mbc alcyane

BASIC ALCYANE
MANUEL DE REFERENCE

Ce document n'est pas un document contractuel

DECEMBRE 1981

01-BASIC DE BASE

01-1 GENERALITES

01-2 FORMAT DES VARIABLES

01-3 ENTREES / SORTIES

01-4 BRANCHEMENTS

01-5 GESTION DES FLAGS

03-GESTION ECRAN

04-GESTION CLAVIER

05-GESTION IMPRIMANTE

06-AIDE A LA MISE AU POINT

07-TRAITEMENT VARIABLES/PROGRAMMES

08-GESTION DE SOUS-PROGRAMMES

09-TRAITEMENT DES FONCTIONS

16-FICHIER SEQUENTIEL

17-TRAITEMENT DE TEXTE

18-MULTIPOSTE ET MULTIALCYANE

19-COMPATIBILITE DISQUE IBM

20-TELEMAINTENANCE

21-GESTION HORLOGE

22-ERREURS BASIC

01

BASIC DE BASE

DEFINITIONS :

. LIGNE BASIC

La ligne se compose d'au maximum 192 caractères et se termine toujours en appuyant sur la touche "ENTER".

Le caractère ":" permet de séparer plusieurs instructions par ligne, la structure générale d'une instruction se résume comme suit :

n D E

ou n D E : D' E' : D" E"...

n représente l'étiquette de l'instruction 1<n<9999
D est la directive, le type de l'instruction
E est l'expression (ou les expressions) composée de formules algébriques contenant des nombres, des variables, des opérations arithmétiques et logiques.

Exemples :

```
10 INPUT I,J
20 DISP "***";I+J;"***"
30 GOTO 10
40 END

100 INPUT I,J : DISP I*J : GOTO 100
```

L'ordre d'entrée des lignes peut être quelconque.
Une ligne non précédée d'un numéro est exécutée immédiatement (mode calculateur).

Le contenu d'une ligne peut être affiché sur l'écran en ne tapant que son numéro de ligne.

. DIRECTIVE DU BASIC

On peut classer les directives du langage BASIC en deux groupes différents :

- les instructions BASIC : elles composent l'ensemble d'un programme.

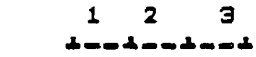
- les commandes BASIC : utilisées pour la création, la mise à jour et la manipulation des programmes.

. LES CONSTANTES

Une constante peut être de deux types différents :

- constantes arithmétiques :
  Une constante arithmétique est un nombre ou une valeur numérique d'au maximum 14 chiffres significatifs compris entre 10^{-63} et 10^{63}.

Exemple :
```
 123
-248
 3.005
-.4206
+4E2
-.52E-27
```
la représentation interne d'une telle constante est la



exposant mantisse : 7 octets
signe mantisse

- constantes littérales :
  Une constante littérale BASIC est une suite de caractères alphanumériques comprise entre deux guillemets ou non. Nous verrons plus loin l'utilité de ces deux représentations.

  Exemple :
  "ZERO",SOMME,"NOMBRE 221"

. VARIABLES

  Trois types de variables sont utilisées :

  . Les variables simples arithmétiques
  . Les variables indicées arithmétiques

  - une variable simple arithmétique représente un nombre. Elle se formule par une suite de caractères lettres ou chiffres non limitée, débutant toujours par une lettre.

    X1 A2 Y TVA SOMME

  - une variable arithmétique indicée est utilisée pour la représentation des tableaux (1,2 ou 3 dimensions). Elle se formule comme la variable simple suivie d'un de deux ou de trois indices (ces derniers étant séparés par des virgules) entre parenthèses.
    A (5,6) TBL (I,J,K)

    A une variable indicée correspond une valeur numérique unique. Les indices peuvent être des expressions arithmétiques, des variables simples ou des nombres. L'omission d'indice permet la représentation complète d'un tableau (utilisé pour les entrées/sorties).

Exemples :

B(5)	représente le 5ème élément du tableau B
MAT(7,J)	se réfère à l'élément de la ligne 7 et de la colonne J de la matrice MAT.
TBL()	désigne le contenu complet du tableau

- Une variable chaine de caractères se représente de la même manière qu'une variable arithmétique suivie du caractère "$"
  A$ LIB$ A7$

  Une variable chaine de caractères est un tableau à une ou deux dimensions et plusieurs formes sont possibles dans chacun de ces deux cas :

- Tableau à une dimension :

A$	représente la chaine de caractères réelle contenue dans le tableau A$
A$(3)	représente le 3ème caractère de la chaine A$
A$(I;J)	se réfère à la sous chaine de caractères comprise entre Ième et Jème caractères inclus (I J)
A$()	désigne le tableau complet

- Tableau à deux dimensions :

A$	représente la chaine de caractères contenue dans A$
A$(K)	désigne la sous chaine de caractères formée par la Kème ligne de A$
A$(I,J)	se réfère au caractère de la ligne I et de la colonne J de A$
A$(I,K;10)	représente la sous chaine de caractères comprise entre Kème et le 10ème caractère sur la Ième ligne du tableau A$.
A$() ou A$	désigne le tableau complet.

RUN

FONCTION

EXECUTION D'UN PROGRAMME

FORME GENERALE

RUN [n]

REGLES

.Si le numéro de ligne n'est pas précisé, le programme sera exécuté à partir du plus petit numéro de ligne.
.Toutes les variables du programme sont initialisées à "zéro" pour les numériques, et à "blanc" pour les chaines de caractères.
.Vérification des branchements en cours de programme.

EXEMPLES

```
0010 I=0
0020 I=I+1
0030 DISP TAB I;I
0040 IF I<4 THEN 0020

0050 END
1000 I=4
1010 I=I-1
1020 DISP TAB I;I
1030 IF I#1 THEN 1010
1040 END

RUN
1
 2
  3
   4
RUN 1000
   3
  2
 1
```

CONTINUE

FONCTION

Reprise d'un programme.

FORME GENERALE

CONTINUE [n]

REGLES

. Si le numéro de ligne n'est pas précisé, le programme reprend l'exécution à la ligne où il a été précédemment arrêté, par :
  - l'instruction STOP
  - "BREAK"
  - une erreur
. Les variables ne sont pas modifiées.
. Les branchements sont vérifiés en cours de programme.
. Pendant l'arrêt du programme toute opération sur les instructions nécessite de préciser le numéro de ligne de reprise de l'ordre CONTINUE (signalé par ERREUR 10)

EXEMPLES

CONTINUE	reprise à la ligne suivante, là où le programme a été arrêté
CONTINUE 160	reprise de l'exécution à la ligne 160

STOP

FONCTION

ARRET DE L'EXECUTION D'UN PROGRAMME

FORME GENERALE

[n] STOP [message]

REGLES

. l'exécution de l'instruction STOP provoque un arrêt du programme avec conservation de l'état des variables et mémorisation de l'instruction suivante à exécuter (reprise ultérieure possible par CONTINUE sans numéro de ligne).

. le message pouvant figurer après STOP sera visualisé avant l'arrêt du programme. Ce message peut se composer de toute combinaison de caractères.

EXEMPLES

```
100 STOP

150 STOP ARRET DU PROGRAMME
```

END

FONCTION

FIN D'UN PROGRAMME

FORME GENERALE

[n] END

REGLES

- l'instruction END met fin à l'exécution d'un programme
- lorsque la commande SEQUENCE est active, END permet d'annuler ce mode.
- après exécution de l'instruction END, les variables sont détruites et ne peuvent donc plus être consultées.

EXEMPLES

```
SEQ'
0010 I=0
0020 I=I+1
0030 DISP I
0040 IF I<100 THEN 20
0050 END
```

MEM

FONCTION

PLACE MEMOIRE RESTANTE EN NOMBRE D'OCTETS

FORME GENERALE

[n] DISP MEM ou V=MEM

REGLES

. Le résultat de cette fonction peut être rangé dans une variable (10 A=MEM) ou testé en cours de programme (20 IF MEM < 1000 THEN 910) pour des protections automatiques de mémoire.

EXEMPLES

```
MEM
02856

DEL 100,120

LIST P
0010 INPUT A,B
0020 DISP A,B
0030 END
0125 DIM A$ [10]
0130 INPUT A$
0135 INPUT I,J
0140 DISP A$ [I;J]
0145 END

MEM                    reste 2925 octets après écriture du
02925                  programme

RUN
 $
MEM
02889                  reste 2889 après la prise en compte
                       des variables
```

NBR

FONCTION

CONNAISSANCE DU NUMERO DE LA DERNIERE LIGNE
EXECUTEE DANS UN PROGRAMME

FORME GENERALE

[n] V=NBR ou DISP NBR

REGLES

- en mode calculateur NBR permet de connaitre où en est un programme arrêté après un BREAK ou activation de la clé INT.
- le numéro de ligne de l'instruction en cours d'exécution est affiché sur l'écran
- le résultat de cette fonction peut être rangé dans une variable (1020 A = NBR) A = 1020

EXEMPLES

```
NBR                 connaissance de la ligne ou le programme
                    est arrêté

100 A=NBR
    -----
    -----
    GOTO A          branchement à la ligne 100
```

DEL

FONCTION

SUPPRESSION DE LIGNES DE PROGRAMME

FORME GENERALE

[n] DEL n1[,n2]

REGLES

- Lorsque deux numéros de ligne sont précisés, il y a suppression de la partie de programme comprise entre ces deux numéros et y compris ces deux numéros.
- Cette instruction peut être programmable (utilisable avec PROG/UNPROG)

EXEMPLES

```
DEL 10
LIST P
0020 DISP A,B
0030 END
0125 DIM A$ [10]
0130 INPUT A$
0135 INPUT I,J
0140 DISP A$ [I;J]
0145 END

DEL 20,30

LIST P
0125 DIM A$ [10]
0130 INPUT A$
0135 INPUT I,J
0140 DISP A$ [I,J]
0145 END
```

SCRATCH

FONCTION

REINITIALISATION DU BASIC
ANNULATION DES PROGRAMMES ET VARIABLES

FORME GENERALE

SCRATCH

REGLES

Cette instruction remet la memoire du systeme dans l'etat dans laquelle elle se trouvait lors du chargement du BASIC

WAIT

FONCTION

MISE EN ATTENTE D'UN PROGRAMME

FORME GENERALE

[n] WAIT Ea

REGLES

- l'expression Ea représente un temps en dixièmes de secondes pendant lequel le calculateur restera en attente.
- cette instruction peut être interrompue par "BREAK"

EXEMPLES

```
10 WAIT 100          attente de 10 secondes

110 WAIT (I*(L-1))
```

REM

FONCTION

INSERTION DE COMMENTAIRES DANS UN PROGRAMME

FORME GENERALE

[n] REM toute combinaison de caracteres

REGLES

- Les premiers caractères "blanc" ne sont pas pris en compte.

EXEMPLES

```
SEQ'
0010 REM PROGRAMME 1
0020 INPUT I
0030 DISP I
0040 IF I#99 THEN 20
0050 REM FIN [I=99]
0060 END
```

FIXED

FONCTION

AFFICHAGE DE NUMERIQUE EN FORMAT FIXE

FORME GENERALE

[n] FIXED [*] Ea

REGLES

. "l'astérisque" indique que l'affichage se fait sans arrondi.

. l'expression donne le nombre de décimales affichées.

. par défaut deux décimales sont affichées

EXEMPLES

10 FIXED 3	affichage de trois décimales avec arrondi
20 FIXED*2	affichage de deux décimales sans arrondi

FLOAT

FONCTION

AFFICHAGE DE NUMERIQUES EN FORMAT FLOTTANT

FORME GENERALE

[n] FLOAT [*] Ea

REGLES

- "l'astérisque" indique que l'affichage se fait sans arrondi.
- l'expression donne le nombre de décimales affichées.
- par défaut deux décimales sont affichées

EXEMPLES

```
20 FLOAT   4     quatre décimales avec arrondi
40 FLOAT*  6     six décimales sans arrondi
                 4.123456  E8
```

DIM

FONCTION

RESERVATION MEMOIRE DES TABLEAUX ET DECLARATION DE LEURS DIMENSIONS MAXIMALES

FORME GENERALE

[n] DIM T(C1[,C2[,C3]])[C4],V$(C5[,C6]),...

REGLES

- les tableaux numériques peuvent avoir une, deux ou trois dimensions
- les valeurs des constantes de dimensions (C1,C2,C3,C5,C6) sont limitées à 65536
- chaque élément du tableau par défaut C4=8

  1<C4<8 détermine le nombre de chiffres par élément suivant la formule (C4 octets-1)*2
- L'encombrement de chaque élément du tableau est d'un octet pour le signe et l'exposant plus 2 chiffres par octet. Si C4 vaut 6, le nombre sera composé de 10 chiffres
- les tableaux alphanumériques peuvent avoir de 1 à 2 dimensions
- au cours d'un programme aucune modification de la dimension d'un tableau n'est permise
- la valeur numérique de l'indice d'un tableau ne pourra dépasser la valeur spécifiée dans la constante de dimension
- un indice nul n'est pas légal
- si une variable indicée n'est pas dimensionnée il y aura un message d'erreur (erreur 14).

EXEMPLES

```
10 DIM A$(20) , T(50)
```
création d'un tableau alphanumérique de 20 caractères et d'un tableau numérique de 50 nombres sur 8 octets chacun.

```
20 DIM B$(10,5),T2(10,2)4
```
création d'un tableau alphanumérique de 10 lignes de 5 caractères et d'un tableau numérique de 10 lignes 2 colonnes chaque nombre étant sur 4 octets soit 6 chiffres possibles en précision.

LET

FONCTION

AFFECTATION D'UNE VALEUR A UNE OU PLUSIEURS VARIABLES

FORME GENERALE

```
[n] Vi[=V2[V3...]]=E
```

REGLES

- Dans de nombreux BASIC cette instruction est appelée instruction LET. Sur ALCYANE, le mnémonique LET ne figure pas. (LET implicite).
- Le type de l'expression doit concorder avec le type de variables auxquelles elle est assignée (constantes alphanumériques entre guillemets).

EXEMPLES

```
10 A=B=C=D=0

100 A$(I;J)=W$(K;L)

120 C$(A)="ABCDEFGHJ"

150 X=INT(ABS(Y+Z)+T(K))
```

INIT

FONCTION

INITIALISATION D'UNE VARIABLE OU D'UN TABLEAU PAR UN OU PLUSIEURS CARACTERES

FORME GENERALE

[n] INIT T TO V

REGLES

- T = Tableau numérique ou chaine à initialiser
  V = Valeur d'initialisation

- Toute variable chaine de caractères est initialisée à "Blanc" et les tableaux à zéro par l'ordre DIM.
  Cette instruction permet de remplir tout ou partie de la chaine avec d'autres caractères spécifiés dans l'instruction et d'initialiser un tableau à une valeur donnée.

EXEMPLES

```
10 INIT A$ TO "*"          A$ = "***..."
20 INIT B$ TO "ABC"        B$ = "ABCABCABC..."
30 INIT C() TO 0           C(1) = C(2) =...= 0
40 INIT D$ TO E$
```

EQUIVALENCE

FONCTION

AFFECTATION D'UNE MEME ADRESSE MEMOIRE A DEUX OU PLUSIEURS VARIABLES

FORME GENERALE

[n] EQUIVALENCE (V1,V2),(V3,V4),...

REGLES

- V1,V2,V3,V4 sont des variables numériques ou alphanumériques. V1 et V3 doivent être connues V2 prendra la même adresse que V1 et V4 l'adresse de V3.

- La place réservée aux variables est allouée à partir de la fin de la mémoire, c'est à dire qu'elles seront classées dans le sens inverse de leur réservation. Les dernières crées seront les premières en mémoire.

- Si la variable qui reçoit son adresse par équivalence est un tableau, elle doit être dimensionnée après l'ordre équivalence.

EXEMPLES

```
0010 DIM A$(10),B$(20),C$(2)
0020 X=10
     en mémoire   X     C$     B$     A$
                  +-----+------+------+
0030 EQUIVALENCE (X,E$)
0040 DIM E$(30)                        E$ regroupe les variables
                                       X,C$,B$
0050 LOAD DATA#(3)"FICH",(1)E$         X,C$ et B$ sont affectés
                                       par cet ordre LOAD DATA#
```

01-3 ENTREES / SORTIES

INPUT

FONCTION

INTRODUCTION DE DONNEES EN COURS DE PROGRAMME
AVEC POSSIBILITE DE TABULATION

FORME GENERALE

[n] INPUT [(E1[,E2])] [texte,]V1[,V2...]

REGLES

. E1, E2 positionnent (ligne, colonne) le curseur pour l'instruction INPUT

. un "?" à l'exécution apparait sur l'écran précédé du texte s'il existe. Introduire alors les constantes satisfaisant en nombre et au mode d'information définis dans la liste de variables de l'instruction INPUT en cours de traitement.

. les constantes devront être séparées par des virgules (voir instruction SEPAR), les constantes alphanumériques ne sont pas mises entre guillemets.

. quatre flags sont liés à cette instruction.

FLAG 16 = 0 (initial)
- message d'erreur si le nombre des constantes est inférieur à la liste

FLAG 16 = 1
- attente jusqu'à satisfaction complète de la liste de l'instruction INPUT

FLAG 15 = 0 (initial)
- les variables non renseignées par des constantes sont remises à zéro

FLAG 15 = 1
- les variables non affectées conservent leur valeur précédente

FLAG 14 = 1
- pas de sortie en erreur si il y a débordement de la saisie par rapport à la dimension de la variable (la saisie est tronquée) dans le cas de chaine de caractère

FLAG 12 = 1
- le contenu de la liste des variables est inchangé si aucune valeur n'est saisie

SORTIE DU MODE INPUT

Pour revenir en mode calculateur ou programmation, répondre à l'affichage de "?" donné par l'instruction INPUT, par le caractère "$".

EXEMPLES

```
10 INPUT "VALEURS",A,B
   affichage du texte VALEURS et saisie de A et B

55 INPUT (10,20) A$
   saisie de la chaine A$ en ligne 10 colonne 20
```

DISP

FONCTION

VISUALISATION DE TEXTES ET DE VALEURS SUR L'ECRAN

FORME GENERALE

[n] DISP liste

REGLES

- voir PRINT
- en mode calculateur (sans numéro de ligne) DISP est implicite.

EXEMPLES

```
30      DISP "ADRESSE" ; AD$ ; "CODE POSTAL" ; CP$

DISP    MEM ou MEM

DISP    2*124/3 ou 2*124/3
```

PRINT

FONCTION

IMPRESSION DE TEXTES ET DE VALEURS

FORME GENERALE

[n] PRINT (np) liste

REGLES

- une expression algébrique quelconque peut figurer dans une liste. Sa valeur sera imprimée lors de l'exécution du programme.
- de même, une chaine de caractères apparaissant dans une liste sera restituée à l'impression.
- le format d'impression d'une ligne pourra être contrôlé par l'emploi de virgules, de points-virgules ou de la fonction TAB.
- chaque ligne d'impression est décomposée implicitement en zones de 16 caractères chacune. La virgule aura pour effet de positionner le prochain caractère à imprimer au début de la zone suivante. Le point-virgule laissera 1 caractère "blanc" entre le prochain caractère d'impression et la dernière entité imprimée.
- la fonction TAB permettra de positionner l'impression du premier caractère à la colonne désirée.

  TAB (Ea), où ea est une expression arithmétique, provoquera un saut en avant à la colonne dont le numéro est la partie entière de Ea dans la ligne d'impression.

- Les nombres imprimés sont toujours justifiés à gauche.

- np représente le numéro de périphérique de sortie (np = 4 est pris par défaut)

EXEMPLES

```
0100  PRINT I,J          impression sur imprimante
0200  PRINT "VALEUR";A
0210  PRINT (2) TAB 3;A$ visualisation de A$
                         le 1er caractère en
                         colonne 3
```

MODIF

FONCTION

AFFICHAGE DE VARIABLES AVEC POSSIBILITE DE TABULATION ET MODIFICATION POSSIBLE DE LEUR CONTENU

FORME GENERALE

[n] MODIF [(E1[,E2])][texte,] liste

REGLES

- liste représente une suite de variables du même type que les ordres DISP ou PRINT
- lors de l'affichage, le curseur vient se positionner en tête de la ligne ou en tête du contenu de la variable toutes les commandes de gestion du clavier sont disponibles pour modifier le contenu des variables.
- la touche RET affecte les valeurs affichées aux variables, cette suite de valeurs doit répondre aux mêmes spécifications que l'instruction INPUT.
- le caractère "$" en tête de la ligne permet de sortir du programme (comme pour INPUT)
- cette instruction peut être utile pour modifier des enregistrements sur disque, sans avoir à les retaper entièrement.
- E1 et E2 permettent d'insérer la tabulation de l'écran (ligne, colonne).
- Texte est une chaine de caractère qui sera visualisé devant le contenu des variables de la liste

EXEMPLES

```
0010 DIM A$ [20]
0020 A$="********************"
0030 MODIF A$
0040 DISP "A$=";A$
0050 GOTO 0030
0060 END
0170 MODIF (10,15) "ADRESSE", AD$
     positionnement du curseur en ligne 10 colonne 15
     de l'écran avec affichage du texte ADRESSE suivi
     du contenu de AD$
```

SEPAR

FONCTION

DEFINITION DU SEPARATEUR DE DONNEES POUR LES INSTRUCTIONS INPUT, MODIF ET DATA A L'INTERIEUR D'UNE CHAINE

FORME GENERALE

[n] SEPAR "2 caracteres hexadecimaux"

REGLES

- Le caractère de séparation est donné sous la forme de son code ASCII codé hexadécimal.
- Cette instruction est utile pour pouvoir saisir des virgules dans les textes

EXEMPLES

30 SEPAR "21"	le "!" servira de séparateur
40 SEPAR "2C"	la "," retrouve sa fonction initiale

WRITE

FONCTION

ECRITURE DE VALEURS SUR UN PERIPHERIQUE ASSOCIE AU FORMAT AVEC SORTIE FIN DE LISTE, ERREUR

FORME GENERALE

[n] WRITE(np,[nf[,n1[,n2]]])liste

REGLES

np : numéro du périphérique
(4 imprimante, 2 écran)
nf : numéro de la ligne auquel correspond les spécifications d'écriture de la liste ou, expression (sauf élément de tableau numérique), donnant le numéro de ligne ou, variable chaine de caractères renfermant les spécifications du format.
n1 : pas de signification
n2 : numéro de la ligne auquel se branchera le programme si une erreur de liaison intervient
liste : liste de variables numériques ou alphanumériques

. np,nf,n1,n2 peuvent être des expressions
. L'absence de "format" provoque une écriture en binaire de la liste des variables. Dans ce cas la liste ne peut être que numérique et la valeur des nombres à écrire doit être comprise entre 0 et 255 (1 octet).

