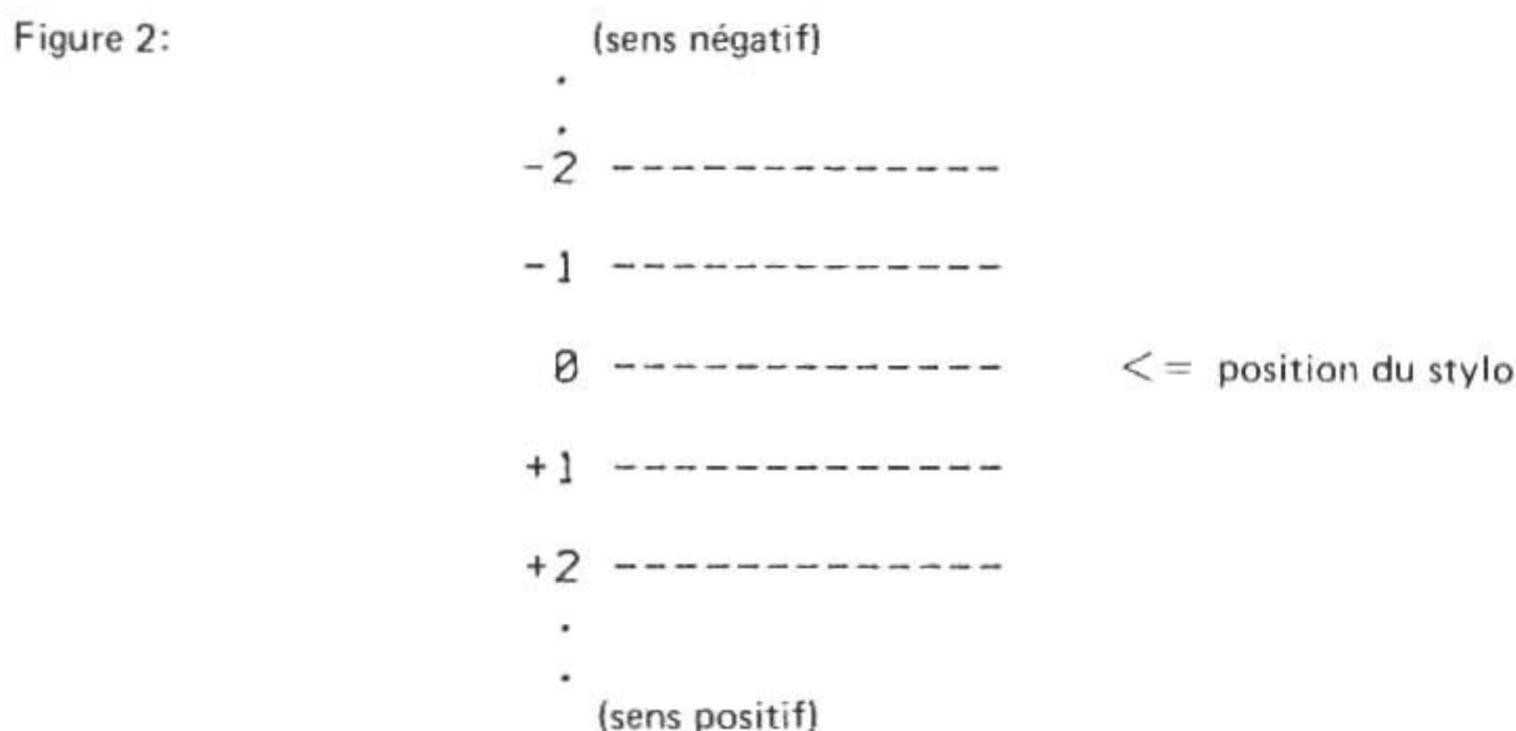
Sur le mode TEXT, l'exécution de la commande COLOR aura pour conséquence le remplacement du stylo à l'extrémité gauche du papier. Sur le mode GRAPH, le stylo retourne sur sa position antérieure.

C.6.4. La Commande LF

La commande LF sert à faire avancer ou reculer le papier dans l'imprimante. La forme de cette commande est la suivante:

LF expression
(sur le mode TEXT uniquement)

Si l'expression consiste en un nombre positif (max. 32767), le papier avance du nombre de lignes spécifié par l'expression. Si le nombre est négatif, le papier reculera du nombre de lignes spécifié par la valeur absolue de ce nombre. La figure 2 illustre l'opération:



L'étendue réelle du mouvement du papier dépend de la taille des caractères en usage lorsque la commande LF est lancée.

Quand le papier se déplace en sens inverse (c'est-à-dire qu'il est tiré en arrière) un dispositif de comptage interne l'empêche de reculer de plus de 10,24 cm.

NOTE: N'essayez jamais d'insérer du papier durant le fonctionnement du mécanisme d'alimentation papier. Vous risqueriez d'endommager l'imprimante.

C.6.5. La commande LPRINT

La commande LPRINT est la principale commande servant à afficher un texte sur l'imprimante. Elle ressemble en nature à la commande PRINT du PC-1500A et elles ont plusieurs formes en commun. Cependant, dû aux caractéristiques supplémentaires de l'imprimante, l'effet des instructions LPRINT est plus complexe. Nous concentrerons donc notre attention sur les finesses qu'implique l'utilisation de l'instruction LPRINT.

L'examen de la commande LPRINT qui suit se base sur un fonctionnement sur le mode TEXT seulement. Quoique les formes données soient possibles sur le mode GRAPH, leur opération sera différente.

L'impression d'un élément unique reste en général la même:

LPRINT élément

format dans lequel l'élément consiste en une expression, une chaîne de caractères, un nombre ou le nom d'une variable dont le contenu est à imprimer. Comme d'habitude, les caractères sont cadrés à gauche et les nombres, cadrés à droite.

De la même manière que le curseur sur l'affichage, si le stylo n'est pas placé à l'extrémité gauche du papier, l'impression commencera à partir de la position du stylo. Il est possible de changer la position du stylo à l'aide de l'instruction LCURSOR ou la proposition TAB (voyez ci-dessous).

Essayer d'imprimer un élément trop long pour une ligne à cause de la taille des caractères causera des difficultés. Si l'élément est constitué par un nombre, un message d'erreur ERROR 76 se produira. Si l'élément est constitué par une chaîne de caractères, l'impression de la chaîne continuera sur la ligne suivante.

Le souci de la taille d'un élément imprimé joue un rôle important dans l'instruction LPRINT à deux éléments, dont la forme est la suivante:

LPRINT élément1, élément2

L'utilisation de CSIZE 1 garantit l'impression sur une ligne de deux éléments numériques. Dans ce cas, les deux éléments seront cadrés de la manière ordinaire dans les deux moitiés de la ligne imprimée. Le format devient plus compliqué dans le cas d'instruction LPRINT impliquant des chaînes. Si deux éléments entrent dans la ligne, ils seront cadrés de façon ordinaire et imprimés entre deux lignes. Dans le cas de caractères de tailles plus grandes, les deux éléments sont imprimés sur deux lignes successives.

On peut aussi se servir du point-virgule dans l'instruction LPRINT. Sa fonction est à la fois d'indiquer l'espace minimum entre éléments et de grouper à la fin d'une instruction les éléments imprimés en succession sur une ligne. Dans les deux cas, si la longueur totale des éléments dépasse la capacité de la ligne imprimée, l'impression des éléments se fera sur autant de lignes successives que nécessaire. L'instruction LPRINT avec le point-virgule se présente sous la forme suivante:

LPRINT élément1 ; élément2 ; ... (etc)

ou encore la forme:

LPRINT liste d'éléments

Parmi les formes mentionnées ci-dessus, plusieurs sont employées dans le programme de démonstration qui suit:

```

10 A$ = "ABCDEFGG"
20 B = 123456
30 FOR I = 1 TO 3
40 CSIZE I
50 LPRINT A$
60 LPRINT A$, B
70 LPRINT A$; B
80 LF 5
90 NEXT I

```

Dont la sortie est:

```

ABCDEF G
ABCDEF G      123456
ABCDEF G 123456

```

```

ABCDEF G
ABCDEF G
                123456
ABCDEF G 123456

```

```

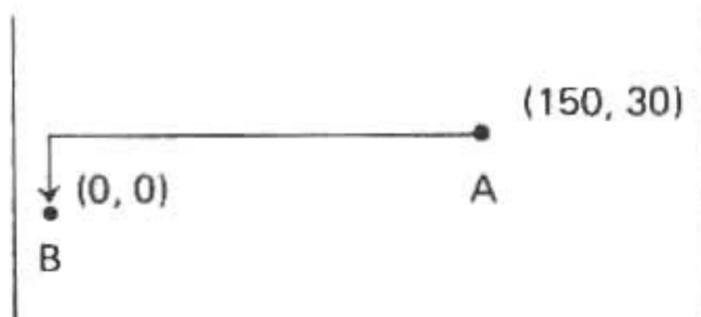
ABCDEF G
ABCDEF G
                123456
ABCDEF G 1234
56

```

Il est possible d'utiliser la commande LPRINT sans spécifier les éléments et cela sur le mode TEXT ou GRAPHIC:

LPRINT

Utilisée de cette manière, elle déclenchera un retour de chariot et (le passage à la ligne suivante). **Si cela ne se produit pas, réglez les compteurs du mode GRAPHIC.** Ainsi, dans l'exemple suivant, quoique l'on se soit servi de l'instruction LPRINT pour déplacer le stylo du point A au point B, l'imprimante se croit encore aux coordonnées (150, 30) et exécute les commandes suivantes comme s'il en était ainsi.



L'instruction LPRINT incorpore aussi une proposition USING qui fonctionne de la même manière que dans l'instruction PRINT. La proposition USING n'est permise que dans une instruction LPRINT effectuée sur le mode GRAPH.

C.6.6. L'instruction LCURSOR

L'instruction LCURSOR sert à placer le stylo sur le papier, d'une manière semblable à l'instruction CURSOR qui déplace le curseur sur l'affichage. Sa forme est la suivante:

```
LCURSOR position  
(sur le mode TEXT uniquement)
```

La position de caractère sur laquelle on peut placer le stylo dépend, bien sûr, de la taille de caractère utilisée. En général, on peut placer le stylo à un espace de moins que le maximum pour la taille de caractère en question. Reportez-vous à la commande CSIZE dans ce chapitre pour la liste des largeurs de lignes imprimées pour chaque taille de caractère.

C.6.7. L'instruction TAB

L'instruction TAB est identique à LCURSOR sauf que l'on peut l'utiliser dans une instruction LPRINT. Ce type d'instruction LPRINT se présentera sous la forme suivante:

```
LPRINT TAB position; liste d'éléments
```

Les remarques faites au sujet de l'expression de la position dans le cas de LCURSOR s'appliquent aussi à TAB. Si la liste d'éléments est vide, le résultat net de l'instruction listée ci-dessus sera un passage à la ligne suivante.

C.6.8. La commande SORGN

La commande SORGN sert à établir l'origine du système de coordonnées X-Y pour les commandes d'exécution de graphiques ultérieurs. La commande SORGN établit la position présente du stylo comme l'origine. On utilisera donc, en général, cette commande immédiatement après des instructions qui placent le stylo à un point donné du papier. La forme de la commande SORGN est simplement:

```
SORGN  
(sur le mode GRAPH seulement)
```

NOTE: L'imprimante CE-150 permet de spécifier la position du stylo à un point situé hors de la gamme de positions à partir desquelles il est possible de dessiner. Dans ce cas, le stylo se déplace aussi loin que possible puis son mouvement semble être "coupé". Si le stylo entre dans ce champ imaginaire et que l'on lance la commande SORGN, les instructions d'impression ou de dessin ultérieures resteront sans effet. Ce phénomène se perçoit comme une erreur du programme ou comme si l'interface était endommagée.

