1. Mise en service

Pour utiliser l'interpréteur du Basic 128, il suffit de mettre le MO5 sous tension et de répondre 1 au menu affiché.
Le message

```
BASIC 128 v1.0 (c) Microsoft 1985
110  127bytes free
OK
```

apparaît; l'interpréteur Basic est actif.

L'interpréteur du Basic 128 est entièrement résident. Toutes les commandes, instructions et fonctions sont disponibles, y compris celles qui permettent de gérer des fichiers sur disquette.
Les réponses 1 ou 2 au menu entraînent le chargement à partir d'un lecteur de disquettes du programme AUTO.BAT, s'il existe, et son exécution.

2. Clavier et éditeur intégré

L'interpréteur Basic comporte un éditeur plein écran qui simplifie considérablement l'écriture et la modification de programmes.
Il est possible de modifier une ligne affichée à l'écran sans la retaper. Après la modification, appuyer sur la touche ENTREE revient à introduire la nouvelle ligne dans le programme.

Clavier

Les touches spécifiques du clavier sont les suivantes:

STOP Arrête l'exécution du programme en cours. Rien n'est modifié sur l'écran. La reprise se fait en appuyant sur n'importe quelle touche sauf CNT , . et STOP.
RAZ Efface l'écran et ramène le curseur en haut à gauche.

CNT S'utilise simultanément avec une autre touche:

CNT-C Interrompt l'exécution du programme ou de la commande en cours. Cette interruption n'intervient qu'à la fin de l'instruction en cours, ce qui peut durer quelque temps avec l'instruction PLAY.
Interrompt la numérotation automatique commandée par AUTO. La ligne courante n'est pas mémorisée.

CNT-X Efface les caractères, depuis la position du curseur jusqu'à la fin de la ligne.

CNT-W Place un lien logique entre la ligne où se trouve le curseur et la ligne qui suit. Cette facilité est utilisée pour rassembler deux lignes de programme en une seule.

 Déplacement du curseur

 Place le curseur en haut à gauche.

INS Bascule le mode d'insertion. Au passage en mode insertion, la caractère qui se trouve au-dessus du curseur passe en vidéo inverse. Les caractères tapés ensuite viennent se placer immédiatement à gauche du curseur. Un deuxième appui sur la touche INS ramène au mode normal.

EFF Supprime le caractère placé au-dessus du curseur.

DEL Supprime le caractère placé à gauche du curseur.

Modification d'une ligne

Une ligne de programme ou une ligne de commandes ou d'instructions en mode direct peut comporter jusqu'à 255 caractères. Au fur et à mesure de la frappe, quand le curseur arrive au bord droit de l'écran, le retour au début de la ligne suivante se fait automatiquement.
Pour modifier une ligne présente entièrement à l'écran, il suffit de placer le curseur sur la partie à corriger et de taper les corrections soit en réécrivant sur les caractères à modifier, soit en supprimant, soit en insérant, soit les trois à la fois.
La nouvelle ligne est prise en compte en appuyant sur ENTREE, sans qu'il soit nécessaire d'amener le curseur en fin de ligne.

Création et modification d'un programme

Toute ligne commençant par un nombre compris entre 0 et 63999 est prise comme une ligne de programme. Son introduction dans le programme se fait quand on appuie sur ENTREE.
S'il existe déjà une ligne de même numéro, la nouvelle ligne la remplace.
Si la nouvelle ligne ne comporte que le numéro suivi ou non de caractères "espace", la ligne est supprimée.

Lors de l'entrée d'un programme, on peut obtenir une numérotation automatique des lignes par la commande AUTO.

La visualisation de tout ou partie d'un programme résidant en mémoire est faite avec la commande LIST.
La modification d'une ligne existante peut être faite soit en la réécrivant, soit en la modifiant directement si elle est visualisée.
La suppression d'une partie des lignes du programme se fait avec la commande DELETE.
La suppression de la totalité du programme est faite par la commande NEW.

3. Modes de fonctionnement

Quand l'interpréteur Basic est actif, le message "OK" apparaît. Dans cette situation, on peut utiliser Basic de deux manières différentes:

- Mode d'exécution directe

Ce mode permet d'effectuer des calculs, de donner des valeurs à des variables, d'appeler des fonctions ou d'exécuter un grand nombre d'instructions et toutes les commandes.

Chaque ligne est terminée par ENTREE et le résultat est obtenu immédiatement.

Ce mode peut servir à calculer comme avec une calculette évoluée. à tester certaines instructions ou à consulter certaines valeurs lors de l'arrêt d'un programme.

Exemples:

```
PRINT 12*12
144
Ok
P=2*35
Ok
PRINT P*1.1
24.145
Ok
```

- mode programme

Pour entrer ou modifier un programme, il faut commencer chaque ligne par un numéro. Le numéro de ligne doit être compris entre 0 et 63999 inclus. Dans ce cas, la ligne est mémorisée et insérée dans le programme présent en mémoire.

L'exécution est obtenue par la commande RUN.

Une ligne de programme peut contenir des instructions, des commandes ou des commentaires. Cependant l'exécution de certaines commandes interrompt le déroulement du programme.

Exemple:

```
10 PRINT 3*24
RUN
 72
Ok
```

4. Structure d'une ligne

Une ligne de programme Basic doit avoir la forme générale suivante :

numéro instruction : instruction : instruction : ...

Le numéro de ligne est un nombre entier compris entre 0 et 63999.
La ligne de programme doit comprendre au moins une instruction.
Lorsqu'il y a plusieurs instructions, elles sont séparées par le caractère deux-points (:).
La longueur de la ligne ne doit pas dépasser 255 caractères. Il convient de tenir compte du fait que les minuscules accentuées sont codées sur trois caractères.
Les caractères qui suivent l'instruction REM ou le caractère apostrophe (') qui lui est équivalent, ne sont pas considérés comme des instructions et peuvent servir de commentaires. Des commentaires peuvent être ajoutés en fin de ligne après le caractère apostrophe (') ou après l'instruction REM.
Les espaces sont facultatifs sauf quand une variable précède un mot clé du langage. Il peut y avoir un nombre quelconque d'espaces entre un mot clé, une constante ou une variable.
Les mots clés et les variables peuvent être écrits indifféremment en majuscules ou en minuscules sans accents. L'interpréteur Basic transpose automatiquement les minuscules en majuscules.

exemple :

```
10 REM Je compte de 1 à 10
20 FOR I=1 to 10
30   PRINT I, 'affiche le nombre
40 next I
50 PRINT "c'est fini" : END
```

5. Alphabet utilisé

L'interpréteur Basic utilise le jeu de caractères ASCII comprenant 96 caractères affichables dont le code est compris entre 32 et 127. Les 32 premiers caractères peuvent être employés comme données en les introduisant directement au clavier (touche CNT + lettre) ou par la fonction CHR$.

Les minuscules accentuées et le ç sont utilisables à partir du clavier (frappe ACC + lettre ou ACC + signe + lettre) et sont codés sur trois caractères.

Les codes 128 à 255 sont destinés à recevoir les caracteres définis par l'utilisateur au moyen de l'instruction DEFGR$, ils sont ensuite appelables par la fonction GR$(n) ou CHR$(128+n) avec n variant de 0 à 127.

6. Constantes

Il existe deux types de constantes :
— les constantes numériques,
— les constantes "chaînes de caractères".

Constantes numériques

Les constantes numériques sont des nombres positifs ou négatifs. Il existe trois types de constantes différents suivant la précision.

Entiers

Tous les nombres entiers positifs ou négatifs compris entre −32768 et +32767.
exemples :
13
−5273

Réels en simple précision

Tous les nombres positifs ou négatifs représentés sous la forme m*10^e où la mantisse m a sept chiffres significatifs et où l'exposant e est compris entre - 38 et +38. Dans la pratique, la précision exacte du nombre varie entre six et sept chiffres. On peut préciser, mais ce n'est pas obligatoire, qu'une constante est en simple précision par le signe ! à la fin.
exemples:
−1.6 E−19
6.02 E23
2.7828
12345!

Réels en double précision

Ils sont identiques aux réels en simple précision mais avec une mantisse de dix-sept chiffres significatifs. Dans la notation d'un nombre en double précision, la lettre marquant l'exposant est D au lieu de E. On peut, quand il n'y a pas d'exposant, préciser qu'une constante est en double précision par le signe # à la fin.
exemples:
3.1415926535#
.54321 D−27

Notation des constantes numériques

Les constantes numériques s'écrivent soit sous la forme décimale (12.3456), soit sous la forme "scientifique" (.123456E2).
Les nombres entiers peuvent être écrits en utilisant une base non décimale.

Notation hexadécimale
Une constante hexadécimale est un nombre exprimé en base 16. Il est précédé des caractères &H. Les chiffres sont 0,1,2,3,...,9,A,B,C,D,E,F.
exemples:
&HFFFF équivalent à 65535
&H32F

Notation octale
Une constante octale est un nombre exprimé en base 8. Il est précédé des caractères &O ou & seulement. Les chiffres sont 0,1,2,3,4,5,6,7.
exemples:
&O622
&1234

Notation binaire
Une constante binaire est un nombre exprimé en base 2, donc représenté uniquement de 0 et de 1. Il est précédé des caractères &B. Cette notation est particulièrement intéressante pour représenter les bits d'un octet ou d'une paire d'octets, spécialement lors de l'utilisation des opérateurs logiques (NOT, AND, OR, XOR, IMP, EQV).
exemples:
&B10101010
&B100000001

Constantes chaînes de caractères

Une constante chaîne de caractères est une suite de caractères encadrée par des guillemets ("). Le nombre de caractères de la chaîne, appelé longueur de la chaîne, doit être compris entre 0 et 255. Une chaîne qui ne contient aucun caractère est une chaîne vide.
Tous les caractères (du code 0 au code 255) peuvent être inclus dans une chaîne. Il faut remarquer que les minuscules accentuées et le ç quand ils sont introduits au clavier occupent trois caractères; il faut en tenir compte pour le calcul de la longueur.

exemples:
"QUE J'AIME A FAIRE CONNAITRE UN NOMBRE UTILE AUX SAGES"
"La concierge est dans l'escalier"

7. Variables

Les variables sont des emplacements mémoire qui possèdent un nom propre et qui sont destinés à recevoir les différentes données utiles au déroulement d'un programme. Une variable peut recevoir la valeur d'une constante ou le résultat d'un calcul ou d'une transformation de constantes et de variables.
Il y a quatre types de variables correspondant aux quatre types de constantes:
— variables numériques entières,
— variables numériques en simple précision,
— variables numériques en double précision,
— variables chaînes de caractères.

Nom des variables

Les noms de variables sont composés d'une lettre au moins, suivie éventuellement de lettres et de chiffres. Le nom d'une variable peut comporter jusqu'à 255 caractères. Toutefois, seuls les seize premiers sont significatifs et serviront à distinguer les variables entre elles. Deux variables dont les noms ont leurs seize premiers caractères identiques sont donc considérées comme identiques.
Le nom d'une variable ne peut pas commencer par un mot clé de Basic (ces mots sont appelés mots réservés, leur liste est donnée en Annexe 5). Ainsi TOTAL, FORCE, VALEUR ne peuvent pas être des noms de variable car ils commencent par TO, FOR, VAL.

exemples:
SOMME
A2
RECAPITULATIFDESDEPENSESDUMOIS

Le nom d'une variable doit être suivi de l'un des caractères (%, #, $) pour en préciser le type (entier,double précision ou chaîne de caractères). Le caractère! peut être utilisé pour indiquer une variable en simple précision. Toute variable qui n'est pas terminée par %, #, $ est considérée comme étant en simple précision.
La signification de ces caractères est la suivante:
% variable numérique entière
! variable numérique en simple précision
# variable numérique en double précision
$ variable chaîne de caractères

exemples:
QUOTIENT variable simple précision
A3! variable simple précision
PI# variable double précision
NOBJET% variable entière
ADR$ variable chaîne de caractères

Le type d'une ou de plusieurs variables peut être également précisé par l'une des instructions DEFINT, DEFSNG, DEFDBL, DEFSTR.

Tableaux

Le Basic offre la possibilité de regrouper un grand nombre de variables de même type sous le même nom dans un tableau.
Un tableau peut avoir une ou plusieurs dimensions. Chaque variable est accessible en donnant une valeur à chacun des indices et constitue un élément du tableau. On utilise quelquefois le terme variable indicée.
Tous les tableaux qui possèdent plus de 10 éléments doivent être déclarés au préalable par l'instruction DIM.
exemple:

```
DIM T(12)
```

Cette instruction crée un tableau numérique à une dimension de treize éléments. En effet, l'indice varie de la valeur 0 au maximum

indiqué par DIM. T(3) désigne le quatrième élément du tableau T.

DIM PHS(2,10)

Cette instruction crée un tableau chaîne de caractères à deux dimensions. Le premier indice varie de 0 à 2, le second de 0 à 10. Il comporte donc 33 éléments.

La syntaxe du Basic accepte des tableaux ayant jusqu'à 255 indices, chaque indice pouvant aller de 0 à 32767.
Si l'évaluation d'un indice conduit à une valeur qui n'est pas entière, cette valeur est arrondie à l'entier le plus proche.
exemples:
A(3.5) est équivalent à A(4)
A(3.49) est équivalent à A(3)

Les noms des tableaux obéissent aux mêmes règles que les variables de même type:
% entiers
! simple précision
# double précision
$ chaîne de caractères.

Volume occupé par une variable ou un tableau

Chaque type de variable correspond à une manière de représenter une valeur en mémoire. Le volume occupé en fonction du type de la variable est le suivant:

entier	2 octets
simple précision	4 octets
double précision	8 octets
chaîne de caractères	longueur de la chaîne en octets

Ainsi pour un tableau, il faut multiplier le volume occupé par un élément par le nombre total d'éléments du tableau.
exemple:
DIM A#(19) réserve un tableau numérique en double précision dont les valeurs vont occuper 20*8=160 octets en mémoire.

La taille d'un tableau est limitée à 64 k-octets.

Conversions

Il est interdit de mélanger nombres et chaînes de caractères dans une expression.
Le Basic accepte des expressions numériques "mixtes", c'est-à-dire comprenant des variables et des constantes numériques de type différent.
L'évaluation d'une expression se fait alors en convertissant s'il y a lieu les valeurs dans le type le plus précis.
exemples:

```
? 12/23.5
 5.106383
```

L'entier 12 est converti en simple précision et la division est faite en simple précision.

```
? 2*3.14159265358#
 6.283185307
```

Après conversion de 2 en double précision, le calcul et son résultat sont en double précision.

Les opérations arithmétiques sont faites en double précision si l'un des opérandes comporte plus de sept chiffres significatifs ou s'il est noté en double précision.

Les fonctions mathématiques du langage (SIN, ATN, LOG,...) sont calculées en simple précision si leur argument est en simple précision et en double précision si leur argument est en double précision.
exemple:

```
?LOG(2.000001)
 .693197
?LOG(2.000001#)
 .6931971793099887
```

Lors de l'affectation d'une valeur à une variable de précision différente, plusieurs cas sont à envisager:

— la valeur est de précision supérieure
Elle est convertie par arrondi (et non pas par troncature) à la précision de la variable.
exemple :

```
[illegible]
[illegible]
[illegible]
```

— la valeur est de précision inférieure
Dans le cas d'un entier, il est converti en conservant sa valeur exacte. Par contre, un nombre en simple précision affecté à une variable en double précision est complété par des chiffres non significatifs mais non nuls.
exemple :

```
[illegible]
[illegible]
[illegible]
```

La valeur n'est améliorée en aucune mesure!
On peut remplir les derniers chiffres significatifs par des 0 en passant par la conversion des nombres en chaînes de caractères.
exemple :

```
[illegible]
[illegible]
[illegible]
```

8. Expressions

Expressions numériques

Une expression numérique peut être une constante, une variable ou encore une combinaison de constantes, variables et fonctions reliées entre elles par des opérateurs.

Les opérateurs numériques sont classés en trois catégories :
— opérateurs arithmétiques,
— opérateurs de relation,
— opérateurs logiques.

Opérateurs arithmétiques

Voici la liste des opérateurs arithmétiques classés par ordre de priorité décroissante :

^	élévation à la puissance	X^Y
	négation	−X
*, /	multiplication, division	X*Y, X/Y
\	division entière	X\Y
MOD	modulo ou reste de la division entière	X MOD Y
+ −	addition, soustraction	

La priorité entre opérateurs signifie que dans une expression où deux opérateurs sont appliqués sans parenthésage, celui qui est prioritaire est évalué en premier. A priorité égale, l'évaluation commence par celui de gauche.
exemple :

```
[illegible]
[illegible]
[illegible]
```

Division entière \

Cette opération n'est possible que si les opérandes sont compris entre −32768 et 32767. Avant la division, les deux opérandes sont arrondis au besoin. Le résultat est le quotient tronqué de la division.
exemples :

```
[illegible]
[illegible]
[illegible]
[illegible]
```

[illegible]

Cas d'erreurs

— Si un calcul amène au dépassement de la capacité dans le type considéré (valeur inférieure à – 32768 ou supérieure à 32767 en entier, valeur inférieure à 10^{-38} ou supérieure à 10^{38} en réel) le message d'erreur Overflow apparaît.
— Une division par zéro est signalée par:
Division by zero
Dans les deux cas, l'exécution est interrompue.

Opérateurs de relation

Les opérateurs de relation sont utilisés pour comparer deux valeurs. Le résultat de la comparaison est une valeur "booléenne" VRAI ou FAUX. Par convention, VRAI a pour valeur –1 et FAUX a pour valeur 0.

Les opérateurs de relation sont les suivants:

=	égal à	X=Y
<>	différent de	X<>Y
<	inférieur à	X<Y
>	supérieur à	X>Y
<=	inférieur ou égal à	X<=Y
>=	supérieur ou égal à	X>=Y

Le résultat d'une relation peut être utilisé dans une instruction de branchement:

[illegible]

Il importe de bien distinguer l'opérateur de relation = destiné à tester l'égalité de deux valeurs, de l'instruction – qui affecte une valeur à une variable.
exemple:

[illegible]

signifie:
Si la valeur de A est 3 alors affecter 2 à B.

Puisque le résultat d'une comparaison est une valeur "booléenne" représentée par un nombre entier (0 ou –1), il peut être affecté à une variable numérique.
exemple:

[illegible]

Opérateurs logiques

Les opérateurs logiques permettent d'effectuer des opérations booléennes et par conséquent, de combiner plusieurs relations dans une condition.
Six opérateurs logiques sont disponibles:
NOT NON logique
AND ET logique
OR OU logique
XOR OU exclusif
IMP implication
EQV equivalence
NOT est le seul opérateur qui ne s'applique qu'à un seul opérande.
En prenant V pour VRAI et F pour FAUX, la table des opérateurs est la suivante:

X	Y	NOT X	X AND Y	X OR Y	X XOR Y	X IMP Y	X EQV Y
V	V	F	V	V	F	V	V
V	F	F	F	V	V	F	F
F	V	V	F	V	V	V	F
F	F	V	F	F	F	V	V

Noter que:
— l'expression X IMP Y est équivalente à (NOT X) OR Y
— l'expression X EQV Y est équivalente à NOT (X XOR Y)

exemples:

```
IF A=B AND C=D THEN 500
IF NOT A OR B THEN 400
```

Fonctionnement des opérateurs logiques

En pratique, les opérateurs logiques travaillent sur des entiers (entre −32768 et 32767), c'est-à-dire sur des suites de seize bits.
Les opérations sont effectuées sur les bits de même position (un bit à 1 est VRAI, un bit à 0 est FAUX).
exemple:

```
? 127 AND 16
 16
```

127 est représenté en binaire par 1111111
16 est représenté en binaire par 10000
le résultat vaut donc 10000

Cette particularité peut être utilisée pour extraire la valeur d'un bit dans un ou deux octets à la fois, à l'aide d'un masque.
Le masque &B11110000 ou encore 240 en décimal permet d'extraire avec AND les quatre bits de poids fort d'un octet.
Pour savoir si un nombre X% est pair, on peut faire:
X% AND &B1111111111111110 = X%
Le résultat vaut −1 (VRAI) si X% est pair,
0 (FAUX) si X% est impair.

Priorité des opérations

Lors de l'évaluation d'une expression numérique, l'ordre de priorité d'évaluation est le suivant:
1 - fonctions (ABS, SIN, INT,...)
2 - exponentiation ^
3 - négation −
4 - multiplication, division *,/
5 - division entière \
6 - modulo MOD
7 - somme, différence +,
8 - relations >,>,=,<=,>=,<>
9 - opérations logiques NOT, AND, OR, XOR, EQV, IMP
On peut modifier l'ordre de priorité avec des parenthèses, les opérations situées dans les parenthèses les plus internes sont évaluées en premier.

Fonctions numériques

Les fonctions numériques disponibles (argument numérique, résultat numérique) sont les suivantes:

ABS(x)	valeur absolue de x
ATN(x)	arctangente de x
CDBL(x)	conversion en double précision
CINT(x)	conversion en entier
COS(x)	cosinus de x
CSNG(x)	conversion en simple précision
EXP(x)	exponentielle de x
FIX(x)	partie entière de x
INT(x)	plus grand entier inférieur ou égal à x
LOG(x)	logarithme népérien de x
MAX(x,y,...)	maximum de x,y,...
MIN(x,y,...)	minimum de x,y,...
RND(x)	générateur de nombres aléatoires
SGN(x)	signe de x
SIN(x)	sinus de x
SQR(x)	racine carrée de x
TAN(x)	tangente de x

Expressions chaînes de caractères

Une expression chaîne de caractères peut être une constante, une variable ou encore une combinaison de constantes, variables et fonctions reliées entre elles par des opérateurs.

