NOTICE DU

K-Basic v.5

POUR **MZ-700**

B. KOKANOSKY

INTRODUCTION

Le K-BASIC est un interpréteur destiné à la programmation de l'ordinateur SHARP MZ-700 . Il a été obtenu par modification et extension du S-BASIC livré avec l'ordinateur et , par suite, sera en grande partie compatible avec celui-ci.

Le K-BASIC se présente sous deux aspects :

+ C'est tout d'abord un interpréteur Basic "standard" ,possédant toutes les instructions du S-BASIC avec ,de plus :
  ELSE pour les tests,
  INKEY et KEY complètant l'ordre GET,
  MAX,MIN,OR,AND,XOR,DIV,MODULO,ASN,ACS pour les calculs,
  GRAPH,DRAW,POINT pour le semi-graphisme sur l'écran,
  SWAP,NULL,COPY pour manipuler des tableaux (à moins de I28 dimensions ...),
  WAIT et PWAIT pour ralentir l'éxécution des programmes,
  INV$,IN$,STRING$,INSTR pour manipuler des chaînes de caractères,
  VPOS et HPOS pour trouver la position du curseur,
  DISP pour afficher,en programme,une ligne de programme,
  EVAL permettant d'appliquer l'évaluateur de formules au contenu d'une chaîne de caractères.

+ C'est surtout un interpréteur destiné à écrire des programmes selon les règles de la "programmation structurée".Dans ce type de programmation,bien connue des utilisateurs de langages de haut niveau (comme PASCAL),le programme sera construit de façon modulaire,chaque module pouvant être mis au point séparément. Avec un peu d'habitude,on pourra alors écrire le longs programmes,faciles à mettre au point,à modifier et dont la structure (et par suite,le mode de fonctionnement) se reconnaît au premier coup d'oeil sur un listing.Surtout,on ne trouvera plus ces nombreux GOTO renvoyant d'un bout à l'autre du programme et le rendant incompréhensible à tout autre qu'à l'auteur ,et même à celui-ci , quelques mois plus tard !
  Que le lecteur se rassure ! .Le K-BASIC permettant les deux types de programmation ,il pourra toujours écrire un programme en suivant les "mauvaises" habitudes dûes à l'utilisation des Basics standards,puis le modifier,l'améliorer pour lui donner une forme plus structurée.Ainsi,petit à petit,il lui sera possible de s'habituer à la programmation structurée et de se rendre compte de la grande puissance de cette méthode de programmation.

Ce type de programmation est permis par la présence de :

3 types de boucles : FOR NEXT
REPEAT UNTIL
WHILE WEND

2 types de tests : IF.....ELSIF....ELSIF.....ELSE.....ENDIF
CASE..OF..WHEN...WHEN....OTHERWISE...ENDCASE

(Ces deux structures peuvent s'étendre sur un nombre quelconque de lignes)

2 types de sous-programmes (en plus des GOSUB-RETURN):
les procédures définies par DEF PROC et appelées par PROC
les fonctions définies par DEF FN et appelées par FN
et permettant toutes deux le passage de paramètres par valeur et par référence avec,bien entendu,l'utilisation possible de variables locales par LOCAL.
Ces deux structures permettent évidemment la Récursivité.
(Remarque: DEF FN existe en S-Basic mais sous une forme beaucoup moins puissante qu'en K-Basic.)

Nous allons maintenant voir plus en détail comment fonctionnent ces diverses structures.Je précise cependant qu'il n'est pas question ici de faire un cours de programmation structurée et qu'après avoir compris les quelques exemples donnés dans cette notice,le lecteur devra expérimenter par lui même,les possibilités (structurées j'espère !) du K-BASIC .

Amiens,le I^{er} Septembre I984

B. Kokanosky

NOTE IMPORTANTE

Juste après le chargement du K-Basic (grâce à la commande L),lorsque la sonnerie se fait entendre,on appuiera sur CR pour lancer l'éxécution du Basic.La sonnerie signifie que le K-Basic attend l'option choisie par l'utilisateur :

CR : éxécution du Basic
Shift Break ; retour au moniteur en mémoire morte
S : réenregistrement du Basic sur cassette.Pour cette option,voir les indications de la page 6 .

80.GOTO	A7 MOVE	CE WOPEN	F5 >	FE 9D	FF 94
81.GOSUB	A8 RMOVE	CF CLOSE	F6 <	FE 9E	FF 95 EOF
82.DISP	A9 TRON	DO ROPEN	F7 +	FE 9F	FF 96
83 RUN	AA TROFF	DI.GRAPH	F8 -	FE AO	FF 97
84 RETURN	AB INP#	D2.DRAW	F9.DIV	FE AI	FF 98
85.RESTORE	AC.INKEY	D3.SWAP	FA.MODULO	FE A2 MUSIC	FF 99
86 RESUME	AD GET	D4.NULL	FB /	FE A3 TEMPO	FF 9A
87 LIST	AE PCOLOR	D5.COPY	FC *	FE A4 CURSOR	FF 9B
88.ENDPROC	AF PHOME	D6.EXC	FD ↑	FE A5 VERIFY	FF 9C
89 DELETE	BO HSET	D7.WAIT		FE A6 CLR	FF 9D
8A RENUM	BI GPRINT	D8.PWAIT	FE 80	FE A7 LIMIT	FF 9E JOY
8B AUTO	B2.KEY	D9 KILL	FE 81 SET	FE A8	FF 9F
8C.ENDCASE	B3 AXIS	DA.LOCAL	FE 82 RESET	FE A9	FF AO CHR§
8D FOR	B4 LOAD	DB.CASE	FE 83 COLOR	FE AA	FF AI STR§
8E NEXT	B5 SAVE	DC.WHEN	FE 84	FE AB	FF A2 HEX§
8F PRINT	B6 MERGE	DD.OTHERWISE	FE 85	FE AC	FF A3
90.ELSIF	B7.OF	DE.EXIT	FE 86	FE AD	FF A4
91 INPUT	B8 CONSOLE	DF.SPOKE	FE 87	FE AE BOOT	FF A5
92.ENDIF	B9.PROC	EO TO	FE 88		FF A6
93.IF	BA OUT #	EI STEP	FE 89	FF 80 INT	FF A7
94 DATA	BB CIRCLE	E2 THEN	FE 8A	FF 81 ABS	FF A8
95 READ	BC TEST	E3 USING	FE 8B	FF 82 SIN	FF A9
96.DIM	BD PAGE	E4.PI ou ΠΤ	FE 8C	FF 83 COS	FF AA.INV§
97 REM	BE.REPEAT	E5	FE 8D	FF 84 TAN	FF AB ASC
98 END	BF.UNTIL	E6 TAB	FE 8E	FF 85 LN	FF AC LEN
99 STOP	CO ERASE	E7 SPC	FE 8F	FF 86 EXP	FF AD VAL
9A CONT	CI ERROR	E8.VARPTR	FE 90	FF 87 SQR	FF AE
9B CLS	C2.ELSE	E9.MAX	FE 91	FF 88 RND	FF AF
9C.DO	C3.USR	EA.MIN	FE 92	FF 89 PEEK	FF BO
9D ON	C4 BYE	EB.OR	FE 93	FF 8A ATN	FF BI.EVAL
9E LET	C5.WHILE	EC.AND	FE 94	FF 8B SGN	FF B2
9F NEW	C6.WEND	ED.XOR	FE 95	FF 8C LOG	FF B3 ERN
AO POKE	C7.DEF	EE ><	FE 96	FF 8D	FF B4 ERL
AI OFF	C8.RESULT=	EF <>	FE 97	FF 8E PAI	FF B5 SIZE
A2.MODE	C9.BEEP	FO =<	FE 98	FF 8F RAD	FF B6.VPOS
A3 SKIP	CA	FI <=	FE 99	FF 90.ASN	FF B7.HPOS
A4 PLOT	CB	F2 =>	FE 9A	FF 91.ACS	FF B8
A5 LINE	CC	F3 >=	FE 9B	FF 92	FF B9
A6 RLINE	CD	F4 =	FE 9C	FF 93	FF BA LEFT§

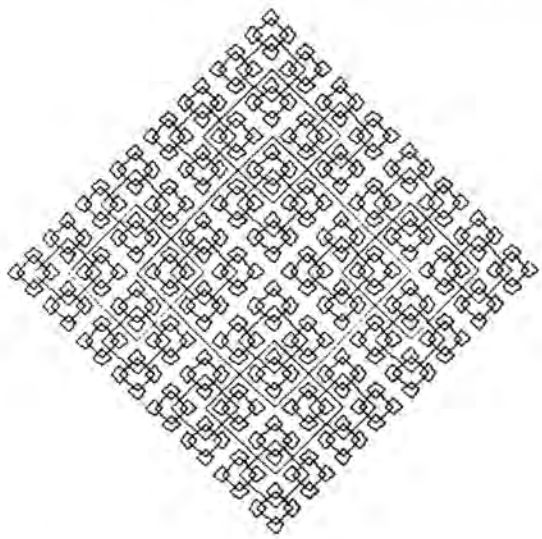
Suite des mots clés et de leur code

FF BB RIGHT§	FF C0	FF C5
FF BC MID§	FF C1	FF C6
FF BD.IN§	FF C2	FF C7.FN
FF BE.POINT	FF C3.STRING§	FF C8
FF BF.INSTR	FF C4 TI§	FF C9

Un point suivant le code signifie que le mot clé est nouveau ou que son mode de fonctionnement a été modifié ,par rapport au S-Basic.

On trouvera des renseignements supplémentaires à partir de la page 29,où commence la description des mots clés du K-Basic.Cependant,lorsqu'il n'y a eu aucune modification,par rapport au S-Basic,on se contentera de renvoyer le lecteur au manuel de l'ordinateur.

++



```
10 MODE GR:MODE 1
20 PROC"carre"(240,-240,180)
30 MODE TN
40 END
50 REM ***********************************
100 DEF PROC"carre"(X,Y,C)
110 IF C>10 DO
120 : C=C/2
130 : PROC"carre"(X+C,Y,C)
140 : PROC"carre"(X-C,Y,C)
150 : PROC"carre"(X,Y-C,C)
160 : PROC"carre"(X,Y+C,C)
170 : MOVE X,Y+C
180 : LINE X+C,Y,X,Y-C,X-C,Y,X,Y+C
190 ENDIF
200 ENDPROC
210 REM ***********************************
```

Numéro	Message
I	Plus de mémoire
2	Type de variable illégal
3	Erreur de pile
4	Nombre non entier
5	Valeur incorrecte
6	Dépassement de capacité
7	Opération illégale
8	Chaîne trop longue
9	Division par 0
I0	Pas d'argument
II	Syntaxe incorrecte
I2	Il manque ')'
I3	Il manque ','
I4	Il manque '('
I5	Mauvais nom de variable
I6	Argument négatif
I7	Mauvais dimensionnement
I8	Indice trop grand
I9	Tableau inconnu
20	Trop de paramètres
2I	Ligne trop longue
22	RESUME impossible
23	Pas de ':' ou fin de ligne
24	Il manque '='
25	Mot-clé inconnu
26	6 chiffres pour TI$
27	Ce n'est pas un chiffre
28	Ligne inconnue
29	Argument nul
30	UNTIL sans REPEAT
3I	NEXT sans FOR
32	Mauvaise variable
33	Il manque ']'
34	CONT impossible
35	Il manque ';'
36	Pas de DATA
37	WHILE sans WEND
38	Ce n'est pas un nombre
39	N° de ligne trop grand
40	RETURN sans GOSUB
4I	Il n'y a pas eu d'erreur
42	Mode imprimante incorrect
43	Mauvais type de fichier
44	Caractère illégal (USING)
45	Fin de fichier
46	Fichier déjà ouvert
47	Erreur de lecture
48	Imprimante non connectée
49	Pas d'ENDIF
50	Mode IF incorrect
5I	Il manque DO
52	WEND sans WHILE
53	Il manque OF
54	PROC ou FN inconnue
55	Mauvais nombre de paramètres
56	ENDPROC sans PROC
57	RESULT sans FN
58	Il manque WHEN
59	Pas d'ENDCASE

Remarques: Les erreurs I et 3 sont "fatales" en ce sens qu'elles conduisent à l'effacement du catalogue des FN/PROC et de la pile Basic.

Comme le lecteur l'aura remarqué ,les messages d'erreur sont assez nombreux et précis. Cependant,dans certains cas, ils peuvent surprendre l'utilisateur non averti :

Exemples: PRINT ASN(2) donnera l'erreur I6 (argument négatif) car le Basic calcule ASN(2) par : ATN(2/SQR(I-2*2)) et l'erreur se produit dans SQR .

De même l'oubli d'un ENDIF,en mode I ,donnera l'erreur : Pas d'ENDIF en NNNN " où NNNN est le numéro de la dernière ligne du programme puisque le Basic a cherché ENDIF dans tout le programme et ne l'a pas trouvé,en arrivant à la dernière ligne.

Note : Après une erreur,il suffira de faire LIST . (ou L..) pour que le Basic affiche la ligne où s'est produite l'erreur.

MODIFICATIONS OU ENREGISTREMENT DU K-BASIC

Un interpréteur Basic étant un programme en langage machine très complexe,bien que beaucoup de vérifications aient été faites et que beaucoup de précautions aient été prises lors de sa construction,il serait illusoire de penser qu'il ne peut se produire aucune erreur lors de son fonctionnement.Le K-Basic étant écrit en mémoire vive,il sera toujours possible de le modifier ou de le corriger.

Afin que cela soit plus aisé , 3 zones vides ont été prévues à des endroits "stratégiques" : zone I : de FREEI à TKTBLI (non comprise)
zone 2 : de FREE2 à TKTBL2 (non comprise)
zone 3 : de FREE3 à TABNOR (non comprise)

(Les adresses correspondantes seront trouvées dans les tables (voir fin)).
On remarquera que ces zones sont situées juste après chaque table définissant les mots-clés du K-Basic,ce qui pourrait permettre à un utilisateur averti et astucieux de créer ses propres mots clés avec leur interprétation.

La taille de ces zones peut varier avec la version de l'interpréteur.Les deux premières zones ont environ 500 octets tandis que la troisième a environ 450 octets.

Pour effectuer des corrections sur le K-Basic,il faut le charger en mémoire sans l'éxécuter .Pour celà ,une boucle d'attente (matérialisée par une sonnerie), se produit juste après le chargement du Basic.

Pour lancer le Basic,on appuiera sur CR simplement.

Par Shift Break,on retournera au moniteur de la mémoire morte,sans effacer le Basic,ce qui permettra d'utiliser les commandes du moniteur pour modifier éventuellement le Basic.On retournera à la boucle d'attente par J 8670 (soit STARTB+I200H)

Par appui sur S ,et après avoir placé une cassette dans le magnétophone,on pourra ré-enregistrer le Basic (cependant,il ne faudra surtout pas avoir modifié les mémoires de IOFO à II07 (Hexa)).

+ Les noms de variables peuvent être de longueur quelconque mais ne peuvent contenir que des lettres majuscules ,des chiffres et le caractère ← (comme en S-Basic).Cependant,ici,tous les caractères seront significatifs.Le nom de la variable devra com encer par une lettre majuscule et les lettres du début ne peuvent former un mot-clé du Basic : par exemple TABLEAU sera refusé à cause de TAB .Par contre MONNAIE sera acceptée bien que ON soit un mot clé. Il faudra donc prendre garde à ne pas écrire des expressions du genre : IFA=BTHENPRINT..... car le Basic prendra BTHENPRINT com e une seule variable. (en conclusion : éviter d'écrire des programmes illisibles ,c'est à dire sans espaces entre les mots-clés !!).
+ Il est possible d'indenter les programmes pour mieux faire ressortir leur structure,en com ençant les lignes par :

exemple : 10 FOR I=I TO 10
20 : PRINT I
30 NEXT I

+ Les mots clés du Basic doivent être écrits sans espaces :GO TO sera refusé .
+ Il est possible d'imprimer " dans une chaîne de caractères.Par exemple : PRINT "Impression de "" et d'autre chose" écrira sur l'écran :
Impression de " et d'autre chose .
+ Pour interrompre momentanément l'éxécution d'un programme,laisser BREAK enfoncée.
+ Les touches du MZ-700 sont à répétition.Lorsqu'on laisse le doigt sur une touche, après une temporisation A,le caractère correspondant est affiché plusieurs fois, chaque affichage étant séparé du précédent par une temporisation B.