Le programme suivant établit le point d'origine à 100 unités plus haut et 100 unités plus à droite que la position actuelle du stylo. Il dessine ensuite un carré de 10 unités dont l'un des angles constitue le nouveau point d'origine:

```
10 GRAPH  
20 LINE (0, 0) - (100, 100), 9  
30 SORGN  
40 LINE (0, 0) - (10, 10), 0, 0, B  
50 TEXT  
60 END
```

C.6.9. L'instruction GLCURSOR

L'instruction GLCURSOR place le stylo à n'importe quelle coordonnée X-Y sans laisser de trace sur le papier. La forme de l'instruction GLCURSOR est la suivante:

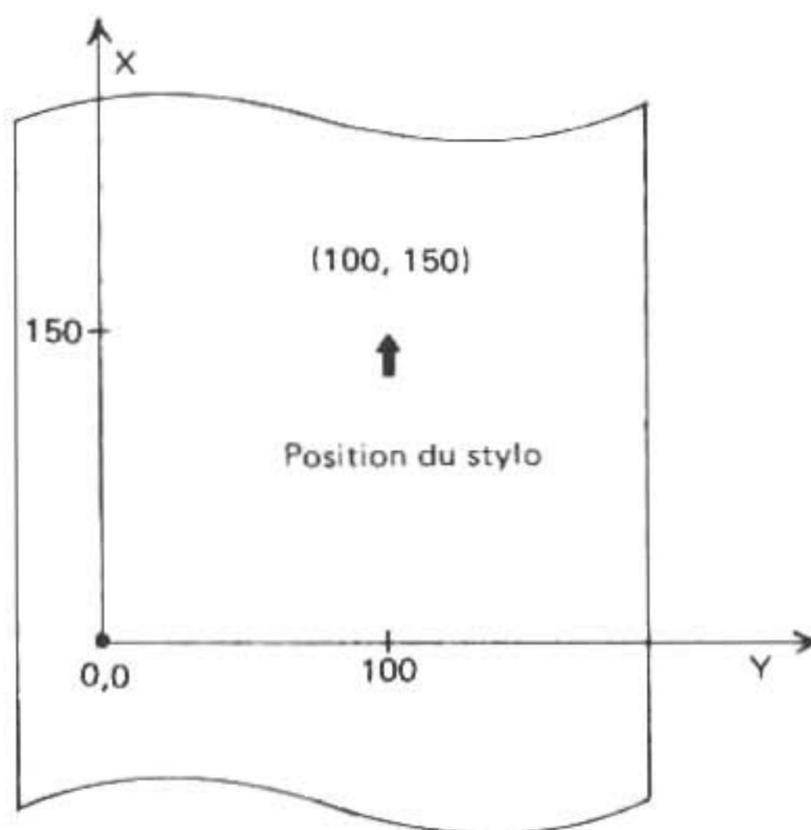
GLCURSOR (expression 1, expression 2)

Les deux expressions doivent consister en un nombre situé dans la gamme allant de -2048 à +2047. L'expression 1 représente la distance X parcourue jusqu'au point de destination et l'expression 2 la distance Y parcourue jusqu'au point de destination.

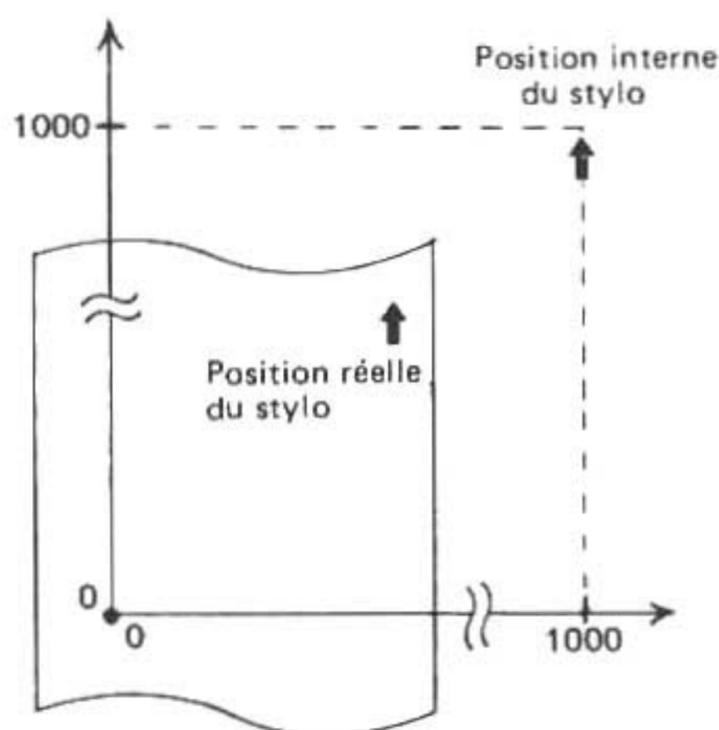
NOTE: Si le point de destination se situe hors du champ de points dans lequel le stylo peut se déplacer réellement, le stylo s'arrêtera sur le bord du papier, mais, intérieurement, le compteur qui contrôle le mouvement du stylo continue abstraitement vers son but.

Les exemples ci-dessous illustrent l'utilisation de l'instruction GLCURSOR:

Dans l'exemple à droite le stylo se déplace vers la position (100, 150).



Dans cet exemple, le stylo est incorrectement placé dans la région "imaginaire" en position (1000, 1000). En fait, le stylo se déplace vers le bord droit et fait revenir le papier. Quand elle atteint le côté droit, elle continue vers le haut jusqu'à ce qu'elle atteigne la limite de recul de 10 cm où elle s'arrête.



C.6.10. La commande LINE

La commande LINE constitue la commande principale du mode GRAPH. Elle spécifie le mouvement du stylo d'un point à l'autre. Si le stylo descend sur le papier dans son mouvement, elle trace une ligne. La commande LINE, permet aussi de tracer des lignes discontinues avec des traits de huit longueurs différentes. La première forme de la commande LINE est la suivante:

LINE (X1, Y1) - (X2, Y2), type de ligne, couleur

La valeur des expressions X1 et Y1 détermine le point de départ de la ligne. Si les valeurs des expressions X1 et Y1 sont omises, la position en cours du stylo est utilisée comme point de départ (X1, Y1). La valeur des expressions X2 et Y2, son point de destination. Les deux valeurs de X et Y doivent se situer dans la gamme allant de -2048 à +2047. Une spécification de valeur hors de cette gamme aura une erreur pour résultat.

Les paramètres de type de ligne et de couleur sont facultatifs. Si on les omet, les valeurs en effet avant la commande seront utilisées. La couleur consiste, bien sûr, en l'une des couleurs représentés par 0, 1, 2 et 3. Le type de ligne doit être représenté par une expression consistant en un nombre de la gamme allant de 0 à 9. Le tableau 2 révèle l'effet correspondant à chaque nombre donné:

Table 2:

Valeur du type de ligne	Type de ligne qui en résulte	Ø
0	Ligne continue	Ø —————
1	Traits 0,4 mm	1 ···········
2	Traits 0,6 mm	2 - - - - -
3	Traits 0,8 mm	3 - - - - -
4	Traits 1,0 mm	4 - - - - -
5	Traits 1,2 mm	5 - - - - -
6	Traits 1,4 mm	6 - - - - -
7	Traits 1,6 mm	7 - - - - -
8	Traits 1,8 mm	8 - - - - -
9	Stylo levé (pas de ligne)	9

Utilisée d'une autre façon, la commande LINE interprète les spécifications de points comme les extrémités d'une diagonale. Elle passera à l'exécution du tracé d'un carré représenté par la diagonale. Cette commande LINE se présente sous la forme suivante:

LINE (X1, X2) – (Y1, Y2), type de linge, couleur, B

La majuscule B indique que l'on commande le dessin d'un carré. Les autres paramètres sont les mêmes que pour la forme précédente.

La dernière forme de la commande LINE permet de spécifier une multiplicité de points. Après le premier, chaque point représente le point de destination du segment de la ligne suivante qui doit être tracé. On admet que la position actuelle constitue le point de départ du segment de ligne. Cette dernière forme de la commande LINE se présente sous la forme suivante:

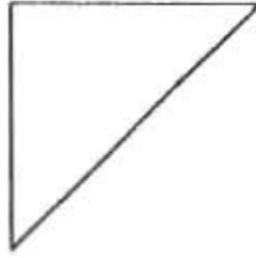
LINE (X1, Y1) – (X2, Y2) – . . . (X6, Y6), type de ligne, couleur

Les trois points servent à indiquer que l'on peut donner une série de spécifications de points, jusqu'à un maximum de six à la file. Remarquez qu'il n'est PAS permis d'utiliser le paramètre B dans cette forme de la commande. Dans l'exemple qui suit, le programme est destiné à faire tracer un triangle à l'aide de quatre segments de ligne. La ligne 15 a simplement pour fonction d'établir l'origine tandis que la ligne 20 exécute le dessin du triangle:

```

10: GRAPH
15: LINE (0, 0)-(10
    0, 0), 9: SORGN
20: LINE (0, 0)-(50
    , 50)-(-50, 50)-
    (-50, -50)-(0, 0
    ), 0, 0
30: TEXT

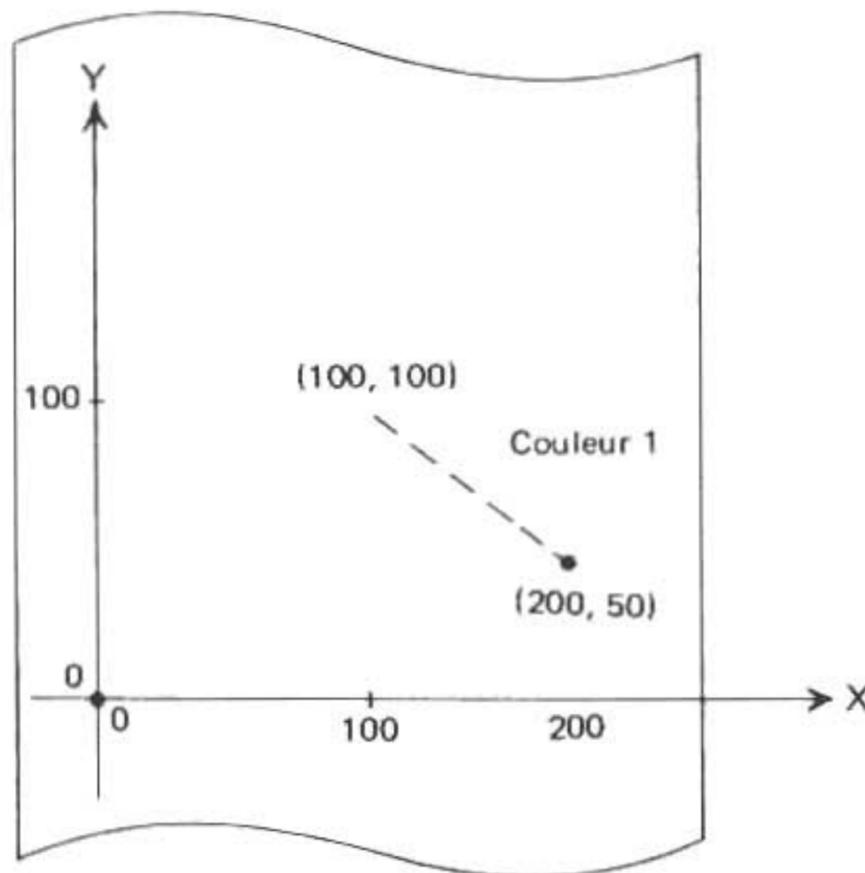
```



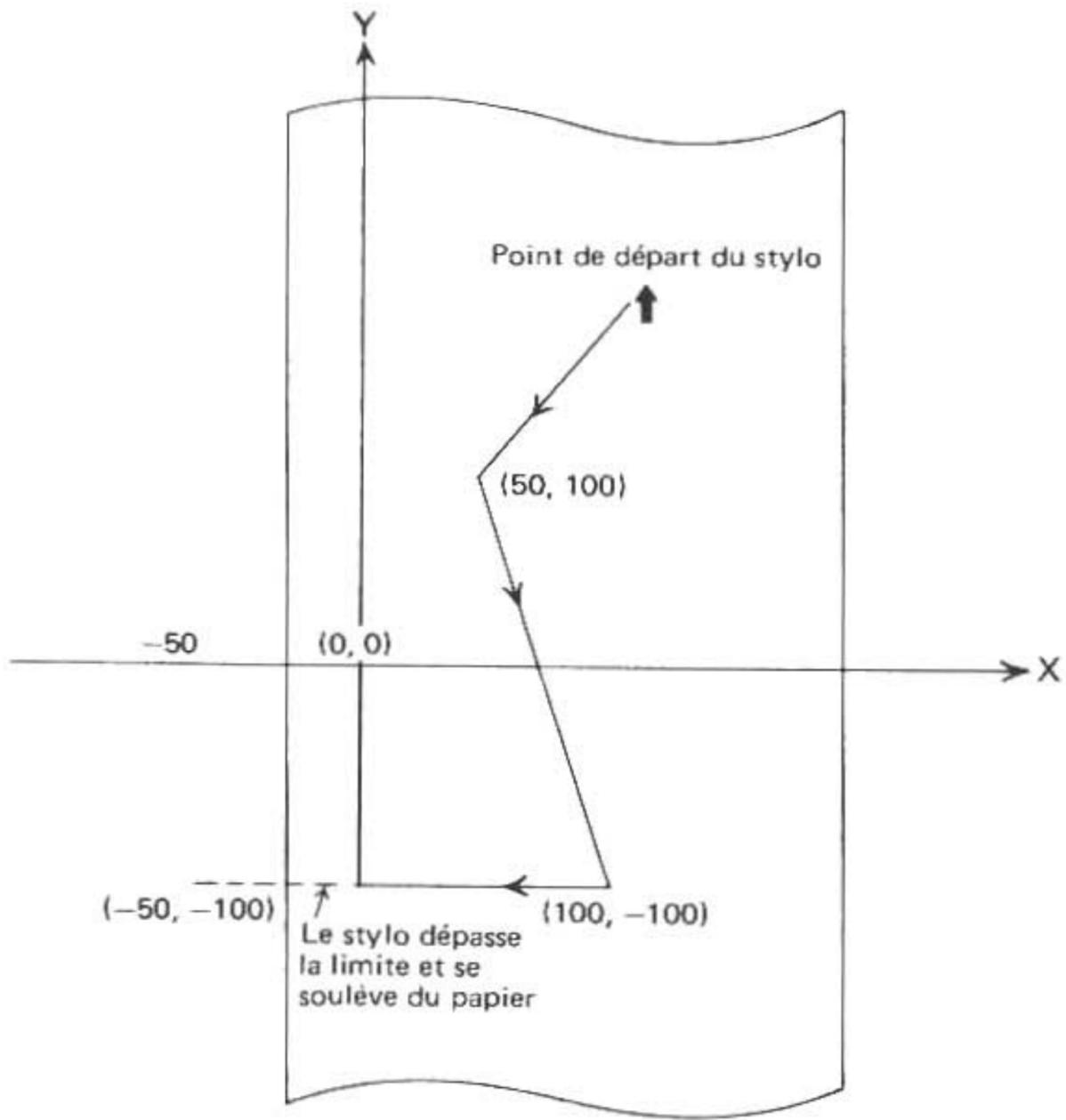
C.6.11. La commande RLINE

La commande RLINE est essentiellement la même que la commande LINE à l'exception du fait que tous les points spécifiés représentent des positions par rapport à la position actuelle du stylo plutôt que par rapport à l'origine. Les formes de commande RLINE sont les mêmes que celles de la commande LINE, à l'exception bien sûr du mot LINE à qui l'on substitue RLINE.