Il n'existe qu'un seul opérateur sur chaîne de caractères :
+ concaténation
Cet opérateur permet de mettre deux chaînes bout a bout pour en faire une troisième.
exemples :

```
? "AUTO" + "ROUTE"
AUTOROUTE
A$="ETRE ";B$="OU NE PAS "
? A$+B$+C$+A$
ETRE OU NE PAS ETRE
```

La concaténation ne place pas d'espace entre les deux chaînes. Il faut donc mettre des espaces a la fin de chaque mot pour faire une phrase.

Comparaison de chaînes de caractères

Les opérateurs de relation peuvent servir à comparer des chaînes de caractères, mais le résultat de la comparaison est logique, donc exprime sous forme numérique : −1 (VRAI) ou 0 (FAUX).
La comparaison se fait caractère par caractère, à partir du début de chaque chaîne. Un caractère est considéré comme plus petit qu un autre si son code dans le jeu de caractères ASCII est inférieur.
Tant que les caractères sont égaux, la comparaison se poursuit. La chaîne qui possède le premier caractère inférieur au caractère correspondant dans l'autre chaîne est considérée comme la plus petite.
Si la fin de l'une des deux chaînes est atteinte avant qu'une différence soit apparue, la chaîne la plus courte est considérée comme plus petite.
Dans une comparaison, tous les caractères sont pris en compte, y compris les espaces et les caractères non visualisables.
exemples :
"AB" < "FG" car A est inférieur à F
"ABCD" < "ABCDEF" car "ABCD" est plus courte
"GRAND" < "GROS" car A est inférieur à O
LOUIS" < "Louis" car O est inférieur à o

Les expressions chaînes peuvent aussi comporter des fonctions dont le résultat est une chaîne (LEFT$, MID$....).

9. Entrées-sorties et fichiers

Entrées-sorties généralisées

La gestion des échanges avec les principaux périphériques (cassettes, disquettes, clavier, écran, imprimante et communication série) est faite par le même jeu d'instructions à usage général. Ces instructions permettent de gérer les transferts concernant programmes, données sous forme ASCII et données sous forme binaire.
L'orientation des informations vers un périphérique ou un autre est faite par le choix du nom de périphérique, de ses options de fonctionnement et du nom de fichier à transférer. Cet ensemble de noms constitue le descripteur de fichier.

Descripteur de fichier

Le descripteur de fichier est une chaîne de caractères qui comprend trois parties organisées de la manière suivante :

Nom du périphérique : (options) nom du fichier

Suivant les cas, chacune de ces trois parties peut être facultative.

Nom du périphérique et options

Les noms de périphériques sont les suivants :

0	1er lecteur de disquettes ou QDD
1	2^{e} lecteur de disquettes
2	3^{e} lecteur de disquettes
3	4^{e} lecteur de disquettes

CASS magnétophone à cassettes
COMM voie de communication série
KYBD clavier
LPRT imprimante parallèle
SCRN écran
Le nom du périphérique, quand il est présent, doit toujours être suivi de deux points (:).
Si le nom du périphérique n'est pas précisé, sa valeur par défaut est:
SCRN: avec la commande LIST et TRON
0: si une unité de disquettes est connectée, pour toutes les autres commandes et instructions
CASS: si seul le lecteur-enregistreur de cassettes est connecté
L'instruction DEVICE permet de modifier le périphérique pris par défaut dans une instruction d'entrée ou de sortie.

Options

Quatre périphériques acceptent des options: l'imprimante parallèle (LPTR:), la voie de communication série (COMM:), les unités de disquettes (0: 1: 2: 3: 4:) et la cassette.

Imprimante parallèle:
L'option indique le nombre de caractères par ligne: entre 0 et 255. Par défaut, ce nombre est fixé à 40. 0 indique une ligne infinie
exemple:
LPRT:(80) désigne une imprimante à 80 colonnes.

Voie de communication série:
L'option comporte obligatoirement trois parties.
— le 1er chiffre indique la vitesse de transmission
1 110 bauds
2 300 bauds
3 600 bauds
4 1 200 bauds
5 2 400 bauds
6 4 800 bauds
7 9 600 bauds
8 19 200 bauds
— le 2e chiffre indique le nombre de bits transmis
7 transmission sur 7 bits
8 transmission sur 8 bits
la parité n'est pas gérée lors de la transmission série; elle doit donc être comptée dans le nombre de bits transmis.
— le 3e chiffre et les suivants indiquent le nombre de caractères par ligne (entre 0 et 255) comme pour l'imprimante parallèle, 0 correspondant à une ligne infinie. Un code de retour à la ligne est émis au bout du nombre de caractères indiqué
La recopie d'écran doit s'effectuer avec une ligne infinie.
exemple:
COMM:(47120) précise une transmission à 1 200 bauds (4) sur 7 bits (7) avec 120 caractères par ligne (120).

Unité de disquettes
L'option est un commentaire associé au fichier lors de son écriture sur la disquette. Ce commentaire est une chaîne de huit caractères maximum.
Ce commentaire est affiché sur la droite de l'écran quand on demande le catalogue de la disquette (DIR).
exemple:
SAVE "13 mai:TEST"
le commentaire 13 mai est associé au fichier TEST.BAS .

Cassette :
CASS:(1) fixe la vitesse d'écriture à 1 200 bauds (MO5).
CASS:(2) fixe la vitesse d'écriture à 2 400 bauds.

Nom du fichier

Le nom d'un fichier est composé de deux parties séparées par un point (.):
nom . suffixe
Le nom comporte huit caractères au plus: tous les caractères sont permis à l'exclusion du code 0, du code 255, de deux parenthèses et du point.
Le suffixe comporte trois caractères au plus avec les mêmes restrictions que le nom. Le suffixe n'est pas obligatoire.
exemples:
ECHECS.JEU

Euclide.tst
ESSAI

En l'absence de suffixe, l'interpréteur Basic ajoute le suffixe suivant:
.BAS pour un programme Basic
.DAT pour un fichier de données
.BIN pour un fichier binaire (zone mémoire ou programme en langage machine)
.MAP pour une image de l'écran
.TRA pour une trace d'exécution de programme

Il existe un seul nom de fichier ayant un usage particulier:
AUTO.BAT
Ce nom est réservé à un programme Basic qui s'exécute automatiquement au chargement du Basic (choix 1 du menu initial).

Autres périphériques

Le son

La génération de sons est programmable en Basic sur une plage de cinq octaves.
Les sons transmis au haut-parleur du téléviseur sont les notes de la gamme (avec dièses et bémols) et le silence.
Pour chaque note émise, il est possible de régler: la durée (de la ronde à la double croche pointée), le tempo (sur une plage de 256 valeurs) et l'attaque qui rend le son plus ou moins amorti.
Les sons sont définis dans des chaînes de caractères. L'ordre d'émission sur le haut-parleur est donné par l'instruction PLAY.

Le crayon optique

Le crayon optique comprend un capteur qui analyse le rayonnement émis par le balayage du tube et un interrupteur qui sert à figer l'instant de la mesure.
La mesure du temps qui s'écoule entre le début du balayage et la perception du rayonnement permet de déterminer l'endroit visé avec une très bonne précision, égale au point élémentaire de l'image.

Le relevé des coordonnées du point visé par le crayon optique se fait par deux instructions:
INPEN relevé à distance
INPUTPEN relevé lors du contact de l'interrupteur
La position de l'interrupteur peut être connue par la fonction PTRIG.
Le contrôle d'une séquence d'instructions par la visée d'une zone au crayon optique est possible en définissant les zones par l'instruction PEN et en faisant les branchements correspondants par ONPEN...GOTO ou ONPEN...GOSUB.
Le noir et le rouge ne permettent pas de viser correctement un point.
En règle générale, la luminosité doit atteindre une valeur minimale pour que le crayon optique puisse fonctionner.
Pour obtenir une mesure précise, il est nécessaire de régler le crayon optique à la mise en route.

Les manettes de jeu

On peut connecter deux manettes de jeu sur le contrôleur de jeu.
La position de chacune de ces deux manettes est donnée par deux fonctions:
STICK position de la manette dans l'une des 8 directions
STRIG état du bouton de la souris "action" de la manette

La souris

Elle se connecte sur la prise gauche des manettes de jeu.
MTRIG état des boutons de la souris

10. Visualisation: mode caractère et mode graphique

La visualisation des données sur l'écran peut se faire sous deux formes différentes:
— *mode caractère:* pour afficher du texte ou des symboles prédéfinis de la taille d'un caractère,
— *mode graphique:* pour afficher des traits, des points, des boîtes ou des objets quelconques.

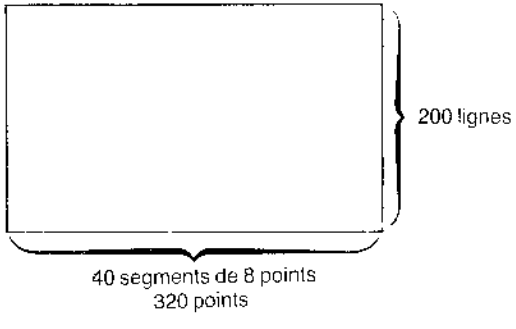
Les deux modes fonctionnent simultanément, ce qui permet d'afficher sur le même écran, du texte et du graphique.

Fonctionnement de la visualisation

Suivant la précision et le nombre de couleurs désirées, il est possible de visualiser caractères et graphiques suivant trois modes distincts:
1- mode standard 40 colonnes
2- mode 80 colonnes
3- mode bitmap 4 ou 16 couleurs
Le mode standard 40 colonnes est celui de la mise en route. Le choix d'un mode particulier se fait par l'instruction CONSOLE (cinquième paramètre).

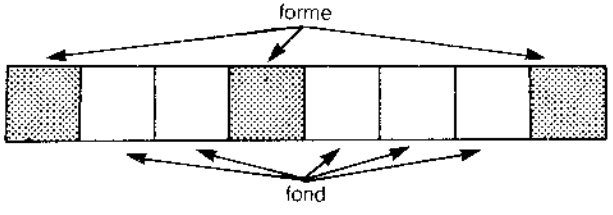
Mode standard 40 colonnes

La partie utile de l'écran est composée de 200 lignes distinctes. Chaque ligne est divisée en 40 segments de 8 points chacun.



A l'intérieur d'un segment, chaque point est rattaché à l'une des deux catégories:
— forme: il s'agit d'un point appartenant à un caractère, à une ligne, à une boîte ou à un objet.
— fond: c'est un point du fond de l'écran, qui n'a été modifié par aucun tracé ou aucun affichage ou qui a été remis au "fond".

exemple d'un segment avec 3 points de forme



Pour un même segment, on ne peut attribuer que deux couleurs:
— une couleur pour les points de la forme,
— une couleur pour les points du fond.

Les deux couleurs d'un segment (forme et fond) sont celles qui ont été fixées par la dernière programmation d'un point de forme ou de fond du segment.

Si aucun point du segment n'a été modifié par une instruction d'affichage ou de tracé, le segment entier appartient au fond de l'écran.

L'appartenance d'un point à la forme ou au fond lors d'un tracé est fixée par le code employé pour la couleur:
— code positif: le point appartient à la forme,
— code négatif: le point appartient au fond.

Si la programmation d'un point de l'écran est faite avec une instruction dans laquelle la couleur n'est pas précisée, le point est traité comme un point de la forme et la couleur prise par défaut est celle qui a été choisie précédemment pour les caractères (instructions SCREEN et COLOR).

D'un segment de huit points à l'autre, le choix des deux couleurs est entièrement libre. Les segments sont indépendants.

Il est possible d'afficher et de tracer sans modifier la couleur de forme des segments atteints par le tracé (troisième paramètre de l'instruction CONSOLE).

L'inversion des couleurs de forme et de fond simultanément sur tout l'écran est possible (quatrième paramètre de l'instruction SCREEN). L'inversion de la suite de l'affichage est possible également (3e paramètre de l'instruction COLOR).

Mode 80 colonnes

Dans ce mode, la précision horizontale de l'écran est doublée: chaque ligne est divisée en 80 segments de 8 points chacun. Deux couleurs seulement sont disponibles; le tracé se fait donc sans tenir compte de la couleur.

Modes bitmap 4 et 16 couleurs

Dans ces modes, la définition de l'écran est de 200 lignes de 320 points en 4 couleurs et de 200 lignes de 160 points en 16 couleurs.

La notion de segment n'existe plus, chaque point est complètement indépendant de ses voisins et peut donc prendre n'importe quelle couleur.

Choix des couleurs

Suivant le mode de fonctionnement, le nombre de couleurs utilisables simultanément est de seize (mode 40 colonnes) de deux, quatre ou seize (modes bitmap).

Ces couleurs peuvent être choisies parmi une étendue de 4 096 couleurs réelles différentes.

L'attribution d'une couleur réelle à un code de couleur se fait par l'instruction PALETTE.

A la mise en route, le code des couleurs est attribué aux couleurs réelles suivantes:

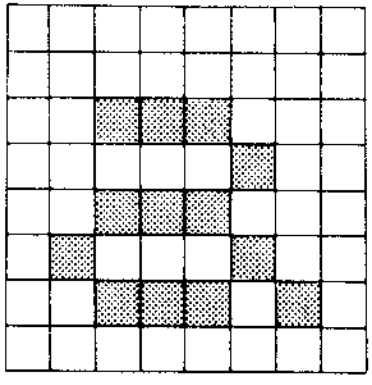
couleur	code forme	code fond
noir	0	-1
rouge	1	2
vert	2	-3
jaune	3	-4
bleu	4	5
magenta	5	-6
cyan	6	-7
blanc	7	-8
gris	8	-9
rouge clair	9	-10
vert clair	10	-11
jaune clair	11	-12
bleu clair	12	-13
magenta clair	13	-14
cyan clair	14	-15
orange	15	-16

Les couleurs claires (de 8 à 14) sont les mêmes que les sept premières mais avec un peu de blanc.

Le code de fond d'une couleur s'obtient facilement par la relation:
code fond = -(code forme + 1)

Mode caractère

Dans les instructions fonctionnant sur le mode caractère, l'écran comporte 25 lignes de 40 (ou 80)caractères chacune. Chaque caractère lui-même est représenté sur l'écran par une matrice de 8*8 points, soit encore huit segments.



matrice du caractère a

En mode caractère, les lignes sont comptées de 0 (ligne supérieure) à 24 (ligne inférieure de l'écran) et les colonnes sont comptées de 0 (la plus à gauche) à 39 ou 79(la plus à droite).

L'instruction LOCATE permet de placer une chaîne de caractères en n'importe quel point de l'écran.

La position du curseur qui marque l'endroit où doit se faire le prochain affichage, est donnée par deux fonctions: CSRLIN pour le numéro de la ligne et POS pour le numéro de la colonne.

Le code ASCII du caractère situé en une position quelconque de l'écran est donné par la fonction SCREEN (à ne pas confondre avec l'instruction du même nom).

La fenêtre d'affichage peut être limitée en hauteur entre une ligne supérieure et une ligne inférieure par l'instruction CONSOLE (premier et deuxième paramètres).

Les caractères peuvent être affichés en double hauteur et double largeur au moyen de l'instruction ATTRB.

Certaines instructions graphiques fonctionnent en mode caractère, c'est-à-dire qu'elles effectuent le même type de tracé mais en prenant un caractère à la place d'un point et avec les coordonnées du mode caractère (colonnes de 0 à 39 ou 79 et lignes de 0 à 24). : PSET, LINE, BOX, BOXF. Les caractères peuvent être utilisés pour remplir une zone en mode graphique. Le caractère est alors pris comme motif de remplissage (instruction PATTERN).

Caractères définis par l'utilisateur

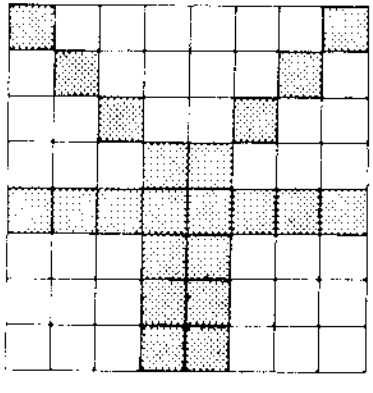
Il est possible de définir 128 caractères de forme quelconque et qui pourront ensuite être utilisés comme des caractères normaux.

Ces caractères sont définis par l'utilisateur au moyen de l'instruction DEFGR$. Le caractère numéro i est ensuite désigné par GR$(i) ou CHR$(128+i). Les numéros de caractères utilisateur s'étendent de 0 à 127.
Ils sont affichés par l'instruction PRINT ou PRINT USING. Ils peuvent être doublés en hauteur et en largeur par ATTRB.

Le nombre maximum de caractères à définir est fixé, le plus souvent en début de programme, dans une instruction CLEAR (troisième paramètre).
Si le nombre de caractères réservés est n, le numéro des caractères s'étend de 0 à n−1.

La définition d'un caractère est faite par huit nombres entiers compris entre 0 et 255. Chaque nombre entier représente l'un des huit segments de huit points qui composent le caractère.

Pour chaque segment, la valeur associée est calculée en binaire en prenant 1 si le point est allumé c'est-à-dire appartient au dessin du caractère, et en prenant 0 si le point appartient au fond.



matrice de points du caractère numéro i

binaire	décimal
1000 0001	129
0100 0010	66
0010 0100	36
0001 1000	24
1111 1111	255
0001 1000	24
0001 1000	24
0001 1000	24

valeur des segments du caratère i

Chaque caractère peut être redéfini en cours de programme.

exemple :

```
10 ' Définition d'un caractère
20 CLEAR ,,1
30 DEFGR$(0) = 129,66,36,24,255,24,24,24
40 PRINT CR$(0)
50 PRINT
60 ATTRB 1,1 : PRINT CHR$(128)
```

Mode graphique

En mode graphique, l'écran est composé de 200 lignes de 320 ou 640 points chacune.

Les coordonnées d'un point graphique visible sont comprises :
— pour les colonnes entre 0 et 319 ou 639
— pour les lignes entre 0 et 199

Les coordonées d'un point sont toujours indiquées avec la syntaxe suivante:
(colonne , ligne)
Le point (0,0) est situé en haut à gauche et le point (319,199) ou (639,199) en bas à droite.

Toutes les instructions graphiques acceptent des coordonnées en dehors de la zone visible. Les valeurs des coordonnées d'un point en colonne et en ligne peuvent varier de −32768 à 32767.

La partie visible des tracés graphiques peut être réduite à une taille inférieure à celle de l'écran ; la fenêtre visible est définie par l'instruction WINDOW.

Si la valeur d'une coordonnée n'est pas entière, elle est arrondie à l'entier le plus proche.

Les instructions de tracé sont les suivantes

PSET	point
LINE	segment
BOX	boîte
BOXF	boîte pleine
CIRCLE	cercle ou ellipse
CIRCLEF	cercle ou ellipse pleins

La couleur du tracé est la dernière fixée par SCREEN ou COLOR ou bien elle peut être précisée dans l'instruction.

Si le code de couleur est négatif, les points tracés appartiennent alors au fond.

Le tracé peut être effectué de quatre manières différentes:

— avec effacement: le tracé efface ce qui se trouvait précédemment au même endroit,

— transparent: le tracé se superpose au tracé précédent, les points de forme du tracé précédent ne sont pas effacés par les points de fond du nouveau tracé,

— avec inversion: chaque point du tracé est inversé : s'il appartenait au fond, il passe à la forme et vice versa. Deux tracés identiques au même endroit rétablissent l'état initial.

— mode ET: le tracé n'apparaît que sur les points en couleur de forme.

De plus, le tracé peut être fait avec ou sans modification de la couleur des points.

Toutes ces modalités sont fixées par le troisième paramètre de CONSOLE.

La fonction POINT permet de connaître la couleur d'un point et son état: le code retourné est positif si le point appartient à la forme et négatif s'il appartient au fond.

Le remplissage d'une surface de couleur homogène avec une couleur quelconque est possible avec l'instruction PAINT. Par défaut, le remplissage est uniforme. Le motif de remplissage (caractère ASCII ou caractère défini par l'utilisateur) est fixé par l'instruction PATTERN.

Plusieurs zones de l'écran peuvent être mémorisées dans un tableau numérique entier puis sauvegardées dans un fichier pour être utilisées ultérieurement (instructions GET, SAVEP, LOADP et PUT). Les coordonnées de chaque zone à sauvegarder doivent correspondre au découpage de l'écran en caractères (colonnes et lignes comptées en caractères).

11. Tortues

On peut définir jusqu'à dix tortues (numérotées de 0 à 9) dont le comportement est semblable à celui de la tortue Logo.

Chaque tortue est caractérisée par:

— sa forme,

— sa taille,

— sa position,

— sa direction de mouvement,

— son orientation,

— sa trace (ou non),

— sa présence (visible ou invisible).

Généralités

Chaque tortue peut être déplacée en translation dans sa direction, en rotation autour de son centre; sa direction de mouvement peut être modifiée.

La forme d'une tortue est définie par des segments visibles ou invisibles avec l'instruction TURTLE.

La taille d'une tortue est modifiée par l'instruction ZOOM, soit en absolu, soit en relatif.

La position d'une tortue, en dehors de tout déplacement, est fixée en coordonnées absolues par l'instruction TURTLE.

Le déplacement d'une tortue dans sa direction est commandé par l'instruction FWD, en avant comme en arrière.

La direction du déplacement est fixée ou modifiée de façon absolue ou relative par l'instruction HEAD.

La tortue peut être placée et déplacée dans un écran virtuel dont les coordonnées varient de −32768 à 32767.

Cet écran s'enroule sur lui-même, le dépassement des bornes d'un côté ramenant la tortue du côté opposé.

L'orientation de la tortue elle-même est fixée ou modifiée par l'instruction ROT, de façon absolue ou relative.

L'instruction TRACE indique si la tortue laisse une trace de ses déplacements.

L'instruction SHOW indique d'une part si la tortue est visible ou invisible et d'autre part si les modifications de taille et d'orientation sont à effet immédiat ou différées au prochain déplacement (FWD) ou placement (TURTLE).