Pour modifier A : POKE 700,N . La valeur normale de N est 96 .On augmente A en augmentant N et inversement.

Pour modifier B : POKE 648,M . La valeur normale de M est 16. On augmente B en augmentant M et inversement.

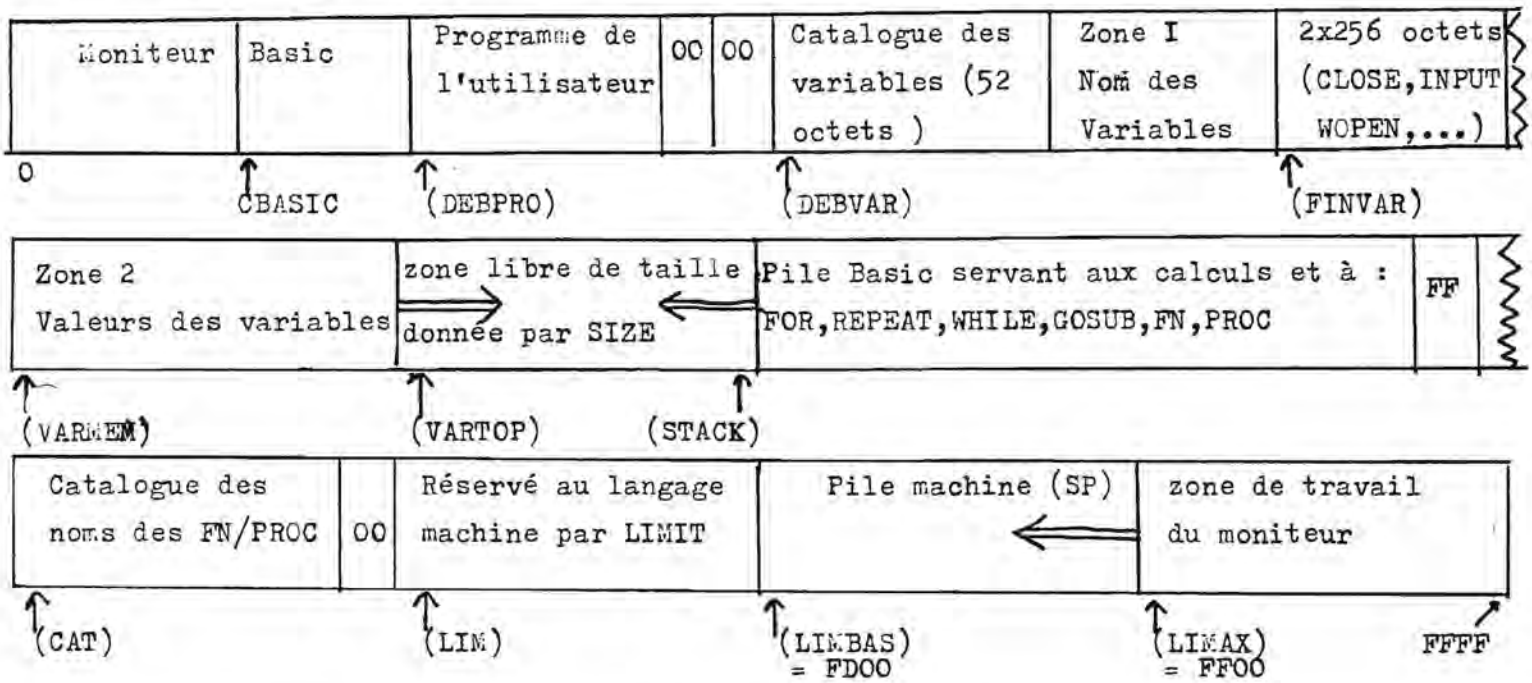
N et M sont compris entre 0 et 255.Il vaut mieux ne pas leur donner des valeurs trop faibles car on ne pourra plus alors contrôler l'ordinateur !!

+ La table des codes de contrôle (CTRL) de la page 28 du manuel Sharp est incomplète En effet CTRL [est équivalent à CTRL M et effectue CR.
En M-Basic ,CTRL G produit un bip sonore.En programme,on pourra faire la même chose avec USR(62) ou BEEP I ou PRINT CHR$(7).Voir page 50 .

QUELQUES INDICATIONS TECHNIQUES

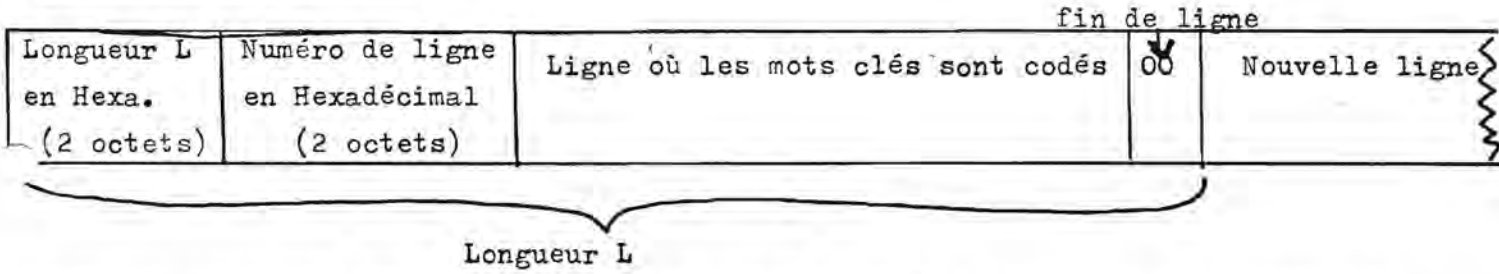
Ces quelques notes sont destinées uniquement au lecteur désirant comprendre le mode de fonctionnement interne du K-BASIC .Elles ne seront pas utilisées dans la suite de cette notice sauf pour la description de l'ordre VARPTR .

CARTE DE LA MEMOIRE :



Les adresses DEBPRO,....,LIMAX sont données dans la table annexée à ces notes.

STRUCTURE DU PROGRAMME :



La fin du programme est marquée par deux octets nuls,à la place de la longueur de ligne.

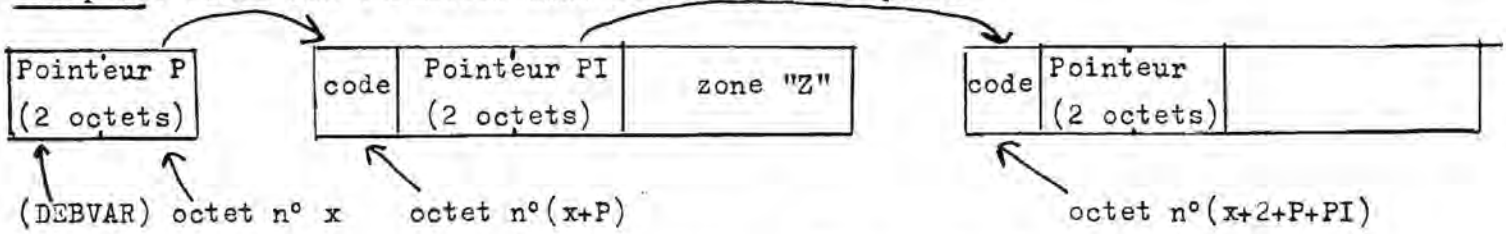
Dans une ligne de programme,les mots clés sont remplacés par leur code (voir les tables),les nombres en virgule flottante sont codés sur 5 octets précédés d'un octet contenant I5 (Hexadécimal),les nombres entrés en hexadécimal (précédés de $) par le programmeur sont codés en hexadécimal sur deux octets précédés d'un octet contenant IIH,les numéros de ligne (utilisés après GOTO par exemple) sont codés en héxadécimal sur deux octets précédés d'un octet contenant OBH (et après une première utilisation du programme,peuvent être remplacés par OCH suivi,sur deux octets,de l'adresse de début de la ligne en question,ce qui accélère les sauts).

ZONE I (Nom des variables) :

Cette zone est organisée sous forme de 26 listes séparées,chaque liste correspondant à l'une des lettres de l'alphabet majuscule.Les divers éléments de chaque liste (les "cellules") sont reliées par des pointeurs relatifs sur deux octets,un pointeur nul signifiant la fin de liste.

La zone I commence par 52 octets contenant les pointeurs relatifs sur les 26 cellules de tête des listes.

Exemple : liste des variables dont le nom commence par A



"code"vaut : 3 pour les chaînes de caractères , 83 pour les tableaux de chaînes
5 pour les nombres , 85 pour les tableaux de nombres

La zone "Z" se présente de la façon suivante :

Nombre ou chaîne :

Longueur du nom	Nom	L	ordre

L est la longueur de la zone de codage (en zone 2).
"ordre" est le numéro d'ordre de cette zone de codage en zone 2.
Il vaut 0 si la variable n'a encore subi aucune affectation.

Tableau :

Longueur du nom	NOM	nb de dimensions	dimension 2 (2 octets)	dimension I (2 octets)	suite

(exemple de 2 dimensions)

I+dimension écrite lors de l'ordre DIM

"suite" se compose de DimI x Dim2 zones de 3 octets constituées de

L	ordre

comme précédemment.

ZONE INTERMEDIAIRE :

Cette zone,dont la constitution n'est pas modifiée,par rapport au S-BASIC,se compose de 256 octets réservés à INPUT suivis de 256 octets utilisés par les ordres CLOSE,ROPEN et WOPEN pour la lecture et l'enregistrement de données sur cassette.

ZONE 2 (Valeurs des variables) :

Cette zone est constituée de cellules repérées par leur numéro d'ordre (de I à FFFF) toutes composées ainsi :

Longueur L	codage sur L octets

(Pour les nombres L=5)

Chaque cellule est donc de longueur L+I

PILE BASIC :

Elle est utilisée pour le stockage temporaire de résultats en cours de calculs :

Nombre flottant : | 5 | codage sur 5 octets |

Chaîne de caractères: | 3 | L | codage sur L octets |

Nombre entier : | 4 | codage sur 4 octets | (calculs avec AND,OR,XOR seulement)

Elle est aussi utilisée pour stocker les informations nécéssaires au bon fonctionnement des boucles,des FN et PROC :

FOR :

2	pointeur ligne suivante	pointeur derrière step	pointeur indice en zone I	codage du STEP	codage de la valeur finale
I	2	2	2	5	5 octets

REPEAT :

9	N° de la ligne du REPEAT	pointeur sur ligne suivante	Pointeur derrière REPEAT
I	2	2	2 octets

GOSUB : Se présente comme REPEAT mais avec le code I en tête

WHILE : Se présente comme REPEAT mais avec le code OAH en tête

Cependant,si le WEND correspondant a déjà été trouvé,on aura :

OD	N° de la ligne du WEND	pointeur sur ligne suivante	Pointeur derrière WEND	.....

les points représentant la zone WHILE normale privée de son code OAH

PROC :

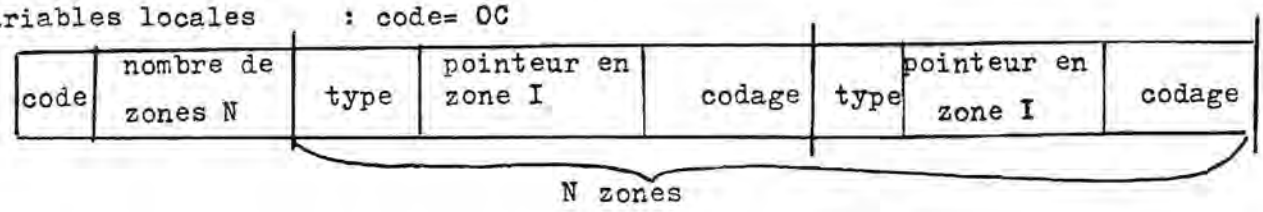
06	6 octets comme REPEAT

FN :

07	Longueur L (2 octets)	Pile machine lors du RET de FN	6 octets comme REPEAT
		L octets	

Paramètres ou variables locales :

Paramètres de FN/PROC : code= 0B
Variables locales : code= 0C



type = 3 ou 5 si variables numériques ou alphanumériques à passage par valeurs
43 ou 45 " " " " " " " " référence
83 ou 85 si tableaux à passage par référence.

CATALOGUE DES FN/PROC :

code = B9 si PROC et C7 si FN

ode	longueur du nom	Nom	pointeur sur le début ligne suivante	pointeur derrière le nom de FN/PROC	code	longueur du nom	..K

Un 00 à la place du code indique la fin du catalogue.
Le catalogue est crée lors de RUN .Les procédures ou fonctions se trouvant en fin de programme seront placées en tête du catalogue.Aussi,en cas de deux procédures de même nom (ou deux fonctions de même nom),seule la dernière pourra être éxécutée.

ZONE LANGAGE MACHINE :

Cette zone,vide à l'initialisation du Basic,sera créee par le programmeur en utilisant l'ordre LIMIT.Elle ne pourra être vidée que par LIMIT MAX .

ONE RESERVEE A LA PILE MACHINE :

Cette zone,allant de FD00 à FEFF,est exclusivement réservée à la pile du microprocesseur.

ZONE DE TRAVAIL DU MONITEUR :

Cette zone qui va de FF00 à FFFF est réservée au moniteur situé en mémoire vive et chargé en même temps que le Basic.

Remarque : Les quelques indications qui viennent d'être données concernent uniquement le K-Basic dont le fonctionnement interne est totalement différent de celui du S-Basic .

Dans ce qui suit , x et y représentent les coordonnées d'un point sur l'écran semi graphique,c'est à dire que x est compris entre 0 et 79 alors que y est compris entre 0 et 99 (pour y entre 50 et 99 ,le point correspondant n'est pas affiché sur l'écran mais dans la partie non visible de la mémoire vidéo).L'origine (0,0) se trouve en haut à gauche de l'écran.

Les ordres graphiques du K-Basic sont les suivants :

SET x,y,c allume (dans la couleur c (facultative)) le point x,y
c varie de 0 à 7 selon la couleur

RESET x,y éteint le point de coordonnées x et y

POINT(x,y) est une variable égale au code couleur du point (x,y) allumé par SET. Dans le cas où ce point est éteint ou s'il s'y trouve un caractère alphanumérique,POINT(x,y) donnera 8 .

GRAPH x,y Place un pseudo-curseur graphique au point (x,y)

DRAW k,x,y,c permet de tracer ou d'effacer des droites sur l'écran.

Pour k=I la droite est tracée dans la couleur c (facultative)
Pour k=0 elle sera effacée.
La droite joindra toujours le point où se trouve le curseur graphique au point (x,y).Après cette instruction,le curseur graphique se trouve en (x,y).

SON

Le K-Basic possède les deux instructions MUSIC et TEMPO du S-Basic.
Il sera aussi possible d'entendre un Bip par USR(62) ou PRINT CHR$(7) (CTRL G) ou BEEP I (BEEP n donnera n bips successifs).

L'instruction BEEP permet d'engendrer des sons plus complexes :
BEEP n,f,d donnera n fois de suite un son de durée d/I00 secondes (approximativement). Si d est absent,la durée sera de I seconde).et de fréquence 895kHz/f .
n doit être entre 0 et 255 ,d et f entre 0 et 65535 . On se gardera de donner à f des valeurs trop faibles,conduisant à des fréquences trop hautes pour que le son soit audible.