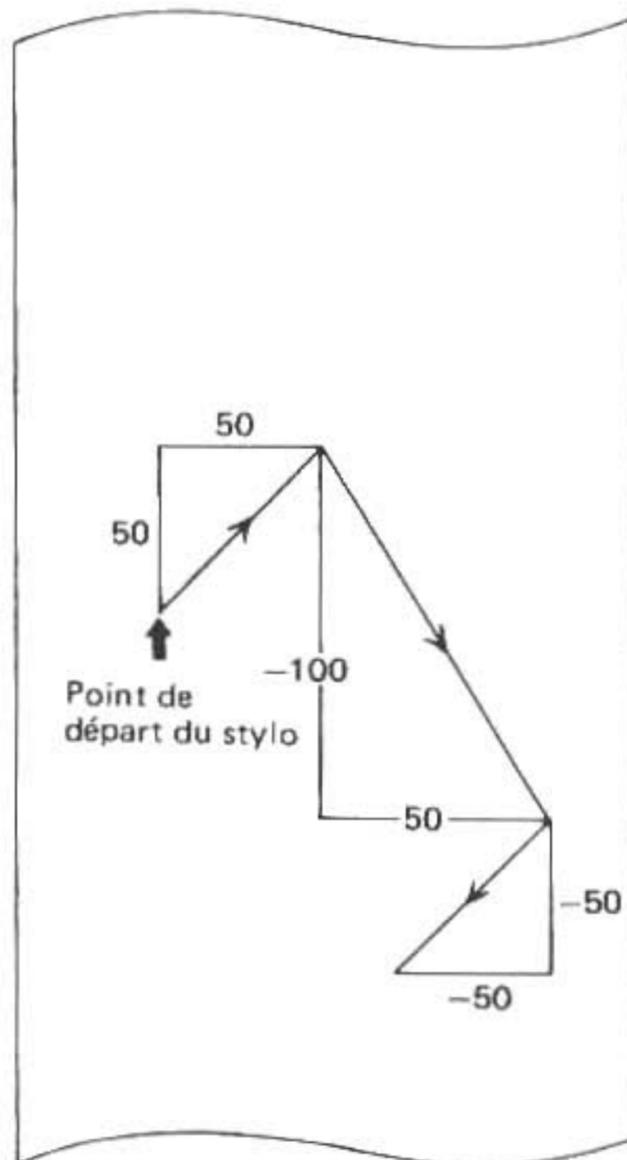
Suivent des exemples: LINE (100, 100) - (200, 50), 2, 1



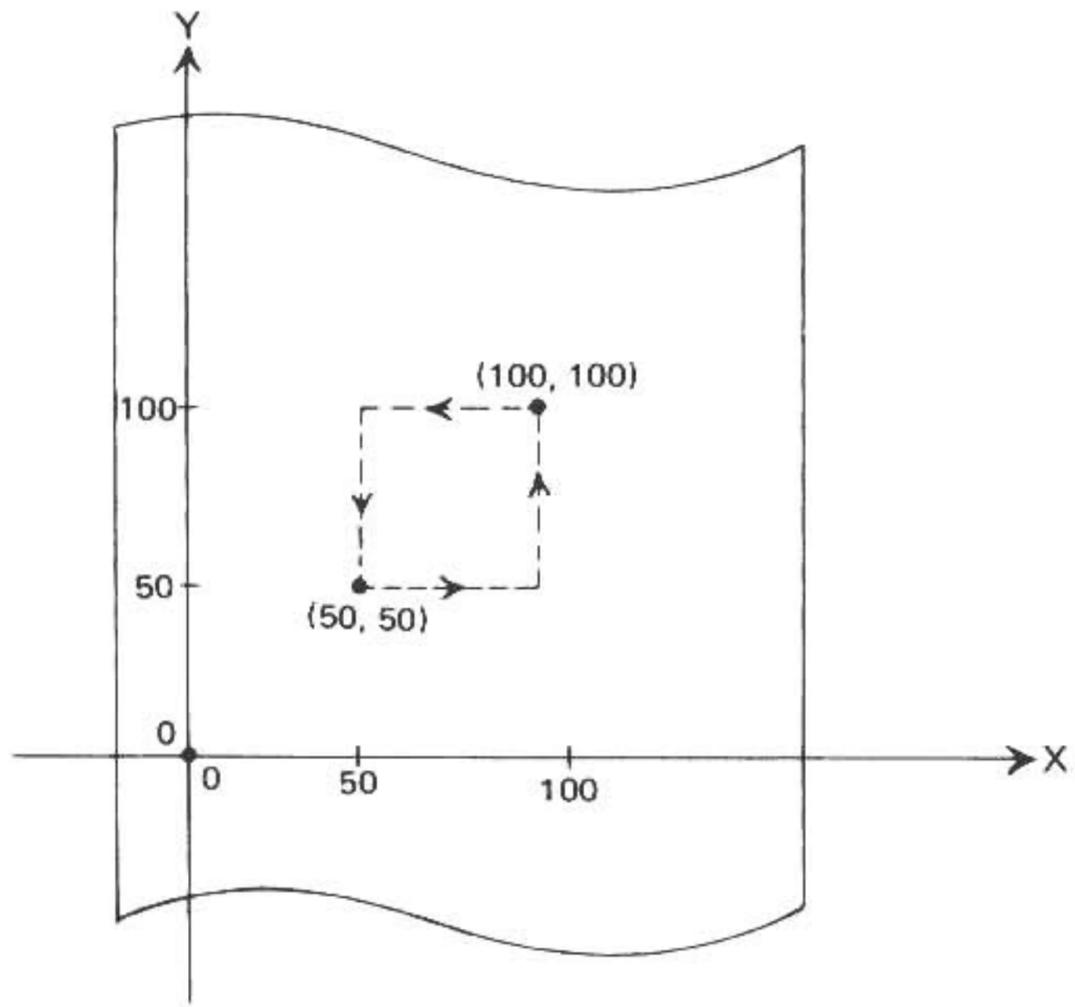
LINE - (50, 100) - (100, -100) - (-50, -100)



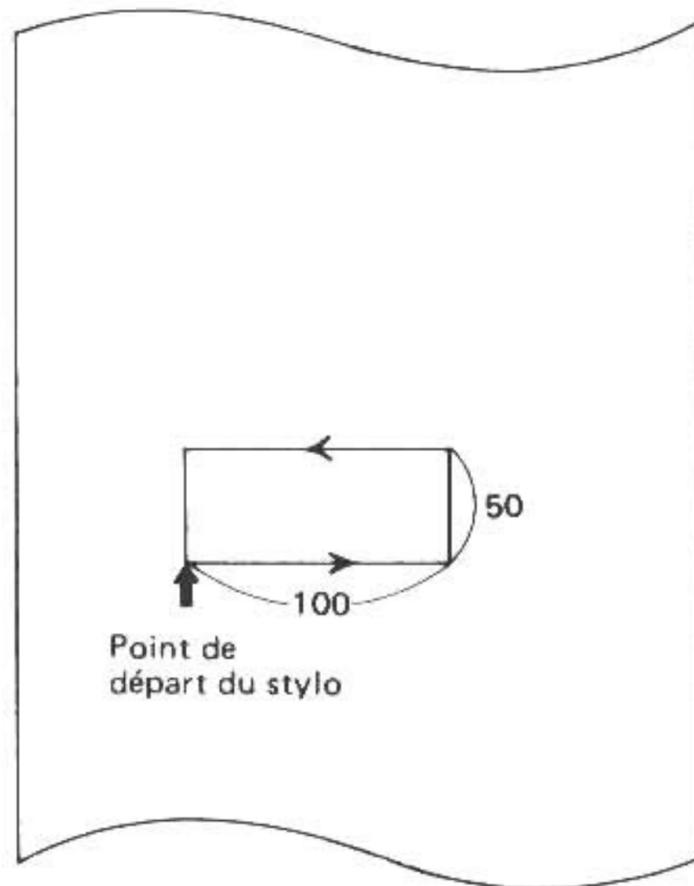
RLINE - (50, 50) - (50, -100) - (-50, -50)



LINE (50, 50) - (100, 100), 2, , B



RLINE - (100, 50) , , , B



VII. LE MODE RESERVE

A. Definition et sélection des touches de réserve

Les six touches à fonctions redéfinissable constituent une caractéristique importante du SHARP PC-1500A en ce qu'elles allègent le travail du programmeur. Ces touches de RESERVE permettent à l'utilisateur de spécifier des phrases ou des mots-clés écrits souvent, phrases et mots que l'on peut alors rappeler en appuyant sur une touche seulement. Les six touches situées dans la rangée supérieure du clavier et marquées des symboles !, ", #, \$, %, et & constituent ces touches de réserve. Il est permis d'affecter jusqu'à trois phrases ou mots-clés à chaque touche ce qui fait un total de 18 possibilités.

Le troisième mode du SHARP, le mode RESERVE est celui qui permet l'affectation de phrases aux touches. Appuyez sur les touches suivantes pour entrer le mode RESERVE:

SHIFT **MODE**

L'indicateur de mode à la partie supérieure de l'affichage indique maintenant RESERVE. Pour sortir de ce mode RESERVE, il suffit d'appuyer à nouveau sur la touche MODE.

Etant donné que chaque touche de réserve peut stocker jusqu'à trois phrases, il doit y avoir une méthode pour choisir celle que l'utilisateur souhaite rappeler. Cette méthode consiste à utiliser une touche située dans le coin inférieur gauche de l'affichage et marquée du symbole (⏏). Cette touche que l'on appelle touche de sélection de réserve fait un choix parmi les phrases mémorisées, qui correspondent actuellement aux touches de réserve. Il est important de prendre bonne note du fait que la touche de sélection de réserve effectue un changement de correspondance pour toutes les touches de réserve à la fois. Cela signifie que la touche de sélection de réserve choisit un groupe de phrases correspondant toutes à une seule touche de réserve. Les chiffres romains (I, II et III) dans la partie supérieure de l'affichage indiquent quel groupe est choisi en ce moment.

Pour affecter une phrase à une touche de réserve, passez sur le mode RESERVE d'abord (en appuyant sur **SHIFT** **MODE**), puis utilisez la touche de sélection de réserve pour choisir le groupe (I, II ou III) sous lequel vous allez stocker la phrase. Appuyez ensuite sur la touche de réserve appropriée (!, ", #, \$, %, ou &). Un affichage du type suivant apparaîtra à ce moment:

```
F 6 : _____ RESERVE •
```

(Le nombre qui vient après le F, 6 dans l'exemple ci-dessus, représente la touche de réserve sur laquelle on a appuyé.) Quand le symbole de guidage fait son apparition, vous pouvez entrer au clavier la phrase à mémoriser. Comme exercice, écrivez ce qui suit:

R **U** **N** **1** **0** **0** **ENTER**

Cette phrase est maintenant associée avec la touche de réserve que vous avez choisi.

Faisons l'essai suivant. Appuyez sur la touche **MODE** pour revenir sur le mode RUN. Appuyez sur la touche de réserve utilisée dans l'exemple précédent. L'affichage indique maintenant la commande mémorisée auparavant:

```
RUN 100_ _____ RESERVE •
```

Si vous appuyez sur la touche ENTER, l'ordinateur essaiera de passer un programme à la ligne 100. L'avantage offert par la touche de réserve consiste à ne pas avoir à écrire une commande complète chaque fois que vous désirez la lancer.