Les valeurs des angles dans les instructions HEAD et ROT sont données en 1/256e de cercle. La valeur 0 correspond à 0°, 128 à 180° et 255 à (360*255)/256 degrés.

Par défaut, les caractéristiques d'une tortue sont les suivantes:

forme	triangle isocèle
taille	échelle normale (16)
position	au centre de l'écran
direction	vers la droite de l'écran (0°)
orientation	identique à la direction (0)
trace	crayon levé
visibilité	invisible
évolution	modifications différées

L'état d'une tortue donnée peut être entièrement connu au moyen d'une instruction:

INPUTTURTLE	position

et de cinq fonctions:

HEAD	direction
ROT	orientation
ZOOM	taille
SHOW	visibilité
TRACE	trace

Les instructions et fonctions s'appliquent à la tortue active: celle-ci est déterminée par la dernière instruction TURTLE.

12. Gestion des erreurs

Les erreurs sont détectées par l'interpréteur Basic au moment de l'exécution des instructions. La détection d'une erreur provoque l'arrêt de l'exécution.
Chaque erreur donne lieu à un court message à l'écran.
exemples:

```
Syntax Error
Division By Zero
Missing Operand
```

Si l'erreur apparaît dans un programme, le numéro de la ligne en cours d'exécution est indiqué.
exemple:

```
Type Mismatch in 150
```

L'affichage de la ligne où l'erreur s'est produite peut être obtenu par:

```
LIST
```

Un pavé en vidéo inverse est inséré automatiquement après l endroit de la ligne où l'erreur a été détectée.

Pour éviter que certaines erreurs prévisibles ne viennent perturber le fonctionnement d'un programme et pour éventuellement y remédier, une instruction ON ERROR ... GOTO permet de déclarer le début d'une séquence de traitement d'erreur.
Dans la séquence de traitement d'erreur, le numéro de l'erreur est contenu dans la variable ERR et le numéro de la ligne où elle s'est produite dans la variable ERL.

La liste des messages d'erreur et de leurs numéros est donnée en Annexe 7.
Après traitement de l'erreur, l'instruction RESUME permet de reprendre l'exécution de la suite du programme soit à l'endroit où l'erreur s'est produite, soit à un endroit différent.

exemple:

```
10 ' DIVISION PAR 0
20 ON ERROR GOTO 100
30 DO
40   INPUT X
50   PRINT 1/X
60 LOOP
90 END
100 IF ERR=11 THEN PRINT "Donner un nombre non nul"
110 RESUME 40

RUN
? 4
 .25
? 0
Donner un nombre non nul
? ...
```

Le signe * indique qu'il s'agit d'une instruction spécifique au Basic 128.
Le signe ♦ indique qu'il s'agit d'une instruction spécifique au Basic 1.

Avant d'entrer dans le détail de chaque commande, instruction ou fonction, rappelons quelques conventions de notations.

— Les éléments facultatifs d'une instruction sont imprimés sur le fond coloré.
— Les principaux paramètres qui changent d'une utilisation à l'autre sont désignés par:

Nombre: représente une constante numérique (3*55) ou une variable numérique (Z) ou une expression dont le résultat est numérique (Z*RND)

Chaîne de caractères: représente une constante ("JULES") ou une variable (P$) ou une expression dont le résultat est une chaîne de caractères (MID$("JULES", 1,1)).

Donnée: représente un nombre ou une chaîne de caractères.

Variable: représente une variable numérique (X) ou une variable chaîne de caractères (Y$).

— Les différents exemples fournis avec chaque instruction illustrent la partie optionnelle de l'instruction. En général, les options sont précisées dans l'explication.

ABS(nombre)
fonction

Donne la valeur absolue du nombre.

Exemple:

```
ABS(-7)
 7
```

ASC(chaîne de caractères)
fonction

Donne le code ASCII du premier caractère de la chaîne.

Si la chaîne de caractères est vide, le message d'erreur Illegal Function Call apparaît.

Remarque: les minuscules accentuées et le ç sont codés par une séquence de trois caractères dont le premier a toujours le code 22. Ainsi, le caractère é est codé par la séquence de codes 22, 66, 101. On obtient chacun de ces codes en utilisant la fonction MID$.

Exemple:

```
PRINT ASC("CELEBRITE")
67
PRINT ASC(MID$("é",2,1))
66
```

ATN(nombre) *
fonction

Donne l'arctangente du nombre.

Le résultat de la fonction est un angle, exprimé en radians, compris entre $-\pi/2$ et $\pi/2$.
Si le nombre est en double précision, le résultat est donné en double précision.

Exemple:

```
[illegible]
```

ATTRB largeur ,hauteur

instruction

Définit la taille des caractères affichés sur l'écran.

Le 1er paramètre indique la largeur des caractères:
largeur = 0, largeur normale
largeur = 1, largeur double

Le 2e paramètre indique la hauteur des caractères:
hauteur = 0, hauteur normale
hauteur = 1, hauteur double

Aucun des paramètres n'est obligatoire. En l'absence de l'un des paramètres, l'attribut correspondant n'est pas modifié.

En double hauteur, l'instruction PRINT affiche sur la ligne courante et sur la ligne précédente; le passage à la ligne provoque un double interligne. En double hauteur, il est recommandé de sauter une ligne vide par un PRINT avant le premier affichage.

L'exécution de INPUT ou LINE INPUT, après l'affichage éventuel du texte associé, provoque le retour à la taille normale. Les fonctions INKEY$ et INPUTS ne modifient pas le mode d'affichage.

Exemple:

```
100 PRINT "DEBUT DU TEXTE"
110 PRINT
120 ATTRB 1,1
130 PRINT "Bonjour"
```

AUTO numéro ,pas

commande

Propose automatiquement les numéros de ligne pour l'écriture de programmes.

Le premier nombre indique le numéro de la première ligne à écrire (10 par défaut).
Le deuxième nombre indique l'écart entre les lignes (10 par défaut).

Les deux paramètres sont facultatifs.

Si la ligne proposée existe déjà, le message "Line not empty" s'affiche avant le numéro de ligne.

Le numéro de ligne proposé peut être modifié , le curseur peut être déplacé sur une autre ligne qui peut être modifiée puis validée: la numérotation automatique se cale alors sur le nouveau numéro. On sort de la numérotation automatique par CNT-C ou en validant une ligne sans numéro.

Exemple:

```
Ok
AUTO
```

BACKUP n° de lecteur 1 TO n° de lecteur 2

commande

Recopie entièrement le contenu d'une disquette sur une autre disquette.

Cette commande recopie intégralement la disquette qui se trouve dans le premier lecteur sur la disquette qui se trouve dans le deuxième lecteur.

Avec un seul lecteur BACKUP n' recopie d'une disquette sur l'autre, en plaçant en alternance la disquette "source" et la disquette "destination" dans le lecteur.

En Basic 1, BACKUP détruit toujours le programme et les variables.

Exemple:

```
BACKUP 0 TO 1        recopie la disquette 0 sur la disquette 1
BACKUP 0             lecteur n° 0 seul.
BACKUP 1             lecteur n°1 seul.
```

BANK *

fonction

Donne le numéro de la banque de mémoire courante.

Ce numéro est un nombre compris entre 1 et 6.
A l'initialisation, c'est la banque de rang le plus élevé qui est sélectionnée.

Exemple:

```
?BANK
2
```

BANK numéro *

instruction

Sélectionne la banque de mémoire dont on précise le numéro.

Ce numéro doit être compris entre 0 et le nombre maximum de banques de mémoire disponibles (6).
Si le numéro est égal à 0, c'est la dernière banque de mémoire existante qui est sélectionnée, comme lors de l'initialisation.
Cette instruction ne concerne que les banques de mémoire situées entre les adresses $H6000 et $H9FFF. Elle a pour effet d'affecter toutes les instructions comportant des références directes à la mémoire: DEF USR, EXEC, CLEAR, LOADM, SAVEM, PEEK, POKE....

Exemple:

```
BANK 1
```

BEEP

instruction

Produit un son bref.

Cette instruction est équivalente à PRINT CHR$(7):.

Exemple:

```
FOR I=1 TO 100: BEEP: NEXT
```

BOX (c1,l1)–(c2,l2) caractère,couleur,fond ,inversion
BOX (c1,l1)–(c2,l2) ,couleur

instruction

Trace un rectangle dont les côtés sont parallèles aux bords de l'écran.

Cette instruction fonctionne en mode caractère quand l'argument caractère est présent. Dans ce cas, les coordonnées en colonne (c1,c2) sont comptées de 0 à 39 (ou 79), et en ligne (l1,l2) de 0 à 24.
Elle fonctionne en mode graphique quand l'argument caractère est absent. Dans ce cas, les coordonnées en colonne et en ligne sont comprises entre –32768 et 32767 en Basic 128. En Basic 1 les coordonnées en colonnes sont comprises entre 0 et 319 et les coordonnées en lignes entre 0 et 199.
Le rectangle tracé a pour sommets opposés les points situés en (c1,l1) et (c2,l2).

Si le premier point (c1,l1) est omis, il est remplacé par le dernier point du dernier tracé ,c'est-à-dire:
— le 2e sommet dans BOX, BOXF ou LINE,
— le point tracé par PSET,
— le point (0,0) si aucun tracé n'a eu lieu.
L'argument couleur est un nombre.
En mode caractère, l'argument caractère est une chaîne de caractères (variable ou constante) dont le premier caractère est utilisé pour le tracé. Les trois paramètres couleur, fond, inversion sont les mêmes que pour l'instruction COLOR et leur effet est identique.
En mode graphique, le tracé n'a lieu qu'à l'intérieur de la fenêtre définie par WINDOW (Basic 128 seulement).
En mode caractère et graphique, le type de tracé est fixé par le 3e paramètre de l'instruction CONSOLE.
Les coordonnées du premier point et les arguments couleur, fond et inversion sont facultatifs.

Exemple:

```
BOX 19,-12,15,,,3,0      mode caractère
BOX 104,81,,216,48,,2    mode graphique
BOX -1000,70
```

Voir STEP.

BOXF (c1,l1)–(c2,l2) caractère,couleur,fond ,inversion
BOXF (c1,l1)–(c2,l2) ,couleur
instruction

Trace un rectangle dont les côtés sont parallèles aux bords de l'écran et le remplit du motif indiqué.

En mode caractere,le remplissage se fait avec le caractère indiqué.
En mode graphique, le motif de remplissage est celui fixé par l'instruction PATTERN, sinon le remplissage est uniforme.

La syntaxe de l'instruction BOXF est identique à celle de BOX, en mode caractère et en mode graphique.

Exemple:

```
BOXF 2,2,-14,14,,-14,0    mode caractère
BOXF 104,81,,216,48,,2    mode graphique
```

Voir STEP.

CDBL(nombre)
fonction

Convertit le nombre en double précision.

Le nombre converti en double précision n'est pas complété uniquement par des 0, ce qui peut conduire à de petites variations.
<si l'argument n'est pas un nombre, le message d'erreur "Illegal Function Call" apparaît.

Exemple:

```
?CDBL(454.67)
 454.6700134277344
```

CHAIN descripteur de fichier,l1-l2,l3 *
instruction

Supprime une partie du programme résident en mémoire, fusionne le programme indiqué et poursuit l'exécution à la ligne indiquée.

Cette instruction permet de fusionner une partie de programme et de l'exécuter sans perdre le contenu de toutes les variables.
La portion de programme comprise entre les lignes l1 et l2 est supprimée (même syntaxe que DELETE). Cette option de suppression n'est pas obligatoire.
Puis le programme indiqué est fusionné au programme résident.
Lors de la fusion, les lignes du nouveau programme remplacent les

lignes du programme résident, le cas échéant. Le programme à fusionner doit avoir été sauvegardé en binaire (et non pas en ASCII comme dans MERGE).
Si l'option ,l3 est présente, l'exécution se poursuit à la ligne l3, sinon l'exécution reprend à partir de la première ligne du programme.
Seules les variables présentes dans la liste de l'instruction COMMON sont conservées.

Exemple:

CHAIN "SUITE",500-900,100 supprime les lignes 500 à 900, fusionne le programme SUITE de la disquette 1 et poursuit l'exécution en ligne 100.
CHAIN "PROG345",500- supprime toute la fin à partir de la ligne 500, l'exécution reprend au début.
CHAIN "PASS2" aucune ligne n'est supprimée, l'exécution reprend au début après la fusion.

CHR$(nombre)
fonction

Rend le caractère de code ASCII correspondant au nombre.

Le nombre doit être compris entre 0 et 255.
Cette fonction permet de construire des chaînes de caractères contenant des codes de contrôle. Ainsi la sonnette (code ASCII 7) peut-elle être mise dans une chaîne par CHR$(7).
Les caractères dont le code est supérieur à 127 sont des caractères définis par l'utilisateur au moyen de l'instruction DEFGR$.
CHR$(128+nombre) est équivalent à GR$(nombre).
Les minuscules accentuées peuvent être introduites par CHR$.
La fonction inverse de CHR$ est ASC.

Exemple:

PRINT CHR$(65) représente A
PRINT CHR$(12) efface l'écran
PRINT CHR$(224)+CHR$(65)+CHR$(130) représente à

CINT(nombre)
fonction

Convertit un nombre réel en un nombre entier.

La conversion se fait par arrondi. Le nombre doit être compris entre −32768 et 32767 sinon l'erreur "Overflow" est déclenchée.

Exemple:

PRINT CINT(45.89)-CINT(45.1)
45

CIRCLE (colonne, ligne) rh, rv, couleur; début, fin
instruction

Trace une portion de cercle ou d'ellipse dont le centre est situé au point (colonne, ligne).

Si rh est présent, cette instruction trace une ellipse de rayon horizontal rh et de rayon vertical rv. Sinon elle trace un cercle de rayon rv.
Si le paramètre couleur n'est pas présent, le tracé se fait dans la couleur courante des caractères.
Si les deux paramètres début et fin sont présents, le tracé ne portera que sur la portion de cercle ou d'ellipse correspondante. Les angles sont comptés en radians; le sens positif étant le sens des aiguilles d'une montre.
Le tracé se fait dans le mode défini par le 3e paramètre de CONSOLE.
En Basic 1, le tracé d'ellipse ou de portions de cercle ou d'ellipse est impossible.

Exemple:

CIRCLE (100,100),80,40,,3.14,6.28 demi-ellipse inférieure centrée en (100,100) de rayon horizontal 80, de rayon vertical 40, de couleur magenta

CIRCLE 1,100,100,90,2 cercle centré au milieu de l'écran, de rayon 90, de couleur verte

CIRCLEF (colonne, ligne) rh, rv, couleur, debut, fin
instruction *

Trace une portion de cercle ou d'ellipse dont le centre est situé au point (colonne, ligne) et la remplit du motif courant.

Le motif de remplissage est celui fixé par PATTERN. Par défaut, le remplissage est uniforme. Si seule une portion de cercle ou d'ellipse est tracée, seul le secteur correspondant est rempli.
La syntaxe de CIRCLEF est identique à celle de CIRCLE.

Exemple:

PATTERN "*" CIRCLEF 100,100,80,40,5,0,3.14/6 demi-ellipse inférieure centrée en (100,100) de rayon horizontal 80, de rayon vertical 40, de couleur magenta et remplie de caractères "*"

CLEAR volume,adresse1,nombre,adresse2
instruction

Fixe plusieurs paramètres de l'espace mémoire utilisateur et remet à zéro toutes les variables.

Cette instruction supprime toutes les variables et tous les tableaux existants et annule l'effet de l'instruction ON ERROR GOTO GOSUB, ON KEY..., ON INTERVAL.
Le 1er paramètre (volume) est un nombre qui fixe le volume réservé aux chaînes de caractères. Ce volume est fixé par défaut à 300 caractères.

Exemple:

CLEAR 2000 réserve un espace de 2000 caractères

Le 2e paramètre (adresse1) permet de réserver une zone mémoire dans l'espace correspondant aux banques de mémoires commutables. Cette réservation affecte la banque sélectionnée par l'instruction BANK et les banques de rang supérieur.
L'adresse précisée est un nombre qui est inférieur à &HDFFF.
En Basic 1, cette réservation n'affecte qu'une seule banque.

Exemple:

BANK 4: CLEAR ,&HBFFF réserve l'espace mémoire &HC000 à &HDFFF de la banque 4 et la totalité des banques 5 et 6
BANK 4: CLEAR ,&H8FFF réserve l'espace mémoire &H9000 à &H9FFF de la zone non commutable et la totalité des banques 4, 5 et 6

Le 3e paramètre (nombre) fixe le nombre maximum de caractères définis par l'utilisateur. Ce nombre n'est pas modifié jusqu'au prochain CLEAR.

Exemple:

CLEAR ,,10 fixe à 10 le nombre des caractères utilisateur

Le 4e paramètre (adresse2) permet de réserver une zone mémoire dans l'espace non commutable, c'est-à-dire hors banques.
L'adresse précisée est un nombre inférieur à &HA000. La réservation ne porte que sur la zone mémoire comprise entre adresse2 + 1 et &H9FFF et n'affecte pas les banques de mémoire.
Ce 4e paramètre n'existe pas en Basic 1.

Exemple:

CLEAR ,,,&H8FFF réserve l'espace mémoire de &H9000 à &H9FFF

Aucun des paramètres de CLEAR n'est obligatoire.
Les réservations effectuées par une instruction CLEAR ne sont modifiables que par un autre CLEAR.
CLEAR seul supprime toutes les variables mais n'affecte pas les réservations.

CLOSE # n° de canal , # n° de canal ...

instruction

Ferme le (ou les) fichier(s) dont on indique le numéro de canal.

Cette opération marque la fin du fichier et inscrit dans le catalogue toutes les informations nécessaires. Si aucun numéro de canal n'est précisé, CLOSE ferme tous les fichiers, sauf si une erreur se produit à l'exécution de la fermeture.
Ne pas oublier qu'un arrêt en cours d'écriture ou de lecture par CNT-C ne ferme pas les fichiers. Tous les fichiers doivent avoir été fermés avant de retirer la disquette du lecteur.

Exemple:

```
CLOSE #1, #3      ferme les fichiers 1 et 3
CLOSE             ferme tous les fichiers ouverts
```

CLS

instruction

Efface la fenêtre de visualisation définie par CONSOLE et place le curseur en haut à gauche de cette fenêtre.

L'effacement se fait en ramenant tous les points à la couleur du fond. Le même effet peut être obtenu par PRINT CHR$(12).

COLOR forme,fond,inversion

instruction

Fixe la couleur de forme et la couleur de fond pour les affichages et les tracés qui suivent

Les deux premiers paramètres sont des nombres. Le 1er fixe la couleur de forme, le second la couleur de fond.
Le 3^{e} paramètre est un nombre qui vaut 0 ou 1. Quand il est présent, il provoque une inversion des couleurs de forme et de fond.
Aucun des trois paramètres n'est obligatoire.

Exemple:

```
COLOR 2,0     caractères et tracés verts sur fond noir
COLOR ,7      sur fond blanc
COLOR ,,1     inverse les couleurs précédentes
```

COMMON liste de variables ★

instruction

Permet de transmettre des variables d'un programme à l'autre lors de l'enchaînement de ces programmes par CHAIN.

Les variables de la liste qui suit COMMON sont protégées lors de l'exécution de CHAIN. Le programme fusionné pourra donc les utiliser.
La liste de variables peut contenir des tableaux avec leur dimension; les tableaux correspondants sont alors déclarés, comme dans DIM.
Les variables de la liste sont détruites par CLEAR, RUN, NEW, LOAD, etc.
A l'exécution de COMMON, toutes les variables existantes sont détruites, à l'exception des variables déjà définies par un COMMON.
Il est donc préférable de placer les instructions COMMON en début de programme, après un CLEAR éventuel.

Exemple:

```
COMMON A,B$,C(100)      protège A,B$ et déclare et protège le tableau C
C=3
COMMON C,D              protège C et D
?C;D
0 0                     le contenu de C est perdu
```

CONSOLE ligne haute, ligne basse, trace, défilement, mode

instruction

Fixe plusieurs paramètres du tracé et de l'affichage.

Tous les paramètres sont optionnels, mais cette instruction ne doit pas se terminer par une virgule et doit comporter au moins un paramètre.
Les deux premiers paramètres fixent la fenêtre d'affichage en mode caractère :
— le 1er indique la ligne supérieure de la fenêtre,
— le 2e indique la ligne inférieure.
Ces deux paramètres sont des nombres compris entre 0 et 24.
Le 3e paramètre précise le mode de tracé graphique :
— s'il est pair, la couleur est modifiée en même temps que la forme,
— s'il est impair, la couleur n'est pas modifiée,
— s'il vaut 0 ou 1, le tracé efface ce qui se trouvait au même endroit sur l'écran (les points du fond du motif sont effacés),
— s'il vaut 2 ou 3, le tracé n'efface pas ce qui se trouvait au même endroit. Un motif agit sur un autre en surimpression.
— s'il vaut 4 ou 5, le tracé inverse les points de la forme existante (OU exclusif). Deux tracés successifs identiques rétablissent l'état initial :
— s'il vaut 6 ou 7, le tracé n'apparaît que sur les points en couleur de forme (mode ET).
Les valeurs 2, 3, 4 et 5 du 3e paramètre sont interdites en Basic 1.
Le 4e paramètre fixe le mode de défilement :
— s'il vaut 0, le défilement a lieu à vitesse normale,
— s'il vaut 1, le défilement a lieu à vitesse lente,
— s'il vaut 2, l'affichage se fait en mode page : quand la fenêtre est pleine, la nouvelle ligne s'affiche en haut de la fenêtre.
Le 5e paramètre définit le mode graphique (n'existe pas en Basic 1) :
— s'il est à 0, c'est le mode 40 colonnes seize couleurs ;
— s'il est à 1, l'écran fonctionne en 80 colonnes avec deux couleurs seulement, le tracé se fait donc sans tenir compte de la couleur :
— s'il est à 2, l'écran fonctionne en quatre couleurs sur tous les points. Dans ce mode, chaque point peut prendre l'une des quatre couleurs, indépendamment de la couleur de ses voisins.
— s'il est à 3, l'écran fonctionne en 16 couleurs sur tous les points. Dans ce mode, chaque point de l'écran en 160 × 200 peut prendre l'une des 16 couleurs, indépendamment de la couleur de ses voisins.
— s'il est à 4, c'est le mode commutation de pages avec visualisation de la page 1 utilisant la couleur 1 et écriture par PRINT en page 2 avec la couleur 2.
— s'il est à 5, c'est le mode commutation de pages avec visualisation de la page 2 utilisant la couleur 2 et écriture par PRINT en page 1 avec la couleur 1.
— s'il est à 6, c'est le mode superposition de deux pages avec écriture par PRINT en page 2 avec la couleur 2.
— s'il est à 7, c'est le mode superposition de deux pages avec écriture par PRINT en page 1 avec la couleur 1. La page 1 est prioritaire sur la page 2.
— s'il est à 8, 9, 10 11, c'est le mode triple superposition avec 8 = plan 1, 9 = plan 2, 10 = plan 3, 11 = plan 4. Chaque plan utilise la couleur correspondant à son numéro. Le plan 4 est prioritaire sur le plan 3, etc.
Dans les modes 4 à 7, la sélection de la page de tracé graphique s'effectue par l'ordre COLOR compris entre 0 et 3.
Dans les modes 4 à 11, la couleur 0 sert de couleur de fond. En mode 3, l'affichage des caractères n'est pas géré. En mode 8 à 11, l'affichage des caractères et du graphique n'est pas géré.