. nf peut être le nom d'une chaine de caractères contenant les spécifications du format entre parenthèses.

EXEMPLES

```
0100 TAB 0 10,1
0110 WRITE(2,120)QTE,PU
0120 FORMAT("QUANTITE :",N5," PRIX UNITAIRE :",N7.2)
0130 RETURN

1000 A$="(5B,"*",N5.2,"*")
1010 WRITE(I,A$)A
```

FORMAT

FONCTION

CONTIENT LES SPECIFICATIONS UTILES AUX INSTRUCTIONS READ,WRITE

FORME GENERALE

[n] FORMAT(specifications)

REGLES

- les spécifications peuvent être ou non séparées par des virgules :
  FORMAT (10B,"**",/,N4.2) ou
  FORMAT (10B"**"/N4.2)
- les spécifications peuvent être répétées plusieurs fois en les précédant d'un coefficient
  5B,3"*"
  au lieu de B,B,B,B,B,"***"
- toute erreur de spécification n'est décelée que lors de l'exécution et le numéro de ligne donné par le BASIC est celui du READ ou du WRITE associé
- les spécifications relatives au mode d'écriture des variables de la liste doivent être de même type que celles-ci
- des spécifications peuvent être regroupées par des parenthèses si elles sont répétitives
  10 FORMAT (/,3(5B,"*"))
- en fin de spécification, si la liste des variables n'est pas épuisée, il y a réexploration du format à partir de la dernière parenthèse gauche
- un débordement de format (spécification inférieure à la valeur demandée) est signalé par des astérisques.

SPECIFICATIONS

Une spécification est un ordre donné au périphérique ou un mode d'écriture relatif à une variable.

Ordre de format :

*	passage à la page suivante
/	passage à la ligne suivante (saut de ligne)

+	surimpression (l'imprimante reste sur la même ligne pour écrire les spécifications suivantes)
"..."	les libellés mis entre double guillemets sont restitués tels quels
'...'	les libellés entre guillemets sont restitués en caractères étendus ou gras suivant le type de l'imprimante et négatifs sur l'écran
B	écriture d'un "blanc" (espace)
-...-	entre deux tirets les spécifications sont inversées (caractères négatifs sur l'écran, étendus sur l'imprimante)
H	Sortie de caractères codés en hexadécimal

Mode d'écriture des variables :
Variables numériques :

Na.b	numérique justifié à droite, a donné le nombre de caractères total du nombre (signe et point compris), b indique le nombre de décimales. N4.2 4 positions xxxx 1 chiffre 2 décimales
Ra .b	identique à N mais calage du nombre à gauche.
Za. b	identique à N mais si la valeur est nulle elle n'est pas éditée
Fa .b	format flottant

Variables alphanumériques :

A n	alphanumérique de n caractères. Si la variable est de longueur supérieure à n, n sera le maximum attribué à cette variable. Dans le cas d'une variable plus courte, le reste est complété par des caractères "blancs".
En	identique à A mais en négatif sur l'écran et étendu ou gras sur l'imprimante.

Commandes pour l'introducteur frontal

I	Input
O	Output
D	Down
U	Up

FORMAT LIBRE

.Un format libre est un format sans spécifications
    FORMAT ()
Il est utilisé pour les entrée/sorties séries ou sur fichier séquentiel pour lire ou écrire une chaine de caractères sans mise en page

EXEMPLES

```
0100 WRITE(4,120)QTE,DES$,PU,MONT
0120 FORMAT(5B,"QUANTITE",5B,"DESIGNATION",5B,"PRIX UNITAI
RE",5B,"MONTANT",/,5B,N5,8B,A10,5B,N7.2,11B,N7.2)
```

KEYIN

FONCTION

SAISIE D'UN CARACTERE SUR LE CLAVIER SANS VISUALISATION

FORME GENERALE

[n] KEYIN V$,n1

REGLES

. l'exécution de cette instruction provoque une scrutation clavier. Si aucune touche n'est enfoncée à ce moment-là le programme continue en séquence. Dans le cas contraire le caractère est stocké dans la variable chaine V$ et le programme exécute l'instruction de numéro n1.

EXEMPLES

Saisie de 5 caractères avec l'ordre KEYIN

```
LIST P
0010 DIM A$[5]
0020 FOR I=1 TO 5
0030 KEYIN A$[I],0040:GOTO 0030
0040 DISP A$
0050 NEXT I
0060 END

RUN
A
AB
ABC
ABCD
ABCDE
```

SUBST

FONCTION

SUBSTITUTION D'UNE SOUS-CHAINE PAR UNE AUTRE DANS UNE VARIABLE CHAINE

FORME GENERALE

[n] SUBST [ALL] chaine1 BY chaine2 IN V$

REGLES

- chaine 1 ou chaine 2 peuvent être soit un ou plusieurs caractères entre guillemets, soit une variable chaine de caractères.
- il y a substitution dans la chaine V$ de tous les caractères si ALL est spécifié ; sinon, les premiers caractères rencontrés de la chaine sont substitués.

EXEMPLES

```
0010 DIM A$[10]
0020 INPUT A$
0030 DIM B$[5]
0040 SUBST "0" BY "*" IN A$ (I;J)
0050 DISP A$
0060 INPUT B$
0070 SUBST ALL " " BY B$ IN A$ (I;J)
0080 DISP A$
0090 END

RUN
0001235
***1235
+
***1235+++
```

SEARCH

FONCTION

RECHERCHE DE LA POSITION D'UNE SOUS-CHAINE DANS UNE VARIABLE CHAINE DE CARACTERES (=,<,>,#)

FORME GENERALE

[n] SEARCH V IN V1$=V2$ [STEP Ea]

REGLES

.V est la position recherchée dans la chaine de caractères V1$ avec la condition <,= ,> ou # à une chaine de caractères V2$

. STEP Ea s'il est présent indique le pas de recherche dans V1$, il peut être positif ou négatif :
Ea >0 recherche à partir du début de V1$
Ea <0 recherche à partir de la fin de V1$

. si STEP Ea n'est pas mentionné la recherche se fera avec un pas d'un caractère

EXEMPLES

```
10 DIM A$(50),B$(50)
20 A$="MONSIEUR MARTIN"
30 B$="MADEMOISELLE DUPONT"
40 SEARCH V1 IN A$("A"
50 SEARCH V2 IN B$("A"
60 DISP V1;V2
70 END
RUN
9.00 13.00
```

01-4 BRANCHEMENTS

GOTO

FONCTION

EFFECTUE UN BRANCHEMENT INCONDITIONNEL VERS UNE AUTRE INSTRUCTION

FORME GENERALE

```
[n] GOTO n1
    GOTO Ea
```

REGLES

- l'exécution du programme passera à l'instruction portant le numéro n1 ou la ligne dont le numéro est la partie entière de l'expression Ea.

- on ne pourra pas sortir d'un sous programme (entrée par GOSUB) sans passer par RETURN, à moins que ce branchement se fasse sur un RETURN CLEAR.

EXEMPLES

```
0010 A=B=0
0015 B=B+1
0020 E=30
0030 GOTO E+10
0040 PRINT TAB B;A
0050 A=B*A
0060 FOR I=1 TO 100
0070 A=A+B
0080 IF B>20 THEN 0100
0090 GOTO 0015
0100 B=A-2
0110 NEXT I
0120 END
```

GOTO OF

FONCTION

EFFECTUE UN BRANCHEMENT CALCULE

FORME GENERALE

```
[n1 GOTO Ea OF n1,n2,n3...
    GOTO Ea OF Ea1,Ea2,Ea3...
```

REGLES

. renvoie l'exécution du programme au numéro de ligne dont le rang est indiqué par la valeur entière de l'expression

EXEMPLES

```
100 GOTO N OF 10,100,150,200
```

si N = 3 branchement à la ligne 150

FOR/NEXT

FONCTION

REPETITION D'UNE SEQUENCE D'INSTRUCTIONS (BOUCLE)

FORME GENERALE

```
[n1] FOR Vs = E1 TO E2 [STEP E3]
[n2] NEXT V$
```

REGLES

- une variable simple vs appelée indice de la boucle
- une valeur initiale E1 qui sera affectée à l'indice dès la première entrée dans la boucle
- un pas E3 dont la valeur de l'indice sera incrémentée chaque fois que l'instruction NEXT sera exécutée
- enfin, une valeur finale E2 qui provoquera la fin d'exécution de la séquence d'instruction de la boucle lorsque la valeur de l'indice aura dépassé cette valeur E2.
- la spécification de l'incrément est optionnelle. Si elle fait défaut, un pas de valeur 1 sera pris implicitement
- un pas négatif est permis dans une spécification d'incrément.
- la valeur de l'indice peut être modifiée dans une boucle FOR. Dans ce cas, la nouvelle valeur de cet indice se substituera à l'ancienne. Elle sera incrémentée ou décrémentée du pas spécifié dans l'instruction FOR. De même, on pourra utiliser cet indice à des fins de calcul au sein de la boucle.
- il est possible de rentrer dans une boucle (ou d'en sortir) par des branchements conditionnels ou inconditionnels.
  Toutefois, on devra procéder avec précaution lorsqu'on se branchera dans une boucle sans initialiser ses paramètres, car les valeurs de ces paramètres seront évaluées si, et seulement si, l'instruction FOR est exécutée.

. une boucle ne sera pas exécutée si les conditions de fin de boucle sont satisfaites à la première entrée, donc si vs>E2.

. plusieurs boucles FOR peuvent s'imbriquer les unes dans les autres. Mais elles ne doivent pas se chevaucher. Chaque boucle doit être un sous-ensemble de l'autre.

EXEMPLES

```
0010 DIM A[25]
0020 FOR I=1 TO 25
0030 A[I]=I
0040 NEXT I
0050 END
```

```
0010 INPUT I,J
0020 FOR K=I TO J
0030 DISP K
0040 NEXT K
0050 END
```

```
0100 FOR X=Z*V/T TO F STEP -3
0110 FOR I=1 TO X STEP 2
0120 FOR N=I TO 100
0130 TBL(I,N)=A(X)
0140 NEXT N
0150 NEXT I
0160 DISP A(X)
0170 NEXT X
0180 END
```

IF ... THEN

FONCTION

BRANCHEMENT CONDITIONNEL

FORME GENERALE

[n] IF expression conditionnelle THEN n1

REGLES

. l'instruction se décompose en deux parties :
  - une partie relationnelle
  - une partie branchement

. si l'expression conditionnelle est vraie, il y aura branchement au numéro de ligne spécifié, dans le cas contraire, le "THEN" restera sans effet.

. la comparaison sur deux chaines de longueur différente se fait sur la plus petite

EXEMPLES

```
10 A=0
20 A=A+1
30 IF A>30 THEN 60
40 DISP A
50 GOTO 20
60 END
```

IF . . . :

FONCTION

EXECUTION DES INSTUCTIONS SUIVANTES DANS LE CAS OU LA CONDITION EST VERIFIEE

FORME GENERALE

[n] IF condition :

REGLES

- si l'expression conditionnelle est vraie, il y aura exécution des instructions situées sur la même ligne, sinon le programme exécutera la ligne de numéro immédiatement supérieure à celui du IF
- la comparaison de deux chaines de caractères de longueurs différentes s'effectue sur la plus petite

EXEMPLES

```
0110   IF A>B : A=0
0120   A = A+1

0540   IF A$ = B$ : DISP A$ : GOTO 510
0545   C$ = A$
```

IF ( )

FONCTION

TESTE LA VALEUR NEGATIVE, NULLE, POSITIVE D'UNE EXPRESSION ALGEBRIQUE

FORME GENERALE

```
[n] IF (Ea) [n1],[n2],[n3]
    IF (Ea) [Ea1],[Ea2],[Ea3]
```

REGLES

. la condition de branchement est la suivante :

```
Ea ( 0  saut à la ligne n1
Ea = 0   "          "    n2
Ea ) 0   "          "    n3
```

. si le numéro de ligne convenant à la valeur de l'expression est absent, le programme continue en séquence.

EXEMPLES

```
0020 IF (A-B) 30,50,120
        si A(B branchement en 30
        si A=B branchement en 50
        si A)B branchement en 120

0060 IF (A-B) 30,,30
0070 DISP A
     Si A)B ou A(B branchement en 30
     Si A=B branchement en 70
```

IF ERR

FONCTION

EXECUTION DE LA LIGNE SPECIFIEE EN CAS D'ERREUR BASIC DANS LE PROGRAMME

FORME GENERALE

```
[n] IF ERR(VS1,VS2) GOTO n1
    IF ERR(VS1,VS2) GOTO Ea
```

REGLES

- après détection d'une erreur BASIC il y a branchement au numéro de ligne spécifié par GOTO avec comme paramètres :
  VS1 = numéro de la ligne où l'erreur s'est produite
  VS2 = numéro de l'erreur rencontrée.

- dans un programme l'instruction IF ERR n'est effective qu'à partir du moment où elle a été rencontrée.

- IF ERR est annulé par l'instruction VALID ERR

EXEMPLES

```
0010 IF ERR [X,Y] GOTO 0100
0020 INPUT A
0030 DISP A
0040 GOTO 0020
0050 END
0100 DISP "ERREUR";Y;"LIGNE";X;"RETAPEZ LA DERNIERE VALEUR"
0110 GOTO 0020

RUN
12
12
ZE
ERREUR 17 LIGNE 20 RETAPEZ LA DERNIERE VALEUR
245
245
10
```

VALID ERR

FONCTION

ANNULATION DE L'EFFET D'UN IF ERR

FORME GENERALE

VALID ERR

REGLES

.Apres l'éxécution de cette instruction si un IF ERR
a été rencontré auparavent il n'est plus operationnel
.Si une erreur basic est rencontréé elle s'affichera sous
la forme: ** ERREUR xxxx **
** LIGNE xxxx **
et le programme s'arretera
.Dés le chargement d'un autre programme, IF ERR est
implicitement annulé, comme lors du chargement de sous
programmes.

EXEMPLES

```
200 VALID ERR
```

GOSUB

FONCTION

APPEL A UN SOUS PROGRAMME RESIDENT

FORME GENERALE

```
[n] GOSUB n1
    GOSUB Ea
```

REGLES

- L'instruction GOSUB passera le contrôle du programme à la ligne définie par le numéro n1 ou la valeur entière de Ea. Le traitement se déroulera jusqu'à la ligne portant l'instruction RETURN.
- Les instructions peuvent être imbriquées les unes dans les autres pourvu qu'une instruction RETURN soit rencontrée pour chaque GOSUB exécuté.
- L'utilisation récursive de cette instruction est possible.

EXEMPLES

```
50 INPUT A
60 GOSUB 1000
70 DISP A
80 GOTO 50

1000 A = INT(V*100+0.5)/100
1010 RETURN
```

GOSUB OF

FONCTION

APPEL A UNE SEQUENCE D'INSTRUCTIONS SUIVANT Ea

FORME GENERALE

```
[n] GOSUB Ea OF n1,n2,n3...
    GOSUB Ea OF Ea1,Ea2,Ea3...
```

REGLES

- identique à GOSUB
- branchement au numéro de ligne dont le rang est indiqué par la valeur entière de Ea (si Ea = 2, branchement a n2).

EXEMPLES

```
100 GOSUB A3 OF 1000,2000,3000
        si A3=1 appel de la séquence 1000
        si A3=2 appel de la séquence 2000
        si A3=3 appel de la séquence 3000

200 GOSUB TEST OF I,J,K
```

RETURN

FONCTION

RETOUR D'UN SOUS PROGRAMME APPELE PAR GOSUB

FORME GENERALE

[n] RETURN

REGLES

- fin de l'exécution d'un sous programme et retour à l'instruction immédiatement après celle de son appel par GOSUB (ou GOSUB OF)

EXEMPLES

```
50 INPUT A
60 GOSUB 1000
70 DISP A
80 GOTO 50

1000 A=INT(V*100+0.5)/100
1010 RETURN
```

RETURN CLEAR

FONCTION

ANNULATION D'UN APPEL EFFECTUE PAR GOSUB

FORME GENERALE

[n] RETURN CLEAR

REGLES

Annulation de l'adresse de retour après exécution d'un sous programme appelé par GOSUB (ou GOSUB OF)

EXEMPLES

```
100 A=B+C
110 GOSUB 2000
120 DISP A

2000 A=INT(V*100+0,5)/100
2010 IF A>33 THEN 2030
2020 RETURN
2030 RETURN CLEAR
2040 GOTO 500
```

BREAK

FONCTION

SUBSTITUTION DU "BREAK" PAR UN SOUS PROGRAMME BASIC

FORME GENERALE

[n] BREAK ON numéro de ligne
expression
exp OF n1, n2,....

[n] BREAK OFF

REGLES

- BREAK ON permet de substituer l'arret d'un programme par "BREAK" par l'exécution d'un sous programme dont le numéro est spécifié dans l'instruction.
- Ce sous programme appelé doit se terminer par RETURN afin de retourner après l'instruction ou s'est produit le "BREAK"
- BREAK OFF remet en état le fonctionnement du "BREAK" BASIC

EXEMPLES

```
10   BREAK ON 1000
                          affichage du message "ALCYANE"
                          chaque fois que le BREAK est
                          actionné
1000 DISP "ALCYANE"
1010 RETURN
5000 BREAK OFF            Le mode normal du BREAK est
                          restitué
```

01-5 GESTION DES FLAGS

SET FLAG

FONCTION

MISE A 1 D'UN OU PLUSIEURS FLAGS

FORME GENERALE

[n] SET FLAG Ea1 [,Ea2[,Ea3...]]

REGLES

- Les expressions représentent les numéros des FLAGS à mettre à un.
- Ces flags sont au nombre de 32 dont certains sont utilisés pour le système (voir FLAG SYSTEME).

EXEMPLES

```
10   SET FLAG 6

60   SET FLAG A,B,C
```

CLEAR FLAG

FONCTION

MISE A ZERO DE FLAGS

FORME GENERALE

[n] CLEAR FLAG Ea1[,Ea2[,Ea3...]]

REGLES

- Les expressions représentant les numéros de FLAGS à mettre à zéro.

EXEMPLES

```
0050 CLEAR FLAG 21,32
```

CPL FLAG

FONCTION

COMPLEMENTATION D'UN OU PLUSIEURS FLAGS

FORME GENERALE

[n] CPL FLAG Ea1 [,Ea2[,Ea3...]]

REGLES

. Les expressions indiquent les numéros du FLAG que l'on veut complémenter.

EXEMPLES

```
310   CPL  FLAG  4

450   CPL  FLAG  7,14,12
```

FLAG

FONCTION

INTERROGATION DES DIFFERENTS FLAGS

FORME GENERALE

FLAG Ea

REGLES

Ea = Expression représentant le numéro du FLAG à traiter.
Résultat = valeur du flag (0 ou 1)

EXEMPLES

```
120  IF FLAG 10 THEN 210

560  DISP FLAG (L*(I-1))
```

FLAGS SYSTEME

FONCTION

CERTAINS FLAGS SONT UTILISES POUR ORIENTER LE SYSTEME.

FORME GENERALE

CES FLAGS SONT GERES PAR L'UTILISATEUR PAR LES FONCTIONS DE GESTION DE FLAG NORMALES (SET FLAG,CLEAR FLAG,CPL FLAG, FLAG)

REGLES

FLAG 6 MIS A 1 IL PERMET L'IMPRESSION OU LA VISUALISATION DES INSTRUCTIONS D'UN PROGRAMME AVEC UNE INSTRUCTION PAR LIGNE

FLAG 7 MIS A 1 IL PERMET DE CONSERVER LES CARACTERES BLANCS LORS D'UNE CONCATENATION DE CHAINES DE CARACTERES

FLAG 8 MIS A 0 LE GENERATEUR DE NOMBRES ALEATOIRES EST REINITIALISE

FLAG 9 MIS A 1 IL CONTROLE LE FORMAT DES VARIABLES

FLAG 10 LORSQU'IL EST A 1 IL INDIQUE LA PRESENCE D'HOMONYMES DANS UNE LECTURE EN SEQUENTIEL INDEXE

FLAG 11 MIS A 1 L'IMPRESSION ET LA VISUALISATION DU Ø EST REMPLACEE PAR O AVEC LES ORDRES PRINT OU DISP

FLAG 12 MIS A 1 GARDE L'ANCIENNE VALEUR D'UNE LISTE DE VARIABLES SUIVANT L'ORDRE INPUT OU MODIF SI ELLES NE SONT PAS RENSEIGNEES LORS DE SON EXECUTION

FLAG 13 MIS A 1 IL INTERDIT LE BREAK

FLAG 14 MIS A 1 IL PERMET DE NE PAS SORTIR EN ERREUR SI IL Y A DEBORDEMENT DE VARIABLE LORS DE L'EXECUTION DES ORDRES INPUT OU MODIF

FLAG 15 MIS A 1 GARDE L'ANCIENNE VALEUR DE LA VARIABLE SUIVANT L'ORDRE INPUT OU MODIF SI ELLE N'EST PAS RENSEIGNEE LORS DE LEUR EXECUTION

FLAG 16 MIS A 1 LE PROGRAMME REVIENT SUR L'ORDRE INPUT SI LOR3 DE LA SAISIE LA VARIABLE N'A PAS ETE RENSEIGNEE

TAB

FONCTION

TABULATION LORS D'UNE IMPRESSION OU VISUALISATION

FORME GENERALE

[n] PRINT(ou DISP) TAB Ea;VN ou V$

EXEMPLES

```
30 MBC$ = "ALCYANE"
50 PRINT TAB 16 , MBC$
Impression de la valeur de MBC$ après 15 blancs sur la ligne
50 PRINT TAB 16;"ALCYANE"
même résultat
```

CVT

FONCTION

CONVERSION NUMERIQUE/ALPHANUMERIQUE

FORME GENERALE

CVT,V

REGLES

- Initialisation d'une chaine ou d'une partie de chaine à la valeur de la variable V
- le nombre de décimales est défini par FIXED
- si l'intervalle, dans la chaine est plus grand que le nombre, celui-ci est calé à gauche

EXEMPLES

```
10 DIM A$(20)
20 INIT A$ TO "X"
30 FIXED 3 : B=2,45
40 A$(5,10)=CVT B
50 DISP A$
60 END
RUN
XXXX2.450 XXXXXXXXXX
```

VAL

FONCTION

CONVERSION ALPHANUMERIQUE/NUMERIQUE

FORME GENERALE

VAL,V$

REGLES

. Conversion de la variable chaine V$ dans une valeur numérique

EXEMPLES

```
10 DIM A$(5)
20 A$="123456789"
30 FOR K=1 TO 9
40 B=VAL A$(K)
50 C=B*B
60 PRINT C
70 NEXT K
80 END
RUN
1
4
9
16
25
36
49
64
81
```

ASC

FONCTION

VALEUR DECIMALE D'UN CARACTERE ALPHANUMERIQUE

FORME GENERALE

ASC V$

REGLES

. Conversion du caractère codé en ASCII en valeur décimale (de 0 à 255).