Il est aussi possible d'engendrer des sons plus complexes en utilisant certains sous-programmes du moniteur :
USR (68) met en marche le haut parleur et produit un son de fréquence 895kHz/f où f=x+256*y x et y étant les contenus des mémoires 0A39 et 0A3A.
USR(7I) arrête le son .

A titre d'illustration,voici un petit jeu utilisant les ordres graphiques du K-Basic et écrit sous forme structurée .

```
10 PROC"presente"
20 REPEAT
30 : PROC"lune"
40 : PROC"jeu"
50 : PROC"resultat"
60 : PROC"encore"
70 UNTIL R$="N"
80 END
90 REM ***********************************
100 DEF PROC"presente"
110 CLR:MODE 1:COLOR ,,1,5:CLS
120 PRINT[1,2]TAB(9)" Arrivee sur la Lun
e "
130 PRINT"▯▯ Vous pilotez un module luna
ire et vous"
140 PRINT"▯ essayez de le poser sur la b
ase."
150 PRINT"▯▯   Les commandes sont: Z = G
auche"
160 PRINT"▯                          X = Dr
oite"
170 PRINT"▯                          ? = Re
tro-fusees"
180 PRINT[4,0]TAB(11)"▯▯Appuyez sur BREA
K"
190 WAIT 65000
200 ENDPROC
210 REM ***********************************
220 DEF PROC"lune"
230 COLOR ,,7,0:CLS
240 CLS:DIM H(79)
250 H(0)=INT(1+6*RND(1))
260 FOR X=0 TO 37
270 : GRAPH X,47:DRAW 1,X,47-H(X),6
280 : H(X+1)=ABS(H(X)+INT(4*RND(1)-2))
290 NEXT X
300 D=2*(H(38) DIV 2)+1
310 FOR X=38 TO 44
320 : H(X+1)=D
330 : GRAPH X,47:DRAW 1,X,47-D,6
340 NEXT X
350 FOR X=45 TO 78
360 : GRAPH X,47:DRAW 1,X,47-H(X),6
370 : H(X+1)=ABS(H(X)+INT(4*RND(1)-2))
380 NEXT X
390 GRAPH 79,47:DRAW 1,79,47-H(79),6
400 FOR X=0 TO 79:H(X)=47-H(X):NEXT X
410 CONSOLE 24,1,0,40:COLOR ,,1,5:CLS
420 PRINT [6,2]TAB(19)"BASE";
430 ENDPROC
440 REM ***********************************
450 DEF PROC"jeu"
460 U=0:X=20:Y=4:H=INT(8*RND(1))
470 REPEAT
480 : PROC"module"(0):A=1
490 : KEY X$
500 : IF X$="?" U=U+2:A=0
510 : ELSIF X$="X" H=H+1
520 : ELSIF X$="Z" H=H-1
530 : ENDIF
540 : U=U-1
550 : X=(X+H) MAX 2 MIN 77
560 : IF X=2 OR X=77 H=0:ENDIF
570 : Y=(Y-U) MAX 2 MIN 47
580 : IF Y=2 U=0:ENDIF
590 : PROC"module"(1):WAIT 10

600 : E=(X>38)*(X<46)*(45-Y<D)
610 : F=(Y>=H(X-1))+(Y>=H(X+1))+(Y>=H(X)
)
620 UNTIL E<>0 OR F<>0
630 ENDPROC
640 REM ***********************************
650 DEF PROC"resultat"
660 CONSOLE:COLOR ,,2,4
670 IF E<>0 DO
680 : IF U<=-3 DO
690 :   BEEP 1,63000,100
700 :   PRINT"▯PAR VOTRE FAUTE,LA BASE E
ST DETRUITE !!"
710 : ELSE DO
720 :   BEEP 10
730 :   PRINT"▯SUPERBE ARRIVEE !!"
740 : ENDIF
750 ELSIF F<>0 DO
760 : IF U<=-3 DO
770 :   BEEP 1,60000,50
780 :   PRINT"▯APRES CE CRASH MEMORABLE,
LA BASE  "
790 :   PRINT"EPARGNEE VOUS REND LES HON
NEURS..."
800 : ELSE DO
810 :   BEEP 2,3000,20
820 :   PRINT"▯ARRIVEE A ";ABS(40-X)/2;"
 KM DE LA BASE"
830 : ENDIF
840 ENDIF
850 ENDPROC
860 REM ***********************************
870 DEF PROC"module"(M)
880 GRAPH X-1,Y
890 DRAW M,X+1,Y,6:DRAW M,X,Y-1,6
900 IF A=0 DRAW M,X,Y+1,6:ENDIF
910 ENDPROC
920 REM ***********************************
930 DEF PROC"encore"
940 PRINT"▯▯  Voulez vous recommencer (O
/N) ?";
950 REPEAT
960 : INKEY R$
970 UNTIL R$="O" OR R$="N"
980 ENDPROC
990 REM ***********************************
```

OPERATEURS MATHEMATIQUES

Les divers opérateurs arithmétiques et mathématiques,opérant sur des nombres ou des chaînes de caractères sont indiqués dans le tableau ci-dessous,avec leur ordre de priorité :

Par ordre de priorité croissante : →

MAX MIN	OR XOR	AND	Tests d'égalité ou non	+ - (soustraction)	* / DIV MODULO	↑	- (changement de signe)	( )	Fonctions

Dans chaque colonne,les opérateurs ont la même priorité:ils seront donc traités selon leur ordre d'apparition dans l'expression en cours d'évaluation.Il est bien entendu,possible de modifier la priorité des divers opérateurs en utilisant des parenthèses.

Quelques remarques :

Parmi ces opérateurs,seuls MAX,MIN,+,les tests et certaines fonctions travaillent sur des chaînes de caractères.

AND,OR,XOR convertissent leurs arguments en entiers signés sur 4 octets,puis font l'opération octet par octet,et reconvertissent le résultat en virgule flottante (format utilisé par le S-Basic).

Le lecteur remarquera que le K-Basic ,suivant en celà les conventions communément admises,donne à ↑ une priorité inférieure à - ,opérateur de changement de signe.Cela signifie,par exemple ,que - 5 ↑ 2 vaudra 25 en K-Basic , alors que le S-Basic donne - 25 .Il y a là une légère incompatibilité avec le S-Basic mais elle me semble si peu gênante que je n'ai pas hésité à me conformer aux conventions habituelles.

On pourra aussi remarquer que le K-Basic accepte l'élévation de nombres négatifs à des puissances entières mais on n'oubliera pas que les calculs sont quand même effectués par les logarithmes et exponentielles ce qui peut conduire à quelques imprécisions dues aux calculs en virgule flottante.

EVALUATION DES CONDITIONS LOGIQUES

Les tests de conditions logiques sont effectués par le K-Basic dans les instructions IF,ELSIF,UNTIL et WHILE

Une condition "élémentaire" (du genre A>B) est considérée par le Basic comme une variable valant 0 si elle est fausse et -I si elle est vraie.Cette variable pourra être utilisée dans un calcul à condition de figurer entre parenthèses : exemple : A=-2*(X>0) +2*(X<=0) vaudra -2 si X est négatif ou nul et +2 si X est strictement positif.

Dans les conditions "composées" ,c'est à dire contenant ET (* en S-Basic ou AND en K-Basic) ,OU (+ en S-Basic ou OR en K-Basic) ou le OU exclusif (XOR en K-Basic), les calculs seront faits en considérant chaque condition élémentaire comme une variable valant 0 ou -I .Lorsque le résultat obtenu est 0,la condition composée est décrétée fausse par le Basic.Sinon,elle sera considérée comme vraie.
Il en résulte qu'il est tout à fait possible d'écrire :

```
IF N THEN ..... (si N=0 ,condition fausse,sinon vraie)
UNTIL 0         (Boucle "infinie")
```

Remarque : Il sera préférable,dans l'adaptation au K-Basic de programmes écrits en S-Basic,de remplacer + par OR et * par AND (dans les tests !).En effet,comme de nombreux basics,le S-Basic peut se comporter de façon semble-t-il anormale lors de l'évaluation de conditions composées.
Par exemple,soit la ligne de programme :
IF (A<>0) + (B<>0)*(B<>I) THEN ...(I) ELSE ...(2)
Supposons que A=I et B=2 .La condition vaudra,pour le S-Basic :
-I + (-I)*(-I) = 0 donc les instructions (2) seront éxécutées.
Or,la condition est : (A<>0) OU((B<>0) ET (B<>I)) ,car on n'oubliera pas que * a priorité sur + lors des calculs .Avec A=I et B=2 ,la condition est vraie et le Basic devrait effectuer (I) .

En K-Basic,on écrira (sans parenthèses à cause des priorités des opérateurs) : A<>0 OR B<>0 AND B<>I .
Avec A=I et B=2,cela donnera -I OR -I AND -I .Comme AND et OR convertissent leurs arguments en entiers sur 4 octets,nous aurons :
FFFFFFFF OR FFFFFFFF AND FFFFFFFF . L'opération AND est effectuée la première et donne : FFFFFFFF AND FFFFFFFF = FFFFFFFF puis OR est effectuée et donne FFFFFFFF soit -I c'est à dire : Condition vraie .

Avant de terminer,il est peut être bon de rappeler au lecteur qu'en langage machine, les opérations AND,OR,XOR,se font "bit à bit" en suivant les tables suivantes :

OR	0	I
0	0	I
I	I	I

XOR	0	I
0	0	I
I	I	0

AND	0	I
0	0	0
I	0	I

Par suite FF (hexa)=IIIIIIII (binaire)donnera bien FF AND FF=FF et par suite,comme nous venons de la dire FFFFFFFF AND FFFFFFFF = FFFFFFFF .

Nous avons vu que le K-Basic possédait trois sortes de boucles :

I/ FOR .... NEXT dont la syntaxe est la même qu'en S-Basic :

```
FOR indice=valeur initiale TO valeur finale STEP pas .....
.....
NEXT indice
```

- indice est le nom d'une variable numérique comme J ou A(2,3)
- Valeur initiale et valeur finale sont des valeurs numériques calculées une fois pour toutes ,ce qui est important surtout pour la valeur finale.
- STEP pas est facultatif.En son absence;pas vaudra I .

Remarque : une boucle FOR-NEXT est toujours parcourue au moins une fois.

2/ REPEAT .. UNTIL de syntaxe : REPEAT:(instructions):UNTIL condition
Cette boucle effectue la partie instructions jusqu'à ce que la condition soit vérifiée.On trouvera ci-dessous son organigramme.

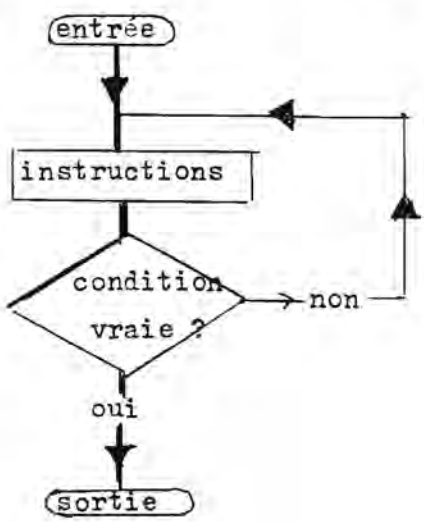
Remarque : une boucle REPEAT-UNTIL est toujours parcourue au moins une fois.

3/ WHILE .. WEND de syntaxe : WHILE condition DO:(instructions):WEND
Cette boucle effectue la partie instructions tant que la condition suivant WHILE est satisfaite.

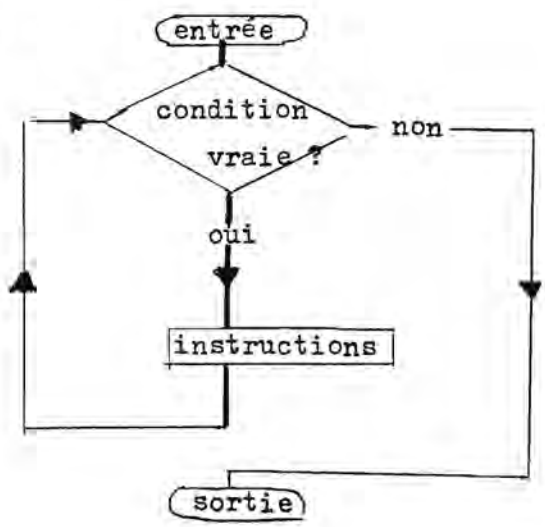
Remarque : une telle boucle peut très bien ne pas être parcourue.En effet si lors du premier passage à WHILE,la condition n'est pas satisfaite,le Basic sautera derrière WEND.

Organigrammes de REPEAT-UNTIL et WHILE - WEND

REPEAT :



WHILE :



Remarques : Pour ces trois boucles,les niveaux d'imbrication sont illimités,dans la mesure de la mémoire disponible !

Il est déconseillé de sortir ou d'entrer dans une boucle REPEAT - UNTIL ou WHILE - WEND par un GOTO ,ce qui est contraire à l'esprit de la programmation structurée et provoquera tôt ou tard l'arrêt du programme sur un message d'erreur.

Il en va de même pour les boucles FOR - NEXT .Cependant,le S-Basic autorisant la sortie d'une ou plusieurs boucles FOR - NEXT situées dans un sous-programme,par RETURN,sans avoir parcouru la totalité de la ou des boucles,il sera possible de faire de même en K-Basic,mais en utilisant une instruction supplémentaire : EXIT .EXIT ou EXIT I permet d'éliminer de la "pile Basic" une boucle alors que EXIT n permet d'éliminer n boucles.Il faudra ensuite que le programmeur réalise lui même la sortie de boucle par un GOTO ou un RETURN.

```
10 CLR:CLS:MODE 1
20 REPEAT
30 :  PRINT"■■FACTEURS PREMIERS"
40 :  PRINT STRING$(17,"=")
50 :  REPEAT
60 :    INPUT"■N entre 2 ET 2E+9:";N
70 :  UNTIL N>1 AND N<2E+09 AND N=INT(N)
80 :  PRINT
90 :  PROC"Facteurs"(2)
100 : PROC"Facteurs"(3)
110 : I=5:A=INT(1+SQR(N))
120 : WHILE I<=A DO
130 :   PROC"Facteurs"(I):I=I+2
140 :   PROC"Facteurs"(I):I=I+4
150 : WEND
160 : IF N>1 DO
170 :   PRINT N;"↑1":PRINT
175 : ENDIF
180 UNTIL 0
190 REM ********************************
200 DEF PROC"Facteurs"(I)
210 S=0
220 WHILE N MODULO I=0 DO
230 : N=N/I:S=S+1
240 WEND
250 IF S>0 DO
260 : PRINT I;"↑";S:A=INT(1+SQR(N))
270 ENDIF
280 ENDPROC
290 REM ********************************
```

LES DEUX TYPES DE STRUCTURES IF ... THEN ... ELSE

En MODE 0 : On utilisera la structure classique IF ... THEN ... ELSE ... écrite sur une seule ligne, la partie ELSE ... étant facultative.
La syntaxe est la suivante :
IF condition THEN instructions I ELSE instructions 2
La condition est d'abord évaluée .Si elle donne un résultat nul ,elle sera considérée comme fausse et le programme se poursuivra après ELSE (ou à la ligne suivante en son absence).Par contre,si la condition donne un nombre non nul,elle sera considérée comme vraie et la partie "instructions I" sera éxécutée.

Remarques :THEN GOTO n° de ligne peut s'écrire THEN n° de ligne
GOTO n° de ligne
ELSE GOTO n° de ligne peut s'écrire ELSE n° de ligne
Mais ,le n° de ligne doit être donné explicitement en non par une expression à calculer.
Dans tous les autres cas THEN est facultatif .Par exemple,au lieu de IF A=B THEN A=C on pourra écrire : IF A=B A=C (ne pas oublier de placer un espace entre B et A !!).