CONT

commande

Reprend l'exécution d'un programme après un arrêt.

L'arrêt du programme peut avoir été provoqué par le caractère CNT-C, ou par l'exécution de STOP ou END, ou par une erreur. L'exécution reprend à l'endroit où elle à été interrompue. Si CNT-C a été envoyé au cours de la réponse à une instruction INPUT, à la reprise le message associé apparaît à nouveau et la réponse doit être fournie une deuxième fois.
La commande CONT est utile pour la mise au point de programme en liaison avec l'instruction STOP. Quand l'exécution est suspendue, le contenu des variables peut être affiché et même modifié par une affectation. La commande CONT permet de reprendre l'exécution à l'endroit où elle s'était arrêtée.
La commande CONT ne peut plus fonctionner si le programme a été modifié (message d'erreur Can't Continue).

COPY descripteur de fichier 1 TO descripteur de fichier 2

commande

Permet la copie intégrale d'un fichier (en changeant éventuellement son nom) sur le même support ou sur un autre support.
Avec deux lecteurs de disquettes, COPY permet de copier un fichier de l'un sur l'autre.
Avec un seul lecteur de disquettes, COPY permet de copier un fichier:
— sur la même disquette, en changeant son nom,
— sur une autre disquette, sans changer son nom.
Dans ce cas, on place alternativement la disquette source et la disquette destination dans le lecteur. On n'indique alors dans la commande que le nom du fichier à recopier (COPY nom de fichier).
En Basic 512, le nom du fichier peut comporter un commentaire.
COPY ne permet pas de transfert d'une disquette vers un autre périphérique.

Exemple :

```
COPY "O:DESS.DAT" TO "[illegible]
COPY "MENU.BAS" TO "O:MENU.SAS
COPY "BUDGET.DAT"
```

COS(nombre)

fonction

Donne le cosinus de l'angle exprimé en radians.

En Basic 512, le résultat est en double précision si l'angle est en double précision.

Exemple:

```
?COS(0.2)
.980067
```

CRUNCH$(chaîne de caractères) *

fonction

Analyse la chaîne donnée en argument et rend une chaîne de code évaluable par la fonction EVAL.

Cette fonction permet de coder une chaîne de caractères qui respecte la syntaxe d'une expression du langage Basic. Cette expression, une fois codée par CRUNCH$, peut être évaluée par EVAL.
La chaîne à coder doit être une expression numérique ou une expression chaîne.

Exemple:

```
10 CLS
20 LINE INPUT "Expression ";A$
30 PRINT EVAL(CRUNCH$(A$))
40 LOCATE 0,4
RUN
Expression ? 3+4*5
 23
Expression ? RIGHT$("DEMAIN",4)
MAIN
```

CSNG(nombre)

fonction

Convertit le nombre en simple précision.

La conversion se fait après arrondi.
Les autres fonctions de conversion sont CINT et CDBL.

Exemple:

```
?CSNG(975.341817#)
 975.342
```

CSRLIN
fonction

Donne le numéro de la ligne où se trouve le curseur.

Le résultat est un nombre entier compris entre 0 (ligne du haut) et 24 (ligne du bas).
La fonction POS permet de connaître la position du curseur sur la ligne.

Exemple:

```
LOCATE 5,8 'CSRLIN affiche 8 en colonne 5, ligne 8
```

CVD (variable chaîne de caractères)
fonction

Assure la conversion du contenu de la variable (chaîne de caractères) en un nombre en double précision.
La variable doit contenir des données qui ont été enregistrées par une conversion à l'aide de MKD$( ).
Cette fonction est appliquée le plus souvent à une variable de champ définie dans un FIELD, après lecture d'un enregistrement par GET#.

Exemple:

```
A#=CVD(PRIX$)
```

CVI (variable chaîne de caractères)
fonction

Cette fonction analogue à CVD( ) assure la conversion du contenu de la variable en un nombre entier.
La variable doit contenir des données enregistrées par une conversion à l'aide de MKI$( ).

Exemple:

```
A%=CVI(C$)
```

CVS (variable chaîne de caractères)
fonction

Cette fonction analogue à CVD( ) assure la conversion du contenu de la variable en un nombre en simple précision.

La variable doit contenir des données enregistrées par une conversion par MKS$( ).

Exemple:

```
A!=CVS(TAUX$)
```

DATA liste de constantes
instruction

Cette instruction permet de déclarer une liste de constantes qui pourront être lues par l'instruction READ.

Les instructions DATA ne sont pas exécutables et peuvent se trouver n'importe où dans le programme. L'instruction DATA peut contenir autant de constantes que la longueur de la ligne le permet et il n'y a pas de limite au nombre des instructions DATA contenues dans un programme.
Les constantes de la ligne peuvent être de type entier, réel, double précision ou chaîne de caractères. Les expressions ne peuvent pas figurer dans cette liste.
Les chaînes de caractères doivent être encadrées par des guillemets si elles contiennent un séparateur (virgule, deux-points, ou espaces significatifs en début et en fin). Dans les autres cas, les guillemets peuvent être omis.
Les diverses constantes de la liste sont séparées les unes des autres par une virgule.
L'ensemble des constantes contenues dans les instructions DATA d'un programme peuvent être lues par des instructions READ de façon séquentielle. Le type des constantes lues doit correspondre au

type des variables des instructions READ qui reçoivent les valeurs ainsi lues.
Les données d'une même instruction DATA peuvent être lues plusieurs fois si l'instruction RESTORE fait remonter à la ligne de DATA correspondante.

Exemple :

```
100 READ A,B,C#,A$
110 PRINT A,B,C#,A$
...
200 DATA 1,2,3,4567890'2,"VOS BEAUX YEUX"

RUN
 1  2  3.456789  0'2 VOS BEAUX YEUX
```

DEF FN nom (variable,variable...) = expression
instruction

Définit une nouvelle fonction qui porte sur les variables indiquées entre parenthèses.

Le nom de la fonction suit exactement les mêmes règles qu'un nom de variable. La partie à droite du signe = est l'expression qui définit la fonction. Elle utilise les mêmes variables que celles indiquées dans la partie gauche. Les noms de variables peuvent être identiques à ceux déjà utilisés par ailleurs, sans qu'il y ait d'interaction.
La définition de la fonction doit se faire avant son utilisation. Les fonctions ainsi définies s'utilisent comme les fonctions ordinaires de Basic. On les appelle par leur nom : FN nom (variable, variable,...).
Les fonctions définies peuvent donner un résultat numérique ou un résultat chaîne de caractères.

Exemple :

```
DEF FN DIST(X,Y) = SQR(X^2 + Y^2)
A = FNDIST(3,5)
DEF FNCH$(I) = MID$(STR$(I),2)
A$ = FNCH$(25)
```

*DEFDBL lettre-lettre, lettre-lettre,... **
DEFINT lettre-lettre, lettre-lettre,...
DEFSNG lettre-lettre, lettre-lettre,...
DEFSTR lettre-lettre, lettre-lettre,...
instruction

Déclare le type d'un groupe de variables commençant par la même lettre.

Ces instructions de déclaration affectent toutes les variables qui ne comportent pas de type à la fin du nom, c'est-à-dire qui ne se terminent pas par %, !, # ou $ et dont le nom commence par une lettre qui se trouve dans l'une des plages indiquées.
DEFDBL déclare le type double précision
DEFINT déclare le type entier
DEFSNG déclare le type réel simple précision
DEFSTR déclare le type chaîne de caractères
En l'absence d'instruction de déclaration de type, ou de caractère marquant le type, toute variable est considérée comme numérique en simple précision.
Les instructions de déclaration de type doivent être exécutées avant l'utilisation des variables qu'elles concernent. Il est préférable de les placer en début de programme.

Exemple :

```
DEFDBL A,C,W-Z      toutes les variables dont le nom commence par
A,B,C,W,X,Y,Z sont de type double précision
DEFINT I,J          les variables dont le nom commence par I ou J sont
entières
```

DEFGR$(n°) = liste de huit nombres

instruction

Définit le caractère utilisateur de n° indiqué.

Le numéro du caractère est un nombre compris entre 0 et 127.
La définition de caractères utilisateur n'est possible que s'ils ont été réservés par une instruction CLEAR précédente (3e paramètre).
Le numéro du caractère ne peut être supérieur au nombre de caractères réservés dans CLEAR moins un.
La définition des caractères se fait par huit segments superposés de huit points chacun.
A chaque point d'un segment on fait correspondre le nombre binaire :
1 s'il est marqué,
0 s'il n'est pas marqué.
A chaque segment, on fait correspondre le nombre binaire ainsi obtenu. Sa valeur, en décimal, est donc comprise entre 0 et 255.
Dans la définition d'un caractère, on indique, à droite du signe =, la liste des huit nombres correspondant aux huit segments qui forment le caractère, en commençant par le segment supérieur. Ces huit nombres peuvent être donnés sous forme de constantes, de variables ou d'éléments de tableau. Le caractère ainsi défini est désigné par GR$(n°) ou par CHR$(128+n°).

Exemple :

```
10 CLEAR,,2
20 DEFGR$(0) = 128,64,32,16,24,36,66,129
30 DEFGR$(1) = &B11111111,&B10000001,&B10000001,&B10000001,&B10000001,&B10000001,
&B10000001,&B11111111
40 PR\TGR$(0);
50 PR\TGR$(1)
```

DEF USRn = adresse ★

instruction

Définit la fonction utilisateur en langage machine, de numéro n, commençant à l'adresse mémoire indiquée.

Il peut y avoir dix fonctions utilisateur simultanées, numérotées de 0 par défaut à 9.
L'adresse indique le début de la fonction en langage machine. Une zone mémoire destinée aux fonctions utilisateur peut être réservée par : CLEAR, adresse ou CLEAR,,adresse.
Le module en langage machine doit se terminer par l'instruction RTS du 6809 et ne doit pas modifier les contenus des registres S et DP.
Le passage des paramètres utilise, à l'appel comme au retour, les registres suivants :
Accumulateur A : 2 nombre entier
(type de la variable) 3 chaîne de caractères
4 simple précision
8 double précision
Index X : 2,X nombre entier
(pointeur sur la valeur) 0,X réel simple ou double précision
0,X descripteur de chaîne de caractères
Accumulateur B : longueur de la chaîne de caractères
Registre U : adresse du 1e octet de la chaîne

Les nombres entiers sont codés sur deux octets (notation en complément à deux), les réels simple précision sur quatre octets (1er octet : exposant, reste : mantisse), les réels double précision sur huit octets (idem).
Le descripteur de chaîne de caractères comprend trois octets : le premier contient la longueur de la chaîne, les deux suivants l'adresse du 1er caractère.

Exemple :

```
DEF USR2=&H9F80      définit la fonction numéro 2 à l'adresse &H9F80
W=USR2(V)            appel de la fonction
```

DELETE plage de n° de ligne, ligne

commande ou instruction

Supprime des lignes du programme et poursuit l'exécution à la ligne indiquée.

Les lignes du programme à supprimer sont indiquées de la manière suivante:

DELETE	ligne supprime la ligne
DELETE .	supprime la ligne courante
DELETE ligne1-ligne2	supprime toutes les lignes de la ligne1 à la ligne2 incluses
DELETE ligne1-	supprime toutes les lignes de la ligne1 à la fin
DELETE -ligne2	supprime toutes les lignes du début jusqu'à la ligne2
DELETE -	supprime tout le programme

Quand le 2e paramètre est présent, il indique la ligne à exécuter après la suppression.
Cette possibilité peut servir à éliminer une partie déjà exécutée pour libérer l'espace nécessaire aux variables qui suivent.
Le 2e paramètre est interdit en Basic 1.

Exemple:

DELETE 100-490,50	supprime les lignes 100 à 490 et poursuit l'exécution en ligne 50
DELETE 80	supprime la ligne 80
DELETE -	supprime tout le programme

DENSITY n° de lecteur,densité ♦

instruction

Fixe la densité d'enregistrement pour le lecteur désigné.
Le numéro de lecteur est un nombre compris entre 0 et 4.
La densité est indiquée par:
— 1 pour la simple densité.
— 2 pour la double densité.

Exemple:

DENSITY 0,2 met le lecteur 0 en double densité

Remarque :
A la mise en service, avec un lecteur double densité, le MO6 se commute automatiquement dans la densité de la disquette présente dans le lecteur. Pour changer de densité en Basic 128, il est nécessaire de réinitialiser avec une disquette dans la bonne densité.

DEVICE "nom de périphérique"

instruction

Détermine le nom du périphérique qui sera pris par défaut dans un descripteur de fichier qui ne le précise pas.

A l'initialisation, le périphérique implicite est le lecteur 0 (Basic 128) ou le lecteur de cassettes (Basic 1 sans DOS).
exemple:

DEVICE "1" le lecteur 1 devient le périphérique pris par défaut
DEVICE "CASS"

DIM liste de tableaux
avec tableau = nom de variable (liste d'indices)

instruction

Réserve la place nécessaire aux tableaux à une ou plusieurs dimensions.

Les indices sont des nombres arrondis à l'entier le plus proche qui fixent la valeur maximum de chaque indice du tableau. La valeur minimum que peut prendre un indice est zéro.
L'instruction DIM est exécutable et lors de son exécution, les éléments d'un tableau numérique sont initialisés à la valeur 0 et les éléments d'un tableau de chaînes sont initialisés à la valeur "chaîne vide".

Si, à l'exécution, un indice est en dehors des limites spécifiées dans DIM, l'erreur "Bad Subscript" est détectée.
Il est possible d'utiliser des tableaux sans les déclarer. Dans ce cas, la valeur numérique des indices ne doit pas dépasser 10.

Exemple:

DIM A(15,12) tableau à deux dimensions dont le premier indice peu varier de 0 à 15 et le second de 0 à 12
DIM C$(14) tableau de chaînes de caractères de quinze éléments

DIR "n° de lecteur: nom . suffixe"

instruction

Liste le répertoire de la disquette relatif aux fichiers précisés.

Si le n° de lecteur n'est pas précisé, on obtient le répertoire de la disquette du lecteur par défaut ("0:" sauf modification par DEVICE).
Si le nom n'est pas précisé, la liste porte sur tous les fichiers ayant le suffixe indiqué. Inversement, si le suffixe n'est pas précisé, la liste porte sur tous les fichiers de ce nom.
Si le nom est incomplet, la liste porte sur tous les fichiers commençant par la racine indiquée.
L'en-tête du répertoire indique la densité de la disquette, le n° du lecteur, son nom s'il existe et le nombre de k-octets libres.
En Basic 1, on ne peut indiquer que le n° de lecteur.
Pour chaque fichier listé, les renseignements du répertoire comprennent six parties:

1. Nom du fichier (huit 8 caractères)

2. Suffixe
BAS: programme Basic
DAT: données
BIN: binaire
MAP: image
TRA: trace

3. Type de fichier
B: programme Basic
D: données Basic
A: programme en assembleur
M: programme en langage machine

4. Type de données
A: ASCII
B: Binaire

5. Taille du fichier en nombre de k-octets

6. Commentaire (huit caractères) s'il existe

L'exécution de DIR pour un périphérique autre qu'un lecteur de disquettes provoque une erreur "Illegal Function Call".

Exemple:

DIR "0:TRICTRAC.BAS" liste le fichier TRICTRAC.BAS dans le répertoire du lecteur 0
DIR "1.DAT" liste tous les fichiers de données du lecteur 1
DIR "T" liste tous les fichiers commençant par T sur le lecteur courant
DIR liste le répertoire complet du lecteur courant

DIRP "n° de lecteur: nom . suffixe" *

instruction

Liste le répertoire relatif aux fichiers précisés sur l'imprimante parallèle.

La syntaxe est identique à celle de DIR.
Le listage est obtenu sur l'imprimante parallèle.

Exemple:

DIRP "2:" liste sur imprimante parallèle tous les fichiers du lecteur 2

DO...LOOP *

instruction générale d'itération

Toutes les instructions comprises entre DO et LOOP sont répétées. On ne peut sortir normalement de la boucle d'itération que par l'instruction EXIT. L'exécution de EXIT fait sortir de la boucle d'itération; l'exécution se poursuit alors à la première instruction qui suit LOOP.

Les instructions de branchement (GOTO, ON...GOTO et THEN n° de ligne) permettent une sortie à un numéro de ligne différent de la sortie normale. Ce mode de sortie n'est utile que pour les cas exceptionnels (erreur, arrêt du programme,...) car il fait perdre les avantages de la structuration du programme.
Il n'est pas possible de rentrer au milieu d'une boucle DO...LOOP par une instruction GOTO. L'interpréteur Basic, à l'exécution de LOOP, cherche le DO correspondant. Quand le DO correspondant n'a pas été exécuté, le message d'erreur "Loop Without DO" apparaît.
Il est possible de construire des boucles emboîtées en prenant soin de placer la boucle interne entièrement à l'intérieur de la boucle externe.
Dans les cas de plusieurs boucles emboîtées, on peut sortir de plusieurs boucles par une seule instruction EXIT. Le nombre qui suit EXIT indique le nombre de boucles dont il faut sortir.

Exemples:

```
10 DO: PRINT "BOUCLE": LOOP          itération infinie
10 DO
20 INPUT A
30 IF A=0 THEN EXIT
40 PRINT A
50 LOOP
60 PRINT "FIN"
```

itération des lignes 20 à 40. Quand la réponse vaut 0, sortie de la boucle

DOS ♦

commande

Retourne à la page en-tête en perdant le contexte pour charger le DOS du Basic 1.

DRAW chaîne de caractères ♦

instruction

Trace le dessin défini dans la chaîne de caractères. Les ordres élémentaires du langage graphique exécuté par DRAW sont les suivants:

U (*Up*), D (*Down*), R (*Right*), L (*Left*), E, F, G, H suivis d'un nombre: déplacement du nombre indiqué dans la direction fixée. Si le nombre vaut 1, on peut ne pas l'indiquer.

D 30

Move: M colonne, ligne
déplacement au point situé en (colonne, ligne)

M 100, 1

M ± C, ± L
déplacement relatif de ± C en colonnes et de ± L en lignes. Le signe est obligatoire dans ce cas.

V – 10, – 10

Scale: S nombre
changement d'échelle (de 1 à 63). L'échelle normale est 4 (rapport 1, 1)

S 16

Arrow: A nombre
fixe l'orientation générale des déplacements ultérieurs, effectue une

rotation sur les déplacements ultérieurs. Le nombre de 0 à 3 correspond à : Up (0), Right (1), Down (2), Left (3).

A2

Color: C couleur
fixe la couleur du tracé de −16 à 15.

C3

Blank B
placé devant un ordre effectue le déplacement sans tracer.

BM100,150

No Update: N
placé devant un ordre permet de revenir au point de départ après un déplacement.
XJ20

Exécute : X variable chaîne;
effectue le tracé défini dans la variable chaîne de caractères.

XAS

On peut introduire la valeur d'une variable numérique par: = variable;

U = 8

Les ordres élémentaires peuvent être séparés par un point-virgule ou un espace. On peut également mettre un espace entre un ordre et son argument.

DRAW "C1;BM150,150;C2;R20G20"
DRAW "U=N81;L=N82;"
DRAW "L20 R30 G30"
DRAW "U30"

DSKF (n° de lecteur)
fonction

Donne le nombre de k-octets libres sur la disquette située dans le lecteur indiqué.

Le numéro de lecteur est obligatoire.

Exemple :

DSKF(0) volume libre sur la disquette du lecteur 0

DSKI$ (n° de lecteur, piste, secteur)
fonction

Effectue la lecture d'un secteur pour lequel on donne le numéro, le numéro de la piste et celui du lecteur où il se trouve.

Le résultat est une chaîne de caractères de 128 octets en simple densité ou de 255 octets en double densité. Le numéro de lecteur est compris entre 0 et 4; le numéro de piste entre 0 et 79, le numéro de secteur entre 1 et 16 inclus. Toute tentative de lecture en dehors de ces valeurs provoque une erreur "Illegal Function Call".

Exemple :

DSKI$(0,20,2) lecture du secteur 2 de la piste 20 du lecteur 0

DSKINI n° de lecteur [,facteur d'entrelacement] [,nom]

Formate la disquette qui se trouve dans le lecteur précisé.

La disquette ne doit pas être protégée en écriture.
La souris ne doit pas être déplacée pendant la durée du formatage d'une disquette 2 × 160 Ko, sous peine d'un formatage défectueux.