EXEMPLES

```
10 DIM A$(7)
20 A$="ALCYANE"
30 FOR K=1 TO 7
40 PRINT(2) ASC A$(K) ou V=ASC A$(K)=DISP V
50 NEXT K
60 RUN
```

Visualisation de

65
76
67
89
65
78
69

BIN

FONCTION

CONVERSION D'UN NOMBRE DECIMAL DE 0 A 255
EN BINAIRE SUR UN OCTET

FORME GENERALE

```
BIN V
```

REGLES

. La valeur de V (de 0 à 255) est transformée en caractère ASCII.

EXEMPLES

```
10 DIM A$(10)
20 INIT A$ TO BIN 42
30 PRINT A$
40 END
RUN
**********
```

NUM

FONCTION

LONGUEUR, A PARTIR DU DEBUT DE LA CHAINE DE CARACTERES, DE LA PARTIE NUMERIQUE

FORME GENERALE

```
NUM V$
```

REGLES

- Le résultat de cette fonction représente le nombre de numériques contenu en tête de la chaine de caractères désignée (les caractères "+","-" et "." sont considérés comme numérique)

EXEMPLES

```
10 DIM A$(50)
20 A$="145 AVENUE DE PARIS"
30 B=NUM A$
40 PRINT A$(B+2;50)
50 END
RUN
AVENUE DE PARIS
```

HEX

FONCTION

CONVERSION ASCII/HEXADECIMAL

FORME GENERALE

HEX"caracteres hexadecimaux" ou HEX V$

REGLES

. compactage d'une chaine alphanumérique codée en hexadécimal

EXEMPLES

```
10 DIM A$(7),B$(14)
20 B$="414C4359414E45"
30 A$=HEX B$
40 PRINT A$
50 END
RUN
ALCYANE
```

LEN

FONCTION

LONGUEUR D'UNE CHAINE DE CARACTERES

FORME GENERALE

LEN V$

REGLES

.le résultat de cette fonction donne le nombre de caractères contenu dans la chaine

EXEMPLES

```
10 DIM A$(20)
20 A$="MBC"
30 L=LEN A$
40 A$(L+1;20)="ALCYANE"
50 L2=LEN A$

65 PRINT L2
70 END
RUN
MBC ALCYANE
11
```

INT

FONCTION

PARTIE ENTIERE D'UNE EXPRESSION ALGEBRIQUE

FORME GENERALE

ABS Ea

REGLES

. L'expression est calculée et affectée du signe +

EXEMPLES

```
100 PI=3,14
180 EX=2,71
190 TTL=PI+EX          ici TTL vaut 5,85
200 TTL=INT TTL        ici TTL vaut 5,00
```

ABS

FONCTION

VALEUR ABSOLUE D'UNE EXPRESSION ALGEBRIQUE

FORME GENERALE

REGLES

EXEMPLES

```
100 A=140

160 B=80

200 C=ABS(B-A)
210 D=A-B                          ici C et D valant 60
```

SGN

FONCTION

SIGNE D'UNE EXPRESSION ALGEBRIQUE

FORME GENERALE

SGN Ea

REGLES

. Le résultat vaut (+1 SI Ea)=0,-1SI Ea(0)

EXEMPLES

```
100 A=-1
100 B=256*SGN A                    ici B vaut -256
120 B=B*SGN A                      ici B vaut +256
```

CAP

FONCTION

CONVERSION MINUSCULE/MAJUSCULE

FORME GENERALE

```
CAP V$
```

REGLES

. tous les caractères de la chaine V$ sont transformés en majuscule

EXEMPLES

```
10 DIM A$(6),B$(6)
20 INPUT A$
30 DISP A$
40 B$ = CAP A$
50 DISP B$
60 END
RUN
a b c d e
a b c d e
A B C D E
```

LEFT

FONCTION

CADRAGE A GAUCHE D'UNE CHAINE DE CARACTERES

FORME GENERALE

V1$=LEFT V2$

REGLES

- V1$ et V2$ sont deux chaines de caractères
- Après exécution de la fonction, V1$ contient la valeur de V2$ mais cadrée à gauche (aucun blanc en tête de V1$)

EXEMPLES

```
10 DIM A$(10)
20 INPUT A$
30 A$=LEFT A$
40 DISP A$
50 STOP
RUN
   123.5
123.5
```

RIGHT

FONCTION

CADRAGE A DROITE D'UNE CHAINE DE CARACTERES

FORME GENERALE

V1$=RIGHT V2$

REGLES

. V1$ et V2$ sont deux chaines de caractères

. Après exécution de la fonction, V1$ contiendra la valeur de V2$ mais cadrée à droite (aucun blanc à la fin de la chaine V1$)

EXEMPLES

```
10 DIM A$(10),B$(10)
20 INPUT A$
30 B$=RIGHT A$
40 DISP B$
50 STOP
RUN
123.5
      123.5
```

01-7 OPERATEURS

LISTE DES OPERATEURS

FONCTION

OPERATEURS ARITHMETIQUES OU LOGIQUES

FORME GENERALE

1- OPERATEURS LOGIQUES

OR	'ou' logique
AND	'et' logique
NOT	négation
>	plus grand que
<	plus petit que
>=	plus grand ou égal
<=	plus petit ou égal
=	égal
#	différent que

2- OPERATEURS ARITHMETIQUES

*	multiplication
/	division
+	addition
-	soustraction
&	concaténation

REGLES

- Les opérateurs logiques sont utilisés dans les expressions conditionnelles (IF .. THEN , IF (exp) ,...). Le resultat de ces expréssions peut avoir deux valeurs: VRAI / FAUX.
- Les operateurs arithmetiques sont utilisés dans les expréssions de calcul arithmétiques.
- L'opération de concaténation (&) s'éffectue entre deux chaines de caractères. Le résultat de cette opération sera aussi une chaine de caractéres

EXEMPLES

```
0100 INPUT A,B
0110 IF A=B THEN 0150
0120 DISP "A#B"
0130 GOTO 0100
0150 DISP "A=B"
0160 GOTO 0100
0200 INPUT A
0210 IF A<10 OR A>100 THEN 0250
```

02

GESTION DISQUE

02-1 INSTRUCTIONS GENERALES

CAT

FONCTION

TRANSFERT DANS UNE CHAINE DE CARACTERES DU CATALOGUE DU DISQUE DESIGNE

FORME GENERALE

[n] V$=CAT(nd,nv)

REGLES

. nd : numéro de drive
  nv : numéro de disque

. le contenu du catalogue est transferé dans la chaine de caractères V$ sous la forme :
  - 6 caractères pour le nom de fichier
  - 5 caractères numériques pour la taille du fichier
  - 5 caractères numériques pour la longueur de l'enregistrement
  - 5 caractères numériques pour la position du fichier

EXEMPLES

```
CATAL#(3)
     PROG   00002   0000   00051
```

```
0010 DIM A$(100)
0020 A$=CAT(3)
0030 DISP A$(1;LEN A$)
0040 END
```

```
>RUN
     PROG   00002000000051DATA   00010000050053
```

CATAL#

FONCTION

LISTE DES FICHIERS SUR UN DISQUE

FORME GENERALE

[n] CATAL#[(nd[,nv])]

REGLES

. nd = numéro du drive
  nv = numéro du disque

. Les différentes colonnes indiquant :

- le nom du fichier
- le nombre de secteurs d'un fichier programme ou le nombre d'enregistrements pour un fichier de données
- la longueur de l'enregistrement (nulle pour les fichiers programmes)
- le numéro du premier secteur du fichier

EXEMPLES

```
PROG   00002   00000   00051
DATA   00250   00080   00053
```

Première ligne : fichier programme PROG (troisième colonne à zéro)
occupant deux secteurs sur le disque et implanté à partir du secteur 51

Deuxième ligne : fichier de données DATA de 260 enregistrements de longueur 80 octets chacun et implanté à partir du secteur 53

FMT#

FONCTION

INITIALISATION D'UN DISQUE

FORME GENERALE

[n] FMT#[nd,nv]

REGLES

- nd = numéro du drive
  nv = numéro du disque

- Initialisation du disque placé dans le drive nd et lui attribue le numéro nv (5 chiffres maximum)

- Initialisation d'un support disque s'effectue en trois temps
  1 - écriture des entêtes de chaque secteur du support
  2 - relecture séquentielle et contrôle de ces entêtes
  3 - écriture et informations sur chacun des secteurs et contrôle de parité

- les opérations 2 et 3 s'accompagnent de l'édition, dans le coin supérieur droit de l'écran du numéro du secteur en cours ; si un problème d'écriture ou de lecture se manifeste, le BASIC le signale par affichage du numéro du secteur défaillant, sous le premier compteur ; si l'erreur est récupérée l'opération de formatage continue. Dans le cas d'une erreur persistente, le BASIC le signale en arrêtant l'opération de formatage et en affichant erreur 51. Dans ce cas le disque présente une anomalie et par conséquent n'est pas utilisable.

- cette opération de formatage est indispensablee préalablement à tout emploi du support.

EXEMPLES

```
FMT#(7)
        formatage sur drive 7
FMT#(2,450)
        formatage sur drive 2 avec le numéro de disque 450
```

FREEDISK

FONCTION

CONNAISSANCE DE LA PLACE LIBRE EN OCTETS SUR DISQUE

FORME GENERALE

[n] V=FREEDISK(nd[,nv])

REGLES

- nd représente le numéro d'unité disque sur laquelle la place restant libre est demandée
- nv représente le numéro du disque
- le résultat obtenu par cette fonction représente le nombre d'octets non encore réservés sur le disque pour des fichiers données ou programmes.

EXEMPLES

```
) FREEDISK(4)                place disponible sur le disque 4
103282
)
120 T=FREEDISK(3)            rangement dans la variable T de la
                             place disponible du disque 3
```

DISKNB

FONCTION

CONNAISSANCE DU NUMERO DE DISQUE (VOLUME)

FORME GENERALE

[n] V=DISKNB [nd]

REGLES

- nd : numéro d'unité de disque
- le résultat obtenu par cette fonction donne le numéro de disque affecté lors du formatage (FMT#)

EXEMPLES

```
)FMT#(3,212)
)DISKNB(3)
212

150 A=DISKNB(7)     le numéro de la disquette sur l'unité
                    7 est rangé dans A
```

PROTECT

FONCTION

PROTECTION D'UN DISQUE EN ECRITURE

FORME GENERALE

[n] PROTECT ON (ou OFF) [(nd[,nv])]

.nd = numéro d'unité de disque

. cette protection est liée au support disque, toutes les instructions disques d'écriture sont interdites (erreur 64) sauf l'instruction d'initialisation du disque.

. le déverrouillage s'effectue par PROTECT OFF (nd)

EXEMPLES

PROTECT ON(3) — interdiction d'écriture sur le disque dans l'unité 3. Pour un emploi ultérieur en écriture il faudra exécuter PROTECT OFF(3).

CONTROL

FONCTION

ANNULATION OU ACTIVATION DES CONTROLES DE PARITE SUR DISQUES

FORME GENERALE

[nI CONTROL ON (ou OFF) (nd)

REGLES

- cette instruction n'est à employer qu'avec beaucoup de précautions lors de problèmes de lecture de disques (parité; erreur 95)
- cette instruction est liée au lecteur lui même et un changement de disque n'intervient pas sur cette instruction.

EXEMPLES

CONTROL OFF(7)	inibition des contrôles de parité sur l'unité 7

DRIVE

FONCTION

MISE EN MARCHE OU ARRET D'UNE UNITE DE DISQUE

FORME GENERALE

DRIVE ON [nd] DRIVE OFF [nd]

REGLES

. nd = numéro de l'unité de disque

. sur toutes les instructions disques un DRIVE ON est implicite

EXEMPLES

```
DRIVE ON(3)
mise en marche du disque 3

DRIVE OFF(4)
arrêt du disque 4
```

COPY ON#

FONCTION

DUPLICATION PHYSIQUE COMPLETE D'UN DISQUE

FORME GENERALE

[n] COPY ON#(nd1,nv1)/(nd2,nv2)

REGLES

- Cette duplication ne se fait que sur le même type de support disque
- Si un fichier est protégé sur le disque original, la copie ne se fait pas (erreur 74).

EXEMPLES

COPY ON# (3)/(4)
copie physique du disque 3 sur le disque 4

FILE

FONCTION

TEST DE PRESENCE OU NON PRESENCE D'UN NOM DE FICHIER SUR UN SUPPORT DISQUE

FORME GENERALE

```
[n] V=FILE(nom,nd,nv)
```

REGLES

- nom est une variable alphanumérique ou chaine contenant le nom du fichier recherché.
- nd, nv sont les numéros d'unité et de support sur lequel la recherche doit se faire.
- si le résultat de la fonction est égal à 0, le fichier n'existe pas sur le support. S'il est différent de 0 il représente alors l'adresse du fichier sur le disque et indique qu'il est bien présent.

EXEMPLES

```
250 A=FILE("FICH",3)
260 IF A=0:DISP"FICH N'EST PAS EN LIGNE":GOTO 200
```

RECORD#

FONCTION

ECRITURE D'UN PROGRAMME SUR DISQUE

FORME GENERALE

```
[n] RECORD#[(nd[,nv])] nom
```

REGLES

- nd = numéro du drive
  nv = numéro du disque
  nom = nom du fichier (texte ou variable chaine de 6 caractères au plus)

- un contrôle est effectué sur le nom du fichier lorsqu'il est chargé par l'ordre "LOAD#" si ce nom est différent, l'erreur 55 apparait, il suffit d'appuyer à nouveau sur la touche "ENTER" pour forcer l'écriture du programme ou rectifier ce nom erroné.

EXEMPLES

```
RECORD#(3)"PROG"

RECORD#(2,5340)"PAIE"

RECORD#(2)"FACT"
```

LOAD#

FONCTION

LECTURE D'UN PROGRAMME SUR DISQUE

FORME GENERALE

[n] LOAD#[(nd[,nv])] nom

REGLES

. nd = numéro du drive
  nv = numéro du disque
  nom = nom du fichier (texte ou variable chaine de caractères)

. le chargement d'un programme réinitialise la mémoire si le programme n'existe pas l'erreur 58 est signalée

. une mauvaise structure de programme est détectée par l'erreur 35

EXEMPLES

```
LOAD#"PROG"          lecture sur le disque 1 du programme PROG

LOAD#(1,5340)"PAIE"  lecture sur le disque 1 numéro 5340
                     du programme PAIE

LOAD#(2)"FACT"       lecture sur le disque 2 du programme FACT
```

LOAD/RUN

FONCTION

CHARGEMENT ET EXECUTION D'UN PROGRAMME AVEC ELIMINATION DES VARIABLES EXISTANTES

FORME GENERALE

[n] LOAD#(nd,[nv])nom/RUN [n1]

REGLES

. nd = numéro du drive
nv = numéro du disque

nom = nom du fichier (texte ou variable chaine de caractères)

n1 (optionnel) est le numéro de ligne de départ de l'exécution de programme

EXEMPLES

1000 LOAD#"PAIE"/RUN
chargement du programme PAIE sur le disque 1 et exécution

LOAD/CONTINUE

FONCTION

ENCHAINEMENT D'EXECUTION DE PROGRAMME AVEC CONSERVATION DES VARIABLES COMMUNES

FORME GENERALE

[n] LOAD#(nd,[nv],nom/CONTINUE [n1]

REGLES

nv = numéro du disque
nom = nom du fichier (texte ou variable chaine de caractères)

- lors du chargement du nouveau programme la table des variables du programme précédent est gardé en mémoire
- les variables du nouveau programme sont renseignées par les valeurs du programme précédent, les variables non réutilisées sont abandonnées lors d'un nouveau LOAD/CONTINUE.

EXEMPLES

```
1050 LOAD#(3)"PRSUIT"/CONTINUE 50
```

chargement du programme PRSUIT et exécution à la ligne 50 en sachant que les valeurs des variables connues sont conservées.

SAVE#

FONCTION

SAUVEGARDE D'UN PROGRAMME EN SOURCE SUR DISQUE

FORME GENERALE

[n] SAVE#[(nd[,nv])]nom[,(C1,C2)]

REGLES

- nd = numéro du drive
  nv = numéro du disque
  nom = nom du fichier
- un sous ensemble du programme peut être sauvegardé en précisant les numéros de ligne

- associé à l'instruction RECALL# cet ordre permet de construire de nouveaux programmes avec des parties d'autres programmes.

EXEMPLES

```
SAVE#(3)"SOURCE",(10,20)
```
sauvegarde en code symbolique sous le nom SOURCE des lignes 10 à 20 du programme contenu en mémoire.

RECALL#

FONCTION

LECTURE D'UN PROGRAMME SOURCE ECRIT SUR DISQUE
PAR SAVE#

FORME GENERALE

[n] RECALL#[(nd[,nv])nom

REGLES

. nd = numéro de drive
  nv = numéro du disque
  nom = nom du fichier (texte ou variable chaine de
        caractères)

. cette instruction n'initialise pas la mémoire, c'est à dire que lorsqu'un programme est présent l'ordre RECALL# ajoute les nouvelles lignes si les numéros sont différents, ou écrasent celles qui portent les mêmes numéros.

. en cas d'erreur syntaxique, la ligne s'affiche à l'écran avec possibilité de correction au clavier, en enfonçant la touche ENTER (CR) une fois la ligne corrigée, le processus reprend.

EXEMPLES

RECALL#(3)"PROG"
lecture du programme PROG sur le lecteur disque 3

CREAT#

FONCTION

CREATION D'UN FICHIER DE DONNEES DE LONGUEUR FIXE ET D'ACCES DIRECT OU SEQUENTIEL INDEXE

FORME GENERALE

```
CREAT#(nd[,nv])nom,(Ea1,Ea2)
```

REGLES

- nd = numéro de drive
  nv = numéro de disque
  nom = nom du fichier (texte ou variable chaine de caractères)

- Ea1 représente le nombre d'enregistrements du fichier de données d'un type accès direct

- Ea2 représente la longueur de chaque enregistrement de ce fichier

- Ea1 et Ea2 ne peuvent pas dépasser la valeur 65535

EXEMPLES

```
CREAT#"DATA",(100,5)
CREAT#(1,100)"FICH",(500,10)
1030 CREAT#(PE,MOIS)A$,(K,L)
```

RECORD DATA#

FONCTION

ECRITURE SUR DISQUE DE DONNEES

FORME GENERALE

RECORD DATA#[(nd],nv])]nom,(E)liste

REGLES

. nd = numéro de drive
  nv = numéro de disque
  nom = fichier de données accédé

. E = numéro de l'enregistrement
  Liste = variables contenant les informations à stocker sur disque

. le système fait une relecture pour contrôle de l'enregistrement (erreur 88)

. la place disponible restant en mémoire (donnée par la fontion MEM) doit être au moins égale à deux fois la taille de l'enregistrement.

EXEMPLES

```
Ø100 RECORD DATA#(3)"DATA"(I)A$,N
```
écriture sur le drive 3 de l'enregistrement I du fichier DATA du contenu des variables A$ et N

LOAD DATA#

FONCTION

LECTURE DE DONNEES SUR DISQUES

FORME GENERALE

[n] LOAD DATA#[(nd[,nv])] nom,(C)liste

REGLES

- nd = numéro de drive
  nv = numéro de disque
  nom = nom du fichier (texte de 6 caractères au plus ou variable chaine de caractères)
  C = numéro de l'enregistrement accédé
  liste= liste de variables alphanumériques ou numériques dans lesquelles seront rangées les valeurs lues sur disque de même type que celle donnée dans l'instruction RECORD DATA#

- le contrôle de type de variables (alphanumériques ou numériques) peut être utilisé en positionnant le flag 9 à 1. Dans ce cas l'erreur 35 signale la non validité des données.

- la place disponible en mémoire (donnée par l'instruction MEM) doit au moins être égale à la taille de l'enregistrement

EXEMPLES

```
100 LOAD DATA# (3)"DATA",(I)A$,N
```

lecture sur le drive 3 de l'enregistrement I du fichier DATA et affectation des variables A$ et N.

CHG#

FONCTION

MODIFICATION D'UN NOM DE FICHIER

FORME GENERALE

[n] CHG#[(nd[,nv])] nom1/nom2

REGLES

- nd = nom du drive
  nv = nom du disque
  nom = nom du fichier (texte ou variable chaine de caractères)

- cette instruction peut être utile lors des créations de clés de protection d'un fichier ou pour l'attribution d'une nouvelle clé.

EXEMPLES

```
CHG# (3) "FICH"/"DATA"
CHG# (7) "FICH"/"FICH : CLE"          affectation de la clé :
                                      CLE
CHG# (8) "FICH : CLE"/"FICH : NOM"    modification de la clé
```

KILL#

FONCTION

DESTRUCTION DU NOM D'UN FICHIER AU CATALOGUE D'UN DISQUE

FORME GENERALE

KILL#[(nd[,nv])] nom

REGLES

- nd = numéro du drive
  nv = numéro du disque
  nom = nom du fichier (texte ou variable chaine de caractères)
- La place sur disque qu'occupait le fichier n'est pas récupérée. Ceci sera fait lorsqu'un COMPACT# aura été effectué sur le disque.

EXEMPLES

```
KILL#"DATA"
KILL#[2]"FICH1"          destruction des fichiers DATA,
                         FICH1 et FICH

KILL#[1,532]"FICH"
```

RECRT#

FONCTION

RESTITUTION D'UN FICHIER DETRUIT PAR KILL#

FORME GENERALE

[n] RECRT#[(nd[,nv])]"nom"

REGLES

. nd = numéro du drive
  nv = numéro du disque
  nom = nom du fichier (texte ou variable chaine de caractères)

. le nom du fichier doit être le même que sur l'ordre KILL# et il ne doit pas exister d'autre fichier du même nom.