En MODE I : Il est alors possible d'utiliser des structures de test plus complexes et s'étendant sur plusieurs lignes (et même sur la totalité du programme).La syntaxe est alors la suivante :

```
IF Condition I  THEN instructions I
ELSIF Condition 2 THEN instructions 2
ELSIF Condition 3 THEN instructions 3
.....
ELSIF Condition n THEN instructions n
ELSE instructions n'
ENDIF
```

Lors de l'éxécution du programme,la condition I est d'abord évaluée; Si elle est vraie,instructions I sont éxécutées puis le programme saute derrière le ENDIF.
Si elle est fausse,c'est la condition 2 qui sera évaluée.
Si elle est vraie,les instructions 2 sont éxécutées puis on passe derrière ENDIF.
Sinon,condition 3 est évaluée .... etc ...
Si toutes les conditions sont fausses,les instructions n' seront éxécutées.

Remarques : Les instructions ELSIF et (ou) ELSE sont facultatives.

Dans les "instructions i" ,il est aussi possible d'utiliser des structures IF....ENDIF ,ce qui signifie que des imbrications de niveau quelconque sont permises (il ne faudra pas cependant oublier un ou plusieurs ENDIF).

La structure présentée précedemment peut aussi s'écrire :

```
IF Condition I DO
   instructions I
ELSIF Condition 2 DO
   instructions 2
.....
ELSE DO
   instructions n'
ENDIF
```

ce qui rend le programme plus lisible,en occupant cependant,un peu plus de mémoire.

La structure CASE....ENDCASE

Cette structure,empruntée au PASCAL est analogue à la structure IF ... ENDIF mais ne peut tester que des égalités.A cause du mode de fonctionnement interne du Basic, elle n'a pas en K-Basic,la simplicité et l'élégance qu'elle possède en PASCAL.Sa syntaxe est la suivante :

```
CASE  expression OF
WHEN liste I  DO : instructions I
WHEN liste 2  DO : instructions 2
.....................................
OTHERWISE : instructions n
ENDCASE
```

Le Basic calcule,une fois pour toutes,"expression" puis compare sa valeur avec les valeurs de la "liste I" (les diverses valeurs doivent être séparées par des virgules).Si on trouve une égalité,les instructions I sont éxécutées puis il y a ut du programme derrière ENDCASE .Sinon,le Basic recommence les comparaisons avec les éléments de la "liste 2" .....
Si le Basic ne trouve aucune égalité avec les éléments des diverses listes,l'exécution du programme se poursuivra après OTHERWISE .

Remarques : Après OF ,le Basic attend soit : suivi de WHEN soit la fin de la ligne. Dans ce cas,la ligne suivante doit commencer par WHEN ou par : suivi d'espaces suivi de WHEN (ceci pour permettre l'indentation du programme).
Les "listes" doivent être suivies de DO: ou de DO suivi de la fin de la ligne.
OTHERWISE : instructions n peut très bien être absent .
OTHERWISE doit être suivi de : ou de la fin de la ligne.
Les structures CASE .... ENDCASE peuvent très bien être imbriquées .

Contrairement à la plupart des Basics,le K-Basic possède deux structures inspirées du PASCAL et généralisant la notion de sous-programme : les Procédures et les Fonctions.

Pour présenter les choses simplement,disons qu'une Procédure produit un effet comme une modification de variable ou le tracé d'un graphe alors qu'une fonction produit un résultat numérique ou alphanumérique.(Cette présentation est très simplifiée !!).

La différence entre Procédures et Fonctions est bien marquée dans l'appel de ces structures et dans la façon de revenir au programme principal.

- Une procédure est appelée par PROC"nom"(argument,argument,....).Le nom de la Procédure est écrit entre guillemets (obligatoires tous les deux),pour éviter que le Basic code certains groupes de lettres de ce nom.On trouve ensuite,séparés par des virgules et placés entre parenthèses,les arguments numériques ou alphanumériques nécéssaires au bon fonctionnement de la Procédure.On remarquera qu'il peut très bien ne pas y avoir d'arguments et que ceux ci peuvent être quelconques (éléments de tableaux ou résultats d'une fonction).Le rôle des arguments sera explicité un peu plus loin.

- Par contre,une Fonction est appelée par FN"nom"(argument,argument,....).Une telle expression est cpnsidérée par le Basic comme une variable numérique ou alphanumérique et peut donc être traitée comme telle.

- Les Procédures seront définies ainsi :

```
DEF PROC"nom"(parametre,parametre,....)
LOCAL paramètre I,paramètre 2,.....
instructions
ENDPROC
```

Le nom de la Procédure doit toujours être entre guillemets ,obligatoires.Les paramètres (il peut ne pas y en avoir !) sont placés entre parenthèses et sont séparés par des virgules.Ce sont des NOMS de variables numériques,alphanumériques ou d'éléments de tableaux.

On trouve ensuite,derrière LOCAL,le NOM des variables locales (terme explicité plus loin) à la Procédure (il peut ne pas y en avoir).Il faudra toujours écrire LOCAL juste après DEF PROC"nom"(.....) ou à la ligne suivante mais surtout,jamais dans une boucle FOR,REPEAT ou WHILE qui serait située dans la définition de la procédure.

ENDPROC est le "RETURN" d'une Procedure.

- Pour les Fonctions,la syntaxe sera analogue :

DEF FN"nom"(paramètre,paramètre,....)
LOCAL paramètre I,paramètre 2,....
instructions
RESULT= expression

La seule grande différence avec les Procédures se fait pour le retour par RESULT= suivi d'une expression qui donne le résultat retourné par la Fonction.

Remarque : Le S-Basic possède ce type de structure,mais sous une forme beaucoup plus simple.Pour assurer le maximum de compatibilité entre S et K-Basic, dans ce dernier,la syntaxe : DEF FN"nom"(.....)= expression sera acceptée et considérée comme : DEF FN"nom"(....):RESULT=expression.Il n'y a cependant pas compatibilité totale ne serait ce que parce que le nom de la fonction doit ici,être entre guillemets (Voir aussi la remarque suivante).

Il est conseillé d'écrire les définitions des Procédures et Fonctions en fin de programme.Pour accélérer le fonctionnement des PROC et FN,un catalogue contenant leurs noms et leurs positions dans le programme est établi lors du "RUN".On trouvera des renseignements sur le catalogue dans le chapitre "Quelques indications techniques".

Remarques : Pour accélérer la construction du catalogue,seuls sont testés les débuts de lignes de programme.Cela signifie que les instructions DEF FN ou DEF PROC ne seront prises en compte que si elles suivent immédiatement le numéro de ligne.

Toute modification du programme effacera le catalogue.

Les arguments et paramètres utilisés lors de l'appel et de la définition d'une FN ou PROC doivent être en même nombre et ,pour les arguments de même rang,de même type (numérique ou alphanumérique).

Après cette description,nous allons voir,pour le lecteur les rencontrant pour la première fois,l'utilité de ces structures.

PARAMETRES ET VARIABLES LOCALES

Supposons que nous voulions un sous-programme traçant une ligne de 5 étoiles sur l'écran.Nous écrirons : I00 FOR I=I TO 5 :PRINT "*"; : NEXT I : RETURN et pour l'utiliser,nous ferons GOSUB I00 .
Or,il est possible que nous ayons besoin,un peu plus tard,de tracer une ligne de 6 étoiles.Plutôt que d'écrire un nouveau sous-programme,il sera plus simple de faire : I00 FOR I=I TO L : PRINT "*"; : NEXT I : RETURN et pour tracer 5 étoiles, on écrira : L=5:GOSUB I00 et pour 6 étoiles : L=6:GOSUB I00 .

Une telle syntaxe est un peu lourde et il serait plus simple d'incorporer L à l'appel du sous-programme c'est à dire de lui "passer" la valeur de L .Pour ce faire,on utilisera une procédure :

```
I00  DEF PROC"trace"(L)
II0  FOR I=I TO L : PRINT "*"; : NEXT I
I20  ENDPROC
```

Pour tracer 5 étoiles,il suffira d'utiliser : PROC"trace"(5) et pour 6 étoiles : PROC"trace"(6) .

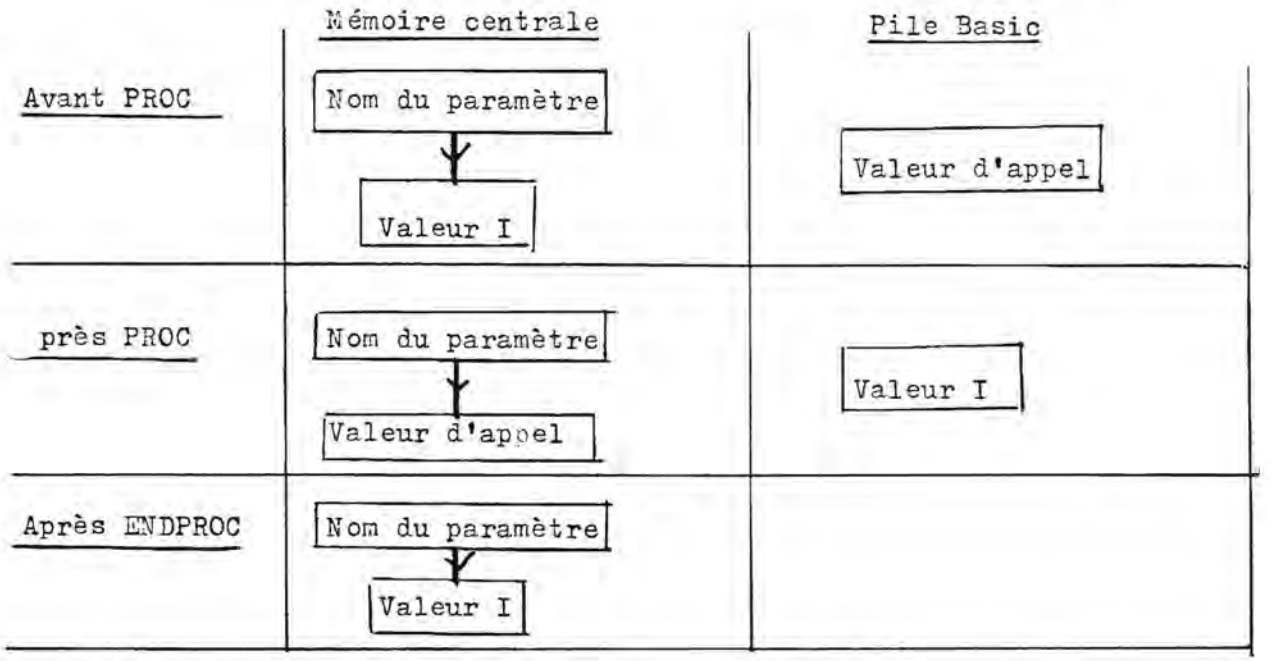
A ce stade,le lecteur comprend certainement comment les choses se passent : Le Basic affecte au "paramètre" L utilisé dans l'en-tête de la procédure (ligne I00), la "valeur d'appel" 5 puis effectue les lignes II0 et I20.On dit qu'il y a là,passage d'un paramètre "par valeur".En fait,cette description masque un autre point essentiel.Ajoutons à notre sous programme la ligne II5 PRINT L et faisons aussi : I0 L=I:PROC"trace"(5):PRINT L : END
Ce nouveau programme trace 5 étoiles puis imprime 5 et enfin I .Autrement dit,dans la procédure,L vaut 5 comme nous l'avions vu précedemment,mais au retour ,à la fin de la ligne I0,L reprend sa valeur initiale qui était I .Cela signifie qu'avant de donner à L la valeur 5,le Basic sauve (dans la "pile Basic") la valeur initiale de L,ce qui permet de lui redonner,lors du ENDPROC,sa valeur initiale.La variable L, utilisée dans la Procédure,ne voit donc pas sa valeur perturbée par la Procédure. Tout se passe comme si le Basic connaissait deux variables L ,l'une utilisée dans la procédure (et uniquement là) et l'autre utilisée en dehors de la Procédure (cependant,dans la Procédure,cette variable n'est plus accessible).On exprime tout cela en disant que L est une variable locale à la Procédure.

En résumé,lors de l'utilisation d'une Procédure (ou d'une fonction),tous les paramètres voient leurs valeurs stockées dans la Pile Basic,puis les valeurs d'appel sont calculées les unes après les autres et enfin,ces valeurs sont affectées aux paramètres.

Supposons maintenant que nous voulions tracer 5 étoiles sur l'écran,mais que notre programme utilise I comme variable essentielle.Que se passera t-il ? .Pour le voir,faisons : I0 I=2:PROC"trace"(5):PRINT I : END .Ce programme va donner I=6 et non 2.En effet,au sortir de la boucle de la ligne II0,I vaut 6 et donc sa valeur initiale (2) a été perturbée par la Procédure.Si nous voulons conserver I ,

il faudrait qu'elle soit traitée comme l'a été L,c'est à dire que sa valeur à l'entrée de la Procédure,soit conservée pour être restituée lors du ENDPROC.C'est tout à fait possible grâce à l'ordre LOCAL .On ajoutera donc : II5 LOCAL I . Cette fois notre programme redonnera 2 comme valeur de I .On remarquera qu'avec LOCAL,il n'y a pas passage de valeurs,donc à l'entrée dans la Procédure,I garde sa valeur initiale,ce qui peut être utile dans certains cas.

Remarque : Tout ce qui vient d'être vu est aussi valable pour les Fonctions.
Voici,en résumé ce qui se passe lors d'une PROC ou FN :



Les procédures et fonctions sont aussi très utiles pour faire de la "Récursivité", c'est à dire faire fonctionner des fonctions (ou procédures) s'appelant elles-mêmes.

Factorielle :

```
10 MODE 1:CLS
20 PRINT"Factorielles"
30 PRINT"-----------"
40 REPEAT
50 : REPEAT
60 :   INPUT "N=";N
70 : UNTIL N>=0 AND N=INT(N)
80 : PRINT N;"!=";FN"Fact"(N):PRINT
90 UNTIL 0
100 REM ****************************
110 DEF FN"Fact"(P)
120 LOCAL R
130 IF P<2 R=1
140 ELSE R=P*FN"Fact"(P-1)
150 ENDIF
160 RESULT=R
```

Coefficients du binôme :

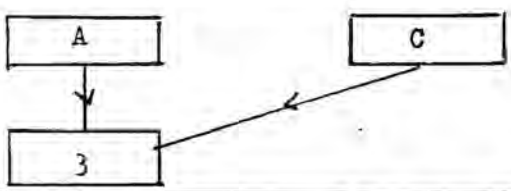
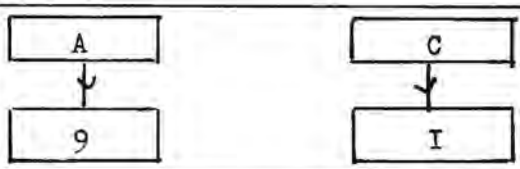
```
10 CLS:MODE 1
20 PRINT"Coefficients du binome"
30 PRINT"-----------------"
40 REPEAT
50 :  REPEAT
60 :    INPUT"N=";N
70 :  UNTIL N>=0 AND N=INT(N)
80 :  REPEAT
90 :    INPUT"P=";P
100 : UNTIL P>=0 AND P=INT(P)
110 : PRINT"C(";N;",";P;")=";
120 : PRINT FN"Bin"(N,P)
130 UNTIL 0
140 REM ************************
150 DEF FN"Bin"(N,P)
160 LOCAL R
170 IF P>N R=0
180 ELSIF P=0 OR P=N R=1
190 ELSE R=FN"Bin"(N-1,P)+FN"Bin"(N-1,P-1)
200 ENDIF
210 RESULT=R
```

PASSAGE DE PARAMETRES PAR REFERENCE

Dans la plupart des cas,le passage des paramètres par valeurs est suffisant.Cependant,grâce au mode de stockage,hérité du S-Basic, des noms de variables et de leurs valeurs dans deux zones différentes de la mémoire,il a été possible de donner au K-Basic le "passage par référence".Voyons tout de suite ce que cela signifie,sur un exemple très simple:

```
5  C=I
IO A=3:PROC"CARRE"(#A):PRINT A:END
IOO DEF PROC"CARRE"(#C)
IIO C=C*C
I2O ENDPROC
```

Le symbole # signifie:passage par référence et doit toujours être suivi du nom d'une variable.Pour le moment,nous la supposerons non dimensionnée.Le programme nous donne 9 .Voyons sur un diagramme ce qui s'est passé en supposant donc C=I et A=3 .