Le formatage d'une disquette déjà enregistrée provoque son effacement intégral et toutes les données qu'elles contenait sont perdues.
Si la disquette est défectueuse, le message Input/Output Error apparaît.
Le facteur d'entrelacement, qui n'est ni obligatoire ni très utile, permet de choisir le nombre de secteurs compris entre deux secteurs de numéros consécutifs (implicitement fixé à 7 en l'absence de précision).
Le numéro du lecteur est obligatoire.
Le nom de la disquette, qui n'est pas obligatoire, est une chaîne de huit caractères maximum.
Le nom de la disquette est interdit en Basic 1.
Si l'on a fait au préalable VERIFY ON, l'initialisation a lieu avec vérification et dure beaucoup plus longtemps.

Exemple:

DSKINIT 1,6,"TRAVAIL" formate la disquette qui se trouve dans le lecteur 1 avec un entrelacement de 6 et nomme cette disquette TRAVAIL
DSKINIT 0 formate la disquette qui se trouve dans le lecteur 0

DSKO$ n° de lecteur, piste, secteur, chaîne de caractères

instruction

Dépose la chaîne de caractères dans le secteur dont on donne le numéro, le numéro de la piste et le numéro du lecteur sur lequel il se trouve.

Le numéro de lecteur doit être compris entre 0 et 4, le numéro de la piste entre 0 et 79, le numéro du secteur entre 1 et 16. La chaîne de caractères doit avoir une taille de 128 caractères maximum en simple densité ou 255 en double densité. Si sa taille est inférieure, le secteur sera rempli de caracteres de code ASCII 0 dans la partie vide après le dernier caractère.

Exemple:

DSKO$ 0,16,3,HAS
DSKO$ 0,20,1,"JEUX" donne le nom JEUX à la disquette du lecteur 0

END

instruction

Termine l'exécution d'un programme.

Elle provoque d'abord la fermeture de tous les fichiers ouverts immédiatement au moment de son exécution, puis le passage en mode commande qui se traduit par le message OK sur l'écran.
L'instruction END ne modifie ni le contenu des variables ni les options de CLEAR.
Elle peut être placée n'importe où dans le programme pour terminer l'exécution. Elle n'est pas indispensable à la dernière ligne du programme.

EOF (n° de canal)

fonction

Retourne –1 (VRAI) si la fin du fichier est atteinte ou 0 (FAUX) sinon.

Cette fonction évite l'erreur "Input Past End " dans la lecture d'un fichier.

Exemple:

10 'LECTURE
20 OPEN"I",#1,"DONNEES"

```
10 DO
20 IF EOF(1)=-1 THEN EXIT
30 INPUT#1,A$:PRINT A$
40 LOOP
```

variables ERL et ERR

ERL et ERR sont des variables particulières qui permettent, lorsqu'une erreur est détectée pendant l'exécution d'un programme, de connaître le numéro de la ligne de l'erreur détectée et le numéro de l'erreur afin de la traiter.
Elles sont réservées à cet usage et toute tentative d'affectation de valeur à ces variables au moyen d'une instruction d'affectation ou d'une instruction de lecture est interdite et détectée comme une erreur de syntaxe.
Lorsqu'une erreur est détectée, deux cas se présentent:
— Aucun traitement d'erreur n'est indiqué; l'exécution du programme est interrompue et le message d'erreur est affiché à l'écran.
— Un traitement d'erreur a été mis en état de veille au moyen de l'instruction ON ERROR. Dans ce cas, la partie de programme chargée du traitement de l'erreur est exécutée. Le numéro de la ligne où s'est produite l'erreur est contenu dans la variable ERL et le numéro de l'erreur est contenu dans ERR. Ainsi les instructions du module de traitement de l'erreur peuvent-elles connaître le type de l'erreur.
La liste de codes d'erreur est fournie en Annexe 7.

Exemple:

```
5 ON ERROR GOTO 100
10 DO
20  INPUT X
30  A=1/X
40  PRINT "L'inverse est :";A
50 LOOP
100 'module de traitement d'erreur
110 IF ERR=6 THEN PRINT "Dépassement numérique":RESUME 20
120 IF ERR=5 THEN PRINT "Paramètre incorrect":RESUME 20
130 RESUME 30
```

ERROR nombre

instruction

Simule une erreur de code existant ou non.

Le nombre est converti en entier par arrondi et donne le numéro de l'erreur.
Cette erreur simulée pourra soit provoquer l'édition du message d'erreur, soit être traitée par le module désigné par ON ERROR.
Si aucune instruction ON ERROR n'est exécutée, le message d'erreur s'affiche ou, s'il s'agit d'un code inconnu de l'interpréteur Basic, le message "Undefined Error".

Exemple:

```
10 S=10
20 T=5
30 ERROR S+T 'simulation de l'erreur 15
RUN
String Too Long in 30
```

EVAL(chaîne de caractères) *

fonction

Rend la valeur fournie par l'évaluation de la chaîne codée par CRUNCHS.

Le résultat de l'évaluation est le même que celui obtenu par la même expression dans un programme.

Si la chaine n'est pas une expression codée par CRUNCH$, le message Syntax Error apparaît.

Exemple:

```
10 LINE INPUT A$
20 A$=CHR$(CRUNCH$(A$))
RUN
?32 - 22
25
```

EXEC adresse, liste de paramètres

instruction

Permet d'exécuter un module écrit en langage machine résident en mémoire centrale.

Si l'adresse n'est pas précisée, c'est celle du dernier EXEC ou l'adresse d'exécution du dernier LOADM qui est prise par défaut.
Si l'adresse est comprise entre &H6000 et &H9FFF, c'est la banque de mémoire fixée par BANK qui est adressée ou à défaut la banque de rang le plus élevé.
Des paramètres peuvent être passés au module. Ils sont mentionnés après l'adresse, séparés par des virgules.
On peut les évaluer en appelant, dans le programme binaire, les modules d'analyse de l'interpréteur Basic. L'appel se fait par JSR adresse. Chaque appel évalue un paramètre.
La liste des adresses des modules d'analyse est fournie en Annexe 4.
Le passage de paramètres est interdit en Basic 1.
Le module en langage machine doit se terminer par l'instruction RTS pour provoquer le retour au programme Basic. Les registres S (pointeur de pile) et DP (page directe) ne doivent pas avoir été modifiés.

EXIT nombre *

instruction

Instruction de sortie de boucle DO...LOOP ou FOR...NEXT.

L'exécution de cette instruction provoque le branchement à la première instruction qui suit LOOP ou NEXT.
Le nombre indique de combien de boucles il faut sortir. Par défaut, il est pris à 1. Un nombre à 0 ne fait pas sortir de la boucle.

Exemple:

```
10 I=0
20 DO
30 I=I+1:PRINT I
40 IF INKEY$<>"" THEN EXIT
50 LOOP
```

La boucle (incrément d'une variable, affichage de sa valeur) se termine quand on appuie sur une touche.

EXP(nombre)

fonction

Retourne l'exponentielle du nombre soit e^nombre.

Rappelons que e vaut 2.712828.
En Basic 128, le résultat est en double précision si l'argument est en double précision.
Le dépassement de capacité est atteint dès que le nombre dépasse 58.02969.

Exemple:

```
?EXP(2
7.38906
```

FIELD # n° de canal, longueur 1 AS variable de champ 1 ,longueur 2 AS variable de champ 2...

instruction

Définit l'organisation en champs de l'enregistrement du fichier à accès direct dont on donne le numéro de canal.

Pour chaque champ, on précise sa longueur en caractères et le nom de la variable de champ qui le désigne. Cette variable est toujours une variable chaîne de caractères.
La somme des longueurs ne doit pas dépasser la longueur de l'enregistrement du fichier qui a été fixée au moment de l'ouverture (OPEN). Si la longueur n'a pas été fixée dans OPEN, elle est de 255 caractères (ou 128 en simple densité).
L'instruction FIELD est en général exécutée une fois, après l'ouverture du fichier par OPEN. Tous les enregistrements sont alors lus et écrits suivant ce format. Cependant il est possible d'exécuter des instructions FIELD différentes pour lire des enregistrements de formats différents ou même d'utiliser plusieur FIELD pour lire ou écrire dans le même enregistrement sous des formats différents.
Les variables de champs ainsi définies ne peuvent pas être manipulées comme n'importe quelle variable. Leur contenu est toujours utilisable pour être affiché, transformé ou affecté à une autre variable. Par contre, on ne peut y déposer une valeur que par les instructions LSET, RSET et MID$()=.
Ces variables chaînes de caractères peuvent également recevoir des nombres après que ceux-ci ont été convertis par les fonctions MKI$ (nombres entiers), MKS$ (nombres réels en simple précision) ou MKD$ (nombres en double précision). La taille d'un nombre converti par ces fonctions est de:
— 2 octets s'il est entier
— 4 octets en simple précision
— 8 octets en double précision

Exemple:

```
FIELD # 2,15 AS NOMS,4 AS QS
```

définit, pour le canal 2, deux champs: le premier de quinze caractères et de nom NOMS, le second de quatre caractères et de nom Q$; on pourra affecter un nombre en simple precision à QS.

FILES nombre de canaux, longueur, taille cache

instruction

Réserve le nombre de canaux utilisables simultanément dans un programme.

Ce nombre ne peut pas être supérieur à 15.
Chaque canal réservé occupe une zone de 281 octets (ou 153 en simple densité) pour les informations concernant le fichier et la zone de transit des enregistrements.
A l'initialisation, le nombre de canaux réservés est fixé à 2. Si l'instruction FILES comporte une indication de longueur (celle-ci n'est pas obligatoire), cette longueur fixe la somme des longueurs des enregistrements des fichiers à accès direct ouverts simultanément. Elle est fixée à 256 octets à l'initialisation (deux fichiers avec enregistrements de 128 octets).
Par exemple, pour pouvoir ouvrir trois fichiers à accès direct dont les longueurs d'enregistrement sont respectivement 30, 40 et 50 caractères, il faut réserver une longueur d'au moins 120 caractères.
L'instruction FILES remet à zéro toutes les variables existantes. Elle doit donc être exécutée en mode direct ou en tout début de programme.

La taille cache réserve en mémoire une zone tampon de n pistes, n pouvant varier de 0 à 25, pour optimiser les accès physiques au QDD ou à la disquette. Par défaut, n=3 avec un QDD et 0 pour une disquette (voir page 120).

Exemple:

```
FILES 3,120
```

FIX(nombre)

fonction

Rend la partie entière du nombre.

La partie entière est obtenue par troncature. Le résultat est un nombre réel.
Le résultat de FIX est équivalent à celui de INT pour les nombres positifs, il en diffère pour les nombres négatifs.

Exemple:

```
?FIX(3.4):INT(3.4)
 3  3
?FIX(-3.6):INT(-3.6)
-3 -4
```

FKEY$(numéro) *

fonction

Rend le code de la touche fonction de numéro indiqué.

Le numéro est un nombre compris entre 1 et 10.

Exemple:

FOR...NEXT

FOR variable = valeur initiale TO valeur finale STEP pas

...

NEXT variable,variable ,...

Instruction d'itération permettant de répéter une séquence d'instructions un nombre fixé de fois.

La variable qui suit FOR est appelée variable de contrôle. Elle est numérique de type entier ou simple précision.
A toute instruction FOR doit correspondre une instruction NEXT. mais une instruction NEXT peut correspondre à plusieurs instructions FOR. Dans ce cas, les variables de contrôle de chacune des instructions FOR sont indiquées après NEXT.

Au départ, la variable de contrôle prend la valeur initiale indiquée. Puis l'interpréteur teste si la variable de contrôle est supérieure à la valeur finale.
— Si oui, les instructions de la boucle ne sont pas exécutées et l'exécution se poursuit après NEXT.
— Si non, les instructions comprises entre FOR et NEXT sont exécutées.
La rencontre de NEXT provoque l'incrémentation de la variable de contrôle de la valeur pas, et l'on revient au test précedent.
La boucle d'itération est exécutée jusqu'à ce que la réponse au test soit positive.
Dans le cas d'un pas négatif, l'interpréteur teste si la variable de contrôle est inférieure à la valeur finale et la variable de contrôle est décrémentée de pas.
Si l'option STEP n'est pas précisée, l'incrément par défaut est 1.
En Basic 128, il est possible de sortir de la boucle par l'instruction EXIT. L'exécution se poursuit alors après NEXT, mais en général, la variable de contrôle n'a pas atteint la valeur finale.

Boucles emboîtées

Les boucles FOR peuvent être emboîtées, c'est-à-dire qu'une boucle FOR peut contenir une autre boucle FOR utilisant une variable de contrôle différente.
Si plusieurs boucles ont la même instruction terminale, il est possible d'utiliser un NEXT multiple pour toutes ces boucles:
NEXT variable1,variable2,...
Si la variable de contrôle est omise, l'instruction NEXT correspond à la boucle FOR la plus proche qui n'est pas appariée.

Exemples:

```
10 K=10
20 FOR I=1 TO K STEP 2
30 K=K+10
40 PRINT K
50 NEXT I
```

la valeur finale de la variable de contrôle reste 10 bien que K ait été modifiée en cours d'exécution.

```
10 I=0: X=0
20 FOR I=3 TO 2
30 X=X+1
40 NEXT I
50 PRINT I;X
```

la boucle n'a pas été exécutée et la variable de contrôle conserve la valeur initiale (3).

```
10 DIM A(20,10)
20 FOR I=1 TO 20
30 FOR J=1 TO 10
40 A(I,J)=1
50 NEXT J
```

deux boucles emboîtées pour l'initialisation d'un tableau.

```
100 FOR I=1 TO 20
110 IF A(I)=0 THEN EXIT
120 NEXT I
130 IF I=21 THEN PRINT "PAS D'ELEMENT NUL"
```

recherche d'un élément nul dans un tableau et sortie par EXIT.

FRE(donnée)

fonction

Permet de connaître l'espace mémoire libre.

Quatre informations sont disponibles:
1. FRE(0) donne la place totale disponible en mémoire.
2. FRE(1) donne la place disponible dans la zone de travail de l'interpréteur (entre &H6100 et &H9FFF).
3. FRE(2) donne la place disponible pour le programme et les données (entre &HA000 et &HDFFF dans les banques de mémoire).
4. FRE(A$) donne la place disponible pour les chaînes. Cet espace est fixé à 300 caractères à l'initialisation.

Noter que FRE(0) = FRE(1) + FRE(2)
Basic 1 ne fournit pas FRE(1) ni FRE(2).

Exemple:

```
10 ?FRE(0),FRE(1),FRE(2),FRE(A$)
20 X$="0123456789"
30 ?FRE(0),FRE(1),FRE(2),FRE(A$)
```

FWD nombre *

instruction

Déplace une tortue dans sa direction de la quantité indiquée.

Le nombre doit être compris entre --255 et 255.
Si le nombre est négatif, la tortue recule.
Si la tortue est visible, elle est effacée de son ancienne position et affichée dans la nouvelle.
Si la trace est active, le déplacement est tracé sur l'écran.
Toutes les modifications de taille, de direction ou de rotation sont prises en compte avant le déplacement.
La tortue déplacée est la dernière désignée par TURTLE.
Si aucune instruction TURTLE n'a été exécutée, l'erreur Illegal Function Call est déclenchée.

Exemple.

FWD 30 déplace la tortue de trente points dans sa direction

*GET (colonne 1, ligne 1)–(colonne 2, ligne 2) ,élément de tableau **

instruction

Mémorise dans un tableau numérique la portion d'image graphique contenue dans le rectangle défini par les deux sommets (colonne 1, ligne 1) et (colonne 2, ligne 2).

La portion d'image est obligatoirement composée d'un nombre entier de caractères. Les coordonnées des deux sommets sont donc données en caractères : colonnes de 0 à 39 (ou 79) et lignes de 0 à 24. Si le deuxième sommet n'est pas précisé, c'est le sommet en bas à droite qui est pris par défaut (39,24) ou (79,24) en 80 colonnes.
Le tableau numérique doit avoir été déclaré au préalable et il doit être de type entier.
Plusieurs images peuvent être mémorisées dans un seul tableau. Elles sont systématiquement compactées et rangées depuis le bas du tableau, c'est-à-dire de l'indice le plus élevé vers les indices plus faibles.
L'élément de tableau désigné dans l'instruction GET est l'élément d'indice le plus élevé pour cette image. L'image est mémorisée à partir de l'élément précédent. Après l'exécution, l'élément désigné contient la valeur de l'indice du premier élément libre du tableau.
C'est cet élément qui pourra servir pour une nouvelle instruction GET ou LOADP avec le même tableau.
L'image est compactée dans le tableau, le volume qu'elle occupe est dépendant de son contenu.
Si le tableau est trop petit, une erreur "Out of Memory" est détectée.
La portion d'image ainsi mémorisée peut être restituée en un autre endroit de l'écran par PUT.
Elle peut être conservée en fichier par l'instruction SAVEP.

Le mode de compactage est compatible avec les outils COLORPAINT et PRAXITEL.

Exemple :

DIM T%(1000) déclare le tableau de réception
GET (0,0)-(639,199),T%(1000) mémorise la portion d'image à partir de la fin du tableau
T%(0)=1000 contient le premier élément libre
817 c'est le 817
GET (0,0)-(639,199),T%(817) mémorise une deuxième portion à partir du premier élément libre
T%(817) nouveau premier élément libre
760 c'est le 760 ... et ainsi de suite

GET (colonne 1, ligne 1)–(colonne 2, ligne 2) ,tableau ♦

instruction

Mémorise dans un tableau numérique la portion d'image graphique contenue dans le rectangle défini par les deux sommets (colonne 1, ligne 1) et (colonne 2, ligne 2). Le tableau numérique (entiers ou simple précision) doit avoir été déclaré au préalable et sa dimension suffisante pour recevoir l'image, compte tenu du fait qu'un octet peut recevoir un seul point de l'image et qu'un élément de tableau comporte 2 ou 4 octets suivant qu'on est en entiers ou en simple précision. La portion d'image ainsi mémorisée peut être restituée en un autre endroit de l'écran par PUT.

DIM T(625)
GET (150,80)-(190,140),T

GET # n° de canal, n° d'enregistrement

instruction

Transfère un enregistrement d'un fichier à accès direct (dont on

nonne le numéro de canal) vers la zone tampon en mémoire centrale.

La récupération du contenu de l'enregistrement peut alors se faire par INPUT#, ou directement dans les variables de champ si un FIELD a été exécuté au préalable.
Si le numéro d'enregistrement n est pas précisé, GET lit l'enregistrement suivant du fichier ou le premier si aucun enregistrement n'a encore été lu ou écrit.

Exemple:

lit le 10e enregistrement du fichier n° 2

GOSUB numéro de ligne
instruction

Permet le branchement à un sous-programme

L'exécution d'un sous-programme appelé par GOSUB débute à la première instruction de la ligne indiquée par son numéro. L'exécution du sous-programme se termine par l'instruction RETURN, elle provoque le retour à l'instruction qui suit immédiatement le GOSUB appelant.
Un sous-programme peut avoir plusieurs points d'entrée. De même, il peut contenir plusieurs instructions RETURN.
Un sous-programme peut appeler un autre sous-programme et ainsi de suite jusqu'à encombrement complet de la mémoire. Un sous-programme peut s'appeler lui-même.
Une tentative d'exécution de GOSUB vers une ligne inexistante est détectée par l'erreur "Undefined Line".
Un sous-programme peut être placé n'importe où dans un programme, mais il est préférable de le séparer du programme principal par une instruction END.

Exemple:

calcul récursif de factorielle N

GOTO numéro de ligne
instruction

Permet le branchement inconditionnel à une ligne de programme.

Si la ligne désignée contient une instruction exécutable, celle-ci sera exécutée, puis les instructions suivantes.
Si la ligne désignée ne contient pas d'instruction exécutable (REM ou DATA), l'exécution se poursuit après cette ligne.
Si la ligne désignée n'existe pas, l'erreur "Undefined Line" est détectée.
En mode d'exécution directe, cette instruction permet de poursuivre au numéro indiqué l'exécution d'un programme interrompu. A la différence de "RUN numéro de ligne", elle n'initialise pas les variables et ne ferme pas les fichiers ouverts.
L'écriture d'une boucle est plus facile avec DO...LOOP.

Exemple:

```
10 PRINT "UNE BOUCLE INFINIE"
20 GOTO 10
```

boucle infine avec GOTO

GR$(nombre)

fonction

Rend le caractère utilisateur de numéro indiqué.

Les caractères utilisateurs sont numérotés de 0 à 127. Ils doivent avoir été déclarés au préalable par CLEAR (3e paramètre) et définis par DEFGR$.
GR$(i) est équivalent à CHR$(128+i).

Exemple:

```
?GR$(23)      affiche le caractère utilisateur n° 23
```

HEAD *

fonction

Rend la direction de la tortue active.

Le résultat est un nombre compris entre 0 et 255.
La tortue active est la dernière fixée par TURTLE.

Exemple:

```
?HEAD
62
```

HEAD TO nombre *

instruction

Fixe ou modifie la direction du mouvement de la tortue active.

Si l'option TO est présente, la direction est fixée de façon absolue, sinon elle est modifiée de façon relative.
Le nombre indiquant la direction est compris entre - 255 et 255. S'il est plus grand, seul le reste de la division par 256 (modulo) est pris en compte.
Un nombre positif signifie vers la droite, un nombre négatif signifie vers la gauche.
La tortue active est la dernière désignée par TURTLE.

Exemple:

```
ROT 64        modifie la direction de 90° vers la droite
ROT 320       idem
ROT TO -32    fixe la direction à 45° vers la gauche
```

HEX$ (nombre)

fonction

Donne une chaîne de caractères représentant le nombre dans la base hexadécimale (16).

Si l'argument n'est pas entier, il est tronqué. Sa valeur doit être comprise entre 65536 et 65535.