Une notation spéciale permise sur le mode RESERVE aurait pu rendre plus aisée l'opération précédente. On peut, en effet, se servir du symbole @ (à) pour représenter. La commande ENTER. Si nous avons affecté la phrase "RUN 100@" à la touche de réserve, l'exécution du programme aurait commencé dès que nous aurions appuyé sur cette touche de réserve après être revenu sur le mode RUN. Pour illustrer cela, entrons les instructions suivantes comme ligne 222:

```
222 BEEP 5,50 : END
```

Maintenant passez sur le mode RESERVE et utilisez les touches suivantes pour définir l'une des touches de réserve:

```
[G] [O] [T] [O] [2] [2] [2] [•] [ENTER]
```

(Notez qu'il est encore nécessaire d'appuyer sur [ENTER] pour définir la touche de réserve elle-même.)

Revenez sur le mode RUN et appuyez sur la touche de réserve que vous venez de définir. Vous remarquerez qu'il est superflu maintenant de vous servir de [ENTER] après avoir appuyé sur la touche de réserve.

En fait, nous aurions pu accomplir ce bruyant exemple en affectant la phrase suivante:

```
BEEP 5,50@
```

directement à une touche de réserve. Faites-en l'essai.

B. Identification des touches de réserve

Plus vous utiliserez les touches de réserve, plus il deviendra difficile de vous rappeler quelle fonction vous avez affecté à quelle touche. Pour parer à ce problème, le PC-1500A vous permet de stocker trois chaînes de caractères (une pour chaque groupe de touches de réserve) qui identifient les fonctions des touches. Ces chaînes ressemblent aux remarques du mode PROgramme.

Ces chaînes d'identifications qu'on appelle "gabarits" sont créées sur le mode RESERVE. Passez sur ce mode et choisissez le groupe de touches de réserve approprié à l'aide de la touche de sélection de réserve. Au lieu d'appuyer sur une touche de réserve ensuite, comme vous le feriez ordinairement, écrivez un gabarit et appuyez sur [ENTER]. Le gabarit sera alors mémorisé en association avec le groupe.

In guise d'exemple, nous allons prétendre que nous avons affecté aux touches de réserve de un à six (dans le group I) les noms des fonctions trigonométriques (Sinus, Cosinus, Tangente, Arc Cosinus, Arc Sinus et Arc Tangente). Pour nous rappeler quelle touche correspond à quelle fonction nous allons spécifier un gabarit. Pour effectuer cette opération, passons sur le mode RESERVE, en appuyant sur [SHIFT] [MODE], et utilisons la touche de sélection de réserve pour choisir le groupe 1 (le un romain apparaîtra sur l'affichage. Maintenant, écrivez:

```
"SIN COS TAN ACS ASN ATN "
```

Touches utilisées:

SHIFT " S I N SPACE C O S SPACE
T A N SPACE A C S SPACE A S N
SPACE A T N SPACE
SHIFT " ENTER

Vous avez maintenant défini et mémorisé le gabarit.

Revenez sur le mode RUN à l'aide de la touche **MODE**. Pour vous rappeler de la signification des touches de réserve, appuyez simplement sur la touche **RCL** (rappel) et les voilà! Appuyez une deuxième fois sur la touche **RCL** et le symbole de guidage fait sa réapparition sur l'affichage.

Il est possible de créer des gabarits pour chaque groupe de touches de réserve, gabarits, qui peuvent atteindre jusqu'à 26 caractères de longueur. Remarquez que le gabarit vous sert seulement de memento: les mots et les lettres que vous stockez dans le gabarit n'ont pas de sens pour l'ordinateur.

C. Effacement des programmes de réserve

1. Comme vous devez déjà le savoir, on remet toutes les mémoires de réserve à zéro en entrant **N E W ENTER**. Remarquez cependant que cette opération doit être effectuée sur le mode RESERVE.
2. Pour effacer une mémoire réserve, utilisez la touche **SPACE** ou bien les touches **SHIFT DEL** comme cela a été expliqué précédemment.

Exemple: Effacez A * A séquence qui se trouve sur la touche F3 (**#**) du groupe 1.

Marche à suivre	Affichage	Remarques
↓ # □	F3: A * A	Jusqu'à l'affichage du symbole 1, on répète la pression de cette touche
▶ OU ◀	F3: A * A	
SPACE SPACE SPACE	F3: —	
OU SHIFT DEL SHIFT DEL SHIFT DEL	(F3: —)	
ENTER	F3:	

VIII. DEMARRAGE DE L'EXECUTION DU PROGRAMME

A. La touche DEF

La touche DEF (définir en abrégé) permet de prendre des raccourcis qui font gagner du temps.

A.1. Passer des programmes DEFInissables

La touche **DEF** fournit la troisième méthode pour commencer un programme. Comme nous l'avons vu dans la section à propos de la commande RUN, il est possible d'étiqueter un programme avec une lettre. La touche **DEF** sert à amorcer rapidement un programme étiqueté. Cette opération consiste à appuyer d'abord sur la touche **DEF**, puis sur la touche alphabétique qui correspond à l'étiquette du programme. Les seules touches alphabétiques qu'il soit permis d'utiliser de cette manière sont les suivantes:

A, S, D, F, G, H, J, K, L, Z, X, C, V,
B, N, M, SPACE et =

A titre d'exercice, entrez les instructions suivantes pour créer trois programmes étiquetés:

Listage du programme:

```
10 "Z" : GOSUB 500
20 PRINT " Z"
30 END
140 "A" : GOSUB 500
150 PRINT " A"
160 END
270 " " : GOSUB 500
280 PRINT " B"
290 END
500 CLS : PAUSE "VOUS AVEZ ENFONCE LA ";
510 RETURN
```

Essayez maintenant, sur le mode RUN, de commencer chaque programme à l'aide de la touche **DEF**. Remarquez que si vous spécifiez une lettre pour laquelle il n'existe pas de programme étiqueté correspondant, vous obtiendrez un message d'erreur ERROR 11.

A.2. Mots-clé pré-affectés

Un petit nombre de mots-clés fréquemment utilisés ont été affectés chacun, en permanence, à une touche alphabétique située dans la seconde rangée du clavier. Pour retrouver ces mots-clés, sur n'importe quel mode, il suffit d'appuyer sur la touche DEF, puis sur l'une des touches alphabétiques en question. Par exemple, pour retrouver le mot-clé USING, écrivez:

DEF **E**

Le tableau ci-dessous indique les mots-clés disponibles et leur touche alphabétique correspondante:

<u>Touche alphabétique</u>	<u>Mot-clé</u>
Q	INPUT
W	PRINT
E	USING
R	GOTO
T	GOSUB
Y	RETURN
U	CSAVE
I	CLOAD
O	MERGE
P	LIST

On peut se servir de ces mots-clés quand l'ordinateur est connecté à l'interface imprimante/cassette. (CE-150).

Gabarit

INPUT	PRINT	USING	GOTO	GOSUB	RETURN	CSAVE	CLOAD	MERGE	LIST
<input type="checkbox"/>									
<input type="checkbox"/>									
<input type="checkbox"/>									

Deux gabarits vous sont fournis avec l'ordinateur. Utilisez-les pour identifier les fonctions affectées aux touches de définition.

A.3. L'instruction AREAD

Il est possible de donner une valeur unique aux programmes étiquetés amorcés à l'aide de la touche DEF chaque fois que l'on passe le programme sans utiliser une instruction INPUT. La lecture de la valeur est effectuée par l'instruction AREAD qui doit suivre l'étiquette du programme sur la même ligne. L'instruction AREAD se présente sous la forme suivante:

AREAD nom de variable

dans laquelle le nom de variable est un nom de variable numérique ou à caractères permis.

Pour passer une valeur à un programme qui incorpore une instruction AREAD, l'utilisateur écrit la valeur, appuie sur la touche **DEF** puis sur la touche alphabétique correspondant à l'étiquette du programme.

Comme exemple, entrez les deux programmes suivants:

Listage du programme:

```
10 "X" : AREAD TM
20 TIME = TM
30 PRINT "HEURE ACTUELLE"; TM
40 END
100 "Z" : AREAD DS
110 PRINT "LA DATE "; DS
120 END
```

Revenez sur le mode RUN et commencez le programme à l'étiquette X en écrivant au clavier le mois, le jour, l'heure et DEF X:

1 2 3 1 0 2 . 4 0 0 0 DEF X

Ce programme règle l'horloge du système à n'importe quelle heure si elle est spécifiée juste avant l'écriture de DEF (voir la fonction TIME).

Pour commencer le programme à l'étiquette Q, écrivez le jour de la semaine, puis DEF Q:

V E N D R E D I DEF Z

B. Amorçage automatique de programme

Il n'est pas seulement possible de commencer des programmes avec facilité et rapidité à l'aide de la touche DEF, mais encore de les faire démarrer automatiquement quand vous mettez le PC-1500A en marche.

L'instruction ARUN permet de déclencher cette opération. Il faut que cette instruction soit la toute première de la mémoire de programme, sinon elle sera ignorée. En outre plusieurs autres conditions doivent être remplies pour que l'instruction ARUN soit efficace. Il faut que le PC-1500A ait été mis à l'arrêt quand il était sur le mode RUN et qu'aucune erreur ne soit détectée quand on met le PC-1500A en marche.

Le programme suivant se sert de l'instruction ARUN pour saluer l'utilisateur de l'ordinateur:

Listage du programme:

```
10 ARUN
30 CLS
50 BEEP 5,50
70 PRINT "HE, SALUT!"
90 END
```

C. Comparaison des méthodes de démarrage

Quoique les différentes méthodes de démarrage de programme achèvent superficiellement le même résultat, leur opération interne diffère. Dans le but d'exploiter à notre avantage ces différences, il s'avère nécessaire d'examiner la mémorisation des données dans l'ordinateur. Ceci inclut aussi une étude des différentes préparations internes qu'effectue le PC-1500A avant le passage d'un programme.

C.1. L'aire de mémoire fixe

Quoique toutes les variables d'un même type s'utilisent de la même façon, elles ne reçoivent pas le même traitement à l'intérieur. Le PC-1500A comprend une aire de "mémoire fixe" avec un espace de stockage suffisant pour 26 variables numériques et 26 variables de chaînes de caractères (chaînes de 16 caractères). Par conséquent, les variables de A à Z et de A\$ à Z\$ sont affectées en permanence dans cette aire.

Toutes les autres variables, y compris les variables à noms à deux caractères sont affectés dans l'aire de mémoire principale de l'ordinateur. Elles partagent cette aire de mémoire principale avec les directives de programme, quoique les variables soient affectées à l'extrémité opposée aux directives dans la mémoire. Du fait que les directives et les données occupent la même aire, il arrive qu'elles l'encombrent totalement. Dans ce cas un erreur située entre ERROR 177 et ERROR 181 se produit.

Il est important de réaliser que les deux aires de mémoire ne reçoivent pas le même traitement au démarrage d'un programme, comme l'explique le tableau dans la section suivante. Fondamentalement, les variables de la mémoire fixe ne peuvent être effacées que par une instruction CL explicite. Celles stockées dans la mémoire principale sont effacées chaque fois que l'on commence un programme à l'aide de la commande RUN.

Une autre particularité de la mémoire fixe consiste en la possibilité de redéfinir les données de cette aire comme des tableaux dont le nom est le symbole @ (à) pour les variables numériques et @\$ pour les variables à chaîne. La désignation@(1) représente donc le même emplacement de stockage que la variable A et @(26) le même emplacement de stockage que la variable Z. La désignation @\$ (5) s'applique au même emplacement que E\$ et la désignation @\$ (20) au même emplacement que T\$. Pour des raisons évidentes, des indices supérieurs à 26 ne sont pas permis. Remarquez qu'il n'est pas nécessaire de donner une dimension à @ et @\$ avant de les utiliser.