	MEMOIRE	Pile BASIC
Avant PROC		
Après PROC		On conserve dans la pile le n° d'ordre en zone 2 de la cellule valeur de C
Après ligne IIO		idem
Après ENDPROC		

En résumé,passage par référence signifie que paramètre (C) et variable d'appel(A) vont partager la même "zone valeur" et donc que toute transformation sur C va provoquer la même transformation sur A.

On remarquera que C se comporte presque comme une variable locale:ici,on ne conserve dans la pile Basic que le n° d'ordre de sa cellule valeur en zone 2 donc,si la procédure a la possibilité d'accéder à la valeur initiale de C,celle ci sera modifiée en sortant de la procédure.Par exemple,si,à la ligne IO on remplace A par C ,le programme donnera C=9 bien que la valeur initiale de C soit 3 .

Remarques : Comme il a déjà été dit,à la page 9,dans le K-Basic,une variable qui n'a encore subi aucune affectation ne possède aucune zone valeur en zone 2 .Cela se produit ,par exemple,pour A et le tableau T dans le programme: IO DIM T(5) : PRINT A : C=5+A:END .
Cette façon de procéder permet d'économiser un peu de mémoire (éventuellement beaucoup dans des programmes de traitement de matrices ayant beaucoup d'éléments nuls).

La présentation précédente du passage par référence ne peut fonctionner correctement que si A (variable d'appel) possède effectivement une zone valeur.Pour s'en convaincre,on comparera les deux programmes :

```
10 PROC"essai"(#A):PRINT A:END
20 DEF PROC"essai"(#C):C=1:ENDPROC
```

```
10 A=0:PROC"essai"(#A):PRINT A:END
20 DEF PROC"essai"(#C):C=1:ENDPROC
```

D'autre part,le fait que deux variables de noms différents (en général) partagent la même zone de valeurs (ce qui,en Basic,ne devrait jamais se produire) peut conduire à ce qu'on appelle des "effets de bord" c'est à dire à des modifications des variables non voulues par le programmeur.

Le passage par référence est très utile lorsqu'on veut obtenir plusieurs résultats d'un sous-programme.En voici un exemple simple : réduction d'une fraction :

```
10 REPEAT
20 : INPUT"N,P=";N,P
30 : PROC"pgcd"(#N,#P)
40 : PRINT"Reduite:";N;"/";P
50 UNTIL 0
60 REM **********************
70 DEF PROC"pgcd"(#A,#B)
80 LOCAL C,D,E
90 D=A:E=B
100 REPEAT
110 : C=B:B=A MODULO B:A=C
120 UNTIL B=0
130 A=D/C:B=E/C
140 ENDPROC
150 REM **********************
```

```
12/ 144
Reduite: 1/ 12

123456/ 654321
Reduite: 41152/ 218107
```

On remarquera que la Procédure produit deux résultats : A et B .

Cas des variables dimensionnées :

```
10 REPEAT
20 : INPUT"@Nb de colonnes de A:";P
30 UNTIL P>0 AND P=INT(P)
40 REPEAT
50 : INPUT"@Nb de lignes   de A:";N
60 UNTIL N>0 AND N=INT(N)
70 DIM A(P,N),C(P,N)
80 NULL A,C
90 PROC"entree"(#A(0,0),P,N,"A")
100 FOR J=1 TO N
110 : FOR I=1 TO P
120 :   PRINT A(I,J),
130 : NEXT I
140 : PRINT
150 NEXT J
160 END
170 REM ********************************
180 DEF PROC"entree"(#C(0,0),C,L,A$)
190 PRINT"@Entrez la matrice "A$
200 FOR J=1 TO L
210 : PRINT"@Ligne N°"J
220 : FOR I=1 TO C
230 :   INPUT"Coefficient:";C(I,J)
240 : NEXT I
250 NEXT J
260 ENDPROC
270 REM ********************************
```

Dans le programme ci-dessus,il y aura passage par référence de tous les éléments du tableau.Autrement dit,les lignes 90 et I80 signifient que pendant la procédure, A(I,J) et C(I,J) vont partager la même zone de valeurs et ceci pour toutes les valeurs de I et de J .

Remarques : L'instruction NULL A est essentielle car elle affecte à chaque élément du tableau A une zone de valeur .On pourra enlever la ligne 80 et voir ce qui se passe.

Les indices 0,0 des lignes 90 et I80 peuvent être remplacés par deux indices quelconques,mais inférieurs ou égaux aux dimensions maximales P et N.

Le tableau C utilisé dans la procédure doit avoir été dimensionné avant l'appel de cette procédure.

Avant de terminer, voyons un dernier exemple mêlant la Récursivité et le passage de paramètres par référence.

Calcul des Nombres de Bernoulli :

Ces nombres, bien connus des étudiants en mathématiques, seront calculés ici sous forme de fractions irréductibles, grâce à la propriété suivante :
Soit F(n,k) la suite double définie par $F(n,0)=(-1)^n/(n+1)$ et pour $k \geqslant 1$ par :
$F(n,k)=n.F(n,k-1) + (n+1).F(n+1,k-1)$. Alors $B(k)=F(0,k)$ où B(k) est le k-ième Nombre de Bernoulli. Nous représenterons F(n,k) sous la forme (irréductible) : A(n,k)/B(n,k) . Voici le programme et ses résultats (On ne peut pas aller au delà des 21 premiers nombres à cause de la capacité limitée des calculs du Basic).

```
10 CLR:CLS:MODE 1
20 D=21:DIM A(D,D),B(D,D)
30 FOR K=0 TO D
40 : PROC"Bern"(0,K)
50 : PRINT "B(";K;")=";
60 : PRINT A(0,K);"/";B(0,K)
70 NEXT K
80 END
90 REM ****************************
100 DEF PROC"Bern"(N,K)
110 IF B(N,K)=0 DO
120 : IF K=0 DO
130 :   A(N,0)=COS(N*π):B(N,0)=N+1
140 : ELSE DO
150 :   PROC"Bern"(N,K-1)
160 :   PROC"Bern"(N+1,K-1)
170 :   R=N:S=B(N,K-1)
180 :   PROC"Pgcd"(#R,#S)
190 :   U=R*A(N,K-1):V=S
200 :   R=N+1:S=B(N+1,K-1)
210 :   PROC"Pgcd"(#R,#S)
220 :   W=R*A(N+1,K-1):R=V
230 :   PROC"Pgcd"(#R,#S)
240 :   A=U*S+R*W:S=R*S*C:R=A
250 :   PROC"Pgcd"(#R,#S)
260 :   A(N,K)=R:B(N,K)=S
270 : ENDIF
280 ENDIF
290 ENDPROC
300 REM ****************************
310 DEF PROC"Pgcd"(#A,#B)
320 LOCAL U,V
330 U=A:V=B
340 IF U=0 LET V=1
350 ELSE DO
360 : REPEAT
370 :   C=B:B=A MODULO B:A=C
380 : UNTIL B=0
390 : U=U/C:V=V/C
400 : IF V<0 U=-U:V=-V:ENDIF
410 ENDIF
420 A=U:B=V
430 ENDPROC
```

```
B( 0)   = 1/ 1
B( 1)   =-1/ 2
B( 2)   = 1/ 6
B( 3)   = 0/ 1
B( 4)   =-1/ 30
B( 5)   = 0/ 1
B( 6)   = 1/ 42
B( 7)   = 0/ 1
B( 8)   =-1/ 30
B( 9)   = 0/ 1
B( 10)  = 5/ 66
B( 11)  = 0/ 1
B( 12)  =-691/ 2730
B( 13)  = 0/ 1
B( 14)  = 7/ 6
B( 15)  = 0/ 1
B( 16)  =-3617/ 510
B( 17)  = 0/ 1
B( 18)  = 43867/ 798
B( 19)  = 0/ 1
B( 20)  =-174611/ 330
B( 21)  = 0/ 1
```

L'objet de la Procédure "Bern"(N,K) est de calculer A(N,K) et B(N,K). Si cela n'a pas déjà été fait, B(N,K)=0 d'où la ligne 110. La ligne 130 calcule A(N,0) et B(N,0) alors que 150 et 160 calculent F(n,k-1) et F(n+1,k-1). Les lignes 170 à 250 réduisent la fraction résultat.

On remarquera que C n'est pas déclarée comme variable locale de PROC"Pgcd" car sa valeur est utilisée en dehors de la Procédure à la ligne 240.

CODE	: 80
INSTRUCTION	: GOTO

GOTO n effectue un saut inconditionnel à la ligne de numéro n .
n peut être donné directement ou par une expression que le Basic calculera. Dans ce dernier cas,il est conseillé de prendre l'habitude d'écrire cette expression entre parenthèses (En effet,si l'expression commence par un chiffre,le Basic prendra ce chiffre comme le début d'un numéro de ligne écrit explicitement,ce qui provoquera une erreur lors du fonctionnement du programme).

Remarques : Si n vaut zéro,il y aura saut au début du programme
Dans le cas où n est donné explicitement,il est possible d'écrire THEN n ou ELSE n au lieu de THEN GOTO n ou ELSE GOTO n .
GOTO est aussi utilisé dans ON ... GOTO (voir ON)

CODE	: 8I
INSTRUCTION	: GOSUB

GOSUB n envoie le contrôle au sous programme commençant à la ligne n.
Dans cette version Basic,il n'y a aucune limite au nombre de sous programmes imbriqués,sauf celle imposée par la taille de la mémoire disponible (chaque appel de sous programmes consomme 7 octets qui seront récupérés lors du RETURN correspondant).
Lorsque ces 7 octets (servant au RETURN) ont été stockés dans la Pile Basic, il y a saut au programme effectuant le mot clé GOTO .Ainsi,ce qui a été vu précédemment pour l'écriture de n est-il toujours valable.

Remarque : GOSUB est aussi utilisé dans ON ... GOSUB (voir ON)

CODE	: 82
INSTRUCTION	: DISP

DISP n affiche en programme,la ligne de numéro n . n peut être donné explicitement ou par une expression.

Exemple : 10 FOR I=2 TO 40
DISP 10 va afficher : FOR I=2 TO 40

Remarques: Après l'affichage,il n'y a pas saut à la ligne.
DISP 10,20,30 affichera les lignes 10 20 et 30 les unes à la suite des autres.

CODE : 83
INSTRUCTION : RUN

RUN lance l'éxécution d'un programme après avoir mis à zéro toutes les variables. L'éxécution se fait à partir de la première ligne du programme.

RUN n lance l'éxécution à partir de la ligne n sans effacer les variables.Tout ce qui est dit sur n dans GOTO est valable ici.En particulier RUN 0 est équivalent à RUN mais sans effacer les variables.

Remarques : C'est lors de l'éxécution de RUN que se construit le catalogue des PROC et des FN.La pile Basic est ensuite remise à zéro.

CODE : 84
INSTRUCTION : RETURN

RETURN est l'instruction de fin d'un sous programme.Le retour se fait sur l'instruction suivant le GOSUB qui avait donné le contrôle au sous programme.

Remarque : Il est possible d'écrire RETURN n (comme dans le S-Basic bien que le manuel n'en parle pas .....) où n est une expression.Dans ce cas,le retour se fera à la ligne n comme si l'on avait écrit GOTO n .Tout ce qui a été vu dans GOTO au sujet de n est encore valable.

CODE : 85
INSTRUCTION : RESTORE

RESTORE remet le pointeur de lecture des DATA au début du programme.

RESTORE n le remet au début de la ligne de numéro n (n peut être donné explicitement ou sous forme d'expression).

CODE : 86
INSTRUCTION : RESUME

RESUME n'a pas vu son fonctionnement modifié.On se reportera au manuel du MZ 700

CODE : 87
INSTRUCTION : LIST

LIST n'a pas été modifié.Se reporter au manuel de l'ordinateur.

Remarque : Juste après un listing du programme,il est possible de le reprendre là où il s'était arrêté par LIST . ou LIST . - n° (cette possibilité existe dans le S-Basic)

Après une erreur ,LIST . affiche la ligne où il y a eu erreur.

CODE	: 88
INSTRUCTION	: ENDPROC

ENDPROC est l'instruction de fin d'une PROCédure.Le retour se fera sur l'instruction suivant l'appel PROC"...."(.....) qui avait lancé l'éxécution de la procédure.

CODE	: 89
INSTRUCTION	: DELETE

DELETE n'a pas été modifié.Se reporter au manuel de l'ordinateur.

Remarque : Comme pour LIST,il est possible de reprendre une destruction partielle du programme par DELETE . ou DELETE . - n° .Il faut cependant utiliser cette technique avec précautions et surtout sans avoir utilisé LIST entre les deux opérations de destruction.

CODE	: 8A
INSTRUCTION	: RENUM

RENUM n'a pas été modifié.Se reporter au manuel de l'ordinateur.

CODE	: 8B
INSTRUCTION	: AUTO

AUTO n'a pas été modifié.

Remarque : Lors de l'entrée de lignes en mode AUTO,si l'on arrive sur une ligne déjà existante,celle ci s'affiche avec le curseur juste après le numéro. On peut donc la modifier ou la remplacer par une autre ligne ou bien la conserver en appuyant simplement sur RETURN.
Un arrêt du mode AUTO se produit par Shift Break. On peut alors reprendre le mode AUTO par AUTO . sans indiquer d'autre paramètre.

CODE	: 8C
INSTRUCTION	: ENDCASE

ENDCASE indique simplement la fin de la structure CASE...OF...WHEN.....ENDCASE

CODE	: 8D
INSTRUCTION	: FOR

FOR indique le début d'une boucle FOR....NEXT.Il doit toujours être suivi du nom d'une variable numérique (indice de boucle) : exemple : FOR A=.... ou FOR A(0,3,5,7)=.....

Dans cette version Basic,il n'y a aucune limite au nombre de boucles FOR... NEXT imbriquées sauf celle imposée par la mémoire disponible (une boucle FOR.NEXT consomme I7 octets).
Dans ce type de boucle,le test de sortie se fait lors du NEXT .Par suite, la boucle sera toujours parcourue au moins une fois.
Il est possible de sortir d'une ou plusieurs boucles FOR..NEXT imbriquées grâce à l'instruction EXIT (cf cette instruction)

Remarque : Lors d'un RETURN,le S-Basic élimine les boucles FOR..NEXT qui avaient été ouvertes dans le sous programme.Ce mode de fonctionnement n'existe pas dans cette version Basic (puisqu'alors,pour des raisons de pure lôgique,il aurait fallu aussi éliminer les REPEAT..UNTIL ,WHILE...WEND , les PROC et les FN !!!) .Pour l'adaptation des programmes,il faudra donc étudier leur fonctionnement et utiliser l'instruction EXIT .

CODE	: 8E
INSTRUCTION	: NEXT

NEXT indique la fin d'une boucle FOR..NEXT
L'indice de boucle n'est pas obligatoire après NEXT mais,pour la bonne compréhension du programme,il est conseillé de l'écrire.
Remarque : On peut écrire NEXT I,J au lieu de NEXT I:NEXT J

CODE	: 8F
INSTRUCTION	: PRINT

PRINT n'a pas été modifié.
Remarque : Une temporisation (existant dans le S-Basic mais accessible uniquement par POKE) permet de ralentir l'affichage : voir PWAIT

CODE	: 90
INSTRUCTION	: ELSIF

ELSIF est utilisé dans les structures IF...ENDIF uniquement en mode I .

CODE	: 9I
INSTRUCTION	: INPUT

INPUT n'a pas été modifié.Voir le manuel de l'ordinateur

CODE	: 92
INSTRUCTION	: ENDIF

ENDIF indique la fin de la structure IF...ENDIF en mode I .