Exemple:

```
?HEX$(65535)
FFFF
```

IF condition {GOTO n° de ligne / THEN instructions} {ELSE {n° de ligne / instructions}}

instruction

Permet des branchements conditionnels.

L'exécution de cette instruction se fait de la manière suivante:
la condition est d'abord évaluée.
— Si le résultat est VRAI, alors les instructions qui suivent THEN sont exécutées et, si ELSE est présent, les instructions qui le suivent sur la ligne sont ignorées.
— Si le résultat est FAUX, les instructions qui suivent ELSE sont exécutées ou bien, si ELSE est absent, l'exécution passe à la ligne suivante du programme.
Dans chaque cas, si la liste d'instructions est remplacée par un numéro de ligne, il y a branchement à la ligne indiquée.

Exemple:

```
IF B*B-4*A*C < 0 GOTO 500        si négatif, branchement en ligne 500
```

instructions IF emboîtées

Il est possible de faire figurer à l'intérieur d'une instruction IF une ou plusieurs instructions IF dans la limite d'une ligne logique.
L'interprétation de IF emboîtés est la suivante:
— chaque ELSE est associé au THEN le plus proche qui n'est pas associé à un autre ELSE.

Exemple:

```
IF A<B THEN IF B<C THEN PRINT "A<C" ELSE PRINT "B>=C"
IF A<B THEN IF B<C THEN PRINT "A<C" ELSE PRINT "B>=C" ELSE PRINT
"A>=B"
```

Remarque:
Les tests d'égalité des nombres réels ne sont pas toujours exacts puisqu'ils dépendent de la précision. Il faut donc leur préférer le test d'une différence minimum.

Exemple:

```
IF ABS(A-B)<1E-5 THEN PRINT "A et B égaux"
```

INKEY$

fonction

Retourne le dernier caractère tapé au clavier ou une chaîne vide si aucun caractère n'a été tapé.

Cette fonction ne renvoie pas d'écho sur l'écran.
Tous les caractères, y compris les caractères de contrôle, sauf CNT-C et STOP, sont pris en compte par INKEY$. En particulier, l'appui sur ENTREE rend le caractère Retour Chariot.

Exemple:

```
10 FOR I=1 TO 1000: NEXT I
20 A$=INKEY$
30 PRINT "CARACTERE:" ;A$
```

INMOUSE variable1, variable2 ★
INPUTMOUSE variable1, variable2

instructions

Lit les coordonnées de la souris.

INMOUSE effectue immédiatement la lecture. INPUTMOUSE attend l'appui sur une des deux gachettes.

Exemple:

```
INMOUSE(X)
```

INPEN variable colonne, variable ligne
INPUTPEN variable colonne, variable ligne
instructions

Permettent de lire les coordonnées du point de l'écran visé par le crayon optique.

INPEN effectue immédiatement la lecture, INPUTPEN attend que le contact du crayon optique soit fermé pour effectuer la lecture.
La première variable reçoit la coordonnée horizontale (entre 0 et 319), la seconde variable reçoit la coordonnée verticale (entre 0 et 199).
En mode 80 colonnes, la valeur de la colonne est un nombre pair compris entre 0 et 638.
Quand la mesure ne peut pas être effectuée, les deux variables reçoivent -1. Cette situation se présente en cas de:
— mauvais réglage du crayon optique,
— luminosité insuffisante du point visé (noir ou rouge par exemple),
— visée d'un point hors de la zone utile de l'écran.

Exemple:

```
100 CLS
110 INPUTPEN X,Y
120 IF X<>-1 THEN PRINT X,Y
130 LOCATE 0,0
140 PRINT X,Y
```

INPUT chaîne de caractères; liste de variables
INPUT chaîne de caractères; liste de variables
instruction

Instruction d'entrée de données au clavier.

Le message contenu dans chaîne de caractères est affiché (s'il existe).
Il est suivi d'un point d'interrogation et d'un espace si le délimiteur est un point-virgule. Le point d'interrogation n'est pas affiché si le délimiteur est une virgule.
L'affichage des caractères suivants est forcé à la taille normale.
L'utilisateur doit alors introduire les données attendues par la liste de variables. Ces données doivent être de même type que les variables de la liste, sinon le message "Redo from start" est affiché.
Les différentes données sont séparées par des virgules. L'introduction d'une chaîne doit être encadrée par des guillemets si elle contient une virgule ou des espaces significatifs au début ou à la fin.
Les variables peuvent être des éléments de tableaux mais pas des tableaux.
INPUT ne peut être utilisé en mode direct.

Exemple:

```
10 INPUT "JOUR, MOIS, ANNEE";J%,M$,AN%
```

INPUT# n° de canal, liste de variables
instruction

Lit le fichier dont on précise le n° de canal et affecte les valeurs lues aux variables mentionnées dans la liste.

Le type des données (numérique ou chaîne de caractères) doit correspondre au type des variables. Cette instruction obéit aux mêmes règles que INPUT à partir du clavier.
Si les variables sont plus nombreuses que les données, l'erreur

"Input Past End" est déclenchée. On s'en prémunit en utilisant la fonction EOF.

Exemple:

```
INPUT#2,A$,N          lit les données dans A$ et N successivement
```

INPUT$ (nombre de caractères,# n° de canal)

fonction

Retourne une chaîne de caractères qui contient le nombre de caractères lus à la suite dans le fichier indiqué.

Cette fonction permet de lire un nombre déterminé de caractères sur le canal considéré. Elle permet de connaître de façon exacte (au caractère près) le contenu d'un fichier.
Si le n° de canal n'est pas précisé, la lecture se fait à partir du clavier sans écho à l'écran.
CNT-C et STOP sont transmis et n'interrompent pas l'exécution.
Au clavier, cette instruction permet d'éviter la frappe de la touche ENTREE.

Exemple:

```
A$=INPUT$(3,#1)       lecture de 3 caractères sur le fichier 1
```

INPUTTURTLE variable colonne, variable ligne *

instruction

Permet de connaître les coordonnées de la tortue active.

La première variable reçoit le n° de colonne, la seconde le n° de ligne. Les valeurs reçues s'étendent de −32768 à 32767.
La tortue active est la dernère désignée par TURTLE.

Exemple:

```
10 TURTLE 1
20 DO
30 FWD 10 : READ N
40 INPUTTURTLE C,L : LOCATE C,L : PRINT "■"
50 LOOP
```

A chaque déplacement, la ligne 40 lit les coordonnées de la tortue et les affiche.

INPUTWAIT n° de ligne; durée, chaîne de caractères, liste de variables *
INPUTWAIT n° de ligne; durée, chaîne de caractères; liste de variables

instruction

Permet de lire des données à partir du clavier, en accordant un temps limité pour la réponse.

Le premier paramètre indique la ligne à exécuter si le temps est dépassé.
Le deuxième paramètre indique en secondes le temps maximum autorisé pour taper la réponse.

Le reste de l'instruction est identique à INPUT. En particulier, la chaîne de caractères doit être suivie d'un point-virgule si l'on veut qu'un point d'interrogation soit affiché.
La forme réduite:
INPUTWAIT n° de ligne:durée,"",
permet d'obtenir une temporisation suivie d'un branchement.

Exemple:

```
10 PRINT "DEBUT"
20 DO
30 INPUTWAIT 100:10,"Valeur de A";A
```

```
40 PRINT A
50 LOOP
60 END
100 PRINT "Prochain"
110 GOTO 10          attend 10 secondes la réponse
INPUT WAIT 900,5     temporise 5 secondes et branche en ligne
                     900
```

INSTR (position, chaîne 1, chaîne 2)
fonction

Rend la position de la chaîne 2 dans la chaîne 1.

Si le premier paramètre est présent, la recherche se fait à partir de la position indiquée, sinon elle se fait à partir du premier caractère de chaîne 1.
Le résultat de la fonction INSTR est 0:
— si la chaîne 2 ne figure pas dans la chaîne 1,
— si la chaîne 1 est vide,
— si la position de départ est supérieure à la longueur de la chaîne 1.
Si la chaîne 2 est vide, le résultat de la fonction est la position de départ (1 par défaut).
Si la chaîne 2 est présente dans la chaîne 1, la fonction INSTR rend sa position.

Exemple:

```
?INSTR("CHAINE","A")
 2
?INSTR(6,"CHAINE","A")
 6
?INSTR(3,"CHAINE","A")
 0
```

INT (nombre)
fonction

Retourne le plus grand entier inférieur ou égal au nombre précisé.

Le résultat est un nombre réel. L'argument peut être entier, réel ou double précision.

Exemple:

```
?INT(56.75)
 56
?INT(-56.75)
-57
```

Pour arrondir au nombre entier le plus proche, il faut prendre:
INT(X+.5)

INTERVAL ON
INTERVAL OFF
instructions

Permettent de déclencher ou d'arrêter le compteur de temps défini dans l'instruction ON INTERVAL.

KILL descripteur de fichier
commande

Supprime le fichier en question, c'est-à-dire supprime ses références dans le répertoire et libère la place qu'il occupe sur la disquette.

Dans la description du fichier, le nom du périphérique n'est pas obligatoire mais le suffixe est obligatoire. Le périphérique pris par défaut est le lecteur 0 ou celui fixé par DEVICE. L'instruction KILL ne

fonctionne que pour un fichier sur disquette. Si le fichier n'existe pas, le message "File Not Found" apparaît.

Exemple:

KILL "1:BONJOUR.BAS" supprime le fichier BONJOUR.BAS de la disquette 1

LEFT$ (chaîne , longueur)
fonction

Donne la partie gauche de la chaîne sur la longueur précisée.

Si la longueur est nulle, le résultat est la chaîne vide.
Si la longueur demandée est supérieure à la longueur de la chaîne, le résultat est la chaîne entière.

Exemple:

```
PRINT LEFT$("MELODRAME",4)
VELO
```

LEN (chaîne)
fonction

Donne la longueur de la chaîne de carctères.

Tous les caractères de la chaîne sont comptés.
Il ne faut pas oublier que les minuscules accentuées et le ç sont codés sur trois caractères.

Exemple:

```
PRINT LEN("TROIS")
5
PRINT LEN("été")
7
```

LET variable = expression *
instruction

Permet d'affecter une valeur à une variable.

Le mot clé LET est facultatif.
La variable et l'expression doivent être de même type, sinon l'erreur "Type Mismatch" est détectée.

Exemple:

```
LET A = 12
B = SQR(A) + 3
C$ = "Bonjour"
D$ = A$ + "!"
```

LINE (c1,l1)–(c2,l2) caractère,couleur,fond ,inversion
LINE (c1,l1) – (c2,l2),couleur
instruction

Trace une ligne entre les deux points désignés.

Cette instruction fonctionne en mode caractère quand l'argument caractère est présent.
Sa syntaxe est identique à celle de BOX.

Exemple:

```
LINE (30,5)-(15,20),3,0      trace une ligne de + entre les deux points de couleur 3 et de fond 0
LINE (100,60)-(163,96),2     trace une ligne de couleur 2 entre les deux points graphiques
LINE -(200,200)              la ligne part du dernier point tracé
```

Voir STEP.

LINE INPUT chaîne de caractères;variable chaîne
LINE INPUT chaîne de caractères,variable chaîne
instruction

Permet d'introduire dans une variable chaîne une séquence d'au plus 255 caractères quelconques.

La fin de la réponse est marquée par la touche ENTREE. Tous les caractères frappés, sauf le caratère Retour Chariot sont affectés à la variable chaîne.
La présence de la chaîne de caractères provoque l'affichage du message qu'elle contient.
Que la chaîne soit suivie d'une virgule ou d'un point-virgule, aucun point d'interrogation n'est affiché.
Cette instruction est très pratique pour lire une ligne dont ne connaît pas la structure.

Exemple:

```
10 LINE INPUT "QUOI";A$
20 PRINT A$
```

LINE INPUT # n° de canal, variable chaîne
instruction

Lit le prochain enregistrement du fichier dans une variable chaîne de caractères en prenant tous les caractères.

Exemple:

```
LINE INPUT #2, XS
```

LIST descripteur de fichier, n° ligne 1 - n° ligne 2
commande

Liste le programme contenu en mémoire, sur l'ecran ou sur le fichier désigné.

Les lignes du programme à lister sont indiquées de la manière suivante:

LIST	ligne liste la ligne
LIST .	liste la ligne courante
LIST ligne1-ligne2	liste toutes les lignes de la ligne1 à la ligne2
LIST ligne1-	liste toutes les lignes de la ligne1 à la fin
LIST -ligne2	liste toutes les lignes du début jusqu'à la ligne2

Si le fichier est un fichier sur disque. LIST est équivalent à SAVE avec l'option ,A.
Si aucun fichier n'est précisé, le listage se fait sur l'écran.

Exemple:

```
LIST "LPT:",1000,1990      liste sur l'imprimante parallèle de la ligne 1000 à la ligne 1990
LIST "PG3.BAS"             liste tout le programme sur le fichier PG3.BAS
LIST                       liste tout le programme à l'écran
```

LOAD descripteur de fichier,R
commande

Charge en mémoire le programme dont on a précisé le nom et le support.

La commande LOAD ferme tous les fichiers ouverts, détruit le programme en mémoire puis effectue le chargement proprement dit. Toutes les variables sont détruites.
Avec l'option ,R, le programme chargé est aussitôt exécuté et les fichiers déjà ouverts avant le chargement restent ouverts. Le

périphérique pris par défaut est le lecteur 0, ou celui fixé par DEVICE.

Exemple:

LOAD "2:OTHELLO",R charge le programme OTHELLO de la disquette 2, laisse les fichiers ouverts et lance l'exécution
LOAD "TICTAC" charge le programme TICTAC

LOADM descripteur de fichier, translation ,R

instruction

Charge une zone mémoire en binaire à partir d'un fichier constitué par SAVEM ou par l'assembleur.

Le chargement s'effectue dans la banque courante.
Si une translation est indiquée, les données ou le programme sont décalés de cette translation. La translation décale aussi l'adresse d'exécution.
S'il s'agit d'un programme, l'option R en provoque l'exécution immédiate.

Exemple:

charge HORLOGE.BIN, le décale de 512 octets et lance son exécution à l'adresse donnée dans SAVEM + 512
LOADM "FORME" charge FORME.BIN

LOADP descripteur de fichier, élément de tableau ★

instruction

Charge dans le tableau l'image contenue dans le fichier spécifié.

Le résultat obtenu est similaire à celui de l'instruction GET.
Le tableau numérique doit avoir été déclaré au préalable et il doit être de type entier.

Plusieurs images peuvent être mémorisées dans un seul tableau. Elles sont systématiquement compactées et rangées depuis le bas du tableau, c'est-à-dire de l'indice le plus élevé vers les indices plus faibles.
Si le suffixe du fichier est absent, .MAP est pris par défaut.

Exemple:
Pour récupérer une image Colorpaint:

```
DIM IM%(2000)
LOADP"0:MONALISA",IM%(2000)
PUT (0,0),IM%(2000)
```

charge le fichier image MONALISA.MAP du lecteur 0 dans le tableau IM% à partir de l'élément 999. Si l'image occupe 400 octets, après le chargement, IM%(1000) contient 799.

LOC (n° de canal)

fonction

Indique où en est la lecture (ou l'écriture) dans le fichier ouvert sous le n° de canal considéré.

Si le fichier est à accès séquentiel, LOC indique le nombre de secteurs écrits ou lus depuis l'ouverture du fichier.
Si le fichier est à accès direct, LOC indique le numéro de l'enregistrement qui suit celui qui vient d'être lu par GET# ou écrit par PUT#.

Exemple:

LOC(2) position de la lecture ou de l'écriture en cours sur le fichier 2

LOCATE colonne, ligne, curseur

instruction

Fixe la position du curseur sur l'écran en mode caractère.

Le premier paramètre indique la colonne et doit être compris entre 0 et 39 (ou 79). Le second indique la ligne et doit être compris entre 0 et 24.
Le troisième paramètre rend le curseur invisible (valeur 0) ou visible (valeur 1).
Les trois paramètres sont facultatifs en Basic 128.
La position du curseur par LOCATE peut se faire en dehors de la fenêtre d'affichage définie par CONSOLE.

Exemple:

```
LOCATE 10,15: PRINT "BONSOIR"
```

LOF (n° de canal)
fonction

Donne la position du dernier enregistrement dans le fichier ouvert sous le n° de canal considéré.

Si le fichier est à accès séquentiel, LOF indique le numéro du dernier secteur du fichier. Si le fichier est à accès direct, LOF indique le numéro du dernier enregistrement.

Exemple:

```
LOF(1)      position du dernier enregistrement du fichier 1
```

LOG(nombre)
fonction

Donne le logarithme népérien du nombre (base e).

En Basic 128, le résultat est en double précision si l'argument est en double précision.

Exemple:

```
?LOG(5)
 1.60944
```

*LOOP **
instruction

Marque la fin d'une boucle et déroute l'exécution au DO correspondant.

(Voir l'instruction DO.)

LSET variable chaîne = chaîne de caractères
instruction

Dépose une chaîne de caractères dans une variable chaîne en cadrant à gauche.

La chaîne de caractères est déposée en commençant par la gauche. Si sa longueur est plus petite que la taille de la variable, le reste est complété à droite par des espaces; si sa longueur est trop grande, la fin de la chaîne est éliminée. Les nombres peuvent être déposés dans une variable chaîne après avoir été convertis par:
MKI$( ) pour un entier
MKS$( ) pour un nombre en simple précision
MKD$( ) pour un nombre en double précision *
Cette instruction est utilisée le plus souvent pour déposer des données dans une variable de champ définie par FIELD.
En Basic 1, la variable doit être une variable de champ.

Exemple:

```
LSET IDS = NOM$ + PR$      dépose dans IDS la concaténation de
NOM$ et PR$
LSET Q$ = MKS$(QU)         dépose la valeur numérique de QU après
conversion
```

Liste alphabétique des commandes, instructions et fonctions

MAX(liste de nombres) *
fonction

Retourne le maximum de la liste.

Les nombres peuvent être de précisions différentes.

Exemple :
```
?MAX(10,12,5.6#)
12
```

MERGE descripteur de fichier,R
commande

Charge en mémoire le programme indiqué et l'ajoute au programme résident.

Les lignes du programme chargé sont classées avec celles du programme résidant et les recouvrent si elles sont identiques. Le programme à charger doit avoir été sauvegardé avec l'option A. L'exécution du programme global est lancée à la fin du chargement avec l'option R.

Exemple :
```
MERGE "1:TEST",R
```
charge le programme TEST de la disquette 1, le fusionne avec le programme en mémoire et lance l'exécution.
```
MERGE "MENU"
```
fusionne le programme MENU au programme résident

MID$ (chaîne, position, longueur)
fonction

Extrait une sous-chaine de longueur donnée d'une chaine donnée.

La position est un nombre compris entre 1 et 255, la longueur est un nombre compris entre 0 et 255.
Si la longueur est trop grande ou si ce paramètre est absent, la sous-chaine va jusqu'à la fin de la chaine.
Si la position est supérieure à la longueur de la chaine, le résultat est une chaine vide.

Exemple :
```
?MID$("PATATRAC",3,2)
TA
?MID$("PATATRAC",5)
TRAC
```

MID$ (variable chaîne , début, longueur) = chaîne de caractères
instruction

Permet de remplacer une partie d'une variable chaine par une autre chaîne.

La longueur de la variable chaîne reste inchangée dans tous les cas.
Le 2e paramètre indique la position du 1er caractère a remplacer dans la variable.
Le 3e paramètre, facultatif, indique la longueur de la partie a remplacer.
— Si la chaine de remplacement est plus longue que la partie à remplacer, seul le début de cette chaîne est utilisé.

Exemple :
```
10 A$="MICRO-ORDINATEUR"
20 MID$(A$,6,7)="PROCESSEUR"
30 PRINT A$
RUN
MICROPROCESSEUR
```

— Si la longueur n'est pas précisée, le remplacement se fait jusqu'à la fin de la variable, si la chaîne modifiante est assez longue.

Exemple:

```
10 A$="SAINT-LOUIS"
20 MID$(A$,7)="DENISE"
30 PRINT A$
RUN
SAINT-DENIS
```

Cette instruction peut être utilisée pour modifier la valeur d'une variable de champ définie dans FIELD.

MIN(liste de nombres) *

fonction

Retourne le minimum de la liste.

Les nombres peuvent être de précisions différentes.

Exemple:

```
?MIN(12,3.4,5.6)
 3.4
```

MKD$ (nombre) *

fonction

Assure la conversion d'un nombre en double précision en chaîne de caractères, pour qu'il puisse être déposé par LSET ou RSET dans une variable chaîne.

La longueur de la chaîne de caractères obtenue est de huit octets. Cette chaîne de caractères est codée et non lisible directement.

Exemple:

```
V$=MKD$(TOTAL#)
V$(1)=1 234567890123#
```

MKI$ (nombre)

fonction

Assure la conversion d'un nombre entier en chaîne de caractères, pour qu'il puisse être déposé par LSET ou RSET dans une variable chaîne.

La longueur de la chaîne obtenue est de deux octets.

Exemple:

```
MK$=MKI$(%)
MK$=-12536!
```

MKS$ (nombre)

fonction

Assure la conversion d'un nombre en simple précision en chaîne de caractères pour qu'il puisse être déposé par LSET ou RSET dans une variable chaîne.
La longueur de la chaîne obtenue est de quatre octets.

Exemple:

```
V$=MKS$(X)
V$=6.023E-23!
```

MOTOR ON / MOTOR OFF

instructions

Comme son nom l'indique, met en marche le moteur du magnétophone à cassettes ou l'arrête.

Exemple:

```
MOTOR ON
```

MTRIG(numéro) *

fonction

Rend l'état du bouton de la souris.

Le numéro désigne la gachette: 0 (gauche)
1 (droite)
Le résultat est –1 (VRAI) si la gachette est enfoncée ou 0 (FAUX) sinon.