C.2. Tableau de comparaison des méthodes de démarrage de programme

	<u>RUN</u>	<u>GOTO</u>	<u>DEF</u>
Remise à zéro de l'affichage.	Oui	Oui	Non
Le curseur revient à la 1ère colonne	O	N	N
L'intervalle WAIT est réglé à l'infini.	N	N	N
Le mode trace est altéré.	N	N	N
La mémoire fixe est remise à zéro.	N	N	N
La mémoire principale est remise à zéro.	O	N	N
FOR-NEXT, GOSUB La pile intérieure est remise à zéro.	O	O	O
ON ERROR GOTO est annulé.	O	N	N
Le compteur DATA pour l'opération de READ est rétabli.	O	N	N
Le format USING est annulé.	O	N	N

Notes d'utilisation concernant le CE-159

CE-159: 8K octets RAM en option avec batterie de secours

En utilisant le CE-159,

- 1) Changer la formule pour vérifier la capacité utilisable en tant que variables dans le Manuel d'utilisation (page 153) du CE-159 comme suit:

MEM - (2048 + 16) **ENTER**



MEM - (6144 + 23) **ENTER**

- 2) Changer le programme de rétention de données dans le Manuel d'utilisation (page 154) du CE-159 comme suit:

[Programme de rétention de données]

```

10 STOP
-----
20 "A" : POKE 30873, PEEK 16375,
   PEEK 16376
30 END
-----
40 "B" : POKE 30873, 72, 0
50 DIM ZY$(127), ZZ$(0) * 2
60 END
-----
70 "C" : POKE 16375, PEEK 30873,

```



[Programme de rétention de données]

```

10 STOP
-----
20 "A" : POKE 30873, PEEK 16375,
   PEEK 16376
30 END
-----
40 "B" : POKE 30873, 88, 0
50 DIM ZX$(255), ZY$(127), ZZ$(0) * 2
60 END
-----
70 "C" : POKE 16375, PEEK 30873,

```

* Différences entre PC-1500A et PC-1500

Puisque le PC-1500A dispose d'une MEV (RAM) plus étendue que le PC-1500, certaines descriptions du circuit de sélection à microplaquettes, du plan d'occupation mémoire, et du connecteur à 40 broches (pour modules de mémoire) figurant dans le "SHARP POCKET COMPUTER PC-1500 TECHNICAL REFERENCE MANUAL" (manuel de référence technique de l'ordinateur de poche SHARP PC-1500) nécessitent des modifications pour convenir au PC-1500A. Un sommaire des différences entre le PC-1500A et le PC-1500 est fourni plus loin. Prendre bonne note de ces différences en utilisant le langage machine.

1. Chip select signal (Circuit de sélection à microplaquettes)

Les méthodes de réception de signaux (S0 – S5) sont différentes. (Page 91)

TC40H138F Output	Memory address	PC-1500	PC-1500A
$\overline{Y0}$	4000H 47FFH	S0 STANDARD USER MEMORY	S0
$\overline{Y1}$	4800H 4FFFH	S1	STANDARD USER MEMORY
$\overline{Y2}$	5000H 57FFH	S2	
$\overline{Y3}$	5800H 5FFFH	S3 OPTION USER MEMORY	
$\overline{Y4}$	6000H 67FFH	S4	S1 S2 OPTION USER MEMORY
$\overline{Y5}$	6800H 6FFFH	S5	S3
$\overline{Y6}$	7000H 77FFH	INHIBITED	INHIBITED
$\overline{Y7}$	7800H	7600H STANDARD USER AND SYSTEM MEMORY 7BFFH	7600H STANDARD USER AND SYSTEM MEMORY 7C00H
	7FFFH	INHIBITED	MACHINE LANGUAGE MEMORY

2. Memory map (Plan d'occupation mémoire)

Si un module de mémoire, le CE-151, ou CE-155 est installé dans le PC-1500A, la zone de mémoire utilisateur est accrue d'autant. (Page 93 ~ 94)

When the CE-151 is used.

3800H	
4000H	STANDARD USER MEMORY (RAM) 6KB
5800H	CE-151 (RAM) 4KB
6800H	NOT USED
7000H	
7600H	STANDARD USERS SYSTEM MEMORY
7C00H	MACHINE LANGUAGE MEMORY
8000H	← ME 0 →

When the CE-155 is used.

3800H	CE-155 (RAM) 2KB	} 8KB IN TOTAL
4000H	STANDARD USER MEMORY (RAM) 6KB	
5800H	CE-155 (RAM) 6KB	} 8KB IN TOTAL
6800H		
7000H		
7600H	STANDARD USERS SYSTEM MEMORY	
7C00H	MACHINE LANGUAGE MEMORY	
8000H	← ME 0 →	

3. 40 pin connector (Connecteur à 40 broches)

Les noms de signaux et les descriptions de la broche 5 aussi bien que des broches 16 ~ 18 diffèrent sur le connecteur à 40 broches utilisé pour le montage du module de mémoire. (Page 100)

Pin no.	Signal name	Description
5	S1	Address designation of 5800H ~ 5FFFH
16	S2	Address designation of 6000H ~ 67FFH
17	S3	Address designation of 6800H ~ 6FFFH
18	S4	Not connected

APPENDICES

A. TABLE DES ABREVIATIONS

Commandes Imprimante

COLOR	COL. COLO.	LPRINT	LP. LPR. LPRI. LPRIN.
CSIZE	CSI. CSIZ.	RLINE	RL. RLI. RLIN.
GLCURSOR	GL. GLC. GLCU. GLCUR. GLCURS. GLCURSO.	ROTATE	RO. ROT. ROTA. ROTAT.
GRAPH	GRAP.	SORGN	SO. SOR. SORG.
LCURSOR	LCU. LCUR. LCURS. LCURSO.	TAB	--
LF	--	TEST	TE. TES.
LINE	LIN.	TEXT	TEX.
LLIST	LL. LLI. LLIS.		

Commandes Cassette

CHAIN	CHA. CHAI.	INPUT #	I. # IN. # INP. # INPU. #
CLOAD	CLO. CLOA.	MERGE	MER. MERG.
CLOAD?	CLO.? CLOA.?	PRINT #	P. # PR. # PRI. # PRIN. #
CSAVE	CS. CSA. CSAV.	RMT OFF RMT ON	RM. OF. RM. O.

Instructions

AREAD	A. AR. ARE. AREA		
ARUN	ARU.	GOSUB	GOS. GOSU.
BEEP	B. BE. BEE.	GOTO	G. GO. GOT.
CLEAR	CL. CLE. CLEA.	GPRINT	GP. GPR. GPRI. GPRIN.
CLS	--	GRAD	GR. GRA.
CURSOR	CU. CUR CURS. CURSO.	IF	--
DATA	DA. DAT.	INPUT	I. IN. INP. INPU.
DEGREE	DE. DEG. DEGR. DEGRE.	LET	LE.
DIM	D. DI.	LOCK	LOC.
END	E. EN.	NEXT	N. NE. NEX.
ERROR	ER. ERR. ERRO.	ON	O.
FOR	F. FO.		
GCURSOR	GCU. GCUR. GCURS. GCURSO.		

PAUSE	PA. PAU. PAUS.	STEP	STE.
PRINT	P. PR. PRI PRIN.	STOP	S ST. STO.
RADIAN	RAD. RADI. RADIA.	THEN	T. TH. THE.
RANDOM	RA. RAN. RAND. RANDO.	TRON	TR. TRO.
READ	REA.	TROFF	TROF.
REM	--	UNLOCK	UN. UNL. UNLO. UNLOC.
RESTORE	RES. REST. RESTO. RESTOR.	USING	U. US. USI. USIN.
RETURN	RE. RET. RETU. RETUR.	WAIT	W. WA. WAI.

Commandes

CONT	C CO. CON.	NEW	--
LIST	L. LI. LIS.	RUN	R. RU.

Fonctions

ABS	AB.	MEM	M. ME.
ACS	AC.	MID\$	MI. MID.
AND	AN.	NOT	NO.
ASC	--	OR	--
ASN	AS.		
ATN	AT.	PI	--
CHR\$	CH. CHR.	POINT	POI. POIN.
COS	--	RIGHT\$	RI. RIG. RIGH. RIGHT.
DEG	--	RND	RN.
DMS	DM.	SGN	SG.
EXP	EX.	SIN	SI.
INKEY\$	INK. INKE. INKEY.	SQR	SQ.
INT	--	STATUS	STA. STAT. STATU.
LEFT\$	LEF. LEFT.	STR\$	STR.
LEN	--	TAN	TA.
LOG	LO.	TIME	TI. TIM.
LN	--	VAL	V. VA.

B. REMPLACEMENT DES PILES DU PC-1500A

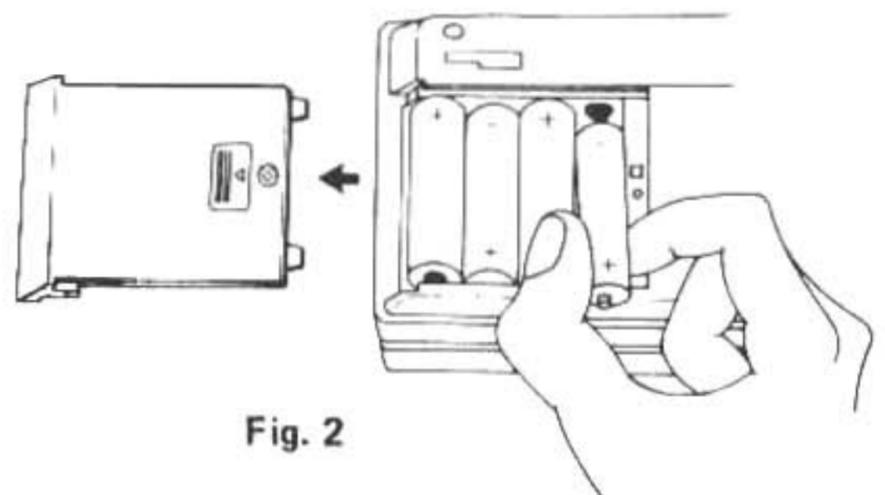
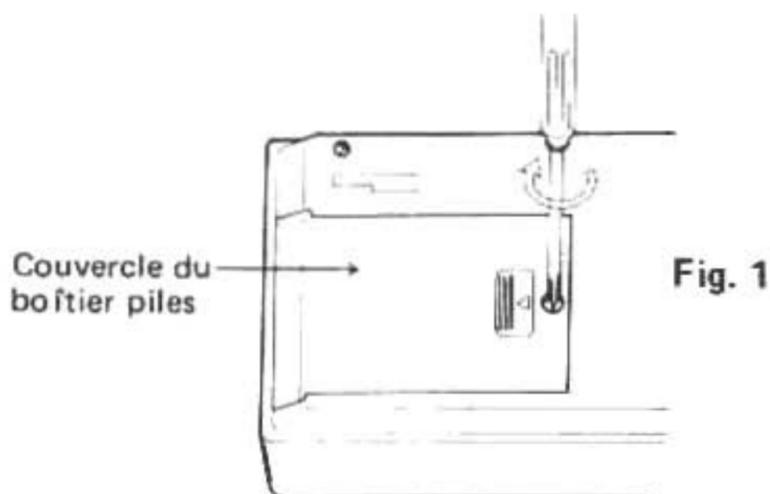
Vous vous éviterez bien des problèmes si vous prenez les précautions suivantes lors du remplacement des piles:

- * Remplacez toujours les 4 piles en même temps.
- * Ne mélangez pas des piles neuves avec des piles usagées.
- * Utilisez exclusivement 4 piles sèches de type AA, R6 ou SUM-3 de 1,5V.

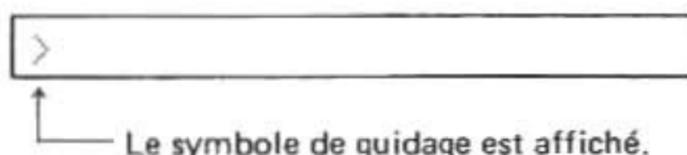
METHODE DE REMPLACEMENT DES PILES

Il est nécessaire de placer des piles dans l'ordinateur dès sa réception et, par la suite, de les remplacer chaque fois que l'indicateur de charge disparaît de l'affichage. Veuillez procéder de la manière suivante:

1. Appuyez sur la touche **OFF** pour mettre l'ordinateur hors tension.
2. Dévissez le couvercle du boîtier piles en vous servant d'une pièce de monnaie ou d'un petit tournevis (fig. 1).
3. Remplacez les quatre piles (fig. 2).



4. Appuyez légèrement sur le couvercle lorsque vous remettez la vis en place.
5. Pour poursuivre, appuyez sur les touches **ON** et **CL**, écrivez **NEW 0** et appuyez sur la touche **ENTER**. Puis, appuyez sur les touches **SHIFT** **MODE** **NEW** **ENTER**.
6. Vérifiez l'affichage qui s'ensuit.



Si l'affichage est vide ou un symbole autre que ">" apparaît, déposez à nouveau les piles, puis remettez-les en place et vérifiez l'affichage.