. cet ordre peut être utile afin d'effectuer des COMPACT# partiels sans pour autant perdre les fichiers du disque

. si plusieurs fichiers portant le même nom ont été détruits c'est le plus ancien qui sera recréé

EXEMPLES

RECRT#(4)"PROG"
regénération au catalogue du fichier PROG

COPY#

FONCTION

RECOPIE DE FICHIER DE DONNEES OU PROGRAMME

FORME GENERALE

[n] COPY#[(nd1[,nv1])]nom1/[(nd2[,nv2])]nom2

REGLES

- nd = nom du drive
  nv = nom du disque
  nom = nom du fichier (texte ou variable chaine de caractères)

- la réservation de place sur le disque pour le fichier nom2 doit être supérieure ou égale à celle du fichier nom1

EXEMPLES

```
COPY#"DATA"/"FICH"

COPY#[2]"DATA"/[1]"FICH"

COPY#[2,100]"DATA1"/[1,20]"FICH2"
```

COMPACT#

FONCTION

REORGANISATION DE DISQUE PAR RECOPIE SUR UN AUTRE SUPPORT

FORME GENERALE

[n] COMPACT#(nd1[,nv1])/(nd2[,nv2])

REGLES

- nd = numéro du drive
  nv = numéro du disque
- cet ordre entraine la copie fichier par fichier d'un disque sur un autre
- les fichiers protégés par une clé ne sont pas recopiés
- la place des fichiers détruits par KILL# est récupérée
- si d'autres fichiers existent sur le disque récepteur les nouveaux fichiers sont mis à la suite et ceux qui portent le même nom sont écrasés par les nouveaux.
- lors de l'exécution, le nom des programmes du fichier en cours de traitement s'affiche sur l'écran.

EXEMPLES

COMPACT#(7)/(3)

copie du contenu du disque 7 sur le disque 3

03

GESTION ECRAN

LIST D

FONCTION

LISTAGE D'UN PROGRAMME SUR L'ECRAN

FORME GENERALE

LIST D [n]

REGLES

. si n est absent, le programme est visualisé à partir du plus petit numéro de ligne, sinon n donne le début.

. trois touches du clavier sont associées à cette commande

M	montée du programme par demi-écran
D	descente du programme par demi-écran
RET	retour à une autre commande

EXEMPLES

```
LIST D

LIST D 1000
```

LINE (2)

FONCTION

DEFINITION DU NOMBRE DE COLONNES AFFICHABLES SUR L'ECRAN DE VISUALISATION

FORME GENERALE

[n] LINE (2) Ea

REGLES

. Cette intruction n'est utilisable que sur les systemes équipés de consoles de visualistion et non de téléviseurs.

. Les deux valeurs possibles de l'expression sont 80 ou 128.

EXEMPLES

LINE(2) 80	Visualisation sur 80 caractères
LINE(2) 128	Visualisation sur 128 caractères

TAB D

FONCTION

TABULATION DE L'ECRAN

FORME GENERALE

[n] TAB D [Ea1[,Ea2]]

REGLES

La première expression indique la ligne, la deuxième la colonne de l'écran où l'on va placer le curseur.
Par défaut, ligne et colonne sont prises à 1.

EXEMPLES

```
0010 FOR I=1 TO 16
0020 FOR J=1 TO 16          affichage de l'astérisque
0030 TAB D I,J              "*" sur la ligne I colonne J
0040 DISP "*"
0050 NEXT J
0060 NEXT I
0070 GOTO 0010
0080 END
```

```
0005 TAB D
0010 FOR I=1 TO 16            position de départ en haut
0020 FOR J=1 TO 16            et à gauche de l'écran
0030 TAB D I,J
0040 DISP "*"
0050 NEXT J
0060 NEXT I
0070 GOTO 0010
0080 END
```

CLEAR D

FONCTION

EFFACEMENT PARTIEL OU TOTAL DE L'ECRAN

FORME GENERALE

[n] CLEAR D [Ea1[,Ea2[,Ea3[,Ea4]]]]

REGLES

Les expressions successives indiquent respectivement les positions ligne-colonne de début et ligne-colonne de fin entre lesquelles aura lieu l'effacement.
Par défaut, les expressions sont prises respectivement à 1, 1, 16, 32 ou 24, 80 selon le nombre de lignes, colonnes de l'écran.

EXEMPLES

```
0005 TAB D
0010 FOR I=1 TO 16
0020 FOR J=1 TO 16
0030 TAB D I,J
0040 DISP "*"
0045 CLEAR D I,J,I,J    effacement de l'astérisque affiché
0050 NEXT J
0060 NEXT I
0070 GOTO 0010
0080 END
```

```
0005 CLEAR D             effacement total de l'écran
0010 FOR I=1 TO 16
0020 FOR J=1 TO 16
0030 TAB DI,J
0040 DISP "*"
0045 CLEAR DI,J,I,J
0050 NEXT J
0060 NEXT I
0070 GOTO 0010
0080 END
```

TOP D

FONCTION

GEL DE LA PARTIE ECRAN AU DESSUS DE Ea

FORME GENERALE

[n] TOP D Ea

REGLES

Ea représente le numéro de ligne à partir de laquelle se fera la remontée de l'écran de visualisation

EXEMPLES

```
0010 FOR I=1 TO 16
0020 TOP D I
0030 FOR J=1 TO 16
0040 DISP TAB J;"*"          blocage sur la ligne numéro I
0050 NEXT J                  de l'écran
0060 NEXT I
0070 FOR I=1 TO 16
0080 TOP D I
0090 FOR J=16 TO 1 STEP -1
0100 DISP TAB J;"*"
0110 NEXT J
0120 NEXT I
0130 GOTO 0010
0140 END
```

LOW D

FONCTION

GEL DE LA PARTIE ECRAN INFERIEURE DE Ea

FORME GENERALE

[n] LOW D Ea

REGLES

. Ea représente la valeur de la ligne à partir de laquelle l'écran sera bloqué

EXEMPLES

0020 LOW D 15

le mode rouleau s'arrête sur la quatorzième ligne de l'écran, de la quinzième au bas de l'écran les informations resteront figées.

04

GESTION CLAVIER

KEY

FONCTION

AFFECTATION A UNE TOUCHE PROGRAMMABLE
D'UNE CHAINE DE CARACTERES

FORME GENERALE

[n] KEY V1$,V2$

REGLES

- les caractères touches de fonctions sont (A,B,C,D,E,F) définis sur le pavé numérique du clavier, sur le système multiposte avec le shift enfoncé ou non, 12 fonctions possibles, et en frappe directe sur 6 fonctions.

- le code "0D" (CR) peut être mis à la fin de la chaine de caractères ce qui déclenchera l'exécution de la fonction

- Le texte défini pour la touche ne peut excéder plus de 63 caractères.

- les doubles guillemets permettent de rentrer ce caractère dans une chaine

EXEMPLES

```
10 KEY "B","NUMERO"
en appuyant sur la fonction B : NUMERO sera généré
 5 DIM A$ (30)
10 A$ ="LOAD# (3)""PROG""/RUN"
12 A$ (21) = HEX "0D"
15 KEY"b",A$
shift B lance le chargement et l'exécution du programme
PROG
```

05

GESTION IMPRIMANTE

PAGE

FONCTION

DEFINIT LE NOMBRE DE LIGNES PAR PAGE POUR L'IMPRIMANTE

FORME GENERALE

[n] PAGE (np) Ea

REGLES

- l'expression Ea représente le nombre de lignes que forme une page sur l'imprimante (utilisé pour les sauts de page (voir WRITE/FORMAT))
- le nombre de lignes est limité à 256
- la valeur implicite est de 66 lignes
- np indique l'imprimante concernée (4 implicite)

EXEMPLES

```
0010  PAGE 30
définition d'une page de 30 lignes
```

LINE

FONCTION

DEFINIT LE NOMBRE DE COLONNES PAR LIGNE SUR L'IMPRIMANTE

FORME GENERALE

[nJ LINE (np) Ea

REGLES

- l'expression Ea représente le nombre maximum de colonnes désiré par ligne
- le nombre de colonnes est limité à 192 sur une imprimante
- np représente le numéro de périphérique sur lequel cette définition doit etre faite (np=4 implicite). Dans le cas d'une console de visualisation (np=2), Ea aura les valeurs 80 ou 128

EXEMPLES

```
0010 INPUT I
0015 LINE I
0020 PRINT "12345678901234567890123456789O"
0025 GOTO 0010
0030 END

1234567890
1234567890                    I=10
1234567890

12345678901234567890
1234567890                              I=20
```

FREELINE

FONCTION

NOMBRE DE LIGNES RESTANT DISPONIBLES

FORME GENERALE

[n] V=FREELINE

REGLES

- Cette fonction nous donne en résultat le nombre de lignes restant à imprimer depuis la position courante jusqu'en bas de la page
- Cette instruction n'est pas implémentée pour les cartes à gestion autonome

EXEMPLES

```
0100 IF FREELINE < 10 THEN 0020
```

Si le nombre de lignes restant sur la page est inférieur à dix (définition par rapport à l'instruction PAGE) il y aura branchement à la ligne 20.

PRINT ON/OFF

FONCTION

IMPRESSION DE TOUT CE QUI EST FRAPPE SUR LE CLAVIER
ET DE TOUT CE QUI EST VISUALISE SUR L'ECRAN

FORME GENERALE

[n] PRINT (ON ou OFF) [(np)]

REGLES

- PRINT ON permet de se mettre dans ce mode et en outre pour l'exécution d'un programme les instructions DISP seront équivalentes à des instructions PRINT.

- PRINT OFF : arrêt de ce mode

- Bien que cette instruction puisse être insérée dans un programme, elle est très utile en mode calculateur pour la mise au point des programmes.
- np permet de spécifier un autre périphérique que l'imprimante à l'adresse 4

EXEMPLES

```
0010 I=0
0020 I=I+1
0030 DISP I
0040 IF I<5 THEN 0020
0050 END

PRINT ON

RUN
1.00
2.00
3.00
4.00
5.00

PRINT OFF
```

06

AIDE A LA MISE AU POINT

FIND

FONCTION

EDITION SUR LE PERIPHERIQUE DESIGNE DES LIGNES CONTENANT LA CHAINE RECHERCHEE

FORME GENERALE

[n] FIND [(np)] "chaine" [, n1 [,n2]]

REGLES

La chaine peut être une variable numérique ou alphanumérique dans ce cas c'est son contenu qui est recherché.

np, n1, n2 peuvent être des expressions numériques

np désigne le périphérique sur lequel on veut le résultat (implicitement c'est l'écran = 2)

n1 et n2 sont les numéros de lignes représentant les bornes de la recherche

EXEMPLES

```
FIND "WRITE"
```
Recherche et visualisation des lignes ou l'instruction WRITE apparait

```
FIND (4) "A(I)"
```
Impression des lignes ou la variable A(I) est utilisée

CHG

FONCTION

CHANGEMENT D'UN NOM DE VARIABLE, INSTRUCTION OU CHAINE DE CARACTERES

FORME GENERALE

[n] CHG "chaine1"/"chaine2" [,n1[,n2]]

REGLES

. Les numéros de ligne du programme ne peuvent être modifiés

. La ligne où figure l'instruction CHG n'est pas modifiée

EXEMPLES

CHG "A$"/"PL$"
Remplacement dans tout le programme de A$ par PL$

CHG "DISP"/"PRINT",100,200
Changement des ordres DISP par des ordres PRINT des lignes 100 à 200 du programme

RENUM

FONCTION

RENUMEROTATION D'UN PROGRAMME

FORME GENERALE

RENUM n1,n2[,pas]

REGLES

. n1 numéro de début de programme à renuméroter

. n2 nouveau numéro de ligne du début de programme

. pas implicitement pris égal à 10

EXEMPLES

RENUM 10,1000,5

Renumérotation en mémoire de la ligne 10 à la fin du programme, les nouveaux numéros de lignes débuteront à 1000 et de 5 en 5.

REORGAN

FONCTION

REORGANISATION DE LA TABLE DES VARIABLES EN FONCTION DU PROGRAMME EXISTANT EN MEMOIRE

FORME GENERALE

REORGAN

REGLES

- la destruction d'une ou plusieurs lignes par l'ordre DEL ne modifiant pas l'état des variables, les lignes du programme sont analysées pour recréer la table des variables.

- si la place mémoire est insuffisante pour exécuter cet ordre procèder avec les ordres SAVE# et RECALL#

TRACE

FONCTION

IMPRESSION DES LIGNES PROGRAMME DANS L'ORDRE D'EXECUTION

FORME GENERALE

[n] TRACE (ON ou OFF) [(np)]

REGLES

- à la rencontre de l'instruction TRACE ON, les lignes exécutées par le programme sont visualisées ou imprimées.
- TRACE OFF permet d'arrêter ce mode
- ces intructions peuvent être utilisées aussi bien en mode calculateur qu'en mode programme
- np permet de modifier le périphérique de sortie
- L'instruction STEP ON peut être associée

EXEMPLES

```
0010 I=0
0020 I=I+1
0030 IF I<4 THEN 0020
0040 END
RUN
TRACE ON
RUN
0010 I=0
0020 I=I+1
0030 IF I<4 THEN 0020
0020 I=I+1
0030 IF I<4 THEN 0020
0020 I=I+1
0030 IF I<4 THEN 0020
0020 I=I+1
0030 IF I<4 THEN 0020
0040 END
```

STEP

FONCTION

EXECUTION D'UN PROGRAMME PAS A PAS

FORME GENERALE

[n] STEP ON
[n] STEP OFF

REGLES

. l'instruction STEP ON provoque un fonctionnement en pas à pas. Pour chaque instruction, son numéro est visualisé sur l'écran et elle n'est exécutée que si l'on appuie sur la touche RET.

. lorsque le numéro de ligne apparait, il est possible de frapper toute instruction BASIC qui sera immédiatement exécutée sans perturber la suite du déroulement du programme (par exemple, visualisation de l'état des variables, en ne frappant que leur nom séparé par , ou ; suivant le format désiré (voir DISP)

. STEP OFF provoque l'arrêt de ce mode et l'exécution normale du programme se poursuit.

EXEMPLES

```
100 STEP ON
              partie du programme à tester
    STEP OFF
```

SEQUENCE

FONCTION

NUMEROTATION AUTOMATIQUE D'UN PROGRAMME

FORME GENERALE

SEQUENCE [n[,pas]]

REGLES

- n désigne le numéro affecté à la première instruction. S'il n'existe pas, il est pris égal à 10.
- pas est pris implicitement égal à 10.
- lors d'une erreur syntaxique, le séquencement automatique n'est pas perdu.
- l'arrêt de ce mode s'effectue soit par l'instruction END soit par le retour à la ligne après avoir effacé le numéro affiché.

EXEMPLES

```
SEQUENCE
0010 INPUT A,B
0020 DISP A,B
0030 END

SEQUENCE 100
0100 I=0
0110 I=I+1
0120 DISP TAB I;I
0130 IF I=10 THEN 110
0140 END

SEQ' 1000,5
1000 DIM A$(10)                    forme abrégée de séquence
1005 INPUT A$
1010 INPUT I
```

LIST P

FONCTION

LISTAGE D'UN PROGRAMME

FORME GENERALE

[n] LIST P [(np)][n1[,n2]]

REGLES

- n1 et n2 représentent les numéros de début et de fin du programme à lister.
  Par défaut, ce sont le plus petit et le plus grand numéro qui sont pris.
- Le flag 6 permet l'édition ventilée du programme
- Le listage peut être interrompu par "BREAK"
- np indique le numéro de périphérique de sortie (par défaut np = 4 : IMPRIMANTE)

EXEMPLES

```
LIST P

LIST P (7) 100,150 sur imprimante adresse 7

10 LIST P (2) sur ecran
```

LIST V

FONCTION

LISTE DES VARIABLES ET FONCTIONS UTILISEES DANS UN PROGRAMME

FORME GENERALE

[n] LIST V [(np)]

REGLES

- l'adresse mémoire exprimée en hexadécimal, le type de la variable et son nom sont donnés sur une même ligne
- types possibles :
  Fn : fonction
  V$: variable alphanumérique
  Vn : variable numérique
  T$: tableau alphanumérique
  Tn : tableau numérique
- cette instruction peut être interrompue par "BREAK"
- np indique le numéro de périphérique de sortie (par défaut np=2 : ECRAN)

EXEMPLES

```
LIST V

LIST V (4) sur imprimante
```

07

TRAITEMENT VARIABLES/PROGRAMMES

PACK

FONCTION

REDUCTION DE VARIABLES NUMERIQUES DANS UNE CHAINE DE CARACTERES

FORME GENERALE

[n] PACK([+]N1[.N2]) V$ FROM liste

REGLES

- si l'encombrement d'un numérique dans V$ est égal à : (N1 + N2)/2 + 1/2 si nombre signé
- chaque octet reçoit deux chiffres, aucune place n'est perdue même si le nombre réduit n'occupe pas un nombre entier d'octets
- si le format ne comporte pas de signe, le nombre conserve sa valeur absolue.

EXEMPLES

```
0010 DIM A$[10]
0020 INPUT A,B
0030 PACK [2.2]A$ FROM A,B
0040 REM
0050 REM
0060 REM
0070 UNPACK[2.2]A$ TO I,J
0080 UNPACK[2.2]A$[1;2] TO X
0090 UNPACK[2.2]A$[3;4] TO Y
0100 DISP I;J;X;Y
0110 END

RUN

1,-3
1.00 3.00 1.00 3.00

0010 DIM A$[20],N[10],M[10]
0020 FOR I=1 TO 10
0030 N(I)=1*-1
```

```
0040 NEXT I
0050 PACK [+2]A$ FROM N[]
0060 REM
0070 REM
0080 UNPACK[+2]A$ TO M[]
0090 WRITE[2.0100]M[]
0100 FORMAT[10N3]
0110 END
RUN
-1 -2 -3 -4 -5 -6 -7 -8 -9 -10
```

UNPACK

FONCTION

TRANSFORMATION DE VARIABLES NUMERIQUES GROUPEES PAR L'INSTRUCTION PACK

FORME GENERALE

[n] UNPACK ([+]N1[.N2])V$ TO liste

REGLES

. Le même format doit apparaitre dans les deux instructions PACK et UNPACK

EXEMPLES

```
0010 DIM A$[10]
0020 INPUT A,B
0030 PACK [2.2]A$ FROM A,B
0040 REM
0050 REM
0060 REM
0070 UNPACK[2.2]A$ TO I,J
0080 UNPACK[2.2]A$[1;2] TO X
0090 UNPACK[2.2]A$[3;4] TO Y
0100 DISP I;J;X;Y
0110 END
RUN
1,-3
1.00 3.00 1.00 3.00
0010 DIM A$[20],N[10],M[10]
0020 FOR I=1 TO 10
0030 N(I)=1*-1
0040 NEXT I
0050 PACK [+2]A$ FROM N[]
0060 REM
0070 REM
0080 UNPACK[+2]A$ TO M[]
0090 WRITE[2.0100]M[]
0100 FORMAT[10N3]
0110 END
RUN
-1 -2 -3 -4 -5 -6 -7 -8 -9 -10
```

PROG

FONCTION

TRANSFORMATION D'UNE CHAINE DE CARACTERES EN LIGNE DE PROGRAMME

FORME GENERALE

[n] PROG V$

REGLES

- la variable chaine doit contenir une ligne BASIC toute erreur syntaxique est détectée.
- lors de l'exécution de cette instruction la ligne est insérée dans le programme au numéro de ligne correspondant ou exécutée en mode calculateur si elle ne comporte pas de numéro.

EXEMPLES

```
10 DIM A$(50)
20 A$="0045 X=A+B"
30 A=20=B=30
40 PROG A$
50 END
RUN
50
LIST D
0010 DIM A$(50)
0020 A$="0045 X=A+B"
0030 A=20=B=30
0040 PROG A$
0045 X=A+B
0050 END
```

UNPROG

FONCTION

TRANSFERT D'UNE LIGNE INSTRUCTION BASIC DANS UNE CHAINE DE CARACTERES

FORME GENERALE

[n] UNPROG V$,Ea

REGLES

- la ligne BASIC transférée dans la chaine de caractères lors de l'exécution de l'instruction UNPROG n'est pas détruite du programme.

EXEMPLES

```
10 DIM A$(40)
20 UNPROG A$,0100
30 MODIF A$
40 PROG A$
50 END
100 DISP MEM
RUN
0100 DISP MEM
0045 DISP MEM
26500
LIST D
10 DIM A$(40)
20 UNPROG A$,0100
30 MODIF A$
40 PROG A$
45 DISP MEM
50 END
100 DISP MEM
```

ØS

GESTION DE SOUS-PROGRAMMES

CALL#

FONCTION

APPEL D'UN SOUS-PROGRAMME BASIC SUR DISQUE
OU EXECUTION D'UN SOUS-PROGRAMME RESIDENT

FORME GENERALE

[n] CALL#(nd,[nv])nom,liste

REGLES

. nd = numéro de drive
  nv = numéro de disque
  nom = nom du fichier (texte ou variable chaine de caractères)

. si le sous programme n'est pas résident, il est chargé en mémoire, exécuté puis détruit

. les tableaux de la liste doivent être dimensionnés

EXEMPLES

```
120 CALL#(3)"SP",A,B$,N
```
Appel sur le disque 3 du sous programme SP avec comme arguments les variables A,B$ et N.

```
160 CALL#(4)"SP2"
```
Appel du sous programme SP2 sur le disque 4

SUBROUTINE

FONCTION

DEFINITION D'UN SOUS-PROGRAMME

FORME GENERALE

[n] SUBROUTINE nom,liste

REGLES

. nom = nom du programme (texte ou variable alphanumérique)

. liste = suite de noms de variables séparées par des virgules

. cette instruction doit figurer en tête de chaque programme

. Les noms des variables peuvent être différents dans la liste CALL# et SUBROUTINE mais la structure de la liste (numérique, alphanumérique...) doit être conservée.

EXEMPLES

```
10 SUBROUTINE"SP",I,V$,J
   I,V$ et J seront affectés par les valeurs transmises par
   l'ordre CALL#
10 SUBROUTINE"SP2"
```

EXTERN

FONCTION

MISE EN COMMUN DE VARIABLES ENTRE DEUX OU PLUSIEURS PROGRAMMES APPELES PAR CALL#

FORME GENERALE

[n] EXTERN liste

REGLES

. Liste = suite de noms de variables séparées par des virgules.

. cette instruction ne figure que dans les sous programmes

. les tableaux désignés dans la liste doivent avoir été dimensionnés dans le programme appelant.
Les variables numériques (autres que tableaux) doivent avoir été initialisées dans le programme principal.