Exemple:

```
IF MTRIG(0) THEN PRINT "BOUM"
```

NAME descripteur de fichier 1 AS descripteur de fichier 2.

commande

Permet de changer le nom d'un fichier sur disquette.

Descripteur du fichier 1 comporte le nom du fichier à renommer. Descripteur de fichier 2 contient le nouveau nom du fichier. Le nouveau nom ne doit pas correspondre à un fichier déjà existant. En Basic 128, un commentaire peut être indiqué avec le nom du fichier.

Exemple:

```
NAME "FANTIC.BAS" AS "FANTAC.BAS"
NAME "TEXTE.DAT" AS "notes:OLD_TXT.DAT"
```

NEXT liste de variables

instruction

Incrémente ou décrémente les variables de contrôle d'une ou plusieurs boucles FOR...NEXT.

(Pour le fonctionnement de cette instruction, voir FOR.)

NEW

commande

Efface le programme présent en mémoire et détruit tous les variables, y compris celles réservées dans COMMON.

Cette commande ne ferme pas les fichiers ouverts et ne modifie pas les options fixées au préalable par CLEAR.

Exemple:

```
NEW
```

OCT$(nombre) *

fonction

Rend une chaîne de caractères exprimant le nombre en octal (base 8).

Si l'argument n'est pas entier, il est tronqué.

Exemple:

```
?OCT$(30)
36
```

ON ERROR GOTO n° de ligne

instruction

Permet le branchement à la ligne indiquée si, dans le déroulement de l'exécution d'un programme, une erreur est détectée.

La séquence de traitement des erreurs est alors effectuée à partir de la ligne indiquée et les interruptions par ON INTERVAL sont suspendues.
Cette séquence doit se terminer normalement par RESUME.
L'instruction ON ERROR GOTO est placée soit en début de programme, soit au début des séquences où une erreur est susceptible de se produire.
ON ERROR GOTO 0 annule l'effet de ON ERROR GOTO n° et ramène à l'état normal: message d'erreur et arrêt de l'exécution.
L'instruction CLEAR annule également l'effet de ON ERROR GOTO n°.

Exemple:

```
10 ON ERROR GOTO 100
20 INPUT X
30 PRINT 1/X
40 ON ERROR GOTO 0
50 END
100 IF ERR=11 THEN PRINT "Division impossible"
110 RESUME 20
```

ON expression GOSUB liste de n° de ligne
ON expression GOTO liste de n° de ligne

instructions

Permet le branchement à une ligne ou à un sous-programme dont la ligne est déterminée par la valeur de l'expression.

L'expression est évaluée et sa valeur, après arrondi à l'entier le plus proche, donne le rang dans la liste du sous-programme à appeler. Si par exemple cette valeur est 4, l'exécution passe à la ligne ou au sous-programme indiqué par le 4° numéro.
Si la valeur de l'expression est nulle ou supérieure au nombre de numéros de ligne de la liste, l'exécution se poursuit à l'instruction suivante.
Les numéros de lignes correspondant aux valeurs de l'expression inutilisées peuvent être omis.

Exemple:

```
10 INPUT "Numero ";N
20 ON N GOSUB 100,200
30 END
100 PRINT "SOUS-PROGRAMME 1"
110 RETURN
200 PRINT "SOUS-PROGRAMME 2
210 RETURN
```

suivant que l'on répond 1 ou 3,
le sous-programme correspondant est exécuté

Exemple:

```
10 INPUT "Nombre ";N
20 ON SGN(N)+2 GOTO 100,200
30 PRINT "Positif"
40 END
100 PRINT "Negatif"
110 END
200 PRINT "Nul"
210 END
```

ON INTERVAL = nombre GOSUB n° de ligne ★
ON INTERVAL = nombre GOTO n° de ligne

instructions

Permet de placer une interruption logicielle sur l'horloge interne.

Le nombre indique en dixièmes de seconde la valeur de l'horloge qui déclenche l'interruption.
L'exécution de INTERVAL ON lance le comptage sur l'horloge interne. L'horloge compte en dixièmes de seconde. Quand le nombre est atteint, l'exécution du programme est suspendue et la séquence d'instructions commençant au numéro indiqué est exécutée.
Si GOSUB est utilisé, la séquence doit se terminer par RETURN qui reprend l'exécution là où elle a été suspendue.
L'exécution de INTERVAL OFF arrête le comptage et par conséquent les interruptions.
L'exécution de la séquence d'instructions peut être interrompue par une autre interruption.
Si la séquence d'instructions est assez longue, il faut exécuter INTERVAL OFF au début pour ne pas être interrompu en cours d'exécution.

Exemple:

```
10 ON INTERVAL=1800 GOTO 100
20 INTERVAL ON
30 DO: LOOP
100 PRINT "C'EST OUT"
110 END
```

Le message est affiché trois minutes exactement après l'exécution de la ligne 20

ON KEY=caractère GOSUB n° de ligne ★
ON KEY=caractère GOTO n° de ligne
instructions

Permet de définir des interruptions logicielles sur la frappe d'un caractère.

Le caractère est indiqué par une constante d'un seul caractère ("A") ou par le premier caractère d'un constante ("AZERTY") ou par son code ASCII (65).
Après l'exécution de cette instruction, l'appui sur la touche correspondant au caractère provoque la suspension de l'exécution et le débranchement à la ligne de numéro indiqué.
Le retour à l'endroit de l'interruption se fait par RETURN si GOSUB est utilisé.
Il est possible de spécifier dix touches différentes avec cette instruction.
Les touches spécifiées ne peuvent plus être lues par INKEY$.
Si le numéro de ligne est égal à 0, l'interruption définie est rendue inactive pour la touche considérée.

Exemple:

```
10 ON KEY="?" GOSUB 100
20 DO: PRINT "N'importe quoi" : LOOP
30 END
100 PRINT "Voici quelques explications"
110 RETURN
```

ONPEN GOSUB liste de n° de ligne ★
ONPEN GOTO liste de n° de ligne
instructions

Permet un branchement conditionnel (GOTO) ou un appel de sous-programme (GOSUB) dépendant de la zone visée par le crayon optique.

Les zones de l'écran doivent avoir été définies au préalable par l'instruction PEN.
L'exécution de ONPEN se déroule de la manière suivante:
— test du contact du crayon optique,
— lorsque le contact est fermé, lecture de la zone visée,
— branchement à la ligne dont le numéro a le même rang dans la liste que celui de la zone visée dans l'instruction PEN.

Le rang dans la liste est compté à partir de 0.
Si le crayon optique vise un point situé hors des zones définies, l'exécution se poursuit en séquence.

Exemple:

```
10 CLS
20 PEN 0,1,0,1,5,4,30,35,1
30 PEN 1,1,40,1,5,4,60,35,1
40 PEN 2,1,70,1,5,4,90,35,1
50 ONPEN GOSUB 100,200,300
70 END
100 PRINT "ZONE 1"
110 RETURN
200 PRINT "ZONE 2"
210 RETURN
300 PRINT "ZONE 3"
310 RETURN
```

OPEN "I", # n° de canal, descripteur de fichier.
OPEN "O", # n° de canal, descripteur de fichier
instructions

Ouvre un fichier séquentiel en lecture ou en écriture sous le n° de canal indiqué.

Toutes les opérations sur ce fichier devront se faire sous ce n° de canal. Le n° de canal doit être compris entre 1 et 16. Le périphérique pris par défaut est le lecteur 0 ou celui fixé par DEVICE.

OPEN "I" ouvre le fichier en lecture. Un même fichier peut être ouvert en lecture sous plusieurs canaux simultanément.

Exemple:

OPEN "I",#2,"1:RESUL.DAT" ouvre en lecture le fichier RESUL-.DAT sur le lecteur 1 sous le numéro 2

OPEN "O" ouvre le fichier en écriture. Un même fichier ne peut être ouvert qu'une fois en écriture. Sur disque, l'ouverture déclenche l'initialisation du fichier; si un fichier existait déjà sous le même nom, il est supprimé.

Exemple:

OPEN "O",#1,"2:TELEPH.DAT" ouvre en écriture le fichier TELEPH.DAT sur le lecteur 2 sous le numéro 1
OPEN "O",#3,"LPRT:" ouvre en écriture le fichier numéro 3 sur l'imprimante parallèle.

OPEN "D", # n° de canal, descripteur de fichier, longueur
OPEN "R", # n° de canal, descripteur de fichier, longueur
instructions

Ouvre le fichier à accès direct dont le nom est indiqué, en lecture et en écriture, sous le n° de canal précisé.

Le n° de canal doit être compris entre 1 et 16.
La longueur, optionnelle, fixe la longueur de l'enregistrement. Cette longueur doit être inférieure ou égale à la longueur indiquée dans FILES. Si la longueur n'est pas précisée dans OPEN, l'enregistrement occupe alors un secteur, soit 128 octets en simple densité ou 255 octets en double densité.
Le périphérique pris par défaut est le lecteur 0 ou celui fixé par DEVICE.
On ne peut ouvrir de fichier à accès direct que sur disquette. Les deux formes OPEN "D", et OPEN "R",... sont équivalentes.

Exemple:

OPEN "D",#2,"1:AGENDA.OLD",32 ouvre le fichier à accès direct AGENDA.OLD sous le numéro 2, avec une longueur d'enregistrement de 32 caractères.

PAINT (colonne, ligne), couleur ★

instruction

Remplit avec le motif en cours une zone de couleur homogène en partant du point situé en (colonne, ligne) avec la couleur précisée.

Si la couleur n'est pas précisée, c'est la couleur courante qui est prise par défaut.
Le remplissage se fait avec le motif défini par l'instruction PATTERN.
Par défaut, le remplissage est uniforme.
Dans tous les cas, le remplissage ne modifie qu'une seule couleur, forme ou fond suivant le signe.

Exemple:

```
PAINT (100,125),3
```
remplissage à partir du point 100,125 par la couleur jaune

PAINT (colonne, ligne), couleur ♦

instruction

Remplit une courbe fermée en partant du point situé en (colonne, ligne) avec la couleur précisée. La couleur doit être la même que celle de la courbe qui délimite la surface à "peindre".

Le dernier paramètre, moiré, qui devra avoir pour valeur 0 ou 1 permet de peindre avec moirage si la valeur donnée est 1. Dans ce cas un point sur deux seulement est allumé, ce qui donne un effet de grisé. Si ce paramètre est omis, c'est la valeur courante qui est prise par défaut. Sa valeur initiale est 0.

Exemple:

```
PAINT (100,125),3
PAINT (30,30)
```

Voir STEP.

PALETTE (numéro) ★

fonction

Rend la valeur réelle de la couleur de code indiqué.

Le résultat est un nombre compris entre −4096 et 4095. Ce nombre correspond à la couleur réelle associée au numéro indiqué.
Si le nombre est négatif, la couleur correspondante est transparente à l'incrustation.

Exemple:

```
?PALETTE(3)
748
```

PALETTE numéro, couleur réelle, incrustation ★

instruction

Affecte une couleur réelle au numéro de couleur indiqué.

Il existe 4 096 couleurs réelles (de 0 à 4095). Seules seize d'entre elles sont utilisables simultanément.
Cette instruction permet d'affecter une couleur réelle (nombre compris entre 0 et 4095) à une couleur utilisable (numéro compris entre 0 et 15).
Le deuxième paramètre peut être indiqué en code positif (couleur de forme) ou négatif (couleur de fond).
Le troisième paramètre, facultatif, indique si la couleur ainsi définie doit être transparente à l'incrustation (valeur 1) ou non (valeur 0). Si ce paramètre est absent, la couleur conserve son statut antérieur.

Exemple:

```
PALETTE 3,748,1
```
affecte la couleur 748 au numéro 3 et la rend transparente à l'incrustation

```
PALETTE 14,2970
```
affecte la couleur 2970 au numéro 14

PATTERN caractère ⋆

instruction

Fixe le motif de remplissage.

Le caractère qui servira de motif peut être donné sous forme de chaîne de caractères (c'est le premier caractère de la chaîne qui est pris) ou par son code ASCII. Si le code ASCII est supérieur à 127, il s'agit alors d'un caractère graphique défini par l'utilisateur.
Le motif est utilisé par la suite dans toutes les instructions de remplissage: BOXF, CIRCLEF, PAINT.

Exemple:

```
le caractère / servira de motif
le 2e caractère graphique servira de motif
   ici également
```

PEEK(adresse)

fonction

Retourne le contenu de l'octet situé à l'adresse indiquée.

L'adresse est un nombre compris entre --65536 et 65535.
Le résultat est un entier compris ente 0 et 255.
En Basic 128, si l'adresse est comprise entre &H6000 et 1H9FFF, l'octet est celui de la banque courante, c'est-à-dire celle fixée par BANK ou, par défaut, la banque la plus élevée.

Exemple:

```
BANK 1 : PRINT PEEK(&H6100)
```

donne le contenu de l'octet situé en &H6100 dans la banque 1.

PEN liste de zones ⋆ *avec liste de zone = numéro,(colonne1,ligne1)-(colonne2,ligne2)* ,...

instruction

Définit des zones rectangulaires sur l'écran et leur attribue un numéro pour l'utilisation de ONPEN.

Les numéros de zones sont compris entre 0 et 7.
Les zones peuvent être définies en mode caractère ou en mode graphique.

• mode caractère:
Chaque zone correspond à un seul caractère; elle est définie par un seul point dont les coordonnées sont données en caractères: colonnes de 0 à 39 et lignes de 0 à 24:

Exemple:

```
PEN 3,30,15),4,32,15)
```

définit les zones 3 et 4 en mode caractère

• mode graphique:
Chaque zone est définie par les deux sommets opposés du rectangle comme dans l'instruction BOX. Les coordonnées sont exprimées en graphique: colonnes de 0 à 319 et ligne de 0 à 199.

Exemple:

```
PEN 0,110,10)-(40,40),1,(50,10)-(80,40)
```

définit deux zones numérotées 0 et 1

Si un numéro de zone est défini sans être suivi des coordonnées, la zone est supprimée et ne peut plus être sélectionnée par ONPEN.
Une instruction PEN sans argument supprime toutes les zones définies.
Si deux zones ont une partie commune, lorsque le crayon optique vise cette partie commune, ONPEN effectue le branchement correspondant à la zone de plus petit numéro.

PLAY liste de chaîne de caractères
Instruction

Joue la musique définie dans les chaînes de caractères.

Les chaînes de caractères ne peuvent comporter que des notes, des silences ou des attributs.
— notes : DO RE MI FA SO LA SI suivies de # ou b si l'on veut un dièse ou un bémol,
— silence : P
— attributs : ce sont l'attaque, la longueur, l'octave et le tempo.

Signification	Attribut	Domaine	Valeur par défaut
amortissement	A	0-255	0 note continue
longueur	L	1-96	24 noire
octave	O	1-5	4
tempo	T	1-255	5

Amortissement :
A0 correspond à un son continu, A225 à un son très amorti.
longueur :
L96 correspond à une ronde, L48 à une blanche,... , L3 à une triple croche .
Octave :
O1 est l'octave la plus grave, O5 la plus aiguë.
Tempo :
T1 est le tempo le plus rapide, T255 le plus lent.
Un attribut A, L, O, T agit sur toutes les notes qui suivent jusqu'à ce qu'un nouvel attribut vienne modifier sa valeur.
Les espaces sont autorisés à n'importe quel endroit dans la chaîne ; le point-virgule peut servir à séparer les notes.

L'interruption d'une séquence PLAY en BASIC 128 peut nécessiter l'appui simultané sur les touches CNT et C pendant plusieurs secondes.

Exemple :

```
10 A$="O5 DOREMI DOREPREV FAFAMIPDOREMIDOREPREMIFASODO"
20 PLAY A$,A$
```

POINT (colonne,ligne)
fonction

Donne le numéro de la couleur du point indiqué.

Les coordonnées sont indiquées en graphique : colonne de 0 à 319 et ligne de 0 à 199.
Le numéro est positif si le point appartient à la forme, négatif s'il appartient au fond.

POKE adresse, nombre
instruction

Dépose le nombre dans l'octet situé à l'adresse indiquée.

Le nombre doit être compris entre 0 et 255.
L'adresse doit être comprise entre - 65536 et 65535.
En Basic 128, si l'adresse est comprise entre &H6000 et &H9FFF, l'octet est celui de la banque courante c'est-à-dire celle fixée par BANK ou, par défaut, la banque la plus élevée.
Cette instruction est interdite sur la mémoire morte.

POS (n° de canal)
fonction

Donne la position du pointeur sur la ligne.

La position la plus à gauche est notée 0.
Le numéro de canal indique le périphérique concerné.
Si ce numéro est égal à 0 ou s'il est omis, la fonction renvoie la position du curseur sur l'écran (nombre compris entre 0 et 39 ou 79).
La fonction CSRLIN permet de connaître le numéro de la ligne où se trouve le curseur.

PRINT liste de données

instruction

Affiche sur l'écran les données dans l'ordre de la liste.

Le mot clé PRINT peut être remplacé par un point d'interrogation.
L'affichage commence à l'endroit où se trouve le curseur.
Un point-virgule permet l'affichage de la donnée suivante juste après la précédente.
Une virgule permet l'affichage de la donnée suivante au début de la tabulation suivante. Les tabulations fixes sont espacées de treize caractères à partir du caractère 0.
Si la liste de données ne se termine pas par une virgule ou un point-virgule, l'affichage se termine par le passage à la ligne suivante.
Si les données occupent plus de 40 (ou 80) caractères, l'affichage se poursuit sur la ligne suivante.
Si une donnée doit être affichée en fin de ligne et qu'il ne reste pas assez de place, l'affichage se fait automatiquement sur la ligne suivante. Dans ce cas, les minuscules accentuées sont comptées comme trois caractères.
Les nombres sont toujours suivis d'un espace. Les nombres positifs sont précédés d'un espace, les nombres négatifs du signe moins.
L'affichage se fait avec exposant si la valeur du nombre est en dehors de l'intervalle 1E−6,1E+6 pour la simple précision, ou 1D-16 1D+16 pour la double précision.
Les chaînes de caractères sont affichées telles quelles.
Avec des caractères en double hauteur, le retour à la ligne provoque un double interligne. Cependant, le premier affichage se fait sur la ligne du curseur et sur la ligne supérieure. Il est donc souvent nécessaire de le faire précéder d'un PRINT vide.

Exemple:

```
10 X=1
20 PRINT X,X+1,X+2
30 PRINT X;X+1;X+2
40 PRINT "X=";X
```

PRINT # n° de canal, liste de données

instruction

Ecrit les données (variables ou constantes) dans le fichier dont on précise le n° de canal.
Les données sont enregistrées de la même manière qu'à l'écran avec PRINT (espaces, séparateurs,...).
Le fichier peut être séquentiel ou à accès direct. Dans ce dernier cas, PRINT # écrit dans le tampon d'enregistrement. Le transfert de l'enregistrement dans le fichier doit se faire par PUT #.

Exemple:

```
PRINT #2, A$, N
PRINT #1, "ARTHUR
```

PRINT USING chaîne de caractères;liste de données

instruction

Permet des affichages selon un format décrit dans la chaîne de caractères.

Les données de la liste sont séparées par des virgules ou des points-virgules.
La chaîne de caractères contient des caractères qui précisent le format d'affichage. Leur signification est la suivante:

format des chaînes de caractères ★

! signifie que seul le premier caractère de la chaîne doit être affiché.
% % signifie que la longueur exacte de la zone d'affichage de la chaîne est la même que celle comprise entre les deux % compris. La chaîne est cadrée à gauche à l'affichage, les caractères en trop étant éliminés ou l'espace restant étant complété par des espaces.
& signifie que la chaîne doit être affichée telle quelle. Ceci permet d'éviter des sauts de ligne pour une chaîne trop longue.

Exemple:

```
10 A$="QU'IMPORTE LE FLACON"
20 PRINT USING "!";A$
30 PRINT USING "%%";A$
40 PRINT USING "&";A$
RUN
Q
QU'IMPORTE
QU'IMPORTE LE FLACON
```

format des nombres

représente la position de chaque chiffre. L'affichage d'un nombre est cadré à droite. Si la zone est trop petite le dépassement est indiqué par l'affichage du caractère % en tête du nombre. Si la zone est trop grande, le reste est complété par des espaces.
. représente le point décimal. Il peut être placé n'importe où pour séparer la partie entière de la partie décimale. Si nécessaire, les nombres sont arrondis avant affichage.

Exemple:

```
10 A$=###.##
20 PRINT USING A$;12
30 PRINT USING A$;-12.345
40 PRINT USING A$;.1234
50 PRINT USING A$;1234.5678"
RUN
12
-12.35
0.12
%1234.57
```

+ placé en début ou en fin de format signifie que le signe + doit toujours précéder ou suivre un nombre positif.

− placé en fin de format signifie que le signe − doit toujours suivre un nombre négatif.

Exemple:

```
PRINT USING "+##.#";12.34
+12.3
PRINT USING "##.#-";  12.34
12.3-
```

** placé en tête de format signifie que les espaces en tête doivent être remplacés par des astérisques.

Exemple:

```
PRINT USING "**##.#";1.23
**  .2
```

$$ placé en tête de format signifie que le signe dollar doit être affiché immédiatement à gauche du nombre.

Exemple:

```
PRINT USING "$$##.#";1.23
 $1.2
```

**$ placé en tête de format combine les deux effets précédents.

Exemple:

```
PRINT USING "**$##.#";1.23
***$1.2
```

, situé à gauche du point décimal signifie que les puissances de 1000 doivent être séparées par une virgule dans la partie entière.

Exemple:

```
PRINT USING "#######, ##";1234567
1,234,570.00
```

^^^^ placés en fin de format signifie que la notation exponentielle doit être utilisée.