CODE	: 93
INSTRUCTION	: IF

IF suivi d'une condition indique le début d'une structure de test en mode O ou en mode I .Se reporter aux premières pages de cette notice.

CODE	: 94
INSTRUCTION	: DATA

DATA n'a pas été modifié.

CODE	: 95
INSTRUCTION	: READ

READ sert à lire les DATA.Son fonctionnement n'a pas été modifié.

CODE	: 96
INSTRUCTION	: DIM

DIM sert à dimensionner (et donc à construire) les tableaux numériques et alphanumériques.
Ici,le nombre de dimensions est limité à 128,ce qui ne veut pas dire qu'il est possible de construire un tel tableau : penser à la taille de la mémoire disponible !
Chacune des dimensions peut être comprise entre 0 et 32767 (7FFF en Hexa).

Remarques : Il est impossible d'utiliser une variable "dimensionnée" (comme A(3,4) par exemple) si le tableau n'a pas encore été défini.De même,toute tentative de redimensionnement ,avec au moins une dimension supérieure à ce qu'elle était déjà,conduira à une erreur.Par contre,le Basic ignorera tout redimensionnement avec des dimensions inférieures.
On peut dimensionner plusieurs tableaux: DIM A(3,7),B$(5,6,7),.....

CODE	: 97
INSTRUCTION	: REM

REM permet d'écrire des commentaires dans le programme.Tout ce qui suit,jusque la fin de ligne ou le prochain : sera ignoré lors de l'éxécution.

Remarque : En début de ligne ' est équivalent à REM .

CODE	: 98
INSTRUCTION	: END

END indique la fin du programme.Voir le manuel de l'ordinateur.

CODE	: 99
INSTRUCTION	: STOP

STOP permet d'arrêter le programme et de le reprendre par CONT (mais on ne doit pas l'avoir modifié).Voir le manuel de l'ordinateur.

CODE	: 9A
INSTRUCTION	: CONT

CONT permet de reprendre l'éxécution d'un programme stoppé par STOP (en programme) ou par Shift Break (au clavier).Voir le manuel de l'ordinateur.

CODE	: 9B
INSTRUCTION	: CLS

CLS permet d'effacer l'écran.Cet ordre est disponible dans le S-Basic bien que le manuel n'en parle pas ....

CODE	: 9C
INSTRUCTION	: DO

DO est utilisé dans WHILE : WHILE condition DO
et dans CASE : WHEN valeur,valeur,....,valeur DO
Son seul effet est de passer à l'instruction suivante.Cependant DO doit toujours être suivi de : ou de la fin de ligne.
DO peut être utilisé partout dans le programme.

CODE	: 9D
INSTRUCTION	: ON

ON est utilisé dans ON ERROR GOTO
dans ON variable GOTO nI,n2,.....
dans ON variable GOSUB nI,n2,....
On se reportera au manuel de l'ordinateur pour le fonctionnement.

CODE	: 9E
INSTRUCTION	: LET

LET est l'instruction (facultative) permettant d'affecter une valeur à une variable. exemple : LET A=67 que l'on écrit plus souvent : A=67

CODE	: 9F
INSTRUCTION	: NEW

NEW permet d'effacer le programme qui se trouve en mémoire.Les variables seront aussi éffacées ainsi que le catalogue des FN/PROC et la pile Basic.Il n'y a,dans ce Basic,aucun moyen de récupérer un programme ainsi détruit.

CODE	: AO
INSTRUCTION	: POKE

POKE permet d'écrire des valeurs dans la mémoire de l'ordinateur.Voir le manuel pour plus de précisions.

Remarque : Le S-Basic (comme le K-Basic) a la propriété suivante : Il accepte la syntaxe : POKE adresse,valeurI,valeur2,valeur3,..... Dans ce cas, valeurI sera placée dans adresse puis valeur2 dans (adresse+I),..etc...

CODE	: AI
INSTRUCTION	: OFF

OFF n'est utilisé que dans PLOT OFF .Voir le manuel du MZ-700.

CODE	: A2
INSTRUCTION	: MODE

MODE est utilisé pour commander l'imprimante : voir manuel de l'ordinateur.
Il est aussi utilisé pour les structures de test IF:
MODE O permet seulement d'utiliser IF ....THEN....ELSE (sur une ligne)
MODE I permet d'utiliser les structures IF...ENDIF .
MODE peut être utilisé en programme d'où la possibilité d'utiliser,dans le même programme ,les deux types de structures de test.

CODE	: A3
INSTRUCTION	: SKIP

SKIP est utilisé avec l'imprimante et n'a pas été modifié.

CODE	: A4
INSTRUCTION	: PLOT

PLOT est utilisé avec l'imprimante et n'a pas été modifié.

CODE	: A5
INSTRUCTION	: LINE

LINE est utilisé avec l'imprimante et n'a pas été modifié.

CODE	: A6
INSTRUCTION	: RLINE

RLINE est utilisé avec l'imprimante et n'a pas été modifié.

CODE	: A7
INSTRUCTION	: MOVE

MOVE est utilisé avec l'imprimante et n'a pas été modifié.

CODE	: A8
INSTRUCTION	: RMOVE

RMOVE est utilisé avec l'imprimante et n'a pas été modifié.

CODE	: A9
INSTRUCTION	: TRON

TRON est utilisé pour la recherche d'erreurs dans un programme.Le numéro des lignes exécutées s'affiche (entre crochets).

CODE	: AA
INSTRUCTION	: TROFF

TROFF supprime le mode TRON.

CODE	: AB
INSTRUCTION	: INP#

INP n'a pas été modifiée.Voir le manuel de l'ordinateur.

CODE	: AC
INSTRUCTION	: INKEY

INKEY nom de variable : attend l'enfoncement d'une touche en faisant clignoter le curseur.

INKEY X : Le Basic attend l'enfoncement d'une touche numérique dont la valeur sera af ectée à X

INKEY X$: La touche enfoncée sera affectée,comme chaîne de caractères" à la variable X$.

Le curseur disparaît dès qu'une touche est enfoncée mais rien d'autre ne sera inscrit sur l'écran.

CODE	: AD
INSTRUCTION	: GET

GET X (ou X$) envoie dans X (ou X$) la "valeur" de la touche enfoncée au moment de l'éxécution de GET (comme INKEY).Cependant,ici,le Basic n'attend pas l'enfoncement de la touche et pour pouvoir être prise en compte,lors d'un nouveau GET,la touche enfoncée doit être relachée puis enfoncée de nouveau. On sera donc obligé de "pianoter".Pour l'éviter,on utilisera KEY.

CODE	: AE
INSTRUCTION	: PCOLOR

PCOLOR est utilisé avec l'imprimante.Voir le manuel de l'ordinateur.

CODE	: AF
INSTRUCTION	: PHOME

PHOME est utilisé avec l'imprimante.Voir le manuel de l'ordinateur.

CODE	: BO
INSTRUCTION	: HSET

HSET est utilisé avec l'imprimante.Voir le manuel Sharp.

CODE : B1
INSTRUCTION : GPRINT

GPRINT est utilisé avec l'imprimante.Voir le manuel Sharp.

CODE : B2
INSTRUCTION : KEY

KEY est utilisé dans KEY LIST et DEF KEY .Le mode de fonctionnement de ces deux ordres n'a pas été modifié.

KEY X (ou X$) fonctionne comme GET mais en "continu",sans que l'on soit obligé de pianoter sur le clavier.Pour bien voir la différence,faire marcher le programme suivant : 10 REPEAT:GET X$:PRINT X$:UNTIL 0 puis remplacer GET par KEY.

CODE : B3
INSTRUCTION : AXIS

AXIS est utilisé avec l'imprimante.Voir le manuel Sharp.

CODE : B4
INSTRUCTION : LOAD

LOAD sert à charger,depuis la cassette,un programme dans l'ordinateur.Voir le manuel Sharp.

CODE : B5
INSTRUCTION : SAVE

SAVE sert à enregistrer sur cassette,le programme se trouvant en mémoire.Voir le manuel de l'ordinateur.

CODE : B6
INSTRUCTION : MERGE

MERGE sert à ajouter au programme en mémoire,un autre programme se trouvant sur cassette.Voir le manuel de l'ordinateur.

CODE	: B7
INSTRUCTION	: OF

OF est seulement utilisé dans CASE....OF

CODE	: B8
INSTRUCTION	: CONSOLE

CONSOLE n'a pas vu son fonctionnement modifié.On se reportera au manuel de l'ordinateur.

CODE	: B9
INSTRUCTION	: PROC

PROC est utilisé pour la définition d'une procédure : DEF PROC
et pour l'appel de cette procédure : PROC
On se reportera à la page 2I (et suivantes) de cette notice.

CODE	: BA
INSTRUCTION	: OUT #

OUT# n'a pas été modifié.Se reporter au manuel Sharp.

CODE	: BB
INSTRUCTION	: CIRCLE

CIRCLE est utilisé avec l'imprimante.Se reporter au manuel Sharp.

CODE	: BC
INSTRUCTION	: TEST

TEST est utilisé avec l'imprimante.Se reporter au manuel de l'ordinateur.

CODE	: BD
INSTRUCTION	: PAGE

PAGE est utilisé avec l'imprimante.On se reportera au manuel Sharp.

CODE	: BE
INSTRUCTION	: REPEAT

REPEAT indique le début d'une structure REPEAT-UNTIL .Voir page I6 de ces notes.

CODE : BF
INSTRUCTION : UNTIL

UNTIL condition indique la fin d'une structure REPEAT-UNTIL.Si la condition est fausse,le Basic reprend au REPEAT correspondant.Si elle est vraie, l'éxécution se poursuit après UNTIL condition.Voir page I6

CODE : C0
INSTRUCTION : ERASE

ERASE est un mot clé du S-Basic qui n'a pas d'interprétation.

CODE : C1
INSTRUCTION : ERROR

ERROR est utilisé dans ON ERROR GOTO . On peut aussi utiliser (comme en S-Basic mais le manuel n'en parle pas) ERROR n seul,ce qui aura pour effet d'inscrire à l'écran le message d'erreur de numéro n.En K-Basic,il y a 59 messages d'erreur donc n devra être I,2,.....,59 .Toute valeur en dehors de ces limites produira une erreur .

CODE : C2
INSTRUCTION : ELSE

ELSE est utilisé dans les structures IF :voir page I8 .
En MODE 0 ,ELSE peut être employé seul;-il fera passer à la ligne suivante.

CODE : C3
INSTRUCTION : USR

USR n'a pas été modifié :Voir le manuel Sharp.
Remarque: Dans le cas USR(ad,X$) ,le manuel dit que BC contient la longueur de la chaîne X$ .C'est vrai en K-Basic mais pas en S-Basic où B et C contiennent chacun la longueur (En K-Basic B=C).

CODE : C4
INSTRUCTION : BYE

BYE donne le contrôle de l'ordinateur au moniteur chargé en même temps que le Basic.
Le retour au Basic se fait par * R

CODE	: C5
INSTRUCTION	: WHILE

WHILE condition DO : ef cctue ce qui est entre DO et WEND tant que "condition" est vraie.Voir page I6

CODE	: C6
INSTRUCTION	: WEND

WEND indique la fin d'une boucle WHILE...WEND.Voir page I6

CODE	: C7
INSTRUCTION	: DEF

DEF est utilisé dans DEF KEY com e en S-Basic
DEF FN et DEF PROC: Voir page 2I de ces notes.

CODE	: C8
INSTRUCTION	: RESULT=

RESULT= expression ,retourne "expression" comme résultat d'une fonction FN .
Dans le cas où la définition de la fonction peut tenir sur une seule ligne, le mot RESULT est facultatif : exemple : DEF FN"A"(X)=3*X + 5

CODE	: C9
INSTRUCTION	: BEEP

BEEP a été expliquée à la page I2 (SON).

CODE	: CA,CB,CC,CD
INSTRUCTION	: néant

Ces codes pourraient être utilisés par un utilisateur voulant créer ses propres mots-clés.

CODE	: CE
INSTRUCTION	: WOPEN

WOPEN sert à ouvrir un fichier sur cassette pour y enregistrer des données.Voir le manuel de l'ordinateur.

CODE	: CF
INSTRUCTION	: CLOSE

CLOSE sert à terminer les opérations commencées par WOPEN ou ROPEN.Voir le manuel.

CODE	: DO
INSTRUCTION	: ROPEN

ROPEN sert à ouvrir un fichier pour transférer des données de la cassette dans l'ordinateur.Voir le manuel Sharp.

CODE	: DI
INSTRUCTION	: GRAPH

GRAPH x,y sert à placer un pseudo curseur graphique en (x,y) sur l'écran.Voir page I2 de cette notice.

CODE	: D2
INSTRUCTION	: DRAW

DRAW k,x,y,c sert à tracer (k=I) ou effacer (k=0) une droite sur l'écran.Voir page I2 pour plus de détails.

CODE	: D3
INSTRUCTION	: SWAP

SWAP A,B où A et B sont les noms de deux TABLEAUX déjà dimensionnés,de mêmes dimensions et de même type (numérique ou nom),permute les éléments des deux tableaux ;C'est l'instruction EXC pour les tableaux .

CODE	: D4
INSTRUCTION	: NULL

NULL A,B$,C met à zéro tous les éléments des tableaux de noms A,B$,C .Si les tableaux sont numériques,les éléments sont mis à zéro.S'ils sont alphanumériques,les chaînes éléments deviennent les chaînes vides.

CODE	: D5
INSTRUCTION	: COPY

COPY A,B où A et B sont les noms de deux tableaux déjà dimensionnés et de même dimensions rend le tableau A identique au tableau B .C'est l'instruction A=B pour les tableaux.

CODE	: D6
INSTRUCTION	: EXC

EXC A,B échange les valeurs de A et B .A et B sont des noms de variables dimensionnées ou non.

CODE	: D7
INSTRUCTION	: WAIT

WAIT n attend environ n/100 secondes avant de reprendre l'éxécution du programme. n est compris entre 1 et 65535 .Il est possible d'interrompre la "boucle d'attente" en appuyant sur Break.Dans ce cas,la mémoire MEMWET (voir son adresse dans les tables de la fin de cette notice) contiendra 0 .Par contre, si la boucle va à son terme,MEMWET contiendra une valeur non nulle.

CODE	: D8
INSTRUCTION	: PWAIT

PWAIT n où n est compris entre 0 et 255 interrompt légèrement le programme à la fin de l'instruction PRINT .Cette possibilité existe dans le S-Basic mais ne peut se faire qu'avec un POKE.dans $3900 (S-Basic seulement !!!).

CODE	: D9
INSTRUCTION	: KILL

KILL est un mot clé du S-Basic ne possédant pas d'interprétation.

CODE	: DA
INSTRUCTION	: LOCAL

LOCAL A,I,H$ déclare A,I,H$ comme variables locales à une PROC ou FN .Voir page 23.

CODE	: DB
INSTRUCTION	: CASE

CASE se trouve en tête de la structure CASE..OF...ENDCASE.Voir page 20

CODE	: DC
INSTRUCTION	: WHEN

WHEN fait partie de la structure CASE...ENDCASE.Voir page 20

CODE	: DD
INSTRUCTION	: OTHERWISE

OTHERWISE fait partie de la structure CASE...ENDCASE.Voir page 20.

CODE	: DE
INSTRUCTION	: EXIT

EXIT n (si n=I on peut écrire EXIT seul) permet d'éliminer de la mémoire du Basic n boucles FOR...NEXT .Il est bon de remarquer que cette instruction ne peut éliminer les autres boucles REPEAT ou WHILE ni les structures GOSUB.Autrement dit,le Basic éliminera effectivement n boucles FOR-NEXT si ,dans la pile Basic,se trouvent n zones "FOR" consécutives.Sinon,le Basic éliminera moins de n zones.
Après EXIT,le programmeur devra réaliser lui même la sortie des boucles.