EXEMPLES

```
20 EXTERN A$,TT(),B
          A$,TT() et B sont définis dans le programme
          appelant
```

RETURN END

FONCTION

FIN DE L'EXECUTION D'UN SOUS-PROGRAMME ET RETOUR AU PROGRAMME APPELANT

FORME GENERALE

[n] RETURN END

REGLES

cette instruction doit figurer dans tous les programmes appelés par les instructions CALL#

EXEMPLES

```
0010 DIM A$[10]
0020 INPUT A$
0030 INPUT I,J
0040 CALL#"SP",I,J
0050 GOTO 0020
0060 END
```

```
0010 SUBROUTINE"SP",A,I
0020 EXTERN A$
0030 DISP A;I;A$
0040 RETURN END
0050 END
```

INCLUDE

FONCTION

CHARGEMENT DE SOUS-PROGRAMMES RESIDENTS

FORME GENERALE

[n] INCLUDE (nd1,nv1)nom1,(nd2,nv2)nom2,...

REGLES

. nd = numéro de drive
  nv = numéro de disque
  nom = nom du fichier (texte ou variable chaine de caractères)

. cette instruction provoque le chargement des sous programmes désignés

. l'appel à ce sous programme sera le même que sur disque pour l'instruction CALL#

EXEMPLES

```
100 INCLUDE(3)"SP",(4)"SP2"
```

Chargement en mémoire derrière le programme principal des sous programmes SP et SP2 respectivement sur les disques 3 et 4. Ils seront résidants tant qu'un ordre RELEASE les affectant ne sera pas rencontré.

RELEASE

FONCTION

ELIMINATION DES SOUS-PROGRAMMES CHARGES PAR INCLUDE

FORME GENERALE

[n] RELEASE nom1,nom2,...

REGLES

. nom = nom du fichier (texte ou variable chaine de caractères)

. tout appel à un sous programme n'existant pas sera signalé par une erreur.

EXEMPLES

1050 RELEASE"SP"
suppression du sous programme SP en mémoire

2070 RELEASE"SP2"
suppression du sous programme SP2

09

TRAITEMENT DES FONCTIONS

DEFINE

FONCTION

DEFINITION DE FONCTION UTILISATEUR

FORME GENERALE

```
DEFINE F(V1,V2,...)=E1
```

REGLES

. F représente le nom de la fonction. Ce nom peut être une combinaison de lettres et de chiffres commençant obligatoirement par une lettre.

. V1, V2 sont des variables numériques ou alphanumériques à partir desquelles l'expression E1 sera calculée.

. la fonction est représentée par une seule expression, pour qu'elle soit connue durant la suite du programme il faut que son instruction DEFINE associée soit exécutée.

. une fonction ainsi définie peut être utilisée dans toute expression du programme . Lors de l'appel de la fonction la liste des arguments doit comporter le même nombre de variables que celle des paramètres donnés dans la commande DEFINE.

EXEMPLES

```
0010 REM ARRONDI D'UNE VALEUR A 2 DECIMALES
0020 DEFINE ARR(X)=(INT(X*100+0,5))/100
0030 INPUT MONT
0040 MONT=ARR(MONT)
0050 DISP MONT
```

READ

FONCTION

AFFECTATION AU VARIABLES DESIGNEES DES VALEURS CONTENUES DERRIERE L'ORDRE 'DATA'

FORME GENERALE

[n] READ V1 [,V2 ...]

REGLES

.V1,V2 sont des variables numeriques ou alphanumeriques

.lors de la première rencontre de la commande "READ" ce sont les premières valeurs données par la commande "DATA" qui seront affectées aux variables désignées. A la deuxième rencontre de l'ordre "READ" ce sont les suivantes qui seront affectées et ainsi de suite.

.les valeurs "DATA" sont séparées entre elles par le séparateur en cours dans le programme, c'est-à-dire données par la commande "SEPAR". Ce séparateur est implicitement la virgule.

.Dés qu'une liste de valeurs contenues dans un ordre DATA est épuisée,le systeme poursuit sa lecture sur l'ordre DATA suivant dans la liste des instructions du programme.

.Dés que toutes les listes DATA sont épuisées une erreur 61 est générée.

EXEMPLES

```
0010 DIM A[10]
0020 FOR I=1 TO 10
0030 READ A[I]
0040 NEXT I
0050 DATA 1,2,3,4,5,6,7,8,9,10
0060 END
```

Après l'exécution de ce programme nous aurons :

A(1) = 1

A(2) = 2

DATA

FONCTION

VALEURS AFFECTEES LORS D'UN READ

FORME GENERALE

[n] DATA Vn,V$,....

REGLES

. Il peut exister dans un programme plusieurs ordres "DATA". L'exploration des listes par la commande "READ" commence toujours par le premier de ces ordres dans la liste des instructions du programme (sauf cas particulier cf RESTORE).

. Lorsqu'une ligne "DATA" A été entièrement lue, le système passe à la suivante et ainsi du suite jusqu'à avoir exploré toutes les listes. Auquel cas une erreur serait détectée.

. Derriere un ordre DATA peuvent figurer des valeurs numériques ou alphanumériques. Il faut bien sur que las variables qui recevront ces valeurs dans l'ordre READ surtout du même type (alphanumérique, numérique).

. une valeur alphanumérique ne se met pas entre guillemets

EXEMPLES

```
0010 DIM A(10)
0020 FOR I=1 TO 10
0030 DATA 1,2,3
0040 READ A(I)
0050 DATA 4,5,6
0060 NEXT I
0070 DATA 7,8,9,10
0080 END

100  READ A$
110  DATA ABCDE,12345
```

RESTORE

FONCTION

RECOMMENCE L'EXPLORATION AU DEBUT DES LISTES DE "DATA"
A PARTIR DE LA LIGNE Ea (SI EXPRIMEE)

FORME GENERALE

[n] RESTORE [Ea]

REGLES

- cette commande permet de recommencer l'exploration au début des listes de "DATA".
- la réinitialisation peut être faite sur une ligne DATA définie
- de la même façon la ligne où doit reprendre l'exploration peut être donnée par une expression ou une variable simple
  RESTORE I + J

EXEMPLES

```
0010 DATA 1,2
0020 DATA 10,11
0030 RESTORE 0020
```

L'exploration se fera à partir de la liste contenue en ligne 20.

RESTORE conditionnel

RESTORE A OF 40, 50, 60	
Si A = 1	la réinitialisation se fait en ligne 40
Si A = 2	la réinitialisation se fait en ligne 50
Si A = 3	la réinitialisation se fait en ligne 60

MIN

FONCTION

PLUS PETITE VALEUR DANS UNE LISTE DE VARIABLES

FORME GENERALE

[n] V=MIN(V1[,V2[,V3...]])

REGLES

. V1, V2, V3 sont des variables numériques

. la fonction MIN restitue la plus petite des valeurs de la liste

EXEMPLES

```
0010 INPUT A,B
0020 DISP MIN (A,B)
0030 GOTO 10
```

MAX

FONCTION

RECHERCHE DE LA PLUS GRANDE DES VALEURS PARMI UNE LISTE DE VARIABLES

FORME GENERALE

MAX(V1 [,V2 [,V3 ...]]

REGLES

. la fonction MAX restitue la plus grande des valeurs de la liste

EXEMPLES

```
0010 INPUT A,B
0020 DISP MAX (A,B)
0030 GOTO 10
```

FONCTION

GENERATION D'UN NOMBRE ALEATOIRE

FORME GENERALE

RND expression

REGLES

- la valeur de cette fonction est donnée entre 0 et 1
- expression = valeur de départ de la génération si flag 8 = 0
- le flag 8=1 sur le second passage de la fonction RND
- pour reproduire une séquence il suffit de positionner le flag 8 à zéro

EXEMPLES

```
 5 FOR I=1 TO 5
10 A=RND 3
15 DISP I,A
20 NEXT I
```

10

FONCTION TRI

TRI INDEX

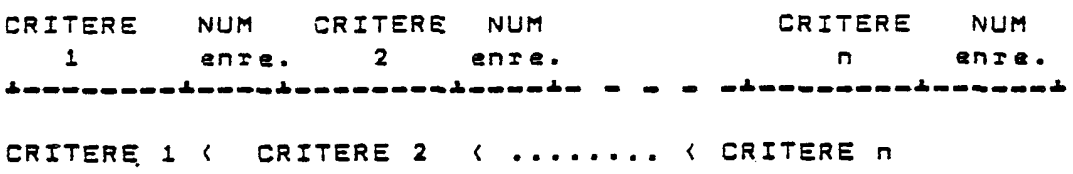
L'utilitaire de tri effectue la construction, à partir d'un fichier de données BASIC, d'un "fichier Index" classé en ordre croissant.

Pour classer un fichier de données, il faut choisir un "critère de classement".

Ex : un fichier client sera classé en ordre croissant sur les numéros de clients.

Le "fichier Index" sera alors une suite ordonnée de ces critères de classement. A chacun de ces critères est associé le numéro de l'enregistrement sur le fichier de données qui contient les renseignements utiles.

SCHEMA DE LA SUITE ORDONNEE



DESCRIPTION D'UN FICHIER INDEX

Ce fichier Index a toutes les apparences d'un "fichier de données BASIC". C'est à dire qu'il est décomposé en enregistrements et qu'il doit être réservé sur le disque par l'ordre "CREAT#".

Les dimensions du fichier index et de ses enregistrements peuvent être quelconques.

L'utilitaire de tri s'adaptera à la configuration rencontrée. Dans chaque enregistrement il mettra un nombre entier de critères, auquel seront associés les numéros d'enregistrements correspondants. Ce nombre étant bien sur supérieur ou égal à 1.

Il faudra réserver un nombre d'enregistrements suffisamment grand pour contenir tous les critères du fichier de données.

Ce fichier Index sera utilisé ensuite par un programme BASIC par l'intermédiaire de l'ordre "LOAD DATA#".

COMPOSITION DES CRITERES DE CLASSEMENT OU CLE

Chaque critère est composé d'une ou plusieurs parties de l'enregistrement du fichier de données correspondant.

Exemple :

Fichier client :
l'enregistrement de ce fichier est composé comme suit :

N client	nom client	adresse rue	départ	ville	comptes deb cred!
5 carac.	20 carac.	20 carac.	2	10	30 carac.

Ce fichier doit être classé en ordre croissant sur le numéro de département et pour chaque département, sur le numéro de client.

Il faut donc indiquer à l'utilitaire de tri que la première partie de la clé se trouve en position 46 sur une longueur 2 et le deuxième critère en position 1 sur une longueur 5.

Une concaténation de ces deux critères est réalisée par le système, pour obtenir un critère général de 7 caractères : département, numéro client.

COMPOSITION DES NUMEROS D'ENREGISTREMENTS OU ADRESSES

Ces numéros sont en binaire sur 2 octets. Une simple expression arithmétique réalise la transformation en décimal.

Exemple :

Un enregistrement du fichier index a été lu dans une chaine de caractères A$, par l'ordre "LOAD DATA#".

On trouve dans cette chaine de caractères de la position 1 à 7 une clé (numéro département et numéro client par exemple précédent).

Dans les positions 8 et 9 on trouve donc le numéro de l'enregistrement correspondant sur le fichier data.

Ce numéro sera relevé comme suit :

N = ASC A$(8) + ASC A$(9)*256

ASC est la fonction du BASIC qui nous donne l'équivalent en décimal d'un octet exprimé en binaire.

La commande :
LOAD DATA# "fich. data", (N) B$
nous donnera dans la chaine de caractères B$ les renseignements correspondant au numéro de client

Si N est nul, la liste des clés est terminée.

DESCRIPTION D'UN ENREGISTREMENT DU FICHIER INDEX

Chaque enregistrement nous l'avons vu contient un nombre entier de clés suivies de leurs adresses.

Mais en plus de cela chaque enregistrement est terminé par un "chainage' en binaire lui aussi sur 2 octets et qui sera donc traité comme les autres variables binaires.

Ce chainage indique le numéro de l'enregistrement du fichier Index sur lequel on peut trouver la suite de la liste ordonnée des clés. S'il est nul alors la liste est terminée.

CALCUL DU NOMBRE DE CLES PAR ENREGISTREMENT INDEX

Soit L = longueur d'un enregistrement index

L aura (L-2) octets pour contenir les clés
(2 octets de chainage)

$$\text{partie entière de } \frac{L-2}{\text{long. d'1 clé+2}} = \text{nombre de clés par enregist.}$$

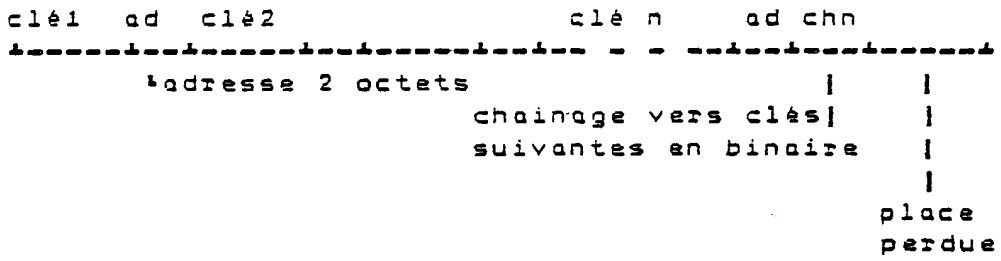
longueur du chainage

(2 sur diviseur = longueur de l'adresse)

Le reste de cette division représente un nombre d'octets perdus qui ne sera pas utilisé par le tri et qui se trouvera tout à la fin de l'enregistrement après le chainage.

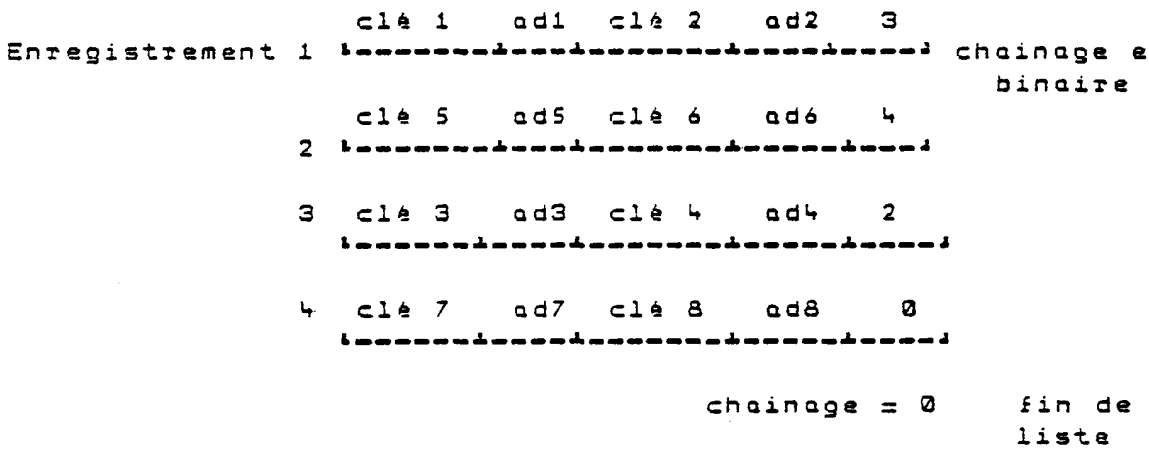
Il faut essayer de bien adapter la longueur des enregistrements à la longueur d'une clé pour que ce nombre soit le plus petit possible, sinon nul.

SCHEMA D'UN ENREGISTREMENT INDEX



SCHEMA DU FICHIER INDEX

Exemple : un enregistrement ne contient que 2 clés



FIN DU FICHIER

L'utilitaire de tri considère implicitement tous les enregistrements du fichier de données. C'est à dire s'arrête après avoir lu tous les enregistrements du fichier de données dont le nombre a été donné par la commande "CREAT#".

Mais on peut demander à l'utilitaire de s'arrêter sur une valeur de clé prédéterminée : on appellera cette valeur "Sentinelle de FIN". Cette valeur sera donnée dans la commande "SORT#" par une fonction spéciale : "END=".

SYNTAXE :

SORT#(np1,nd1)"FICH DONNEE"/(np2,nd2)"FICH INDEX"/P1,L1,P2,
L2,....,Pn,Ln(END = variable ou cste)
nd1 et nd2 des numéros de disques s'il y a lieu

"FICH DONNEE" représente le nom du fichier de données à trier

"FICH INDEX" représente le nom du fichier index résultat.

Notes : ces noms peuvent figurer dans une chaine de caractères, les deux fichiers peuvent figurer sur le même support.

P1,P2..., Pn représentant les positions des différents critères de tri dans l'ordre de leur priorité.

L1,L2,...,Ln sont leurs longueurs respectives

Note : Ces valeurs sont obligatoirement numériques et peuvent être représentées par des expressions arithmétiques.

END = : facultatif :
Cette fonction permet de donner la valeur d'une "Sentinelle de fin de fichier de données".

Si la fonction est absente ou si la sentinelle n'est jamais rencontrée dans le fichier, c'est l'ensemble des enregistrements de données qui sera trié.

Exemple :

Nous désirons construire une table d'index sur un fichier de données "SAI" d'environ 45.000 articles de 82 caractères de long. Le critère de tri se trouvera sur les 6 premiers caractères de chacun des enregistrements.

Le dernier des critères aura la valeur "******" et représentera donc la sentinelle de fin.
Tous les traitements se feront sur une unité de disque lourd, c'est à dire sur le support numéro 3.

1. Réservation du fichier Index :

a) Calcul de la longueur d'un enregistrement Index

Nous désirons par exemple avoir 5 clés par enregistrement du fichier Index.
Une clé occupe 6 octets pour sa valeur propre + 2 octets pour le numéro d'enregistrement correspondant sur le fichier de données "SAI"

Soit 8 octets par clé
5 clés par enregistrement 40 octets auxquels sont ajoutés 2 octets de chainage des enregistrements index.

Pour pouvoir contenir 5 clés de 6 caractères chacune un enregistrement devra avoir une longueur de 42 caractères.

b) Calcul du nombre d'enregistrements

Du fichier de données de 45.000 articles, vont résulter 45.000 valeurs de clés qui devront être contenues par le fichier INDEX

Soit 5 clés par enregistrement INDEX
il faut réserver $\frac{45.000}{5}$ enregistrements soit 9000

Remarque : Pour offrir à l'utilitaire de tri une zone de manoeuvre suffisamment grande, nous ajouterons à cette taille théorique du fichier index, un nombre d'octets égal à environ 1/3 de la mémoire libre au moment de l'exécution de l'ordre "SORT".
Si par exemple la fonction "MEM" du Basic nous donne 15 000 : nous ajouterons environ 5.000 octets soit 5000/42 = 120 enregistrements Index.
La taille définitive de notre fichier sera donc de 9120 enregistrements de 42 octets de long.

c) Création du fichier

CREAT#(3)"INDEX",(9120,42)

2. Construction du fichier Index

SORT#(3)"SAI"/(3)"INDEX"/1,6,END = "******"

Position et longueur de la clé

3. Utilisation du fichier index

Nous voulons traiter les enregistrements du fichier INDEX dans l'ordre croissant des critères :
Programme de traitement des résultats :

```
     A$ contient un enregistrement du fichier index
0010 DIM A$(42),B$(882)----contient un enre. du
                           fichier "SAI"
0020 P=1------------------ début de la liste des clés dans
                           le fichier index : enregist. 1
0030 LOAD DATA#(3)"INDEX",(P)A$--- lecture d'un enregist.
                                   Index
0040 FOR J=1 TO 39 STEP 8-- boucle sur les 5 clés de
                            l'enregistrement par pas de 8
                            (longueur d'une clé)
0050 N=ASC A$(J+6)+ASC A$(J+7)*256
0060 LOAD DATA#(3)"SAI",(N)B$-- lecture d'un enregistrement
                                du fichier de donnée :
                                N = numéro de l'enregis.
0070 GOSUB 1000
0080 NEXT J
0090 P=ASC A$(41)+ASC A$(42)*256-- P=Valeur de chainage
0100 IF P#0 THEN 0030---- si P=0 : fin du fichier INDEX
0110 STOP
1000 REM TRAITEMENT VARIABLE B$
1010 RETURN
1030 END
```

SORT#

FONCTION

UTILITAIRE DE TRI DISQUE

FORME GENERALE

SORT#(nd1,nv1)"fich donnee"/(nd2,nv2)"fichindex"/ P1,L1,
...,Pn,Ln[,END=C ou Eo]

REGLES

. P1,Pn sont les positions des sous clés dans chaque enregistrement du fichier de donnée
. L1,Ln sont les longueurs de chacune de ces sous clés.

EXEMPLES

voir généralités sur tri disque.

11

FONCTION SEQUENTIEL INDEXE

I. GENERALITES

Un fichier de données représente un ensemble d'informations d'un même type pour tous les éléments ou enregistrements, tous les éléments du fichier clients représentent des informations relatives à des clients. Mais chacun de ces éléments se distingue des autres suivant un critère bien précis, chaque client d'un fichier clients se distingue par son numéro au plan comptable.

C'est suivant ce critère qu'il est possible de reconnaitre dans un fichier un élément précis. Il devient donc un critère d'accès, et chaque élément d'un fichier a sa propre clé d'accès. Dans le fichier clients, le numéro de client représente le critère d'accès et chaque clé est contenue par chacun des enregistrements de ce fichier.

Un même fichier de données peut posséder plusieurs critères d'accès, le nom de client peut aussi être composé de plusieurs valeurs dans chaque enregistrement du fichier de données. Le numéro de client associé à une date de commande peut représenter le critère d'accès à un fichier de commandes clients.

Il est possible que certains critères génèrent des clés homonymmes, c'est à dire que plusieurs clés de différents éléments du fichier peuvent avoir la même valeur et chacune d'elles réfère bien sûr des enregistrements distincts du fichier de données.
D'après l'exemple précédent, un même client peut passer plus d'une commande dans une même journée.

II. ACCES SEQUENTIEL INDEXE

Il est important de pouvoir accéder à un enregistrement d'un fichier connaissant la valeur de sa clé d'accès correspondante. Une des solutions est de parcourir séquentiellement tout le fichier, élément par élément, jusqu'à trouver celui recherché. Cette méthode risque d'être très coûteuse en temps. Une autre méthode plus performante est de regrouper dans une table (table d'index ou fichier index) l'ensemble des clés d'un même fichier, où elles seront classées suivant l'ordre désiré. A chacune de ces clés sera associé le numéro de l'enregistrement (index) du fichier de données où se trouve le reste des informations. Des méthodes de recherche dans cette table permettent de retrouver très rapidement un élément du fichier de données bien défini. Le système séquentiel indexé se charge de gérer la complexité d'une telle table d'index.