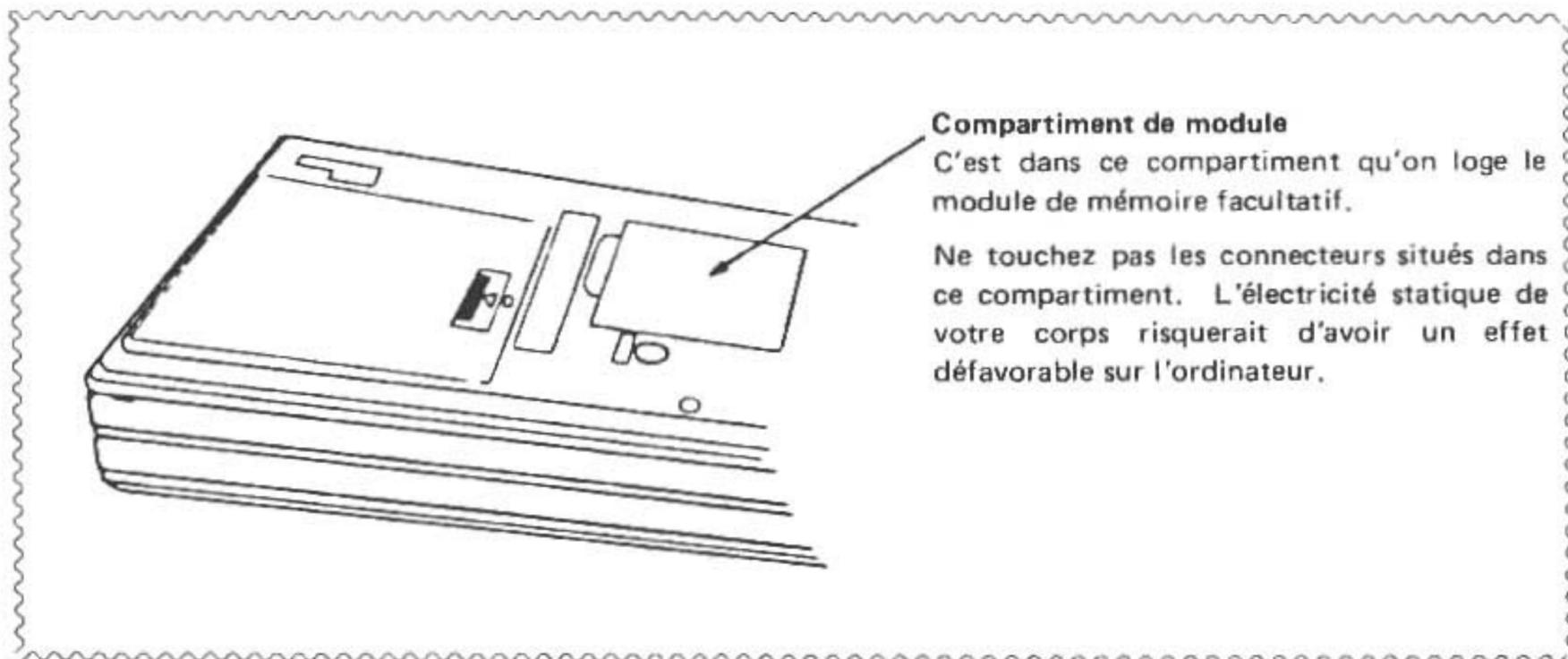
Note: * Otez les piles si vous savez que vous n'allez pas utiliser l'ordinateur pendant longtemps, afin d'éviter des dégâts provoqués par des fuites de liquide des piles.

* De même, retirez promptement une pile morte, à cause de la possibilité d'une fuite de liquide de cette pile, fuite qui risquerait d'endommager l'ordinateur.

* Il n'est pas possible d'utiliser les piles rechargeables sur le PC-1500A.

* L'adaptateur c.a. EA-150 du CE-150 peut aussi être utilisé séparément du CE-150.

(Ne connectez pas le EA-150 sur l'ordinateur PC-1500A quand celui-ci est monté sur l'interface imprimante/cassette CE-150).



C. TABLE DES CODES DE CARACTERES ASCII

Positions supérieures →
de bits b7, b6, b5

000 001 010 011 100 101 110 111

Positions inférieures
de bits
b4, b3, b2, b1



0000

0001

0010

0011

0100

0101

0110

0111

1000

1001

1010

1011

1100

1101

1110

1111

Hexa decimal	0	1	2	3	4	5	6	7
0			SPACE	0	@	P		p
1			!	1	A	Q	a	q
2			"	2	B	R	b	r
3			#	3	C	S	c	s
4			\$	4	D	T	d	t
5			%	5	E	U	e	u
6			&	6	F	V	f	v
7			* []	7	G	W	g	w
8			(8	H	X	h	x
9)	9	I	Y	i	y
A			*	:	J	Z	j	z
B			+	;	K	√	k	* }
C			,	<	L	✳	l	* ;
D			-	=	M	π	m	* }
E			.	>	N	^	n	~
F			/	?	O	* _	o	* ■

*: Les caractères étoilés ne peuvent apparaître que sur l'afficheur, l'imprimante ignore ceux-ci.

PC-1500A LISTE DE CODE DES ERREURS

Code des Erreurs

Explication

1. Erreur de syntaxe: Une erreur résultant d'une faute de frappe comme:
d'une donnée manquante:
10: GOTO
de commandes invalides:
10: 5A = 1 or
10: NEW
(parce que NEW ne peut être une instruction)

< Affichage > **ERROR 1 IN 10**
Supplément: Voir page 171.
2. Cette erreur se produit quand il n'y a pas de commande FOR correspondant à une commande NEXT.

Ex. 10: FOR A = 1 TO 10
 :
 :
 100: NEXT B

< Affichage > **ERROR 2 IN 100**
4. Cette erreur se produit quand il n'y a pas de DATA correspondant à une commande READ.

Ex. 10: READ X, Y
 20: DATA 10
 30: END

< Affichage > **ERROR 4 IN 10**
5. Ce message est lancé quand une variable de tableau est déclarée sans le nom d'une variable en existence.

Ex. 10: DIM A (10, 10)
 20: DIM A (5)

< Affichage > **ERROR 5 IN 20**
6. Ce message est lancé quand une variable de tableau a été utilisée sans instruction DIM (dimension).

Ex. 10: CLEAR
 20: A (3) = 1

< Affichage > **ERROR 6 IN 20**

PC-1500A LISTE DE CODE DES ERREURS

Code des Erreurs

Explication

7. Ce message est lancé quand le nom d'une variable est inapproprié.
 Ex. 10: A\$ = 10 or
 10: FOR A\$ = 1 TO 10
 < Affichage > **ERROR 7 IN 10**
8. Ce message est lancé quand la dimension dépasse 3 dans la déclaration d'une variable de tableau.
 Ex. 10: DIM A (3, 4, 5, 6)
 < Affichage > **ERROR 8 IN 10**
9. Ce message est lancé quand l'indice d'une variable de tableau est plus grand que la taille du tableau énoncée dans la commande DIM.
 Ex. 10: DIM A (3)
 20: A (4) = 1
 < Affichage > **ERROR 9 IN 20**
10. Ce message est lancé quand il n'y a pas suffisamment de mémoire disponible pour créer plus de variables.
 Ex.

<u>Touches utilisées</u>	<u>Affichage</u>
MEM <input type="button" value="ENTER"/>	7
AB = 10 <input type="button" value="ENTER"/>	ERROR 10
11. Ce message est lancé quand la ligne spécifiée ne se trouve pas dans le programme.
 Ex. 10: INPUT "X ="; X: GOTO 5
 < Affichage > **ERROR 11 IN 10**
12. Ce message est lancé quand la commande USING spécifie incorrectement le format.
 Ex. 100: PRINT USING "###A#"; 10
 < Affichage > **ERROR 12 IN 100**
13. Ce message est lancé quand un programme dépasse la capacité de la mémoire de programme ou quand la spécification de la touche RESERVE dépasse la capacité de l'aire de réserve.
 Ex.

<u>Touches utilisées</u>	<u>Affichage</u>
MEM <input type="button" value="ENTER"/>	7
15 A = A + 1 <input type="button" value="ENTER"/>	ERROR 13

PC-1500A LISTE DE CODE DES ERREURS

Code des Erreurs

Explication

14. (1) Les instructions FOR ont été logées trop profondément et la capacité de l'aire d'empilage a été outrepassée.
(2) L'espace-tampon a été outrepassé durant l'analyse d'une expression.
15. (1) Les instructions GOSUB sont logées trop profondément et la capacité de l'aire a été outrepassée.
(2) Durant l'analyse d'une expression, la taille du tampon de chaîne a été dépassé par les chaînes de caractères manipulées.
16. (1) La valeur spécifiée est supérieure à 10 E100 ou inférieure à -10 E100.
Ex. 123E 99
(2) La valeur établie par hexadécimaux dépasse 65535.
Ex. &1FFAB
17. Le type de données (nombres, chaînes de caractères) n'est pas approprié à l'expression du calcul.
Ex. 1 + "A"
18. Le nombre des arguments n'est pas approprié à l'expression.
Ex. LEFT\$ ("ABC")
SIN (30, 60)
19. La valeur numérique spécifiée est hors gamme permise.
Ex. 10: DIM A (256)
< Affichage > **ERROR 19 IN 10**
20. Il n'y a pas de '(' suivante '@' ou '@\$' quand les variables de tableau de mémoire fixe sont spécifiées.
Ex. 100: @\$ = "A"
< Affichage > **ERROR 20 IN 100**
21. Une variable est requise dans l'expression.
Ex. 10: FOR 1 = 0 TO 10
< Affichage > **ERROR 21 IN 10**
22. Ce message est lancé quand on charge le programme et qu'il n'y a pas d'espace de mémoire disponible pour le chargement.

PC-1500A LISTE DE CODE DES ERREURS

<u>Code des Erreurs</u>	<u>Explication</u>				
23.	Ce message est lancé quand l'heure est incorrectement réglée. Ex. TIME = 131005.10 <input type="button" value="ENTER"/>				
26.	Ce message est lancé quand la commande ne peut être exécutée sur le mode actuel. Ex. < RUN MODE > NEW <input type="button" value="ENTER"/>				
27.	Ce message est lancé quand il n'y a pas de programme correspondant à l'étiquette donnée. Ex. <table border="0" style="width: 100%;"> <tr> <td style="text-align: center;"><u>Touches utilisées</u></td> <td style="text-align: center;"><u>Affichage</u></td> </tr> <tr> <td style="text-align: center;"><input type="button" value="DEF"/> <input type="button" value="ENTER"/></td> <td style="text-align: center;">ERROR 27</td> </tr> </table>	<u>Touches utilisées</u>	<u>Affichage</u>	<input type="button" value="DEF"/> <input type="button" value="ENTER"/>	ERROR 27
<u>Touches utilisées</u>	<u>Affichage</u>				
<input type="button" value="DEF"/> <input type="button" value="ENTER"/>	ERROR 27				
28.	Ce message est lancé quand une commande ou un indicatif de fonction a été inséré dans " ", ou quand on essaie de substituer des commandes INPUT ou AREAD pour des variables à caractères. Ex. 10 INPUT A\$ <table border="0" style="width: 100%;"> <tr> <td style="text-align: center;"><u>Touches utilisées</u></td> <td style="text-align: center;"><u>Affichage</u></td> </tr> <tr> <td style="text-align: center;"><input type="button" value="DEF"/> W <input type="button" value="ENTER"/></td> <td style="text-align: center;">ERROR 28</td> </tr> </table>	<u>Touches utilisées</u>	<u>Affichage</u>	<input type="button" value="DEF"/> W <input type="button" value="ENTER"/>	ERROR 28
<u>Touches utilisées</u>	<u>Affichage</u>				
<input type="button" value="DEF"/> W <input type="button" value="ENTER"/>	ERROR 28				
30.	Ce message est lancé quand un numéro de ligne dépasse 65535. Ex. 102235 A = 10 <input type="button" value="ENTER"/>				
32.	Ce message est lancé quand le curseur graphique se trouve entre les colonnes 152 et 153 durant l'exécution de commandes d'entrée (l'indicatif d'entrée ne peut être affiché). Ex. 100: GCURSOR 152 110: INPUT X < Affichage > ERROR 32 IN 110				
131	Immédiatement avant la chaîne de caractères et/ou la variable à caractères se trouve un "+", ou un "-", que l'on considérera en tant que signe.				
177 ~ 181	Durant la fabrication du programme, le programme a débordé dans l'aire des données. Il s'est produit un chevauchement des deux aires.				
0, 224 ~ 241	Des données d'entrée incorrectes ont été introduites durant l'exécution de commandes INPUT, READ ou AREAD. Ex. 10: INPUT A <table border="0" style="width: 100%;"> <tr> <td style="text-align: center;"><u>Touches utilisées</u></td> <td style="text-align: center;"><u>Affichage</u></td> </tr> <tr> <td style="text-align: center;">123 PRINT <input type="button" value="ENTER"/></td> <td style="text-align: center;">ERROR 240</td> </tr> </table>	<u>Touches utilisées</u>	<u>Affichage</u>	123 PRINT <input type="button" value="ENTER"/>	ERROR 240
<u>Touches utilisées</u>	<u>Affichage</u>				
123 PRINT <input type="button" value="ENTER"/>	ERROR 240				

PC-1500A LISTE DE CODE DES ERREURS

<u>Code des Erreurs</u>	<u>Explication</u>
71.	(1) Le papier est retourné en arrière de plus de 10,24 cm. (2) Le papier retournera en arrière de plus de 10,24 cm lors de l'exécution de commandes à venir (sur le mode TEXT).
72.	La valeur donnée n'est pas appropriée à la valeur de TAB ou de LCURSOR.
73.	Une commande a été lancée sur le mauvais mode d'impression (GRAPH/TEXT).
74.	Le nombre de virgules (,) dans une commande LINE ou RLINE est trop grand. NOTE: Une entrée de plus de sept virgules produit une erreur. De même, si l'on omet la première virgule, plus de six virgules produiront une erreur.
76.	Concerne LPRINT: l'impression des résultats d'un calcul ne peut pas se faire sur une ligne (sur le mode TEXT).
78.	(1) On est en train de changer les stylos. (2) On n'a pas répondu à l'avertissement "LOW BATTERY" pile affaiblie. (Voir Note 1) Ce message est lancé quand pour l'une de ces deux raisons, des commandes visant à faire se déplacer le stylo (comme LPRINT et LINE) ne peuvent être exécutées.
79.	L'indicateur de couleur ne s'est pas allumé (Voir Note 2)
80.	Piles affaiblies. (Voir NOTE 3)

NOTES:

- (1) Si le message ERROR 78 est dû à un état affaibli des piles, mettez l'imprimante à l'arrêt (OFF). Après l'avoir rechargé, mettez le CE-150 en marche (ON). Vous pouvez continuer à présent.
- (2) L'indicateur de couleur indique COLORO (couleur 0) et ne s'allume que lorsque le stylo revient sur le côté gauche. Sur cette position, il est possible d'identifier le numéro du stylo utilisé à ce moment.
- (3) Après la recharge, appuyez immédiatement sur le commutateur ON du CE-150 et continuez les opérations en cours.