CODE	: DF
INSTRUCTION	: SPOKE

SPOKE adr,n où adr est compris entre D000 (53248 en décimal) et D7FF (55295 déc.) et n entre 0 et 255 ,place dans la mémoire adr + $800 la valeur n. Cette instruction sera donc très utile pour colorer des caractères placés sur l'écran par des POKE et pour accéder au second générateur de caractères.Par exemple :
SPOKE$D000 ,$2I : POKE $D000,$8I écrit "a" en rouge sur fond bleu en haut à gauche de l'écran .Mais si on écrit $AI au lieu de$2I (On remarquera que$AI = $80 +$2I) ,on aura un petit parapluie au lieu du "a" puisque $8I est le code du parapluie dans la seconde table.

CODE	: EO
INSTRUCTION	: TO

TO est utilisé dans FOR...NEXT

CODE	: EI
INSTRUCTION	: STEP

STEP est utilisé dans FOR...NEXT

CODE	: E2
INSTRUCTION	: THEN

THEN est utilisé dans IF..THEN..ELSE mais,dans la plupart des cas,sera facultatif.
Voir page I8 de cette notice.

CODE	: E3
INSTRUCTION	: USING

USING fonctionne comme dans le S-Basic avec la nouvelle possibilité nouvelle suivante :

```
I0 A=I23                                  affichera : ␣␣FI23
20 PRINT USING "FF####";A
```

CODE	: E4
INSTRUCTION	: $\overline{\text{II}}$ ou PI

Cette valeur (3.I4....) peut être entrée sous la forme PI mais elle sera automatiquement convertie en $\overline{\text{II}}$.

CODE	: E5
INSTRUCTION	: néant

Ce code est libre mais ne peut pas être donné à un mot clé crée par l'utilisateur.

CODE	: E6
INSTRUCTION	: TAB

TAB fonctionne comme en S-Basic.Voir le manuel de l'ordinateur.

CODE	: E7
INSTRUCTION	: SPC

SPC n'a pas été modifié.Voir le manuel Sharp.

CODE	: E8
INSTRUCTION	: VARPTR

VARPTR(nom de variable) donne l'adresse du début de la cellule située en zone 2 et contenant la valeur de la variable.On se souviendra (page I0) que cette cellule commence par un octet contenant la longueur de la "zone de codage". Si la variable n'a pas encore subi une affectation,elle n'a pas de zone de codage en zone 2 et VARPTR retournera 0.

CODES	: E9 et EA
INSTRUCTIONS	: MAX et MIN

A MAX B (resp. A MIN B) donne le maximum (resp. le minimum) des deux variables numériques ou alphanumériques A et B.

CODES	: EB EC ED
INSTRUCTIONS	: OR AND XOR

Ces opérateurs travaillant uniquement sur des variables numériques sont le OU , le AND et le OU exclusif.Leur mode de fonctionnement a été vu à la page I4 .

CODES	: EE à F6
INSTRUCTIONS	: tests

Ces 6 types de tests (< > et > < sont équivalents ,.....) travaillent sur des arguments numériques et alphanumériques.

CODE	: F7
INSTRUCTION	: +

+ est le symbole de l'addition pour les nombres et de la "concaténation" pour les chaînes de caractères.

CODE	: F8
INSTRUCTION	: -

- a deux significations : opérateur diadique : soustraction
opérateur monadique: changement de signe

CODE	: F9
INSTRUCTION	: DIV

A DIV B est égal à INT(A/B) .Cette instruction est surtout utile avec des arguments entiers.

CODE	: FA
INSTRUCTION	: MODULO

A MODULO B est égal à A - B* INT(A/B),ce qui,pour A et B entiers,donne le reste de la division de A par B (alors que DIV donne le quotient).

CODES	: FB FC FD
INSTRUCTIONS	: / * ↑

Il s'agit de la division,multiplication et élévation à une puissance.Dans ce dernier cas,le Basic accepte d'élever un nombre négatif à une puissance entière.

TABLE DES "FE"

Dans cette table,seuls sont utilisés les codes :

FE 81 SET : Voir page I2 et le manuel Sharp.
FE 82 RESET : Voir page I2 et le manuel Sharp.
FE 83 COLOR : Voir le manuel Sharp.
FE A2 MUSIC : Voir le manuel Sharp.
FE A3 TEMPO : Voir le manuel Sharp.
FE A4 CURSOR : Voir le manuel Sharp.
FE A5 VERIFY : Voir le manuel Sharp.
FE A6 CLR : efface toutes les variables en zone I et 2.
FE A7 LIMIT : Utilisation comme dans le S-Basic.Son emploi en programme est déconseillé car il y aura alors effacement du catalogue des FN/PROC et de la pile Basic.Les éventuelles FN/PROC seront alors inaccessibles au programme.
D'autre part,en S-Basic,lorsqu'on écrit LIMIT MAX,le Basic teste la présence de M puis de A puis de X alors que le K-Basic teste seulement la présence du code E9 (celui de MAX).Il y a donc là,une légère incompatibilité avec le S-Basic:il suffira de ré-entrer la ligne de programme contenant LIMIT MAX pour que tout rentre dans l'ordre.
FE AE BOOT : BOOT renvoie le contrôle au moniteur de la mémoire morte.Il n'y a aucun moyen de revenir au Basic,celùi ci étant partiellement détruit.

TABLE DES "FF"

Dans cette table,on trouve les fonctions préprogrammées du K-Basic.

CODE : FF 80
INSTRUCTION : INT

INT(X) fournit la partie entière de X .

CODE : FF 81
INSTRUCTION : ABS

ABS(X) fournit la valeur absolue de X .

CODE : FF 82
INSTRUCTION : SIN

SIN(X) où X est en radians,donne le sinus de X.

CODE : FF 83
INSTRUCTION : COS

COS(X) où X est en radians donne le cosinus de X.

CODE : FF 84
INSTRUCTION : TAN

TAN(X) où X est en radians donne la tangente de X.

CODE : FF 85
INSTRUCTION : LN

LN(X) où X est strictement positif,donne le Logarithme Népérien de X.

CODE : FF 86
INSTRUCTION : EXP

EXP(X) donne l'exponentielle de X.

CODE : FF 87
INSTRUCTION : SQR

SQR(X) où X est positif ou nul,donne la racine carrée de X.

CODE : FF 88
INSTRUCTION : RND

RND seul est équivalent à RND(0)
RND(X) où X est négatif ou nul redonne toujours la même valeur .Voir le manuel de l'ordinateur pour plus de précisions.

CODE : FF 89
INSTRUCTION : PEEK

PEEK(adr) renvoie,en décimal,le contenu de l'octet adr.Voir le manuel Sharp.

CODE : FF 8A
INSTRUCTION : ATN

ATN(X) renvoie,en radians l'Arc tangente de X.

CODE : FF 8B
INSTRUCTION : SGN

SGN(X) vaut : 0 si X est nul
I si X est positif
-I si X est négatif.

CODE : FF 8C
INSTRUCTION : LOG

LOG(X) où X est strictement positif,est le logarithme décimal de X.

CODE : FF 8D
INSTRUCTION : néant

Ce code n'est pas attribué en K-Basic.

CODE : FF 8E
INSTRUCTION : PAI

PAI(X) est ègal à X*PI

CODE : FF 8F
INSTRUCTION : RAD

RAD(X) est égal à X*PI/I80 donc convertit X (écrit en degrés) en radians.

CODE : FF 90
INSTRUCTION : ASN

ASN(X) où X est compris entre +I et -I est l'Arc sinus de X .

CODE : FF 9I
INSTRUCTION : ACS

ACS(X) où X est compris entre +I et -I est l'Arc cosinus de X.

CODE : FF 92 à FF 94
INSTRUCTION : néant

Ces codes ne sont pas attribués.

CODE : FF 95
INSTRUCTION : EOF

EOF (qui doit signifier End of file) est un mot-clé du S-Basic sans interprétation.

CODE : FF 96 à FF 9D
INSTRUCTION : néant

Ces codes ne sont pas attribués dans la version 5 du K-Basic.

CODE : FF 9E
INSTRUCTION : JOY

JOY est utilisé avec les poignées de jeu.Sa syntaxe est JOY(X) où X est compris entre 0 et 7 .

CODE	: FF 9F
INSTRUCTION	: néant

Ce code n'est pas attribué.

CODE	: FF AO
INSTRUCTION	: CHR$

CHR$ (X) est le caractère dont le code ASCII est X .Le manuel Sharp dit qu'on ne peut utiliser une valeur de X inférieure à 33.En fait,c'est tout à fait possible,mais on obtient alors des "caractères de contrôle" .

Par exemple : PRINT CHR$ (X) donne :

si X=0,I,2,3,4	rien	
si X=5	passage en minuscules	(cf CTRL E)
si X=6	passage en majuscules	(cf CTRL F)
si X=7	un bip	(cf CTRL G)
si X=8,9,I0,II,I2	rien	
si X=I3	CR (retour chariot)	(cf CTRL M)
si X=I4,I5	rien	
si X=I6	Delete	(cf CTRL P)
si X=I7	curseur en bas	(cf CTRL Q)
si X=I8	curseur en haut	(cf CTRL R)
si X=I9	curseur à droite	(cf CTRL S)
si X=20	curseur à gauche	(cf CTRL T)
si X=2I	curseur en "HOME"	(cf CTRL U)
si X=22	effacement de l'écran	(cf CTRL V)
si X=23	mode graphique	(cf CTRL W)
si X=24	idem touche INST	(cf CTRL X)
si X=25	passage en majuscules	(cf CTRL Y)
si X=26	rien	
si X=27	CR (retour chariot)	(cf CTRL [)
si X=28,29,30,3I	rien	
si X=32	affiche un espace.	

(Le lecteur aura peut être remarqué que la lettre suivant CTRL est pour chaque ligne,la Xième de l'alphabet:ce n'est pas un hasard car pour calculer CHR$(X) ,le K-Basic utilise la table de CTRL qui commence à 0067H et se termine à 00A6H.En y"pokant" l'adresse d'une routine écrite par lui même,l'utilisateur pourra y avoir accès instantanément par CTRL suivie d'une autre touche ou par CHR$)

Remarque : Comme en S-Basic,on peut aussi écrire CHR$(a,b,c,d) qui est équivalent à CHR$(a)+CHR$(b)+CHR$(c)+CHR$(d)

CODE	: FF AI
INSTRUCTION	: STR$

STR$ (X) renvoie une chaîne de caractères ASCII représentant la valeur de l'expression numérique X.

CODE	: FF A2
INSTRUCTION	: HEX$

HEX $(X) renvoie une chaîne de caractères qui représente la valeur de X écrite en héxadécimal (X de 0 à 65535)

CODE	: FF A3 à FF A9
INSTRUCTION	: néant

Ces codes ne sont pas attribués dans la version 5 du K-Basic.

CODE	: FF AA
INSTRUCTION	: INV$

INV$(A$) renvoie une chaîne de caractère obtenue en inversant l'ordre des caractères de A$. Exemple PRINT INV$("BONJOUR") donne "RUOJNOB".

CODE	: FF AB
INSTRUCTION	: ASC

ASC(A$) renvoie le nombre égal au code ASCII du premier caractère de A$.

CODE	: FF AC
INSTRUCTION	: LEN

LEN(A$) donne la longueur de la chaîne A$.

CODE	: FF AD
INSTRUCTION	: VAL

VAL(X$) convertit en valeur numérique, la représentation en caractères ASCII d'une valeur numérique. Exemple VAL("I.345") donne I.345 .

CODE	: FF AE à FF B0
INSTRUCTION	: néant

Ces codes ne sont pas attribués.

CODE	: FF BI
INSTRUCTION	: EVAL

EVAL (A$) calcule d'abord A$ puis applique l'évaluateur de formules au résultat. Il y a donc deux évaluations successives.
Exemple : PRINT EVAL("SIN"+"(1)") va donner .84147.... soit SIN(1).

Le programme ci-contre il ustre les deux utilisations principales de EVAL :
Entrée d'une fonction en réponse à un INPUT (comme SIN(X) par exemple).
Entrée d'une valeur numérique non calculée en réponse à un INPUT (com e 2* PI par exemple).

```
10 INPUT"Fonction:";F$
20 REPEAT
30 : INPUT"X=";X$:X=EVAL(X$)
40 : PRINT EVAL(F$)
50 UNTIL 0
```

x=λ$
1. Funktion eingeben z.B. SIN(X)
2. X = Wert für x eingeben in X$ EVAL(X$) setzt X
3. EVAL(F$) zeigt Wat für SIN(X)

CODE	: FF B2
INSTRUCTION	: néant

Ce code n'est pas attribué.

CODE	: FF B3
INSTRUCTION	: ERN

ERN est le numéro de la dernière erreur s'étant produite au cours du programme.

CODE	: FF B4
INSTRUCTION	: ERL

ERL est le numéro de la ligne où s'est produite la dernière erreur.

CODE	: FF B5
INSTRUCTION	: SIZE

SIZE donne la taille de la mémoire disponible .Pour le lecteur ayant lu les pages 8 et suivantes,signalons que SIZE=(STACK)-(VARTOP) . Au départ ,SIZE est égal à 34000 .

CODE	: FF B6
INSTRUCTION	: VPOS

VPOS est la position verticale (entre 0 et 24) du curseur.

CODE	: FF B7
INSTRUCTION	: HPOS

HPOS est la position horizontale (entre 0 et 39) du curseur.

CODE	: FF B8 et FF B9
INSTRUCTION	: néant

Ces codes ne sont pas attribués.

CODE	: FF BA
INSTRUCTION	: LEFT$

LEFT$(A$,n) est la chaîne de caractères constituée des n caractères les plus à gauche de A$.
n doit être compris entre 0 et 255 .
Si n=0 ,on trouvera la chaîne vide et pour n supérieur à LEN(A$),on trouvera A$.

CODE	: FF BB
INSTRUCTION	: RIGHT$

RIGHT$(A$,n) est la chaîne de caractère constituée des n caractères les plus à droite de A$; n doit être compris entre 0 et 255.
Si n=0 ,on trouvera la chaîne vide et pour n supérieur à LEN(A$),on trouvera A$.

CODE	: FF BC
INSTRUCTION	: MID$

MID$(A$,n,p) est le chaîne constituée des caractères de rang n,n+I,...n+p-I dans la chaîne A$ .On prend donc dans A$,p caractères,le premier étant le n-ième caractère de A$.
n doit être compris entre I et 255 (0 donnera une erreur).
p doit être compris entre 0 et 255 (0 donne une chaîne vide).

MID$(A$,n) est considéré comme MID$(A$,n,255) et est donc équivalent à : RIGHT$(A$,LEN(A$)+I-n)

CODE	: FF BD
INSTRUCTION	: IN$

IN$(A$,n,p,B$) permet d'insérer la chaîne B$ dans A$ entre les caractères de rang n et de rang p (ces caractères sont conservés tous deux).
n doit être compris entre 0 et L (où L=LEN(A$))
p doit être strictement supérieur à n (et inférieur à 255).
Par exemple,si n=0 et p=1 ,le résultat est B$+ A$.
Si n=L et p=255 ,le résultat sera A$ + B$.
B $peut être vide:cela reviendra à supprimer des caractères dans A$.

CODE	: FF BE
INSTRUCTION	: POINT

POINT a été vue à la page I2 lors de l'étude des ordres graphiques.

CODE	: FF BF
INSTRUCTION	: INSTR

INSTR(A$, B$,n) cherche l'apparition de la chaîne B$ dans la chaîne A$,en commençant la recherche au n-ième caractère de A$.
n doit être compris entre I et 255
Le résultat est 0 si B$n'est pas contenue dans A$,sinon il est égal à la position du premier caractère de B$ dans A$, à partir du n-ième caractère de A$.
Exemple : P. INSTR("COMPUTER","PUT",n) donnera :

4 si n=I
3 si n=2
2 si n=3
I si n=4
0 si n est supérieur ou égal à 5

CODE	: FF C0 à FF C2
INSTRUCTION	: néant

Ces codes ne sont pas attribués.