Il est possible dans le système séquentiel indexé d'effectuer d'une part des traitements séquentiels de fichiers, et d'autre part des accès sélectifs aux enregistrements en fonction de leur valeur de clé.

Dans le cas d'accès sélectif, le système doit connaître la valeur du critère à rechercher et à traiter. Cette valeur de critère sera donnée par une expression numérique ou alphanumérique (qui sera désignée ici par "CLE"). La longueur du résultat de cette expression pourra être inférieure à la longueur normale du critère. La recherche se fera sur la longueur donnée sans tenir compte des caractères complémentaires. C'est donc le premier critère respectant la condition qui sera accédé.

Une table d'index est construite suivant le nom de client dans le fichier clients. Il y a en tête de ce fichier les noms :

ALBERT
ANATOLE
ANTOINE
ARISTIDE

La recherche des valeurs de clé suivantes donnera en résultat :

"A" longueur 1	ALBERT
"AU" longueur 2	inexistant
"AN" longueur 2	ANATOLE
"ANT" longueur 2	ANTOINE

III. INSTRUCTIONS DU SEQUENTIEL INDEXE

Elles sont au nombre de neuf, elles permettent toutes les manipulations désirées sur les fichiers :

BUILD	construction d'une table d'index
USE	description du fichier
INDEX READ	lecture sélective
INDEX READ NEXT	lecture séquentielle
INDEX PROCEED	définition des bornes de lecture
INDEX REWRITE	mise à jour d'un enregistrement
INDEX DELETE	destruction d'un enregistrement
INDEX NUMBER	connaissance du numéro d'enre.donné
INDEX REORGAN	réorganisation d'une table d'index

Elément de la table	
Valeur de clé	Index vers fichier de données
01	10
02	05
03	01
04	14
05	09
06	15
07	07
08	02
09	06
10	12
11	08
12	13
13	03
14	04
15	11
Ordre croissant	

FICHIER DE DONNEES

N enregt	Contenu de l'enregistrement				
	Critere = n client	Informations utiles = NOM	ADRESSE	DEBIT	CREDIT
1	03				
2	08				
3	13				
4	14				
5	02				
6	09				
7	07				
8	11				
9	05				
10	01				
11	15				
12	10				
13	12				
14	04				
15	06				
	ordre alphanu.				

BUILD

FONCTION

CONSTRUCTION D'UNE TABLE D'INDEX A PARTIR D'UN OU PLUSIEURS FICHIERS DE DONNEES

FORME GENERALE

[n] BUILD (P,H),(ND1,NV1) "fich. index" OF (ND2,NV2)"fich. donnee1" [,(ND3,NV3)"fich. donnee2" ...] BY P1,L1,S1,T1 ,END=Ea

REGLES

1) INDEX PRIMAIRE, INDEX SECONDAIRE

Nous aborderons avec plus de détails ce point dans les paragraphes relatifs à la commande d'écriture, qui fait l'adjonction d'un enregistrement dans le fichier et réalise une mise à jour de la table d'index. Seule la table d'index désignée comme primaire est mise à jour. Les tables secondaires devront être reconstruites avant chaque utilisation éventuelle.

Si P = 1 le fichier index sera primaire
Si P = 0 le fichier index sera secondaire

Quel sera le fichier primaire ?

Il est préférable que le fichier d'index primaire soit construit suivant le critère le plus utilisé. C'est à dire le critère de classement de plus haut niveau (le numéro de client pour le fichier clients).

Les index secondaires seront construits suivant des critères qui demandent des traitements moins fréquents.

Un index primaire n'est pas affecté définitivement à un fichier de données : si un fichier de données dispose déjà d'un index primaire, ce dernier peut être remplacé par un autre index si au moment de la commande BUILD l'expression P vaut 1. Un fichier de données peut aussi ne pas avoir d'index primaire du tout.

2) HOMONYMES

Nous appelons homonymes une liste de clés ayant la même valeur mais correspondant à des enregistrements différents.

Les homonymes peuvent être interdits dans un fichier référencé par une table d'index (deux clients ne peuvent avoir le même numéro comptable). Dans ce cas l'expression H donnée dans la commande BUILD devra être nulle.

Si H = 1 les homonymes seront autorisés. (si le critère d'accès à un fichier client est le numéro de département du lieu de domicile du client, il est logique que deux clients différents puissent habiter le même département). Nous verrons dans les commandes d'accès aux fichiers comment traiter ces homonymes.

3) LE FICHIER INDEX

Ce fichier est bien sur destiné à contenir la table d'index. Il a une structure similaire aux autres fichiers de données : c'est à dire qu'il est découpé en enregistrements de longueur fixe et que son emplacement doit être réservé préalablement.

Cette réservation peut être faite par l'ordre CREAT, mais il est préférable de laisser le soin au système de créer lui même par l'ordre BUILD le fichier index dont il calculera lui même l'encombrement. Cette création se fera évidemment si le fichier n'existe pas sur l'unité désignée par ND1.

4) CLAUSE "OF"

Le mot "OF" est suivi de la liste des fichiers de données : leur nom et leur place sur les diverses unités. Plusieurs fichiers de données peuvent être indexés par la même table d'index. Les enregistrements de chacun de ces fichiers peuvent être de longueurs différentes et leur nombre quelconque.

Ces fichiers ont donc un critère de classement qui leur est commun, qui occupe la même position dans tous les fichiers et qui est de longueur identique pour tous. Exemple : les fichiers "Stock de produits finis" et "Stock Matière Première" seront référencés par un numéro d'article interne à la société. Certaines opérations peuvent demander à ce que ces deux fichiers soient regroupés en un seul.

Ces fichiers ont chacun leur nom propre et peuvent être répartis sur plusieurs volumes. C'est à dire qu'un fichier peut figurer sous le même nom sur plusieurs unités différentes. Cette liste de fichiers forme un ensemble. Le début du fichier se trouve sur le lecteur donné par ND2 sur le disque numéro NV2 et il se poursuit sur les lecteurs dont les numéros sont donnés.

Des volumes peuvent être ajoutés à un fichier séquentiel indexé sans avoir à reconstruire la table. Le nombre de volumes possible pour un fichier est limité au nombre d'unités en ligne sur une machine.

5) CLAUSE "BY"

Cette clause est destinée à indiquer au système la longueur des clés, leur position dans les fichiers de données et le sens choisi de classement.

Le critère de classement d'un fichier peut être composé de plusieurs parties (ou sous critères) réparties sur chaque enregistrement.
Exemple : Un fichier commande sera classé suivant le numéro de commande et la date.

Chaque partie d'une clé est décrite par un ensemble de quatre expressions numériques P, L, S, T.
P donne la position de l'enregistrement du sous critère

L donne la longueur du sous critère

S désigne le sens de classement choisi pour le présent sous critère

Si S = 0 le classement se fera en ordre croissant
Si S = 1 le classement se fera en ordre décroissant

T désigne le type du sous critère de classement
Deux types sont possibles :

T = 1 type numérique
T = 0 type alphanumérique cadre à gauche

6) MARQUE DE FIN DE FICHIER

La liste des descriptifs de sous clé peut être terminée par la fonction "END=" suivie d'une expression numérique ou alphanumérique.
Si cette fonction est absente, tous les fichiers de données vont être lus et une clé va être construite à partir de chaque enregistrement de chacun d'eux.

Si cette fonction est présente, la valeur de clé égale à l'expression suivant "END=" marquera la fin de chaque fichier de données. Cette clé ne sera pas incluse dans le fichier index parmi les autres clés jusqu'alors rencontrées.

Dans le cas de multivolumes le passage d'un volume a l'autre se fait automatiquement jusqu'à rencontre dans un des volumes de la marque de fin de fichier.

EXEMPLES

```
0100 BUILD(1,1),(3)"INDEX" OF (3)"DATA1" BY 1,5,0,0,15,8
     ,1,0,END="*****"
```

-Construction d'une table d'index (INDEX) depuis un fichier de donnée (DATA1).Cette table sera primaire et les homonymes seront autorisés.
Sur chaque enregistrement du fichier de donnée, les positions 1 à 5 et 15 à 22 sont les critères de tri et ils sont respectivement alphanumériques croissant et numeriques croissant.
Le premier enregistrement du fichier contenant dans les positions 1 à 5 la valeur "*****" representera la fin du fichier.

USE

FONCTION

DECLARATIONS DU FICHIER INDEX ET INITIALISATIONS
DES POINTEURS

FORME GENERALE

```
[n] USE[(nd[,nv])]"fich.index" OF
(nd1[,nv1],nd2,..)]"fich.de donnees"
```

REGLES

Cette fonction doit être exécutée en début de programme pour chaque fichier index utilisé par la suite dans le programme.

La liste des fichiers et des volumes de chaque fichier ainsi cités doit respecter l'ordre dans lequel ils ont été donnés dans la commande BUILD.

PAR LA COMMANDE "USE" on peut ajouter des volumes à un fichier, c'est à dire que la liste des volumes cités pour chaque fichier de données peut être plus longue que celle donnée par BUILD.

Attention : après l'exécution de la commande USE dans un programme, les supports disques ne doivent pas être échangés dans les différentes unités.

Si une table d'index déjà déclarée par USE dans un programme est reconstruite dans ce même programme, la commande USE devra être réexécutée avant toute utilisation du fichier index.

INDEX READ

FONCTION

LECTURE D'UN ENREGISTREMENT SUIVANT UNE VALEUR DE CLE DONNEE

FORME GENERALE

```
[n] INDEX READ[(nd[,nv])]"fichindex",(cle[,rang])
liste [ERR=Ea]
```

REGLES

1) GENERALITES

Le but du système est de pouvoir lire un enregistrement d'un fichier de données connaissant la valeur de son critère de classement.

La commande INDEX READ répond à cette demande. On accède au fichier de données par l'intermédiaire du fichier index en donnant le nom de ce dernier, le numéro du support sur lequel il se trouve (ND) et s'il y a lieu le numéro du disque (NV).

La valeur du critère recherché est donnée par une expression numérique ou alphanumérique (CLE).

La liste des variables mentionnée dans l'ordre INDEX READ est destinée à contenir l'enregistrement du fichier de données correspondant à la valeur de clé trouvée. Cette liste de variables est similaire à une liste de variables utilisée dans l'ordre RECORD DATA OU LOAD DATA.

Si le critère demandé n'existe pas, deux options sont offertes à l'utilisateur.

Si la clause ERR= est mentionnée, le programme va se poursuivre au numéro de ligne spécifié.

Si la clause ERR= est absente, c'est une erreur 69 qui est signalée par le système.

2) ACCES RELATIF A UNE CLE

Si par le biais d'un accès séquentiel pur, qui est détaillé dans les paragraphes suivants, l'utilisateur découvre la position d'une clé par rapport à une autre (clé repère), l'accès peut être fait en indiquant au système cette dernière et la position (RANG).

Exemple : la variable "CLE" représente la valeur de base. Nous voulons accèder à la clé se trouvant immédiatement après "CLE", c'est à dire en position 2 par rapport à celle ci. Nous aurons donc :

```
                RANG = 2
INDEX READ (ND,NV) "FINDA", (CLE,RANG)A$ ERR = 1000
```

Si "RANG" n'est pas donné, il est implicitement pris égal à 1, et dans ce cas c'est le critère dont la valeur est donnée par "CLE" qui sera considéré.

Dans l'exemple cité si le critère donné par "CLE" n'existe pas dans le fichier, ou s'il se trouve en dernière position du fichier, le programme se poursuivra en ligne 1000.

3) HOMONYMES

Après une lecture sélective d'un fichier, la consultation du FLAG 10 indique si la clé lue a un homonyme ou non.

Si FLAG 10 = 1 il y a un homonyme
Si FLAG 10 = 0 il n'y a pas d'homonymes

Ces homonymes peuvent être traités en séquentiel.

EXEMPLES

```
0010 USE(3)"TABLE1" OF (3)"FICH1"
0020 DIM A$(256),CLE$(8)
0030 INPUT"NUMERO ARTICLE A CONSULTER ",CLE$
0040 INDEX READ(3)"TABLE1",(CLE$(1;LEN CLE$))A$ ERR=0080
0050 DISP"NUMERO :";A$(1;8);" / REFERENCE FOURNISSEUR :
";A$(9;11);A$(12;13)
0060 DISP
0070 GOTO 0030
0080 DISP"NUMERO ARTICLE INEXISTANT":GOTO 0030
0090 END
```

INDEX PROCEED

FONCTION

DEFINITION DES BORNES DE LECTURE SEQUENTIELLE

FORME GENERALE

INDEX PROCEED[(nd[,nv])]"fich index" [FROM cle1] [
TO cle2]

REGLES

- L'expression "CLE 1" représente la valeur de critère à laquelle doit commencer une lecture séquentielle (INDEX READ NEXT). Lorsque le critère "CLE2" a été atteint la fin de fichier sera détectée et l'instruction "INDEX READ NEXT" se continuera par la clause ERR= ou se terminera par une sortie en erreur 69 si celle-ci est absente.

- une des clauses FROM ou TO peut être omise si une seule des bornes doit être définie

- une commande INDEX PROCEED peut être annulée par une autre commande INDEX PROCEED comportant les mêmes clauses FROM TO

EXEMPLES

```
0100 INDEX PROCEED(3)"INDEX" FROM"A"
```

```
0110 INDEX READ NEXT(3)"INDEX",A$
```
lecture du premier enregistrement dont la clé est immédiatement supérieure ou égale à "A"

```
0200 INDEX PROCEED(3)"INDEX"TO"B"
```

```
0210 INDEX READ NEXT(3)"INDEX",A$ ERR=500
```
lecture séquentielle des enregistrements, depuis la position courante et jusqu'à la clé immédiatemnt inférieure à "B". Dès qu'une clé immédiatement supérieure ou égale à "B" sera trouvée, le programme continuera en ligne 500.

INDEX READ NEXT

FONCTION

LECTURE SEQUENTIELLE D'UN FICHIER SUIVANT UNE TABLE D'INDEX

FORME GENERALE

```
[n] INDEX READ NEXT[(nd[,nv])]"fichindex",liste
[ERR=Ea]
```

REGLES

Cette commande réalise la lecture de l'enregistrement de données dont le critère se trouve immédiatement après celui accédé auparavant, l'ordre des critères lus suit l'ordre choisi pendant la construction de la table d'index (BUILD).

L'accès précédant cette commande de lecture séquentielle a pu être fait par le même ordre de lecture séquentielle (INDEX READ NEXT) ou par un ordre de lecture sélective (INDEX READ) ou par un des ordres d'écriture.

Si la commande INDEX READ NEXT est le premier accès réalisé sur le fichier index, c'est la première clé de la table d'index qui sera lue et qui représentera la base pour la lecture suivante.

La clause ERR= sera utilisée dans le cas où le système a atteint la fin de fichier.

TRAITEMENT DES HOMONYMES

Comme toutes les commandes d'accès au fichier, une lecture séquentielle met à jour le FLAG 10. Il est donc possible de savoir s'il y a un homonyme après le critère lu.

S'il y a un homonyme, une lecture séquentielle de plus peut y accéder.

En résumé, une lecture sélective (INDEX READ) permet d'accéder au début d'une liste d'homonymes. Une succession de lectures séquentielles (INDEX READ NEXT) permet l'accès à un élément précis de la série d'homonymes.

EXEMPLES

```
0100 REM LISTAGE COMPLET DU FICHIER
0110 USE(3)"INDEX"OF(3)"DONNEE"
0120 INDEX READ NEXT(3)"INDEX",A$ERR=150
0130 PRINT A$
0140 GOTO 120
0150 PRINT "FIN DE FICHIER"

0200 REM LISTAGE D'UNE LISTE D'HOMONYMES
0210 INPUT CLE$
0220 INDEX READ(3)"INDEX",(CLE$)A$ ERR=300
0230 PRINT A$
0240 IF NOT FLAG 10 THEN 320
0250 INDEX READ NEXT(3)"INDEX",A$
0260 GOTO 230
0300 DISP "CLE INEXISTANTE"
0310 GOTO 210
0320 DISP "FIN DE LISTE"
0330 GOTO 210
```

INDEX WRITE

FONCTION

ADJONCTION D'UN ELEMENT A UN FICHIER DE DONNEES
ET MISE A JOUR DE LA TABLE D'INDEX

FORME GENERALE

[n] INDEX WRITE[(nd[,nv])]"fich data",(cle)liste [
ERR=Ea]

REGLES

C'est le fichier de données qui est désigné dans cette commande d'écriture. La table d'index primaire de ce fichier de données doit obligatoirement être en ligne, c'est à dire qu'elle doit figurer dans une des commandes "USE" exécutées en tête de programme.

La clé correspondant à l'enregistrement écrit est insérée dans la table primaire en bonne place pour permettre de la retrouver par la suite.

Si ce fichier de données a des tables secondaires, elles ne sont pas mises à jour (comme dans le cas de "INDEX DELETE"). Si un traitement du fichier de données est nécessaire suivant une de ces tables secondaires, il faudra en refaire la construction par la commande BUILD.

EXEMPLES

```
0010 USE(3)"TABLE1"OF(3)"FICH1"
0020 DIM A$(256)
0030 INPUT"NUMERO",A$(1;8)
0040 INPUT"REFERENCE",A$(9;13)
0050 INDEX WRITE(3)"FICH1",(A$(1;8))A$ ERR=0070
0060 GOTO 30
0070 DISP "REFERENCE DEJA EXISTANTE"
0080 GOTO 30
```

INDEX REWRITE

FONCTION

MISE A JOUR D'UN ENREGISTREMENT

FORME GENERALE

```
[n] INDEX REWRITE[(nd[,nv])]"fich index",
cle[,rang]) [ERR=Ea]
```

REGLES

. Cette commande ne réalise pas l'adjonction de nouvelles clés. La liste de variables est enregistrée à la place des anciennes valeurs contenues par cet enregistrement.

CAS DES HOMONYMES

. Sur une commande INDEX REWRITE ou INDEX DELETE, si la clef concernée a une valeur différente de celle qui a été traitée auparavant, une recherche en table sera effectuée et dans le cas ou elle appartient à une liste d'homonymes c'est la première trouvée qui sera traitée par contre si la clef est égale à l'accès précédent, c'est sur ce même enregistrement que le traitement se fera.
Si dans l'enregistrement de données, positions représentant la clef sont modifiées par la commande INDEX REWRITE, la valeur de la clef d'accès n'est pas modifiée : c'est à dire qu'il faut continuer à accéder à cet enregistrement par son ancienne valeur de clef.
Pour détourner ce cas particulier il faut faire un INDEX DELETE de l'ancienne clef et un INDEX WRITE de la nouvelle à condition bien sur que la table soit primaire.

EXEMPLES

```
100 INDEX READ(3)"INDEX",(CLE$)A$,NOM$
110 MODIF NOM$
120 INDEX REWRITE(3)"INDEX",(CLES$)A$,NOM$
```

```
0100 INPUT CLE$
0110 INDEX READ(3)"INDEX",(CLES$)A$,NOM$
0120 DISP A$ ; NOM$
0130 INPUT "EST CE L'ENREGISTREMENT RECHERCHE", CHOIX$
0140 IF CHOIX$ = "OUI" THEN 200
0150 IF NOT FLAG 10 THEN 250
0160 INDEX READ NEXT(3)"INDEX",A$,NOM$
0170 GOTO 120
0200 MODIF NOM$
0210 INDEX REWRITE(3)"INDEX",(CLE$),A$,NOM$
0220 RETURN
0250 DISP "ENREGISTREMENT INEXISTANT"
0260 GOTO 100
```

INDEX DELETE

FONCTION

ELIMINATION D'UN ENREGISTREMENT DU FICHIER

FORME GENERALE

INDEX DELETE[(nd[,nv])]"fich index",(cle[,rang])

REGLES

L'enregistrement du fichier de données correspondant à la clé demandée est détruit et l'emplacement libéré sera récupérable par un nouvel enregistrement,

Au moment où la commande INDEX DELETE est exécutée, le fichier index désigné doit être un fichier index primaire, et il a dû être réservé au préalable par l'ordre USE, sans quoi le système signale une erreur 67.

- voir cas des homonymes (INDEX REWRITE)

Une tentative de relecture d'une clé éliminée par l'ordre INDEX DELETE détectera bien sûr une absence de celle-ci et provoquera une sortie en erreur

EXEMPLES

```
100 INDEX READ(3)"INDEX",(CLE$)A$,MONT,TVA ERR=1000
110 IF MONT # 0 THEN 0130
120 INDEX DELETE (3),(CLE$)
130 RETURN
```

INDEX NUMBER

FONCTION

CONNAISSANCE DU NO. D'ENREGISTREMENT, DU NUMERO D'UNITE ET DU NOM DU FICHIER PAR LA CLE

FORME GENERALE

```
[n] INDEX NUMBER(nd)"fich index",(clef,rang)A,B,  V
$ [ERR=Ea]
```

REGLES

- la clef spécifiée dans l'instruction est recherchée dans les tables d'index, et ses coordonnées dans le fichier données sont transférées dans les variables A,B,V$.

  A = numéro d'enregistrement dans le fichier donnée
  B = numéro de drive ou se trouve le fichier de données
  V$ = nom du fichier de données

- les variables B et V$ peuvent être omises

- cette instruction peut servir pour des verrouillages d'enregistrements sur les systèmes multipostes.

EXEMPLES

```
0100 INDEX NUMBER(3)"INDEX",(CLE$)A,B,V$
0110 LOCK(B)V$,(A)
```

INDEX REORGAN

FONCTION

REORGANISATION D'UNE TABLE D'INDEX

FORME GENERALE

[n]INDEX REORGAN [(nd)"fich index]

REGLES

- Après de multiples adjonctions dans un fichier en accès séquentiel indexé, la table d'index peut être amenée à se saturer. C'est à dire qu'il est impossible de faire de nouvelles écritures [ERREUR 70 sur INDEX WRITE]. Ce phénomène est tout à fait normal et seul INDEX REORGAN permettra de débloquer la situation.

- Au moment où la commande INDEX REORGAN est exécutée, les commandes USE des tables à réorganiser doivent avoir été exécutées.

- Si aucun nom de fichier index n'est donné dans la commande INDEX REORGAN, toutes les tables saturées seront réorganisées.