F. LECTURES RECOMMANDÉES

Plusieurs éditeurs proposent des ouvrages dont la lecture pourra vous aider à aller plus loin dans le monde de l'informatique:

- SYBEX 18, rue Planchat 75020 – PARIS
- PSI: 41/51, rue Jacquard – B.P 86
77400 – LAGNY – SUR – MARNE
- MC GRAW-HILL: 28, rue Beaunier 75014 – PARIS

O. ORDRE D'ÉVALUATION D'EXPRESSIONS

Les calculs sont effectués en accordance avec la hiérarchie suivante: expressions entre parenthèses ayant la plus haute priorité et opérations logiques ayant la moindre. Si deux ou plus de deux expressions de la même priorité se trouvent dans la même expression ou sous-expression, leur évaluation s'accomplit de la gauche à la droite.

- 1) Expressions entre parenthèses
- 2) Récupération des valeurs des variables, TIME, PI, MEM, INKEYS
- 3) Fonction (SIN, COS, LOG, EXP, etc.)
- 4) Calcul de puissances (Exemple: $2A^3 = 2 * (A^3)$)
- 5) Signe Arithmétique (+, -)
- 6) Multiplication, Division (*, /)
- 7) Addition, Soustraction (+, -)
- 8) Opérateurs de comparaison (<, <=, =, >=, >, <>)
- 9) Opérateurs logiques (AND, OR, NOT)

NOTES:

– Quand on utilise un signe arithmétique et les puissances dans la même expression, la puissance est évaluée avant le signe.

Exemple: 5^4 est évalué à -625 au lieu de 625

– Les calculs entre parenthèses sont évalués d'abord. Dans les expressions à plusieurs "couches" de parenthèses, le calcul commence avec la paire de parenthèses logée le plus au centre et avance de là vers les parenthèses situées le plus à l'extérieur.

X. DIFFERENCES ENTRE LES COMMANDES DU PC-1211 ET CELLES DU PC-1500A

X.1 Directives communes au PC-1211 et au PC-1500A

1. Fonctions

ABS
ACS
ASN
ATN
COS
DEG
DMS
EXP
INT
LOG
LN
 π (PI)
SGN
SIN
 $\sqrt{\quad}$ (Racine carrée)
TAN
 \wedge (Calcul de puissance)

3. Commandes

CONT
LIST
NEW
RUN

4. Commandes cassette

CHAIN
CLOAD
CLOAD?
CSAVE
INPUT #
PRINT #

2. Instructions

AREAD	NEXT
USING	PAUSE
CLEAR	PRINT
DEGREE	RADIAN
END	REM
FOR-TO-STEP	RETURN
GOSUB	STOP
GOTO	THEN
GRAD	USING
IF	
INPUT	
LET	
MEM	

X-2 COMMANDES PARTICULIERES AU PC-1500A

1. Fonctions

AND
ASC
CHR\$
INKEY\$
LEFT\$
LEN
MID\$
NOT
OR
POINT
RIGHT\$
RND
STATUS
STR\$
TIME
VAL

2. Instructions

ARUN
BEEP (pas sur le PC-1211)
CLS
CURSOR
GCURSOR
GPRINT
DATA
DIM
LOCK
ON ERROR
ON GOSUB
ON GOTO
POINT
RANDOM
READ
RESTORE
TRON
TROFF
UNLOCK
WAIT

3. Commandes

(les mêmes que le PC-1211)

4. Directives cassette

MERGE
RMT OFF
RMT ON

5. Directives imprimante

COLOR
CSIZE
GCURSOR
GLCURSOR
GPRINT
GRAPH
LCURSOR
LF
LINE
LLIST
LPRINT
RLINE
ROTATE
SORGN
TAB
TEST
TEXT

Z. TABLE DE REFERENCE DES COMMANDES

1. Fonctions

Fonction	Abréviations	Remarques
ABS	AB.	Valeur absolue
ACS	AC.	COS^{-1}
AND	AN.	exp. AND exp. (AND de Boole)
ASC	ASC	Convertit des caractères en code ASCII ASC "caractère" variable de caractères
ASN	AS.	SIN^{-1}
ATN	AT.	TAN^{-1}
CHR\$	CH. CHR.	Convertit des caractères en code ASCII
COS	COS	
DEG		Convertit des degrés, minutes et secondes en degrés décimaux.
DMS	DM.	Convertit des degrés décimaux en degrés, minutes et secondes.
EXP	EX.	e^x
INKEY\$	INK. INKE. INKEY.	variable de caractères = INKEY\$ Si l'on appuie sur une touche pendant l'exécution de la commande INKEY\$, le caractère ASCII sera lu dans la variable de caractères.
INT		Réduit une valeur à un nombre entier: INT (10/3) ENTER < Display > 3
LEFT\$	LEF. LEFT.	LEFT\$ (variable de caractères, expression numérique) Prend le nombre de caractères spécifié du côté gauche de la chaîne de caractères spécifiée.
LEN		LEN "caractère" variable de caractères Donne le nombre de caractères de la chaîne de caractères spécifiée.
LOG	LO.	$\log_{10} X$
LN		$\log_e X$
MEM	M. ME.	Affiche le restant des nombres de pas disponibles dans la mémoire. Identique à STATUS 0.

Fonction	Abréviations	Remarques
MID\$	MI. MID.	MID\$ (variable de caractères, exp1 numérique, exp2 numérique); Prend un (ou des) caractère(s) du milieu de la chaîne de caractères spécifiée.
NOT	NO.	NOT exp. [négation booléenne]
OR		exp. OR exp. [OR booléen]
π (PI)		Spécifie le rapport de la circonférence: (= 3,141592654)
POINT	POI. POIN.	POINT expression numérique Rapporte un nombre correspondant au motif de points activés de la colonne donnée.
RIGHT\$	RI. RIG. RIGH. RIGHT.	RIGHT\$ (variable de caractères, exp. numérique) Prend le nombre de caractères spécifié du côté droit de la chaîne de caractères donnée.
RND	RN.	RND expression Engendre des nombres aléatoires.
SGN	SG.	Fonction de signe
SIN	SI.	Sinus d'une expression.
$\sqrt{\quad}$ (SQR)	SQ.	Racine carrée d'une expression.
STATUS	STA. STAT. STATU.	STATUS 0 ou STATUS 1 (0) Nombre de pas disponibles d'un programme. (1) Nombre de pas accomplis d'un programme.
STR\$	STR.	STR\$ expression numérique. Convertit des valeurs numériques en chaîne de caractères.
TAN	TA.	Tangente d'une expression.
TIME	TI. TIM.	(1) TIME = MMDDHH.MMSS (2) TIME (annonce l'heure et la date présentes) La fonction TIME établit et extrait le mois (MM), le jour (DD), l'heure (HH), la minute (MM) et la seconde (SS).
VAL (value)	V. VA.	VAL { "caractère" variable de caractères. } convertit une chaîne de caractères en valeurs numériques.
^		Fonction exponentielle.

2. Instructions

Instruction	Abréviations	Remarques
AREAD (lecture automatique)	A. AR. ARE. AREA.	AREAD variable Lors de l'exécution de programmes au moyen de touches DÉFINIES, AREAD introduit le contenu de l'affichage dans la variable spécifiée.
ARUN (passage automatique)	ARU.	ARUN Ordonne le démarrage automatique de l'exécution d'un programme quand le PC-1500A est en marche.
BEEP	B. BE. BEE.	BEEP exp1, exp2, exp3 Commande de signal sonore. Met en marche et arrête les fonctions génératrices de signaux sonores, et spécifie le ton et la durée des dits signaux.
CLEAR	CL. CLE. CLEA.	Commande l'effacement de toutes les valeurs de variables.
CLS (clears)		Efface l'affichage.
CURSOR	CU. CUR. CURS. CURSO.	(1) CURSOR exp. ($0 \leq \text{exp.} \leq 255$) spécification de la position de départ de l'affichage. (2) CURSOR Ramène le curseur à la gauche de l'affichage.
DATA	DA. DAT.	DATA exp, ext, ... Spécifie des données à l'intérieur d'un programme. Utilisé avec une instruction READ.
DEGREE	DE. DEG. DEGR. DEGRE.	Choisit les degrés comme mode angulaire courant. [°]
DIM (dimension)	D. DI.	(1) DIM nom de variable (exp) (2) DIM nom de variable (exp) * exp3 (3) DIM nom de variable (exp1, exp2) Spécifie la taille et les dimensions d'un tableau. (): exp3:
END	E. EN.	Annonce la fin d'un programme.
FOR TO STEP	F. FO. STE.	(1) FOR variable numérique = exp1 TO exp2 Début de la boucle FOR-NEXT. Utilisée en correspondance avec la commande NEXT. (2) FOR variable numérique = exp1 TO exp2 STEP.exp3 exp 1: expression initiale exp 2: valeur finale exp 3: intervalle à augmenter avec chaque boucle.

Instruction	Abréviations	Remarques
GCursor (graphic cursor)	GCU. GCU. GCU. GCU.	Spécifie la position du curseur graphique.
		GCursor expression (0 ≤ expression ≤ 155) ou (&0 ≤ expression ≤ &9B)
GOSUB	GOS. GOSU.	GOSUB { expression "caractère" variable de caractère. } Démarre l'exécution d'un sous-programme à la ligne ou à l'étiquette spécifiée.
GOTO	G. GO. GOT.	GOTO { expression "caractère" variable de caractère. } Transfère le contrôle à la ligne ou à l'étiquette spécifiée
GPRINT (impression de graphisme)	GP. GPR. GPRI. GPRIN.	Manoeuvre l'affichage LCD pour les graphismes.
		(1) GPRINT "OO OO OO ..." {" " constituent des nombres hexadécimaux) (2) GPRINT O; O; ... (3) GPRINT &O; &O; ...
GRAD	GR. GRA.	Choisit les grades comme mode angulaire courant. ([9])
IF		(1) IF expression conditionnelle (THEN) expression (2) IF expression arithmétique (THEN) expression Evalue les conditions données et soit déplace l'exécution à la ligne suivante soit exécute l'expression. THEN est facultatif.
INPUT	I. IN. INP. INPU.	(1) INPUT nom de variable (2) INPUT "chaîne", variable, "chaîne", variable (3) INPUT "chaîne"; variable, "chaîne"; variable
LET	LE.	(1) LET variable numérique = expression (2) LET variable à caractères = "caractères" (3) LET variable à caractères = variable à caractères LET est facultatif, excepté à l'intérieur de commandes IF.
LOCK	LOC.	Bloque l'ordinateur sur le mode actuel.
NEXT	N. NE. NEX.	NEXT nom de variable Signale la fin de la boucle FOR-NEXT. Le nom de variable doit être le même que le nom de variable de l'instruction FOR.