CODE	: FF C3
INSTRUCTION	: STRING$

STRING$(n,A$) est la chaîne constituée de n répétitions du premier caractère de la chaîne A$. n doit être compris entre Ø et 255 .

CODE	: FF C4
INSTRUCTION	: TI$

TI$ sert à mettre à l'heure l'horloge interne et à la relire.Voir le manuel de l'ordinateur.

CODE	: FF C5 et FF C6
INSTRUCTION	: néant

Ces codes ne sont pas attribués.

CODE	: FF C7
INSTRUCTION	: FN

FN sert à définir une Fonction.Voir page 2I.

+ Il n'y a pas dans le K-Basic de routines permettant l'adaptation de programmes écrits en Basic SP-5025 (Sharp MZ 80 K) .L'utilisateur désirant faire cette adaptation sera donc obligé de le faire grâce au S-Basic et ,ensuite seulement, il pourra utiliser le K-Basic.

+ Les définitions de fonctions :DEF FN.(X)=.... du S-Basic devront être modifiées en écrivant le nom de la fonction entre guillemets et en plaçant DEF juste après un numéro de ligne.De même , FN. dans les calculs devra voir son nom placé entre deux guillemets.

+ LIMIT qui pouvait être utilisé en programme dans le S-Basic ,peut toujours l'être en K-Basic mais il est préférable de ne l'utiliser que directement au clavier. Une ligne de programme écrit en S-Basic et contenant LIMIT MAX devra être ré-entrée dans l'ordinateur (la lister,placer le curseur sur cette ligne et appuyer sur CR).

+ Un programme écrit en S-Basic sans espaces entre les mots-clés sera lu et exécuté correctement par le K-Basic.Cependant toute modification risque de rendre le programme non compréhensible par le K-Basic :voir La première note de la page 7. Il est d'ailleurs illusoire de supprimer les espaces dans un programme en espérant qu'il "tournera" plus rapidement : on y gagnera peut-être quelques microsecondes mais on pourra y perdre des heures en essayant de le mettre au point, le programme devenant illisible .Ne pas oublier que l'un des buts de la programmation structurée est de produire des programmes lisibles !

Cette table a été produite par le "RELOCATABLE LOADER SP 230I" lors de la construction du Basic.Les adresses sont les adresses absolues lors du fonctionnement du Basic et on devra leur ajouter I200H si le Basic a été chargé sans être éxécuté.

#FAC1	72AB	#FAC2	72B0	#FAC3	72B5	#FAC4	72BA
#FAC5	72BF	#FAC6	72C4	#MSG1	1803	#MSG2	180E
#MSG3	3C22	?ADCN	04CD	?BEL0	0A12	?DACN	04B6
?MANG	00FA	?PNT2	05C5	?PONT	05B6	ABS	5B5B
ACS	6124	ADDER	5705	ADRVA2	53C1	ADRVAR	53B8
AIGU	191B	AIGUIL	190F	AMPM	0064	ASC	5221
ASCHEX	43DE	ASN	60E5	ASSEZM	5430	ATN	5BC1
ATTVAR	258E	AUTMEM	72D5	AUTO	2209	AXIS	4AC9
BASIC	1842	BASIC0	1847	BASIC2	0FFC	BASIC3	1F94
BASIC4	2B9D	BASIC5	3A3B	BASIC6	4D62	BASIC7	58C9
BASICT	629F	BEEP	1F9A	BELL	003E	BLOCKM	48B5
BOOT	338C	BRKEY	001E	BUFFER	7316	BUFNB0	731E
BUFNB1	7327	BUFNB2	732F	BUFNB3	7321	BUFNB4	7322
BUFWNB	731D	BYE	13C2	CARCAL	7350	CASE	301D
CASECR	3F4B	CASESP	3F3B	CASWRT	3F34	CAT	73AD
CBASIC	1822	CFLTHL	365B	CHANGE	18C7	CHGLIG	18FA
CHGLIN	18F7	CHIFFR	340E	CHINDC	4D62	CHLASC	21DB
CIRCLE	4B17	CLOSE	3E57	CLR	229D	CLRCAT	2264
CLRMEM	1ED1	CLRSTK	2278	CLS	3185	CM2OCT	4850
CMPFLT	577C	CMPNOM	16EE	CMPSTR	5759	CODCAS	0FFC
CODSTK	72DC	COLGRP	72DA	COLOR	3D87	COLORC	005D
COMPAF	5754	COMPIL	4073	COMPLM	56EC	CONSCK	004E
CONSLX	005B	CONSLY	0056	CONSOL	3320	CONSSP	734C
CONT	2060	CONT0 D	E004	CONT1 D	E005	CONT2 D	E006
CONTF D	E007	CONUNB	3462	COPY	2A9F	COS	5C79
CPARAM	2D24	CSTPT D	E003	CSTR D	E002	CURSOR	6780
DATA	6727	DBLHL	3583	DEBBAS	00E7	DEBCAS	1010
DEBPRO	73A1	DEBVAR	73A3	DEFKEY	3CF9	DEFONC	24C1
DEFTBL	1322	DEJOUU	4750	DEL0	2C64	DEL1	6207
DELETE	6855	DESESP	359C	DETABL	2A40	DETNOM	5403
DIM	55D6	DIM1	55F2	DISP	2D75	DIV	56B0
DIVIS	59BF	DIVPRO	37B8	DIXCOD	36AC	DO	3132
DPRNT	0054	DPRNT0	0055	DRAW	3259	ECRIS	16C8
ECRMES	17E2	EFFPRT	1CA7	ELIMPL	1E25	ELSE	3122
ELSIF	3119	ENDBAS	21EB	ENDP	65D2	ENDUGM	1E0E
ENTFLT	1F03	ERL	5122	ERLM	72E9	ERN	511A
ERN0	511D	ERNM	72F1	ERRADR	2157	ERRM	72EB
ERRM0	72ED	ERROR	214C	ERRT1	2080	ERRT10	2085
ERRT11	20B8	ERRT12	20BB	ERRT13	20BE	ERRT14	20C1
ERRT15	20C4	ERRT16	20C7	ERRT17	20CA	ERRT18	20CD
ERRT19	20D0	ERRT2	20A0	ERRT20	20D3	ERRT21	20D6
ERRT22	20D9	ERRT23	20DC	ERRT24	20DF	ERRT25	20E2
ERRT26	20E5	ERRT27	20E8	ERRT28	20EB	ERRT29	20EE
ERRT3	2087	ERRT30	20F1	ERRT31	20F4	ERRT32	20F7
ERRT33	20FA	ERRT34	20FD	ERRT35	2100	ERRT36	2103
ERRT37	2106	ERRT38	2109	ERRT39	210C	ERRT4	20A3
ERRT40	210F	ERRT41	2112	ERRT42	2115	ERRT43	2118
ERRT44	211B	ERRT45	211E	ERRT46	2121	ERRT47	2124
ERRT48	2127	ERRT49	212A	ERRT5	20A6	ERRT50	212D
ERRT51	2130	ERRT52	2133	ERRT53	2136	ERRT54	2139
ERRT55	213C	ERRT56	213F	ERRT57	2142	ERRT58	2145
ERRT59	2148	ERRT6	20A9	ERRT7	20AC	ERRT8	20AF
ERRT9	20B2	ERRTRT	181F	ESP2PT	1A7F	EXC	2A15
EXIT	5398	EXP	5EC6	EXPM0	7313	EXPSGN	7312
EXSTLN	2CF0	EXTVAR	5623	FAC1/2	33F5	FAC2/1	33FD
FINBAS	72FC	FINPRT	1D71	FINVAR	73A5	FN	62D7
FOR	1AD8	FORBUF	2649	FREE1	6A47	FREE2	6C90
FREE3	6F3C	FVARPT	72F8	GARDPT	72F2	GARDSP	72C9
GET	5791	GETKV	001B	GETL	0003	GETLI	004A

GETVDU	00EA	GNLNAU	225C	GOSUB	2E1A	GOTO	2EEC
GPRINT	4A4C	GRAPH	3238	GRAPHX	72D9	GRAPHY	72D6
GRDLIG	72F6	GRDNLI	72F4	GRHTBL	12DD	GTKEY	0058
HADR	1884	HEX$	517D	HEXASC	36C0	HEXCHF	3452
HEXFLT	3302	HPOS	5115	HSET	4A3F	IAC	72A7
IACCPL	1F49	IF	30A3	IMPRES	2691	IN$	2B10
INC5DE	33ED	INDATN	730E	INDCNV	3B8A	INDGRP	72DB
INDIC	004D	INERR	184D	INKEY	679A	INMESS	21CB
INOR	1864	INP#	6757	INPILE	72E1	INPUT	22FB
INSTR	6686	INT	5B60	INTEGR	35D1	INTEXP	7311
INUR	1868	INVER	29FB	INVINT	3614	JOY	32F8
JOYSTK	0061	KANTBL	129D	KEY	3C38	KEYD	679E
KEYPA D	E000	KEYPB D	E001	KEYPC D	E002	KEYPF D	E003
LEFT$	524D	LEN	5226	LET	194B	LETNL	0006
LIGB0	110E	LIGBUF	110F	LIGDCO	427C	LIGMEM	72DD
LIM	73AF	LIMAX	73B3	LIMBAS	73B1	LIMIT	6618
LINE	4977	LINENB	2DA0	LIST	3A3B	LN	5F94
LNCONV	3BB6	LNDCOL	7348	LNMEM	7314	LOAD	3B03
LOADNM	172C	LOCAL	64C3	LOG	5F88	LONCAS	100E
LONGLN	004F	MAJTBL	121D	MANTSG	5AAA	MELDV	0030
MEM0	2651	MEM1	2655	MEM2	72FA	MEM3	72D2
MEM4	5621	MEM5	72D4	MEM6	72FB	MEM7	72E3
MEM8	72E5	MEMRND	730F	MEMWET	1FFF	MERGE	3AE3
MID$	5287	MIDWNB	7326	MINTBL	125D	MODE	48BA
MODEIF	30C7	MODMEM	72CD	MODPTR	55C0	MODPTS	55B7
MODTST	2A36	MODULO	56B9	MONIT	0000	MOVE	49EB
MSG	0015	MSGX	0018	MSGY	0051	MSTA	0044
MSTP	0047	MULT	58D4	MUSIC	3D53	NAME?	48A6
NB?ENT	1E71	NB?INT	1E79	NBCELL	72D0	NBEGAU	57B3
NBPI	5E5D	NBR3	29E3	NBR?	56F6	NBUF1	7370
NBUF2	7371	NBUF4	7372	NEW	2288	NEXT	1B37
NL	0009	NOLNMS	4606	NULBUF	56E4	NULL	2AE5
NUMFRT	35A3	NUMPTR	72DF	ON	2EA5	OUT#	6740
OVER?	57EF	OVERMM	7353	PAGE	4D2A	PAI	5E54
PARSER	4D80	PCOLOR	49FC	PEEK	5E67	PHOME	4A31
PLOT	3386	PLOTMM	48B7	POINT	3206	POKE	67E7
PRINT	1BF8	PRINTM	1819	PRNT	0012	PRNTS	000C
PRNTT	000F	PRO3	24E1	PROBLD	6136	PROC	62D3
PRODEB	75D7	PRODOM	569A	PROGRF	3955	PROPDM	565A
PROPTR	72EF	PRTCAS	3F55	PRTMES	055E	PTDEBN	736E
PTRADD	7349	PUISS	5ADE	PWAIT	2017	PWRACC	167E
PWRIT	1762	R1	1909	RAD	5E4F	RATIO	0A39
RDDAT	002A	RDINF	0027	READ	2541	READYM	21D0
REM	6727	REM0	6728	RENUM	28A6	REPEAT	2F60
RESET	31AD	RESMEM	72E7	RESNUL	5744	RESTOR	24FB
RESU	572D	RESULT	65E2	RESUME	2E36	RESUMM	72CC
RETURN	2DF4	RIGHT$	5263	RLINE	49E4	RMOVE	49F5
RND	5E81	ROPEN	3FC5	RUN	1A87	SAUERR	1820
SAUTLN	17F9	SAVE	3B8B	SEARCH	162E	SET	318A
SETCAT	63C4	SETHL1	7302	SETHL2	7304	SETHL3	7306
SETHL4	7308	SETHL5	730A	SETHL6	730C	SETLIM	2270
SGN	5E38	SGNIND	734B	SIN	5C8F	SIZE	5102
SKIP	491D	SPAC0	4F41	SPACE	4F42	SPECIA	74D3
SPETBL	131D	SPOKE	629F	SPRO2	3E9F	SPRO3	3E83
SPSAUV	734E	SQR	5BA3	STACK	73A8	STACK0	72CE
STACK1	72D6	STARE0	1BF6	START	00DA	STARTB	7470
STKTRF	1F5E	STOERR	56E0	STOP	202C	STOP0	2024
STR$	5202	STRIN$	1ED7	STRIN?	56D5	SUBBER	5701
SUMEXP	7351	SUNDG D	E008	SWAP	2A8B	SWAP0	2A50
TABA1	3408	TABERR	4414	TABLFE	71A9	TABLFF	7207

TABLOU	543C	TABNOR	70E9	TAKE	360D	TAN	5D7B
TEMP	D E008	TEMPO	3D7S	TEMPO1	4846	TEMPO2	4849
TEST	4D1C	TESTES	3415	TI$	52D3	TIMRD	0038
TIMST	0033	TITLE	0050	TITRMS	73B5	TKTBL0	6867
TKTBL1	6C3B	TKTBL2	6E84	TOKFF	2440	TR1OOD	3405
TRFAC1	3400	TRFSTK	1DDC	TRFVA2	1A00	TRFVAR	19CC
TRFVAS	19D1	TROFF	22CC	TRON	22C8	TRONMM	72D3
TSPARF	538E	TST2PT	4274	TSTPA0	5384	TSTPA1	5391
TSTPA2	5387	TSTPAR	5383	TYPADR	19A7	TYPE0	3133
TYPE?	3137	TYPFAC	1E45	UNCOD	3681	UNTIL	2F33
USINPR	2657	USR	6707	VAL	522A	VARIAB	72CB
VARMEM	73A7	VARPTR	50E6	VARTOF	73A9	VARTYP	7355
VERFY	002D	VERIFY	3BE8	VIDPRO	228E	VIRG?	6736
VIRGU	6733	VIRGUL	6739	VPOS	5110	WAIT	1FE7
WEND	2FDC	WHEN	3014	WHILE	2F7S	WOPEN	3F89
WORK0	7356	WORK1	735E	WORK2	7366	WRDAT	0024
WRINF	0021	WRIT	1703	WRTVDU	00F2	WTNULI	22D1
XTEMP	0041						

TABLE DES MATIERES

INTRODUCTION I

MOTS CLES avec leur code 3

MESSAGES D'ERREUR 5

MODIFICATION OU ENREGISTREMENT DU K-BASIC 6

NOTES DIVERSES 7

QUELQUES INDICATIONS TECHNIQUES 8

GRAPHISME SUR ECRAN I2

SON I2

OPERATEURS MATHEMATIQUES I3

EVALUATION DES CONDITIONS LOGIQUES I4

LES BOUCLES EN K-BASIC I6

LES DEUX TYPES DE STRUCTURES IF ... THEN ... ELSE I8

\ STRUCTURE CASE ... ENDCASE 20

PROCEDURES ET FONCTIONS 2I

PARAMETRES ET VARIABLES LOCALES 23

PASSAGE DE PARAMETRES PAR REFERENCE 25

MOTS CLES CODES SUR UN SEUL OCTET 29

MOTS CLES CODES SUR DEUX OCTETS 47

COMPATIBILITE AVEC LE S-BASIC 55

TABLE DES ADRESSES DU K-BASIC 56