EXEMPLES

```
0100 IF ERR (LG,ER) GOTO 1000
0110 INDEX WRITE (3) "DONNEE",(CLE$)A$
0120 RETURN

1000 IF ER#70 THEN 2000
1010 INDEX REORGAN
1020 GOTO LG
```

12

GESTION ASSEMBLEUR

LOAD AS

FONCTION

CHARGEMENT D'UN PROGRAMME ASSEMBLEUR SUR DISQUE

FORME GENERALE

[n] LOAD AS [(nd[,nv])]nom

REGLES

- nd = numéro du drive
  nv = numéro du disque
  nom = nom du fichier (texte ou variable chaine de

- Après chaque ordre LOAD AS le BASIC est réinitialisé. Le programme vient se mettre physiquement derrière le langage BASIC, ces programmes assembleurs doivent être compilés avec une adresse origine correspondant à la fin du langage BASIC (précisée avec chaque version)

- ADRESS () permet de connaitre l'adresse de fin du BASIC et INIMEM peut servir à forcer cette adresse à une valeur connue

EXEMPLES

LOAD AS "PASM" chargement du programme PASM et réinitialisation du BASIC

CALL

FONCTION

APPEL D'UN SOUS-PROGRAMME ASSEMBLEUR

FORME GENERALE

[n] CALL adresse ou (chaine de caracteres)

REGLES

- adresse exprimée en hexadécimal permet de donner le début du sous programme assembleur qui doit se terminer par RET (0C9H)
- ce sous programme peut être soit chargé par LOAD AS soit être contenu dans une chaine de caractères

EXEMPLES

```
0100 CALL 99AC              appel du sous programme à l'adresse
                            99ACH

0200 A$ = HEX"3E2A320011C9"

0210 CALL (A$)                  écriture d'un astérisque à
                                l'adresse 1100 H c'est à dire
                                sur l'écran
```

PEEK

FONCTION

LECTURE D'UN OCTET EN MEMOIRE

FORME GENERALE

[n] PEEK ad,valeur

REGLES

ad = adresse (en décimal) de la lecture
valeur = nombre positif compris entre 0 et 255

EXEMPLES

0300 PEEK 45900,A	lecture dans A du contenu de la mémoire à l'adresse 45900
0510 PEEK ADRESS V$(4),I	lecture du quatrième caractère de V$ dans I

POKE

FONCTION

ECRITURE D'UN OCTET EN MEMOIRE

FORME GENERALE

[n] POKE ad,valeur

REGLES

ad = adresse (en décimal) de l'écriture
valeur = nombre positif compris entre 0 et 255

EXEMPLES

```
0030 POKE 40503,09        écriture de 09 à l'adresse
                          mémoire 40503

0100 POKE ADRESS A$(10),65      écriture dans le dizième
                                caractère de A$ du caractère
                                "A" (A$(10) = "A")
```

ADRESS

FONCTION

ADRESSE MEMOIRE D'UNE VARIABLE

FORME GENERALE

ADRESS V

REGLES

- l'adresse de tout type de variable peut être recherchée même les variables indicées.
- si V est remplacé par () l'adresse de fin du BASIC est obtenue
- l'adresse est exprimée en décimal

EXEMPLES

```
A = ADRESS V$        adresse de V$

DISP ADRESS ()       adresse de fin du BASIC
```

DEFMEM

FONCTION

DEFINITION DE FIN DE MEMOIRE

FORME GENERALE

[n] DEFMEM valeur

REGLES

- la valeur en hexadécimal doit être un nombre entier mutiple de 256 octets
- le BASIC initialisant toute la mémoire, cette instruction permet de diminuer afin de tester des applications devant fonctionner sur des configurations moins étendues en mémoire
- cette instruction réinitialise le BASIC

EXEMPLES

```
DEFMEM   0C000
         Réduction de la mémoire système de 16K sur
         un système de 64K octets
```

INIMEM

FONCTION

DEFINITION DU DEBUT DE MEMOIRE

FORME GENERALE

REGLES

. pour la compatibilité ASSEMBLEUR/BASIC il peut être interessant de forcer l'adresse de début de mémoire afin quelle soit constante quelque soit la version de BASIC utilisée.

. cette instruction réinitialise le BASIC

EXEMPLES

```
INIMEM    0A000
```

La zone de travail du BASIC sera définie entre A000 et la fin de la mémoire du système

DUMP

FONCTION

CONVERSION HEXADECIMAL/ASCII

FORME GENERALE

REGLES

. Cette fonction ne travaille que sur les variables alphanumériques

. Le résultat donne une chaine de caractères de deux fois la taille de la chaine initiale

EXEMPLES

```
100 DIM A$(7),B$(14)
110 A$="ALCYANE"
120 B$=DUMP A$
130 PRINT B$
140 END
RUN
414C4359414E45
```

13

FONCTIONS TRANSCENDANTES

FONCTIONS DE BASE

FONCTION

OPERATEURS DE CALCUL DES FONCTIONS TRANSCENDANTES

FORME GENERALE

```
SIN     = Sinus
COS     = Cosinus
TAN     = Tangente
ASIN    = Arcsinus
ACOS    = Arcosinus
ATAN    = Arctangente
**      = Elévation à une puissance
LN      = Logarithme népérien
LOG     = Logarithme décimal
EXP     = Exponentielle
SQRT    = Racine carrée
RTD     = Conversion radian/degré
DTR     = Conversion degré/radian
FNPI    = Constante de PI
```

REGLES

Argument en degrés implicitement ou aprés exécution de l'instruction DEGRES, sinon en grades ou radians aprés exécution des instructions GRADES ou RADIANS.

EXEMPLES

```
0010 DISP"CALCUL DE DETERMINANTS"
0020 REM EQUATION AX**2+B*X+C
0030 INPUT"A= ",A
0040 INPUT"B= ",B
0050 INPUT"C= ",C
0060 DET=SQRT(B**2-4*A*C)
0070 DISP DET
0080 GOTO 0010
0090 END
```

RADIANS

FONCTION

POSITIONNE LA MACHINE EN MODE RADIANS

FORME GENERALE

[n] RADIANS

REGLES

Repositionne l'indicateur de mode trigonométrique :
Les fonctions trigonométriques directes SIN/COS/TAN ont leur argument en RADIANS.
Les fonctions trigos inverses ASIN/ACOS/ATAN ont leur valeur en RADIANS.

Cet indicateur n'est pas affecté par SCRATCH

EXEMPLES

```
10 FIXED 12
20 RADIANS
30 X=SIN(FNPI/4)
40 DISP X
50 END
RUN
0.707106781186
```

DEGRES

FONCTION

POSITIONNE LA MACHINE EN MODE DEGRES

FORME GENERALE

[n] DEGRES

REGLES

Repositionne l'indicateur de mode trigonomètrique :
Les fonctions trigonomètriques directes SIN/COS/TAN ont leur argument en DEGRES.

Les fonctions trigos inverses ASIN/ACOS/ATAN ont leur valeur en DEGRES.

Cet indicateur n'est pas affecté par SCRATCH.

DEGRES est le mode implicite lors du chargement du BASIC.

EXEMPLES

```
10 FIXED 12
20 DEGRES
30 X = SIN (SIN 45)
40 DISP X
50 END
RUN
0.707106781186
```

GRADES

FONCTION

POSITIONNE LA MACHINE EN MODE GRADES

FORME GENERALE

[n] GRADES

REGLES

. Repositionne l'indicateur de mode trigonométrique : Les fonctions trigonométriques directes SIN/COS/TAN ont leur argument en GRADES.

Les fonctions trigos inverses ASIN/ACOS/ATAN ont leur valeur en GRADES.

Cet indicateur n'est pas affecté par SCRATCH.

EXEMPLES

```
10 FIXED 12
20 GRADES
30 X = SIN (SIN 50)
40 DISP X
50 END
RUN
0.707106781186
```

14

FONCTIONS GRAPHIQUES

PLOT

FONCTION

DEFINITION D'UN VECTEUR GRAPHIQUE

FORME GENERALE

[n] PLOT X,Y,Z,T,B

REGLES

. B donne la brillance 0<=B<=15
les coordonnées du vecteur sont comprises entre 0 et 255
et cas particuliers B>=16

EXEMPLES

```
[10] SET PLOT 0
[20] INPUT"COORDONNEES POINT 1 :",X,Y
[30] INPUT"COORDONNEES POINT 2 :",Z,T
[40] INPUT"BRILLANCE :",B
[50] PLOT X,Y,Z,T,B
[60] END
```

SET PLOT

FONCTION

INITIALISATION DE LA PAGE GRAPHIQUE

FORME GENERALE

[n] SET PLOT B

REGLES

Valeur de B : Brillance
0<=B<=15

EXEMPLES

100 SET PLOT 1
mise de l'écran au niveau de brillance 1

200 SET PLOT 0
effacement total de l'écran

15

ENTREES/SORTIES

INOUT

FONCTION

PARAMETRAGE DES FONCTIONS D'ENTREE/SORTIE INTEGREES AU BASIC

FORME GENERALE

[n] INOUT I,J,K,L,M,N,P

REGLES

I Numéro de périphérique (10 à 14)

J Vitesse

0	= BREAK	
1	= 110	BAUDS
2	= 300	BAUDS
3	= 600	BAUDS
4	= 1200	BAUDS
5	= 2400	BAUDS
6	= 4800	BAUDS
7	= 9600	BAUDS
8	=19200	BAUDS
20	= EXT/1	
21	= EXT/16	
22	= EXT/64	

K Parité

0 = sans parité
1 = sans parité
2 = parité impaire
3 = parité paire

L Caractère de fin de lecture et de fin de buffer d'écriture

M Longueur de fin de lecture

N Nombre de bits

0 = 5 bits
1 = 6 bits
2 = 7 bits
3 = 8 bits

P Protocole

0 = CTS et RTS non gérés
1 = CTS testé (entrée) et RST émis à 1 (sortie)

ALCYANE MODEM 2 = CTS' émis (sortie) et RTS' indifférent (entrée)

Valeurs par défaut : K=0,L=13,M=0,N=3,P=0

. Un ordre INOUT erroné aura pour résultat une erreur 48 ou 62. Dans ce cas les paramètres définis lors du précédent INOUT ne sont pas réinitialisés, les buffers ne sont pas effacés.

  Sinon, tout autre ordre INOUT REINITIALISE les buffers de réception et d'émission.

. Ces valeurs de J supposent le déplacement de 2 cavaliers sur l'interface et ne sont à utiliser que dans le cas de MODEMS fournissant l'horloge afin de travailler en mode pseudo-synchrone. Ce cas est très rare.

. Parité PAIRE : Nombre pair de bits à 1 dans le mot, parité comprise
  EX : CR(7 bits) 000 1101 parité = 1 = 1000 1101

. Le caractère de fin de buffer d'écriture est normalement inséré automatiquement à la suite des caractères générés par l'ordre WRITE. Cette propriété peut être supprimée en spécifiant une valeur négative pour le caractère de fin. Il sera alors pris en compte à la réception mais ne sera plus inséré à la fin du buffer d'écriture.

  Le caractère de fin de lecture doit être compatible avec le format utilisé (nombre de bits).

  Si la longueur de fin est nulle, la réception se terminera lorsque le caractère de fin de lecture sera reçu, auquel cas il sera le dernier caractère recopié dans la chaine de caractères désignée dans l'ordre READ, ou quand un nombre de caractères de l'ordre de 508 aura été reçu.

  Si la longueur de fin est non nulle, la réception s'arrêtera lorsque ce nombre aura été atteint, le caractère de fin étant alors ignoré.

  Le format le plus couramment utilisé pour des échanges série est le format 7 bits, Parité paire.

  Le cable standard MBC est prévu pour la connexion directe sur un MODEM. Le protocole à utiliser dans ce cas est, soit le protocole 0 (cas le plus courant), soit le protocole 1 (ALCYANE TERMINAL).

La connexion d'ALCYANE sur un terminal impose l'emploi d'un cable permutant les signaux TXD & RXD, CTS & RTS, DSR & DTR. Il peut être nécessaire dans certains cas de connecter le signal DP à un signal actif tel que RTS. Le protocole à utiliser est, soit le protocole 0 (cas le plus courant), soit le protocole 2 (ALCYANE MODEM).

En cas d'erreur de réception (erreur de parité ou désynchronisation), la carte continue à accumuler des caractères dans son buffer jusqu'à rencontrer une condition de fin licite. Quand l'ordre READ sera exécuté, le buffer sera recopié dans la chaine de caractères, et le FLAG 4 sera positionné à 1 pour signaler l'erreur.

. En ce qui concerne les ordres généraux du BASIC, il faut noter que le caractère hexadécimal "FF" n'est plus utilisé comme marque de fin de chaine, et peut donc être présent sans problème dans un calcul d'expression alphanumérique.

En ce qui concerne les entrées-sorties, le caractère hexadécimal "0D" n'est plus utilisé comme caractère de fin de réception ou d'émission. Cela signifie qu'il devient possible de recevoir ou d'émettre des chaines contenant toute combinaison de codes, avec la règle suivante :

- ENTREE

```
0010 READ(PER,20)A$
0020 FORMAT()
```

A$ contiendra l'ensemble des caractères reçus (y compris des "0D" intermédiaires) jusqu'au caractère (éventuellement la longueur) spécifié comme marque de fin dans l'INOUT, ce caractère étant le dernier caractère "utile" de la chaine A$ qui est ensuite complétée avec des blancs.

Il faut toutefois noter que si l'on veut visualiser sans plus de précautions A$ sans la tronquer au premier "0D" il est nécessaire d'exécuter la séquence suivante :

```
DIM C$(1),D$(1):C$=HEX"0D":D$=HEX"0A"
SUBST ALL C$ BY D$ IN A$
DISP A$
```

- SORTIE

L'ordre préférentiel est pour la sortie l'ordre WRITE, avec un format complet ou un format libre. La séquence de caractères transmise par la carte V24 se compose de :

- l'ensemble des caractères générés par le FORMAT, "0D" intermédiaires compris, mais sans adjonction de "0D" final implicite,

- en fin de séquence, (éventuellement) le caractère de fin spécifié dans l'ordre INOUT.

Il est toutefois possible d'effectuer un PRINT généralisé sur la carte V24 (par ex PRINT(10)A$,3,C$) en tenant compte du fait que, contrairement au WRITE, le PRINT génère automatiquement un "0D" à la fin de la séquence juste avant le caractère de fin.

Il est de même possible d'exécuter un LIST P généralisé (ou un SAVE) en suivant la même règle que pour le PRINT.

READ

FONCTION

LECTURE DE DONNEES SUR UN PERIPHERIQUE

FORME GENERALE

[n] READ(np,nf[,n1[,n2]])V$

REGLES

.np représente le numéro de périphérique sur lequel doit se faire la lecture.
.nf représente le numéro de ligne du format associé: Ce format est toujours un format libre - FORMAT () -
.n1 est le numéro de ligne auquel doit se brancher le programme si une donnée a été lue sur le périphérique. Si n1 est absent le systeme attend une donnée et ne sort pas de l'ordre READ.
.n2 est le numero de ligne auquel doit se brancher le programme si une erreur de parité a été décelée pendant la lecture. En l'absence de ce paramètre, aucune erreur n'est signalée.
. V$ chaine de recopie du message. La précision de cette variable doit être suffisante pour recevoir le message (erreur 26).

EXEMPLES

```
0100 READ(10,0110,0200)A$
0110 FORMAT()
0120 DISP "ATTENTE DE MESSAGE "
0130 GOTO 0100
0200 PRINT A$(1;LEN A$)
0210 GOTO 0100
```

WRITE

FONCTION

ECRITURE DE DONNEES SUR UN PERIPHERIQUE QUELCONQUE

FORME GENERALE

[n] WRITE(np[,nf[,n1[,n2]]])liste

REGLES

VOIR REGLES ORDRES WRITE (Chap 01-3)

N.B Un format libre peut être utilisé dans le cas des entrées/sorties généralisées.

SAVE

FONCTION

SORTIE D'UN PROGRAMME SUR UN PERIPHERIQUE

FORME GENERALE

[n] SAVE (np)

REGLES

- le FLAG 6 intervient comme pour le LIST P
- cette instruction facilite le transfert de programme entre deux systèmes connectés en liaison série ou parallèle.
- le transfert s'effectue ligne par ligne et peut être interrompu par BREAK.

EXEMPLES

SAVE(10)
: transfert du programme contenu en mémoire sur le périphérique 10 (ligne asynchrone)

SAVE(4)
: listage du programme sur imprimante

RECALL

FONCTION

LECTURE D'UN PROGRAMME SUR UN PERIPHERIQUE

FORME GENERALE

[n] RECALL (np)

REGLES

- cette instruction permet de remplacer le clavier par un périphérique désigné dans cet ordre
- la réception s'effectue ligne à ligne et les instructions sont rangées en mémoire d'une façon identique à une saisie de programme par le clavier.
- en cas d'erreur syntaxique la ligne s'affiche avec possibilité de correction au clavier, en enfonçant la touche ENTER (CR), une fois la ligne corrigée, le processus reprend.

EXEMPLES

```
RECALL(10)
lecture d'un programme sur le périphérique 10 qui est
une ligne Asynchrone (connexion de deux systèmes ALCYANE)
```

16

FICHIER SEQUENTIEL

ASSIGN

FONCTION

AFFECTATION D'UN NUMERO DE PERIPHERIQUE A UN FICHIER DE TYPE SEQUENTIEL EN LONGUEUR VARIABLE

FORME GENERALE

[n] ASSIGN Vn,(nd,[nv])V$

REGLES

Vn est une variable numérique ou une constante désignant le numéro de périphérique qui sera utilisé dans les commandes READ et WRITE.

14(= Vn (= 17

nd est une variable ou constante numérique représentant le numéro d'unité disque sur lequel se trouve le fichier à traiter

nv numéro de disque

V$ variable ou constante alphanumérique, représente le nom de fichier à accès séquentiel. La première instruction WRITE ou READ qui suit la commande ASSIGN va se porter sur le premier élément du fichier et les instructions suivantes vont séquentiellement se référer aux éléments suivants. Arrivée en fin de fichier l'instruction READ génère une erreur 61, ou sort par la clause END de l'ordre READ.

EXEMPLES

```
0020 DIM FICH$(6),LIGNE$(80)
0030 CLEAR D :INPUT"NOM DU FICHIER SEQUENTIEL A CREER:",
     FICH$
0040 INPUT"NUMERO DE L'UNITE DISQUE:",N
0050 CREAT#(N)FICH$
0060 CLEAR D
0070 ASSIGN 14,(N)FICH$
0080 CLEAR D 4,1,4,80:INPUT"ENTER LA LIGNE:",LIGNE$
0090 IF LIGNE$="                " THEN 0130
0100 WRITE(14,0110,0140)LIGNE$
0110 FORMAT()
0120 GOTO 0080
```

```
0130 END
0140 CLEAR D 23,1,23,80:DISP"ERREUR SUR ECRITURE EN
     SEQUENTIEL"
0150 STOP


0010 DIM FIC$(6),LIGN$(8)
0020 CLEAR D :INPUT"NOM DU FICHIER A LIRE EN SEQUENTIEL:",
     FIC$
0030 INPUT"NUMERO DE L'UNITE DISQUE:",N
0040 ASSIGN 14,(N)FIC$
0050 READ(14,0060,0100)LIGN$
0060 FORMAT()
0070 DISP LIGN$
0080 GOTO 0050
0090 END
0100 CLEAR D 23,1,23,80:DISP"FIN DE LECTURE"
0110 END
```

CREAT#

FONCTION

CREATION D'UN FICHIER DE DONNEES DE TYPE SEQUENTIEL ET DE LONGUEUR VARIABLE

FORME GENERALE

CREAT#[(Nd [,nv])]nom

REGLES

.nd = numéro du drive
nv = numéro du disque
nom = nom du fichier (texte ou variable chaine de caractères)

. Dans ce cas de creation de fichier il n'y a aucune limite dans le nombre et la longueur des enregistrements

. Le fichier sera accessible séquentiellement en lecture par l'ordre READ associé a un ordre FORMAT libre -FORMAT ()- et en écriture par un ordre WRITE associé a un format quelconque -voir FORMAT chap. 01-3 - .

EXEMPLES

```
CREAT#(3)"SORTIE"
CREAT#(1,100)"SORTIE"
1010 CREAT#(ND,NV)F$
```

READ

FONCTION

LECTURE DE DONNEES A PARTIR D'UN FICHIER DISQUE DE TYPE SEQUENTIEL

FORME GENERALE

[n] READ(np,nf[,n1[,n2]])V$

REGLES

- np représente le numéro de périphérique désigné par l'ordre ASSIGN affecté au fichier de donnée a lire.
- nf représente le numéro de ligne du format : ce format est un format libre - FORMAT () -
- n1 est le numéro de ligne ou se branchera le programme dés que le fichier sera entierement lu. Si n1 est absent le système génerera une erreur 61 dés la rencontre de la fin du fichier
- n2 est inutilisé

EXEMPLES

```
0100 ASSIGN 14,(3)"IMPRIM"
0110 READ(14,120,200)A$
0120 FORMAT()
0130 PRINT A$(1;LEN A$)
0140 GOTO 110
0200 STOP FIN DE LISTAGE
```

WRITE

FONCTION

ECRITURE DE DONNEES SUR UN FICHIER DISQUE DE TYPE SEQUENTIEL

FORME GENERALE

[n] WRITE(np[,nf,[,n1[,n2]]])liste

REGLES

VOIR REGLES ORDRE WRITE (Chap. 01-3)

N.B Un format libre peut être utilisé pour l'écriture de données sur un fichier séquentiel.

17

TRAITEMENT DE TEXTE

ADD

FONCTION

CONSTRUCTION DE TEXTE

A$)ou(B$
[n] ADD V1$) (OU ( ) V2$

REGLES

ADD A$)B$ est équivalent à :
B$=B$ & A$ & BIN 13

ADD A$(B$ est équivalent à :
B$=A$ & BIN 13 & B$

Si LEN A$+1+LEN B$) LEN B$ il y a erreur

EXEMPLES

VOIR RECORD TEXTE

```
10 DIM A$(80),B$(240)
20 INPUT "B$=",B$
30 INPUT "A$=",A$
40 ADD A$)B$
50 DISP "B$=B$+A$ EN HEXA = ",DUMP B$
60 ADD A$(B$
70 DISP "B$=A$+B$+A$ EN HEXA =", DUMP B$
80 END
```

TAKE

FONCTION

EXTRACTION D'UNE CHAINE DANS UN TEXTE

A$ > OU < B$
[n] TAKE V1$ > (ou <) V2$

REGLES

TAKE A$<B$: Enlève dans A$ la première ligne de B$
TAKE A$>B$: Enlève dans A$ la dernière ligne de B$

EXEMPLES

```
10 DIM A$(10),B$(300)
20 INIT B$ TO "A"
30 TAKE A$ < B$
40 DISP "A$=",A$
50 DISP"B$=",B$
60 END
```

LOAD TEXTE

FONCTION

LECTURE SUR DISQUE DE LA TOTALITE D'UN TEXTE CONSIDERE COMME UN FICHIER SEQUENTIEL

FORME GENERALE

[n] LOAD TEXTE#(nd,nv)"nom",A$

REGLES

Le texte "nom" est lu dans sa totalité ou jusqu'à occurence de la longueur de A$.