Instruction	Abréviations	Remarques
ON ERROR	O. ER. ERR. ERRO.	ON ERROR GOTO expression Instruction détectrice d'erreurs
ON GOSUB	O. GOS. GOSU.	ON expression GOSUB exp1, exp2, exp3. . . Forme automatisée de l'instruction GOSUB. Démarre l'exécution de l'un des sous-programme par exp1, exp2, exp3, etc. selon la valeur de la variable donnée.
ON GOTO	O. G. GO. GOT.	ON expression GOTO exp1, exp2, exp3 Forme automatisée de l'instruction GOSUB. Démarre l'exécution de l'un des sous-programmes spécifiés par exp1, exp2, exp3, etc., selon la valeur de la variable donnée.
OPN	OP	OPN OPN "Nom du dispositif" Cette commande peut être utilisée quand le PC-1500A est connecté au CE-158 (Interface). Pour plus de détails, se référer au Manuel d'Instructions du CE-158.
PAUSE	PA. PAU. PAUS.	Identique à la commande PRINT. Affiche l'information spécifiée pendant environ 0,85 secondes, puis continue le programme.
POINT	POI. POIN.	POINT expression ($0 \leq \text{expression} \leq 155$) ($\&0 \leq \text{expression} \leq \&9B$) Ramène un nombre représentant le motif de points activés dans la colonne donnée de l'affichage.
PRINT	P. PR. PRI. PRIN.	(1) PRINT { expression "caractère" variable de caractère. } (2) PRINT { expression "caractère" variable de caractère } , { expression "caractère" variable de caractère } (3) PRINT { expression "caractère" variable de caractère } ; { expression "caractère" variable de caractère } ; . . . ; { expression "caractère" variable de caractère } ;
RADIAN	RAD. RADI. RADIA.	Choisit les radians comme mode angulaire courant.
RANDOM	RA. RAN. RAND. RANDO.	Change la semence utilisée par RND pour engendrer des nombres aléatoires.

Instruction	Abréviations	Remarques
READ	REA.	READ variable, variable2, ... etc. Entre des données des instructions DATA dans des variables spécifiées.
REM (remarque)		REM ... information explicative ...
		Spécifie des remarques à l'intérieur d'un programme.
RESTORE	RES. REST. RESTO.	(1) RESTORE expression Change l'ordre des données entrées par l'instruction READ. (2) RESTORE Lit les données à partir de la première instruction DATA.
RETURN	RE. RET. RETU. RETUR.	Continue l'exécution à l'instruction après l'instruction GOSUB qui a appelé le présent sous-programme.
STOP	S. ST. STO.	Commande l'arrêt de l'exécution d'un programme.
THEN	T. TH. THE.	Utilisé dans une instruction IF pour séparer une expression du test conditionnel qui détermine si l'expression sera exécutée ou non. THEN { expression "caractère" variable de caractère }
TRON (trace on)	TR. TRO.	Déclenche le mécanisme de mise au point.
TROFF (trace off)	TROF.	Arrête le mécanisme de mise au point.
UNLOCK	UN. UNL. UNLO. UNLOC.	Annule LOCK pour permettre à l'utilisateur de changer de mode.
USING	U. US. USI. USIN.	(1) USING "###.###^" (2) USING "&&&.....&&&" (3) PRINT USING "Format"; (4) USING (5) PRINT USING; Spécifie le format d'informations affichées par un programme.
WAIT	W. WA. WAI.	WAIT expression (0 ≤ expression ≤ 65535) Spécifie la durée d'affichage d'une information lors de l'utilisation des commandes PRINT. Une instruction WAIT non accompagnée d'un argument annule la spécification antérieure (durée = infini).

3. Commandes

Instruction	Abréviations	Remarques
CONT (continue)	C. CO. CON.	Redémarre l'exécution d'un programme arrêtée temporairement. Efficace sur le mode RUN.
GOTO	GO. GOT.	Démarre l'exécution d'un programme à la ligne spécifiée de ce programme.
LIST	L. LI. LIS.	Effectue le listage des programmes. Efficace sur le mode PRO.
NEW		(1) NEW (2) NEW Ø (Effectif uniquement dans le mode PRO) Dans ce mode, cette fonction remet le programme et toutes les variables à zéro. (3) NEW expression (ou, l'expression ≠ 0) Dans le mode PRO, l'ère entre la première adresse et l'adresse de "expression - 1" est assurée pour l'ère du langage de la machine, et le reste pour le programme et toutes les variables sont remis à zéro.
RUN	R. RU.	(1) RUN (2) RUN expression (3) RUN "caractère" variable de caractères Efficace sur le mode RUN.

4. Commandes Cassette

Instruction	Abréviations	Remarques
CHAIN	CHA. CHAI.	Entre un programme enregistré sur bande magnétique et exécute ce programme. (1) CHAIN "nom de fichier" (CHAIN-1 "nom de fichier") (2) CHAIN "nom de fichier", expression (CHAIN-1 "nom de fichier", expression) (3) Formes abrégées de noms de fichier de fichier de (1) et (2).
CLOAD (chargement de cassette)	CLO. CLOA.	Enregistre le contenu de la mémoire Programme ou Réserve sur bande magnétique. (1) CLOAD (CLOAD-1) (2) CLOAD "nom de fichier" (CLOAD-1 "nom de fichier")

Instruction	Abréviations	Remarques
CLAOD? (chargement de cassette)	CLO.? CLOA.?	Commande de comparaison. (Cette commande compare un programme mémorisé avec un programme enregistré sur bande magnétique. Peut aussi servir à comparer le contenu de la mémoire de réserve. (1) CLOAD? (CLOAD?-1) (2) CLOAD? "nom de fichier" (CLOAD?-1 "nom de fichier")
CSAVE (enregistrement sur cassette)	CS. CSA. CSAV.	Cette commande fait enregistrer sur bande le contenu des mémoires de programme et de réserve. (1) CSAVE (CSAVE-1) (2) CSAVE "nom de fichier" (CSAVE-1 "nom de fichier")
INPUT #	I.# IN.# INP.# INPU.#	Cette commande transfère les données enregistrées sur bande dans les variables spécifiées. Sa forme est la même que celle de la commande PRINT#.
MERGE	MER. MERG.	Cette commande extrait des programmes enregistrés auparavant sur bande magnétique. A la différence de CLOAD, le programme extrait n'est pas inscrit au-dessus du programme logé dans la mémoire.
PRINT #	P.# PR.# PRI.# PRIN.#	Commande d'enregistrement de données. Cette commande enregistre sur bande les données dans le PC-1500A. (1) PRINT# nom de variable, nom de variable . . . (2) PRINT# "nom de fichier", nom de variable, . . . (PRINT#-1 "nom de fichier", nom de variable, . . .)
RMT OFF	RM.OF. RMTOF.	
RMT ON	RM.O. RMTO.	

5. Commandes de l'imprimante

Instruction	Abréviations	Remarques
COLOR	COL. COLO.	Spécifie la couleur de l'impression COLOR expression COLOR expression ($0 \leq \text{expression} \leq 3$)
CSIZE (taille des caractères)	CSI. CSIZ.	Spécifie la taille des caractères imprimés. CSIZE expression ($1 \leq \text{expression} \leq 9$)

Instruction	Abréviations	Remarques
GLCURSOR (graphic cursor)	GL. GLC. GLCU. GLCURS. GLCURSO.	Déplace le stylo du point de départ à un point coordonné X-Y. Valide sur le mode GRAPH uniquement. GLCURSOR (exp1, exp2)
GRAPH (graphisme)	GRAP.	Etablit le mode sur lequel s'effectue le dessin des graphes et des illustrations.
LCURSOR (curseur de ligne)	LCU. LCUR. LCURS. LCURSO.	Déplace le stylo de l'imprimante sur l'une des positions de caractère disponibles.
LF (Alimentation lignes)		Déroule le papier du nombre de lignes indiqué par l'expression. LF expression
LINE	LIN.	Commandes de tracé de ligne (1) LINE (exp1, exp2)-(exp3, exp4) (2) LINE (exp1, exp2)-(exp3, exp4), exp5, exp6 (3) LINE (exp1, exp2)-(exp3, exp4), exp5, exp6, B exp5: spécifie le type de ligne (continue, discontinue) exp6: spécifie la couleur de la ligne. B: (4) LINE (exp1, exp2)-(exp3, exp4)-... ... (exp11, exp12)
LLIST	LL. LLI. LLIS.	Liste une ou plusieurs lignes du programme courant.
LPRINT	LP. LPR. LPRI. LPRIN.	Fait imprimer des informations spécifiées. Valide sur le mode texte uniquement. Sa forme est la même que celle des commandes PRINT.
RLINE (ligne relative)	RL. RLI. RLIN.	Trace des lignes avec des coordonnées à l'emplacement actuel du stylo. Efficace sur le mode GRAPH seulement. Sa forme est la même que celles des commandes LINE.

Instruction	Abréviations	Remarques
ROTATE (pivoter)	RO. ROT. ROTA. ROTAT.	Spécifie le sens dans lequel les caractères seront imprimés (sens d'impression). ROTATE expression ($0 \leq \text{expression} \leq 3$)
SORGN (fixation du point d'origine)	SO. SOR. SORG.	Cette commande établit l'emplacement actuel du stylo comme le nouveau point de départ (point d'origine) du stylo. Valide sur le mode GRAPH seulement.
TAB		Identique à LCURSOR, mais peut être utilisé dans des instructions LPRINT: (1) LPRINT TAB Expression; . . . (2) TAB espression;
TEST	TE. TES.	Teste l'imprimante. Trace un carré de 5 mm de côté dans chaque couleur.
TEXT	TEX.	Spécification du mode TEXTE, sur lequel s'effectue l'impression des caractères et des nombres.

L'instruction IF – Supplément (Voir page 48)

– **Avertissement** –

Cet appareil est capable de calculs pour une mantisse comportant jusqu'à 10 chiffres. Cependant, pour augmenter la précision, la mantisse est calculée jusqu'à 12 chiffres dans l'appareil, et le résultat obtenu est affiché arrondi au 10ème chiffre. Par exemple, $5/9$ et $5/9 * 9$ sont calculés comme suit:

$5/9 \rightarrow 5.5555555555E - 01$
↑ Ceci est arrondi au 10ème chiffre.
Affichage $\rightarrow 5.555555556E - 01$

$5/9 * 9 \rightarrow 4.9999999999E 00$
↑ Ceci est arrondi au 10ème chiffre.
Affichage $\rightarrow 5$

Les calculs peuvent ainsi être effectués pour des mantisses comportant jusqu'à 12 chiffres. Ceci peut provoquer une différence dans les résultats de calculs effectués en succession et indépendamment.

(Exemple 1) $3^2 - 9 =$

Calcul en succession: 3 [SHIFT] [^] 2 [-] 9 [ENTER] $\rightarrow -9E - 11$
Calcul indépendant: 3 [SHIFT] [^] 2 [ENTER] $\rightarrow 9$
[-] 9 [ENTER] $\rightarrow 0$

Même dans l'instruction IF, ces différences risquent de provoquer un fonctionnement imprévu du programme pour tout calcul successif.

(Exemple 2) 10 INPUT A
20 IF A ^ 2 >= 9 THEN 50

⋮

Pour $A = 3$, le résultat de $3 \wedge 2$ est $8.99999999991E 00$, ce qui donne une instruction IF non formulée.

Dans ce cas, reprogrammer le calcul en utilisant des variables de façon à le rendre indépendant, comme suit:

```

10 INPUT A
15 B = A ^ 2
20 IF B >= 9 THEN 50

```

Le résultat de $A \wedge 2$ est remplacé par une variable qui est utilisée pour formuler une expression conditionnelle.

Les calculs de puissance sont fondés sur $\log x$ et 10^x , ce qui a tendance à provoquer une différence dans les résultats par rapport à ceux calculés dans l'ordinateur.

$$A \wedge B \rightarrow 10^{B \log A}$$

Code de Erreur – Supplément

Lorsque l'erreur 1 apparaît, noter les points suivants:

(1) Entrée

- Pour corriger une erreur d'entrée:

```
10 APRINT A$
```



```
10 GPRINT A$
```

← A est remplacé par G.

- Pour remplacer l'instruction CURSOR par l'instruction GCURSOR:

```
10 CURSOR 10
```

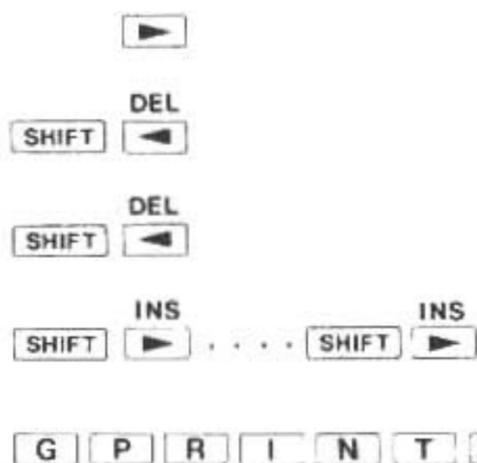


```
10 GCURSOR 10
```

← G est inséré.

- Lorsqu'on utilise **G** **DEF** **W** pour l'entrée GPRINT:

Lorsqu'elle a été corrigée ou entrée en utilisant la méthode ci-dessus, l'instruction ne peut pas être jugée par l'ordinateur. Dans ce cas, l'instruction doit être entrée à nouveau en utilisant les touches alphabétiques.



```

10: APRINT A$
10: A PRINT A$
10: P RINT A$
10: A $
10: [ ] [ ] [ ] [ ] [ ] [ ] A$
10: GPRINT A$

```

Le curseur permet de vérifier si l'instruction a été correctement entrée.

(Entrée correcte)

(Entrée incorrecte)



SHARP CORPORATION

OSAKA, JAPAN