La variable A$ est préalablement initialisée à blanc, avant la rencontre de l'ordre LOAD TEXTE.

EXEMPLES

```
0010 REM LOAD TEXTE/EDIT
0020 DIM FIC$(6),A$(8000)
0030 CLEAR 0 :INPUT"NOM DU TEXTE A EDITER:",FIC$
0040 INPUT"NUMERO DE L'UNITE DU DISQUE:",ND
0050 LOAD TEXTE(ND)FIC$,A$
0060 EDIT A$
0070 END
```

RECORD TEXTE

FONCTION

ECRITURE SUR DISQUE D'UN TEXTE SOUS FORME D'UN FICHIER SEQUENTIEL

FORME GENERALE

[n] RECORD TEXTE#(nd,nv)"nom",A$

REGLES

La chaine A$ est écrite sur disque sous forme d'un fichier séquentiel et peut être de longueur variable.

EXEMPLES

```
0010 REM ADD/RECORD TEXTE
0020 DIM FIC$(6),TX$(8000),A$(90)
0030 CLEAR 0 :INPUT"NOM DU FICHIER A ENREGISTRER:",
     FIC$
0040 INPUT"NUMERO DE L'UNITE DISQUE:",ND
0060 INPUT"CONTENU DE LA LIGNE:",A$
0070 IF A$="            " THEN 0100
0080 ADD A$(TX$
0090 GOTO 0060
0100 RECORD TEXTE(ND)FIC$,A$(1;LEN A$)
0110 END
```

EDIT

FONCTION

MISE EN PAGE, VISUALISATION OU IMPRESSION D'UNE CHAINE DE CARACTERES

FORME GENERALE

[n] EDIT A$,[,n]

REGLES

La chaine A$ doit avoir une structure de texte. Cette chaine est mise en page et imprimée ou visualisée selon que N est spécifié ou non, conformément aux conventions du traitement de texte ALCYANE. La chaine A$ n'est pas limitée.

SPECIFICATIONS :

La chaine doit se terminer par un code BIN 13. Elle peut contenir des directives de mise en page. Chaque directive est caractérisée par un "préfixe de directive" codifié par : BIN 13 & BIN 153 & BIN 153 (3 caractères) suivi de la désignation en clair de la directive.
En mode image ligne les lignes sont séparées par le code BIN 13. Dans les autres modes seules les directives servent de séparateurs. (changement de paragraphes etc...)

L'état initial du système de mise en page est initialisable, à partir de l'état courant pris comme modèle, lors de la rencontre de la directive $IN.

L'enchainement de plusieurs impressions est possible au moyen des directives AS et SU ("à suivre" et "suite").

Les tabulations et le titre sont initialisables au moyen des directives TA et TH :

De façon générale toutes les conventions de mise en page du traitement de texte sont conservées ici, l'ordre EDIT A$ est équivalent à la commande P du traitement de texte avec la totalité du texte dans A$.

PARTICULARITES RELATIVES A L'AFFICHAGE

Lors de l'affichage de la mise en page, l'écran est préalablement effacé et la mise en page est affichée à partir du haut de l'écran considéré comme un début de page. A chaque tranche de 20 lignes affichées et mises en page, ainsi qu'à la fin de l'affichage, la machine attend que l'utilisateur signale qu'il a pris connaissance de ce qui est affiché en pressant la touche CONT.

NOTES CONCERNANT L'IMPLEMENTATION

Dans cette instruction A$ est toujours le nom d'une chaine. Par contre N peut être soit directement une constante soit le nom d'une variable numérique.

EXEMPLES

```
10 DIM LG$(128),TX$(1000),DIR$(2),CE$(4),PA$(4)
20 DIM IL$(4),NPN$(5),OD$(1)
30 DIR$=HEX"9999":CE$=DIR$8"CE"
40 PA$=DIR$8"PA":OD$=HEX"OD"
50 INPUT "ENTRER LA LIGNE :",LG$
60 ADD LG$(TX$
70 ADD CE$(TX$
80 ADD NPN$(TX$
90 ADD OD$)TX$
100 EDIT TX$
110 END
```

LECT

FONCTION

AFFICHAGE DE VARIABLES ET MODIFICATION POSSIBLE AVEC INTERPRETATION DES TOUCHES DU CLAVIER

FORME GENERALE

[n] LECT A$[,V]

REGLES

Même syntaxe que l'ordre BASIC MODIF (affichage des variables et modification possible de leur contenu), mais avec interprétation des touches du clavier de fonction comme sous traitement de texte.

Les touches du clavier de fonction permettent :
- progression et régression par caractère et par mot ((--,--))
- effacement caractère et mot (DEL, CLEAR)
- positionnement en début et fin de zone (C,F)
- conversion majuscule et minuscule, tabulation
- fin d'opération pour les autres touches du pavé de fonction

La variable N, si elle est définie, reçoit le rang de la touche qui a servi à terminer l'opération.

NOTES SUR L'IMPLEMENTATION

- l'affichage a lieu à partir de la position courante, (ou définie par TAB D). Si cette position est plus basse que la 21ème ligne de l'écran, c'est celle-ci qui est prise pour origine.

- la zone de travail de cette instruction est limitée à 240 octets

EXEMPLES

```
10 DIM A$(10000)
20 LOAD TEXTE (nd),"nom"A$
30 EDIT A$,4
40 END
```

18

MULTIPOSTE ET MULTIALCYANE

UTIL

FONCTION

CONNAISSANCE DU NUMERO DE MACHINE, DU NUMERO DE POSTE ET DE LA TACHE EN COURS

FORME GENERALE

[n] UTIL V1,V2,V3

LOCK

FONCTION

VERROUILLAGE D'UN PERIPHERIQUE, FICHIER OU ENREGISTREMENT DE FICHIER

FORME GENERALE

[n] LOCK[*](np)[nom[,(ne)]]

REGLES

- le verrouillage par un utilisateur interdit les accès pour tous les autres sur ce périphérique, ou élément de ce périphérique. Ce rejet est indiqué par un message d'erreur (ERREUR 74).

- le verrouillage d'un enregistrement de fichier, s'étend sur un enregistrement physique du disque (nombre entier de secteurs). Sur un disque de 10M (secteur physique de 256 octets) le verrouillage d'un enregistrement de 128 octets interdira l'accès à l'enregistrement précédent ou suivant selon le cas.

- le numéro logique de l'imprimante est égal à 11 pour les instructions LOCK et UNLOCK

EXEMPLES

UNLOCK

FONCTION

DEVERROUILLAGE D'UN PERIPHERIQUE, FICHIER OU ENREGISTREMENT DE FICHIER

FORME GENERALE

[n] UNLOCK (np)[nom[,(ne)]]

REGLES

- Toute instruction UNLOCK doit s'appliquer sur le même élément considéré dans le LOCK précédent. Dans le cas contraire un message d'erreur (ERREUR 76) l'indique.

- UNLOCK ALL permet de déverrouiller tous les éléments propres à un utilisateur.

EXEMPLES

19

COMPATIBILITE DISQUE IBM

DEFINE UNIT

FONCTION

DEFINITION DE LA FORME DE TRAVAIL SUR UNE UNITE DISQUE

FORME GENERALE

DEFINE UNIT (nd,f[,sf[,t]])

REGLES

. nd represente le numero de l'unité disque sur laquelle doit se faire la définition de forme de travail.
. f représente la forme dans laquelle doit travailler l'unité disque désignée.
. sf représente la sous forme.
. t représente le type.

TABLEAU DES FORMES ET SOUS FORMES DE TRAVAIL

FORME	S/FORME	LONG SECT	TYPE FLOP	NBR FACE	DENSITE S/D	IBM
0	0	256	5 "	1	D	
1	0	256	5 "	2	D	
2	0	128	8 "	1	S	OUI
2	1	256	8 "	1	S	OUI
2	2	512	8 "	1	S	OUI
3	0	256	8 "	2	D	OUI
3	1	512	8 "	2	D	OUI
3	2	1024	8 "	2	D	OUI
3	3	256	8 "	2	S	OUI
4	0	256	8 "	1	D	
5	0	256	8 "	2	D	

. L'unité de disque double densité est equipée d'un buffer de tampon lui permettant de stocker plusieurs secteurs. Ce principe permet d'éviter la relecture de certains secteurs figurant déja dans ce buffer. En ajoutant 4 à la valeur de la sous forme donnée par le tableau, il est possible d'éviter cette mémorisation.

RSECT

FONCTION

LECTURE D'UN SECTEUR IBM

FORME GENERALE

[n] RSECT(nd,s) V$

REGLES

- nd = numéro de drive
  s = numéro du secteur à lire

- le disque sur lequel se fait la lecture doit obligatoirement être en forme IBM (2 ou 3)

EXEMPLES

```
100 DEFINE UNIT (9,2)
110 FOR S=32 TO 200
120 RSECT (9,5)A$
130 RECORD DATA#(3) "COPIE",(I)A$
140 I=I+1
150 NEXT S
160 STOP
```

WSECT

FONCTION

ECRITURE D'UN SECTEUR IBM

FORME GENERALE

[n] WSECT(nd,s)V$

REGLES

. nd = numéro de drive
  s = numéro de secteur à écrire

. le disque sur lequel se fait l'écriture doit être de forme IBM (forme 2 ou 3)

EXEMPLES

```
100 DEFINE UNIT (9,3)
110 FOR S=32 TO 200
120 LOAD DATA#(3) "COPIE",(I)A$
130 WSECT(9,S)A$
140 J=I+1
150 NEXT S
160 STOP
```

20

TELEMAINTENANCE

REMOTE

FONCTION

SUBSTITUTION DU SYSTEME CLAVIER-ECRAN LOCAL PAR UN SYSTEME A DISTANCE

FORME GENERALE

[n] REMOTE

REGLES

- pour effectuer une maintenance à distance le système esclave bascule son modem ON (ou pose son récepteur téléphonique sur son modem acoustique) et dès que les voyants du modem confirment l'établissement de la liaison il tape REMOTE.
  A partir de cet instant le système maitre émule totalement le clavier écran du système esclave et permet toutes les manipulations du système local.

- Pour permettre cette émulation, le système maitre doit avoir auparavant chargé un programme basic spécialement prevu pour cette application.

EXEMPLES

.VOIR DOCUMENTATION TELEMAINTENANCE SPECIFIQUE

LOCAL

FONCTION

RESTITUTION DU SYSTEME CLAVIER-ECRAN LOCAL

FORME GENERALE

[n] LOCAL

REGLES

. cette instruction permet de redonner son autonomie au système esclave, cette instruction est à effectuer par le système maitre qui a le controle complet du système appelant.

21

GESTION HORLOGE

FNTIME

FONCTION

LECTURE DE LA DATE ET HEURE SUR L'HORLOGE INTERNE.

FORME GENERALE

[n] V$=FNTIME

REGLES

. V$ est une chaine de 12 caractères qui contiendra les coordonnées Mois/Jours/Heures/Minutes/Secondes 1/10 de secondes au moment de l'interrogation de l'horloge par FNTIME

. Structure du résultat de la fonction

position 1 à 2	= Mois dans l'année	: 01 à 12
position 3	= Rang du jour dans la semaine	: 01 à 07
position 4 à 5	= Quantième du mois	: 01 à 31
position 6 à 7	= Heure	: 00 à 23
position 8 à 9	= Minutes	: 00 à 59
position 10 à 11	= Secondes	: 00 à 59
position 12	= 1/10 Secondes	: 00 à 09

EXEMPLES

```
0010 DIM MOIS$(12,9),JOUR$(7,8),M$(9),J$(8),A$(12)
0020 FOR I=1 TO 12
0030 READ MOIS$(I)
0040 NEXT I
0050 FOR I=1 TO 7
0060 READ JOUR$(I)
0065 NEXT I
0070 CLEAR D
0080 A$=FNTIME
0090 TAB D 5
0100 J$=JOUR$(VAL A$(3)):D=VAL A$(4;5):M$=MOIS$(VAL A$(1;2)
0110 H=VAL A$(6;7):M=VAL A$(8;9):S=VAL A$(10;11):X=VAL
     A$(12)
0120 WRITE(2,0130)J$,D,M$,H,M,S,X
0130 FORMAT(A8,B,N2,B,A9,"1981"//N2,"HEURES",N2,"MINUTES",
     N2,"SECONDES",N 1,"'")
```

```
0140 GOTO 0080
0150 END
1000 DATA JANVIER,FEVRIER,MARS,AVRIL,MAI,JUIN,JUILLET,AOUT,
     SEPTEMBRE,OCTOBRE,NOVEMBRE,DECEMBRE
1010 DATA LUNDI,MARDI,MERCREDI,JEUDI,VENDREDI,SAMEDI,
     DIMANCHE
```

SET TIME

FONCTION

MISE A JOUR DE L'HORLOGE INTERNE.

FORME GENERALE

[n] SET TIME V$

REGLES

. V$ est une chaine de 10 caractères de long qui contient les coordonnées Année/Mois/Jour/Heure/Minute auxquelles doit être initialisée l'horloge

. contenu de la chaine de caractères

- position 1 : détermine la position de l'année par rapport à la dernière année bissextile

année bissextile	: V$(1)	= "8"
année bissextile +1	:	= 4
année bissextile +2	:	= 2
année bissextile +3	:	= 1

- position 2 à 3 : Rang du mois dans l'année : 01 à 12
- position 4 : Rang du jour dans la semaine : 01 à 07
- position 5 à 6 : Quantième du mois : 01 à 31
- position 7 à 8 : Heure : 00 à 23
- position 9 à 10: Minutes : 00 à 59

. Au moment de l'exécution de la commande SET TIME les secondes sont mises à 0

. Après l'arrêt de la machine, l'horloge est maintenue.

. Après un arrêt prolongé de la machine ayant entrainé la décharge complète de l'accumulateur, il convient d'exécuter cette instruction pour remettre à jour et redémarrer l'horloge.

EXEMPLES

Nous sommes le jeudi 15 octobre 1981
11 heures 53 mn

```
0010 A$="4104151153"
0020 SET TIME A$
```

```
A$(1)    =  "4" 1981 est l'année suivant l'année bissextile
A$(2;3)  = "10" Mois d'octobre
A$(4)    =  "4"  Jeudi
A$(5;6)  = "15" Nous sommes le 15 du mois
A$(7;8)  = "11" il est 11 heures
A$(9;10)= "53" et 53 mn
```

22

ERREURS BASIC

GESTION DES ERREURS

FONCTION

CES ERREURS SONT GENEREES PAR LE SYSTEME A LA SUITE D'INCIDENTS SUR DES PERIPHERIQUES OU A LA SUITE D'ERREURS DE PROGRAMMATION

FORME GENERALE

CES ERREURS GENERENT UNE INTERRUPTION DU PROGRAMME ET L'AFFICHAGE DU MESSAGE: ** ERREUR nnnn **
** LIGNE nnnn ** , SAUF SI LE PROGRAMME EN PREVOIT LA GESTION PAR "IF ERR".

REGLES

1 MAUVAISE SYNTAXE POUR UNE VARIABLE DIMENSIONNEE
2 PAS DE END OU DE STOP EN FIN DE PROGRAMME
3 LIGNE INEXISTANTE
4 MANQUE DES PARENTHESES GAUCHES OU DROITES
5 LIBELLE INCORRECT
6 NUMERO DE LIGNE > 9999
7 ERREUR SYNTAXE DANS UNE VARIABLE OU UNE CONSTANTE
8 EXPRESSION INCORRECTE
9 MAUVAISE AFFECTATION DE VARIABLE
10 REPRISE DE PROGRAMME PAR CONTINUE INCORRECTE
11 MANQUE UNE VIRGULE
12 INDICE DE VARIABLE <=0
13 MAUVAIS INDICAGE D'UNE CHAINE DE CARACTERES
14 VARIABLE MAL INDICEE OU INDICE > A LA DECLARATION
15 VARIABLE MAL DEFINIE OU NON REFERENCEE
16 TABLEAU NE FIGURANT PAS DANS L'ORDRE DIM
17 VARIABLE > A SA DECLARATION
18 NOMBRE INCORRECT
19 MAUVAISE LISTE DE VARIABLES
20 DEPASSEMENT DE CAPACITE
21 ERREUR NUMERO DE "FLAG"
22 DIVISION PAR ZERO
23 MAUVAIS NUMERO DE LIGNE FORMAT
24 ERREUR DE SYNTAXE (INSTRUCTION INCORRECTE OU INCOMPLETE)
25 EXECUTION D'UN RETURN SANS GOSUB PREALABLE OU GOSUB SANS RETURN
26 ZONE MEMOIRE SATUREE
27 DONNEES POUR L'ORDRE INPUT INCOMPLETES

28 SPECIF INCORRECTE DANS FORMAT OU MAUVAISE DEFINITION DE ZONE ALPHANUMERIQUE
29 ORDRE BASIC INCORRECT
30 NOMBRE DE CARACTERES D'UNE LIGNE) 192
31 TOUCHE NON VALIDE (KEY)
32 NUMERO DE LIGNE MANQUANT OU INCORRECT
33 ZONE DE CALCUL EN DEBORDEMENT
34 FIXED N N<0 OU N>13 FLOAT
35 MAUVAISE STRUCTURE DE PROGRAMME (LOAD/RECORD)
36 MAUVAISE DEFINITION DE VARIABLE ALPHANUMERIQUE ET NUMERIQUE
37 MAUVAISE CHAINE ALPHANUMERIQUE EXPRIMEE EN HEXADECIMAL
38 MANQUE NEXT DANS L'INSTRUCTION FOR ASSOCIEE
39 ERREUR TYPE DE VARIABLE (ALPHA OU NUMERIQUE)
40 MAUVAISE DEFINITION DE SOUS PROGRAMME (CF SUBROUTINE)
41 LONGUEUR ENREGISTREMENT DISQUE INCORRECTE
42
43 PANNE HARDWARE SUR IMPRIMANTE
44 PLUS DE PAPIER SUR IMPRIMANTE
45 IMPRIMANTE NON SELECTEE
46 LONGUEUR INCORRECTE (KEY)
47
48 MAUVAIS NUMERO DE PERIPHERIQUE
49 VARIABLE REDIMENSIONNEE
50 LIGNE MAL TERMINEE
51 MAUVAIS FORMATAGE D'UN DISQUE OU DRIVE 600K "NOT READY"
52 TENTATIVE D'UTILISATION D'UNE OPTION NON AUTORISEE SUR LA MACHINE
53
54 NOM DE FICHIER OU NUMERO DE DISQUE INCORRECT
55 ENREGISTREMENT D'UN PROGRAMME SOUS UN NOM DIFFERENT DE CELUI DU LOAD#
56 FICHIER DEJA CREE
57 CATALOGUE SATURE
58 FICHIER INEXISTANT OU ERREUR SUR LE TYPE DE FICHIER
59 CHARGEMENT D'UN PROGRAMME PAR INCLUDE DEJA RESIDENT OU RELEASE D'UN PROGRAMME INEXISTANT
60 DISQUE SATURE
61 NOMBRE OU NUMERO D'ARTICLE D'UN FICHIER DF DONNEES MAL DEFINI
62 ERREUR SUR UN PARAMETRE DE "INOUT"
63 ERREUR DE PARITE SUR "READ" (V24)
64 PROTECTION ECRITURE
65 SATURATION TABLE
66 INDEX PRIMAIRE NON EN LIGNE
67 FICHIER DATA NON EN LIGNE (NON DONNE PAR USE)
68 FICHIER DATA DE LISTE USE NON FOURNI LORS DU BUILD "USE"
69 CLE NON VALIDE
70

```
71
72  DEFAUT DE TRANSMISSION (multialcyane)
73
74  VERROUILLAGE FICHIER PAR "LOCK"
75  TABLE DES VERROUS SATUREE (LOCK)
76  DEVERROUILLAGE D'UN FICHIER NON VERROUILLE
77
78
79
80
81  PAS DE READY
82  ERREUR PARITE SUR DATA "CRC"
83  FAUTE AVANT ACCES PISTE
84  FAUTE APRES ACCES PISTE
85  ERREUR PARITE 10 MILLIONS
86  DESELECTION DU DRIVE
87  FAUTE APRES ACCES SECTEUR
88  LECTURE # ECRITURE
89  SECTEUR NON VALIDE NON REPERTORIE
90  PISTE A ACCEDER ) 76
91  BUTEE PISTE 00
92  LECTURE ENTETE IMPOSSIBLE (LORS DE VALIDATION SECTEUR
    OU PARITE SUR P/S)
93  ACCES PISTE IMPOSSIBLE (ERREUR PARITE SUR P/S OU NUMERO
    P/S # RECHERCHE)
94  ACCES SECTEUR IMPOSSIBLE
95  PARITE SUR DATA
96  ECRITURE DATA IMPOSSIBLE
97
98  ERREUR SUR PLOT
99
100
101 DEFAUT D'ACCES CHIP DU CONTROLEUR DOUBLE DENSITE
102 PISTE INACCESSIBLE
103 FORMATTAGE DU DISQUE IMPOSSIBLE
104 ERREUR D'ACCES AU CHIP CONTROLEUR EN ECRITURE OU EN
    LECTURE
105 DISQUE PROTEGE EN ECRITURE
106 LECTEUR PAS PRET, PORTE OUVERTE
107 IMPOSSIBLE A ECRIRE
108 DEFAUT DANS UNE ENTETE SECTEUR
109 ERREUR DANS LES DONNEES D'UN SECTEUR
110 AUTRE ERREUR EN LECTURE OU EN ECRITURE
111 ORDRE INEXISTANT
112 MEMOIRE INEXISTANTE
113 DEFAUT D'INITIALISATION LIAISON
114 DESYNCHRONISATION ENTRE ALCYANE ET CONTROLEUR
115 ORDRE INEXISTANT
116 ACCES IMPOSSIBLE
117 ORDRE INEXISTANT
```

118 ECRITURE INCORRECTE
119 DEFAUT EN RECHERCHE DE PISTE
120 NON REPERTORIEE
121 LECTURE DE DELETE DATA
122 EXECUTION D'ORDRE NON EFFECTUEE