Exemple:

```
PRINT USING "##.##^^^^";123.45
1.23E+02
```

format général *
Le format doit contenir autant de formats élémentaires qu'il y a de données à afficher. Les formats élémentaires peuvent être séparés par des constantes qui seront affichées entre les données.
Si le format général ne contient pas assez de formats élémentaires, il est repris au début pour l'affichage de la suite des données.

Exemple:

```
PRINT USING "J'AI VENDU ### LOG CIELS A ### ## FRANCS";53;499.50
J'AI VENDU 53 LOG CIELS A 499.50 FRANCS
PRINT USING "##.#";1.23;2.34;3.45
1.2
2.3
3.5
```

PRINT # n° de canal, USING chaîne de caractères ; liste de données
instruction

Fonctionne comme PRINT #, mais les données sont écrites suivant le format précisé par USING.
Les données sont enregistrées de la même manière qu'à l'écran avec PRINT USING.

Exemple:

```
PRINT #1, USING "### ##";120.5
```

PSET (colonne,ligne) ,caractère,couleur,fond ,inversion
PSET (colonne,ligne) ,couleur
instruction

Marque un point situé en colonne,ligne.
Cette instruction fonctionne en mode caractère quand l'argument caractère est présent et en mode graphique quand il est absent.
En mode caractère, la colonne est un nombre de 0 à 39 (ou 79) et la ligne un nombre de 0 à 24.
En mode graphique, colonne et ligne sont des nombres compris entre −32768 à 32767 en Basic 128. En Basic 1, colonne est un nombre compris entre 0 et 319 et ligne un nombre compris entre 0 et 199.
La suite de la syntaxe est identique à celle de BOX.

Exemple:

```
PSET(10,20),"X",3,0
```
marque le caractère X en colonne 10 ligne 20, de couleur jaune sur fond noir

```
PSET(55,115)
```
marque le point graphique situé en 55,115 avec la couleur courante

Voir STEP.

PTRIG
fonction

Rend la valeur VRAI (−1) si le contact du crayon optique est fermé et la valeur FAUX (0) sinon.

*PUT (colonne , ligne) ,élément de tableau **
instruction

Restitue à l'écran la portion d'image conservée dans le tableau à partir de l'élément indiqué.

La portion d'image est obligatoirement composée d'un nombre entier de caractères. Les coordonnées du sommet sont donc données en caractères : colonne de 0 à 39 (ou 79) et ligne de 0 à 24. Le sommet indique le coin supérieur gauche de l'image.
L'image doit avoir été mémorisée dans le tableau numérique entier par une instruction GET préalable, ou chargée dans le tableau par une instruction LOADP.
La portion d'image n'est pas affichée si elle est trop grande.

Exemple :

PUT (10,12), M%(300) restitue la portion d'image conservée dans le tableau à partir de l'élément 299 , au point situé en colonne 10 ligne 12

PUT (colonne 1, ligne 1)–(colonne 2, ligne 2), ♦ tableau

instruction

Restitue à l'écran contenue dans le tableau, dans le rectangle dont on donne deux sommets opposés : colonne 1, ligne 1 et colonne 2, ligne 2. L'image mémorisée dans le tableau numérique par une instruction GET préalable doit correspondre aux mêmes dimensions (nombre de points en colonne et en ligne).

PUT # n° de canal, n° d'enregistrement

instruction

Transfère un enregistrement d'un fichier à accès direct (dont on donne le n° de canal) de la zone tampon vers la disquette.
Le contenu de l'enregistrement doit avoir été déposé par WRITE# ou PRINT#, ou directement dans les variables de champ par LSET, RSET ou MID$()=. Si le numéro d'enregistrement n'est pas précisé, PUT # écrit dans l'enregistrement suivant du fichier, ou dans le 1er si aucun enregistrement n'a été lu ou écrit.

Exemple :
PUT #1,12 dépose le douzième enregistrement du fichier n° 1

READ liste de variables

instruction

Lit les données des instructions DATA et les affecte aux variables de la liste.
Les variables doivent être de même type que les données à lire, sinon une erreur "Syntax Error" est détectée.
Une seule instruction READ peut lire plusieurs lignes de DATA et inversement, une ligne de DATA peut être lue par plusieurs instructions READ.
La première instruction READ commence par lire les données de la première instruction DATA du programme. La suite des lectures se fait en séquence.
S'il n'y a plus assez de données, une erreur "Out of Data" apparaît.
Il est possible de commencer ou de recommencer la lecture à une ligne donnée par une instruction RESTORE.

Exemple :

```
10 READ NOM$,PR$,AGE
20 PRINT NOM$,PR$,AGE
100 DATA MARTIN, LUCIEN, 77
```

REM texte

instruction

Permet d'incorporer le texte de remarques, titres, commentaires,... dans un programme.
Le mot clé REM peut être remplacé par une apostrophe.
Tout ce qui suit REM jusqu'à la fin de la ligne est ignoré par l'interpréteur mais apparaît dans le listage du programme.
Il est possible de brancher à une ligne contenant un REM, l'exécution se fait à la première ligne qui suit.
Des commentaires peuvent être ajoutés en fin de ligne en les séparant des instructions par une apostrophe.

Exemple:

```
10 REM Titre du programme : jan 1984
20 PRINT "Bonjour" : message de bienvenue
30 A=2+2 : calcul
40 PRINT "Bonsoir" : REM message de politesse
```

RENUM nouveau n°, 1re ligne, ancien n°, 1re ligne, ancien n°, pas dernière ligne

commande

Rénumérote un programme ou une partie de programme.
Toutes les indications sont optionnelles et possèdent des valeurs par défaut.
Le premier nombre indique le nouveau numéro (10 est pris par défaut).
Le deuxième nombre indique à partir de quel numéro (ancien) il faut commencer à renuméroter (par défaut, c'est la première ligne du programme).
Le troisième nombre indique l'ancien numéro de ligne de fin de la renumération (par défaut, la fin du programme).
Le quatrième nombre indique l'écart entre chaque nouvelle ligne (par défaut 10).
RENUM change en conséquence, et dans tout le programme, tous les numéros des lignes renumérotées dans les branchements et les appels de sous-programmes. Si dans la ligne L1 (ancien numéro), il est fait appel à une ligne L2 inexistante, le message suivant apparaît:
Undefined line L2 in L1
dans lequel L1 est le numéro de ligne renumérotée.
Les virgules sont indispensables, sauf les dernières.
La renumérotation d'une partie de programme n'est possible que si les nouvelles lignes restent à l'intérieur de l'espace laissé par les lignes inchangées: le nouveau n° de la 1re ligne renumérotée doit rester plus grand que le n° de la ligne inchangée qui le précède, et le nouveau n° de la dernière ligne renumérotée doit rester plus petit que le n° de la ligne inchangée qui suit.

Exemple:

```
RENUM 1500, 1000, 1340, 5      renumérote la partie de programme
```
comprise entre 1000 et 1340, avec un pas de 5, en commençant à 1500
```
RENUM 100, 10      renumérote le programme à partir de 10 avec un
```
pas de 10, en commençant à 100
```
RENUM      renumérote tout le programme avec un pas de 10 en
```
commençant à 10

RESET ★

instruction

Initialise le micro-ordinateur.

Son effet est de ramener au menu initial.

Exemple:

```
RESET
```

RESTORE n° de ligne

instruction

Permet de relire des constantes dans une instruction DATA d'une ligne donnée.

Après l'instruction RESTORE, le premier READ prend ses valeurs dans le premier DATA rencontré à partir de cette ligne.
Si aucun numéro de ligne n'est indiqué, le premier READ prend ses valeurs à partir du premier DATA du programme.

Exemple:

```
10 READ A,B
```

```
20 PRINT A,B
30 RESTORE
40 READ A
50 PRINT A
100 DATA 12,23,34
```

RESUME n° de ligne
RESUME NEXT
instruction

Permet le retour au programme principal après une séquence de traitement d'erreur appelée par ON ERROR GOTO.

L'exécution reprend au numéro de ligne indiqué s'il est présent ou à l'instruction qui a provoqué l'erreur si rien n'est indiqué.
Si l'option NEXT est présente, l'exécution reprend à l'instruction qui suit celle qui a provoqué l'erreur.
Si cette instruction est rencontrée sans appel par ON ERROR GOTO, une erreur "Resume without Error" est détectée.

Exemple:

```
10 ON ERROR GOTO 100
20 INPUT X
30 PRINT SQR(X)
40 ON ERROR GOTO 0
50 END
100 IF ERR=5 THEN "Nombre négatif"
110 RESUME 20
```

RETURN
instruction

Retour au programme principal après un appel par GOSUB.

(Voir instruction GOSUB.)

RIGHT$(chaîne de caractères, longueur)
fonction

Donne la partie droite de la chaîne sur une longueur précisée.

Si la longueur est nulle, le résultat est la chaîne vide.
Si la longueur demandée est supérieure à la longueur de la chaîne, le résultat est la chaîne entière.

Exemple:

```
? RIGHT$("MELODRAME",5)
DRAME
```

RND[(numérique)]
fonction

Donne un nombre réel, pseudo-aléatoire, compris entre 0 et 1 exclus.

Les nombres obtenus avec cette fonction sont différents et répartis uniformément dans l'intervalle ouvert 0,1 si l'argument de la fonction est un nombre positif ou s'il n'y a pas d'argument.
La séquence des nombres générés est réinitialisée à chaque exécution du programme.
Un argument négatif réinitialise la séquence à partir d'une valeur qui dépend de l'argument.

Exemple:

```
10 DO
20  FOR = 1 TO 5
30  PRINT RND
40  NEXT I
50  PRINT RND:2:
60 _LOOP
RUN
```

ROT ★
fonction

Rend l'orientation de la tortue active.

Le résultat est un nombre compris entre 0 et 255. Le sens positif est celui des aiguilles d'une montre.
La tortue active est la dernière fixée par TURTLE.

Exemple:

```
? ROT
167
```

ROT TO nombre ★
instruction

Fixe ou modifie l'orientation de la tortue active.

Si l'option TO est présente, l'orientation est fixée de façon absolue, sinon elle est modifiée relativement à sa valeur antérieure.
La valeur absolue du nombre est comprise entre 0 et 255. Si cette valeur est plus grande, seul le reste de sa division par 256 (modulo) est pris en compte.
Un nombre positif signifie vers la droite, un nombre négatif signifie vers la gauche.

La tortue active est la dernière fixée par TURTLE.

Exemple:

```
ROT TO -32      fixe l'orientation à -45° vers la gauche
ROT 64          modifie l'orientation de 90° vers la droite
ROT 320         en fait autant
```

RSET variable chaîne = chaîne de caractères
instruction

Dépose une chaîne de caractères dans une variable chaîne en la cadrant à droite.

La chaîne de caractères est déposée en commençant par la droite. Si sa longueur est plus petite que la longueur de la variable, le reste est complété à gauche par des espaces; si sa longueur est trop grande, la fin de la chaîne est éliminée. Les nombres peuvent être déposés dans une variable chaîne après avoir été convertis par:
MKI$( ) pour un nombre entier
MKS$( ) pour un nombre en simple précision
MKD$( ) pour un nombre en double précision.
Cette instruction est utilisée le plus souvent pour déposer des données dans une variable de champ définie dans FIELD.
En Basic 1, la variable doit être une variable de champ.

Exemple:

```
RSET DS = NOM$+PR$      dépose dans la variable ID$ le contenu des
deux chaînes NOM$ et PR$ en cadrant le tout à droite
RSET QS = MKI$(1234)    dépose dans Q$ le nombre 1234 après
conversion
```

RUN n° de ligne
RUN descripteur de fichier, R
commande

Lance l'exécution, après chargement éventuel, du programme résident en mémoire.

Si un n° de ligne est indiqué, l'exécution du programme résident commence à la ligne donnée.
Si un fichier est indiqué, le programme qu'il contient est chargé en mémoire puis exécuté à partir du début.
Avec l'option R, les fichiers déjà ouverts avant le lancement restent ouverts.

Exemple:

RUN 100 lance l'exécution en ligne 100
RUN "1 OTHELLO", R charge le programme OTHELLO et l'exécute à partir du début tout en conservant tous les fichiers ouverts.

SAVE descripteur de fichier
SAVE descripteur de fichier, A
SAVE descripteur de fichier, P
instruction

Sauvegarde, sous le nom et sur le support indiqués, le programme présent en mémoire centrale.

Sur disque, si un fichier existait déjà sous le même nom, il est remplacé par le nouveau programme.
Le suffixe par défaut est BAS.
Avec l'option A, le programme est sauvegardé sous une forme littérale, identique au listage à l'écran. Il pourra alors être utilisé par MERGE.

Avec l'option P (protégé), le programme est sauvegardé sous une forme codée. Au chargement suivant, le programme ne pourra être ni listé ni modifié. Il est conseillé de le sauvegarder également sous la forme habituelle.

Exemple:

SAVE "CASS:OTHELLO", P sauvegarde sur cassette le programme résident, sous le nom OTHELLO.BAS, sous forme protégée
SAVE "TICTAC", A sauvegarde sous forme littérale (ASCII)
SAVE "TOE" sauvegarde normale

SAVEM descripteur de fichier, adresse 1, adresse 2, adresse 3
instruction ou commande

Sauvegarde une partie de la mémoire dans un fichier binaire.

L'adresse 1 indique le début de la zone à sauvegarder, adresse 2 en indique la fin, adresse 3 indique l'adresse de lancement de l'exécution au moment du chargement par LOADM avec l'option R ou par EXEC sans paramètres.
Si ces adresses sont comprises entre &H6000 et &H9FFF, la zone mémoire est prise dans la banque de mémoire courante.
SAVEM peut servir à conserver une zone mémoire précise ou un programme en binaire.

Exemple:

SAVEM "HORLOGE", &H8800, &H8900, &H8800 sauvegarde la zone binaire comprise entre &H8800 et &H8900

SAVEP descripteur de fichier, élément de tableau ★

instruction

Sauvegarde dans un fichier une portion d'écran compactée dans un tableau.

Le tableau doit être un tableau d'entiers. Il peut contenir plusieurs images mais une seule est sauvegardée à la fois.
L'image doit avoir été compactée à partir de l'élément indiqué par une instruction GET préalable.
Le suffixe par défaut est MAP.
L'image peut être récupérée ultérieurement par LOADP et affichée par PUT.

Exemple:

SAVEP "EINSTEIN", IM%(300) sauvegarde l'image compactée à partir de l'élément 299 dans le tableau IM% sous le nom EINSTEIN.MAP

SCREEN(colonne,ligne)

fonction

Donne le code ASCII du caractère affiché en colonne et ligne indiqués.

Les coordonnées sont données en caractères: colonne de 0 à 39, ligne de 0 à 24.
Seuls les caracteres du tableau ASCII et les minuscules accentuees sont reconnus. Si aucun caractère n'est reconnu, le résultat est 0.
Les codes suivants sont attribués aux minuscules accentuees et au ç:

128 ç	134 é	140 ï	146 ú	BASIC 128 uniquement
129 à	135 ê	141 ò	147 û	149 '
130 á	136 ë	142 ó	148 ü	150 §
131 â	137 ì	143 ô		151 ß
132 ä	138 í	144 ö		152 Ü
133 è	139 î	145 ù		153 Ä

Exemple:

```
10 LOCATE 2,4: PRINT "A"
20 PRINT SCREEN(2,4)
RUN
 A
 65
```

SCREEN forme, fond, tour, inversion, incrustation

instruction

Fixe les couleurs de l'affichage pour tout l'écran.
Tous les paramètres sont facultatifs mais un au moins doit être présent.
En mode 80 colonnes, seule l'instruction PALETTE permet de modifier les couleurs en Basic 128.
Les trois premiers paramètres indiquent respectivement:
- la couleur des caractères et des tracés,
- la couleur du fond,
- la couleur du pourtour de l'écran.

La présence du quatrième paramètre indique que les couleurs de forme et de fond doivent être inversées.
La présence du cinquième paramètre indique que l'image TV doit être incrustée. Cette option ne fonctionne qu'avec l'interface d'incrustation: elle provoque une erreur fatale en l'absence d'interface.

Exemple:

```
SCREEN 1,3,6    caractères rouges sur fond noir et tour magenta
SCREEN ,,,1     inverse les couleurs précédentes
```

SCREENPRINT

instruction

Recopie le contenu de l'écran sur l'imprimante parallèle.

Cette instruction ne fonctionne que pour les imprimantes PR90-582 et PR90-600.

Exemple:

```
SCREEN\PRINT
```

SEARCH chaîne de caractères, n° ligne 1, n° ligne 2

commande

Affiche sur l'écran toutes les lignes du programme comportant la chaîne de caractères spécifiée, de la ligne 1 à la ligne 2 incluse.

La chaîne de caractères à rechercher peut être donnée dans une constante ou une variable.
On peut effectuer la recherche sur tout ou partie du programme, ou sur la ligne courante, avec une syntaxe identique à celle de LIST.

Exemple:

```
SEARCH "NOM$",1000-2000      recherche la variable NOM$ entre les lignes 1000 et 2000
                             recherche NBJ sur la ligne courante
                             recherche 520 dans tout le programme
```

SGN(nombre)

fonction

Donne le signe du nombre.

Le résultat vaut:
— 1 si le nombre est positif,
— 0 si le nombre est nul,
— −1 si le nombre est négatif.

Exemple:

```
PRINT SGN(-2)
-1
```

SHOW ★

fonction

Indique si la tortue active est visible (résultat −1) ou invisible (résultat 0).

Exemple:

```
? SHOW
-1
```

SHOW visible, libre ★

instruction

Fixe l'état de la tortue active.

Le premier paramètre fixe la visibilité.
S'il vaut 0, la tortue est invisible; s'il vaut 1, la tortue est visible.
Le deuxième paramètre fixe la liberté.
S'il vaut 1, la tortue est affichée à chaque modification.
S'il vaut 0, la tortue n'est affichée qu'à chaque déplacement.

Exemple:

```
SHOW 1,1      la tortue active est visible et libre
```

SIN(nombre)

fonction

Donne le sinus du nombre compté en radians.

En Basic 128, le résultat est en double précision si l'argument est en double précision.

Exemple:

```
PRINT SIN(1.5)
.997495
```

SKIPF "nom de fichier"

instruction

Sur la cassette, arrête la bande après le fichier indiqué, ou le fichier suivant si rien n'est indiqué.

Exemple:

```
SKIPF "TICTAC"
```

arrête la bande après le fichier TICTAC

SPACE$ (nombre)

fonction

Donne une chaîne de caractères composée d'autant d'espaces que le nombre le précise.

Exemple:

```
SPACE$(30)
```

donne une chaîne de 30 espaces

SPC(nombre)

fonction

Cette fonction n'est utilisable que dans un PRINT.

Exemple:

```
PRINT "A";SPC(10);"B"
A          B
```

SQR(nombre)

fonction

Donne la racine carrée du nombre.

En Basic 128, le résultat est en double précision si l'argument est en double précision.
Si l'argument est négatif, l'erreur "Illegal Function Call" est détectée.

Exemple:

```
PRINT SQR(10)
3.16228
```

STEP(déplacement vertical, déplacement horizontal) ★

STEP peut être utilisé dans les instructions: BOX, BOXF, CIRCLE, CIRCLEF, LINE, PAINT et PSET.
STEP permet de définir les coordonnées d'un point relativement à la position du dernier point défini (par défaut, ce point est l'origine (0,0)). Les déplacements peuvent être positifs ou négatifs.

Exemple:

```
BOX(100,100)-STEP(-50,50) : BOX(200,150)-STEP(50,50)
```

est équivalent à

```
BOX(100,100)-STEP(-50,50) : BOX STEP(50,50)-STEP(50,50)
```

STICK(n° de manette)

fonction

Donne la position de la manette de jeu désignée.

Il y a deux manettes de jeu numérotées 0 et 1.
Le résultat vaut:
0 neutre

1.90°
2.45°
3.0°
4. – 45°
5. – 90°
6. – 135°
7.180°
8.135°

Exemple:

ON STICK(0) GOSUB ... branche au sous-programme correspondant à la position de la manette

STOP
instruction

Suspend l'exécution du programme.

Cette instruction peut être placée en n'importe quel endroit du programme.
Lorsqu'elle est rencontrée, le message "Break in ..." apparaît.
La reprise peut être faite par CONT ou GOTO.
Tous les fichiers restent ouverts.

Exemple:

```
10 INPUT A,B
20 C=A^2+B^2
30 STOP
40 PRINT C
```

STR$(nombre)
fonction

Transforme un nombre en sa représentation en chaîne de caractères.
Le résultat a la même forme que l'affichage du nombre par PRINT.
La fonction inverse de STR$ est VAL.

Exemple:

```
PRINT STR$(68.5);STR$(-1234)
 68.5 -1234
```

STRIG(n° de manette)
fonction

Donne l'état du bouton de la manette désignée.

Il y a deux manettes de jeu numérotées 0 et 1.
Le résultat est – 1 si le bouton est enfoncé, 0 sinon.

Exemple:

```
IF STRIG(1) THEN PRINT "TOUCHE"
```

STRING$ (nombre, caractère)
fonction

Donne une chaîne de caractères identiques au caractère indiqué et dont la longueur est égale au nombre indiqué.

Le caractère peut être précisé par son code ASCII.

Exemple:

STRING$(10,"*")	donne une chaîne de 10 étoiles
STRING$(20,42)	donne une chaîne de 20 étoiles

SWAP variable 1, variable 2
instruction

Echange le contenu de deux variables de même type.

Cette instruction est très utile dans les programmes de tris.

Exemple:

SWAP A(I),A(J) échange les valeurs des éléments I et J du tableau A

TAB(nombre)
fonction

Positionne le curseur dans une instruction PRINT ou PRINT#.
Le nombre indiqué, modulo la largeur du périphérique utilisé (écran ou imprimante), indique la position du curseur en colonne.
Si cette position est à gauche de la position courante, un saut de ligne est effectué.
Un nombre négatif ou nul est sans effet.
En double largeur, le positionnement tient compte de la largeur double.
Les règles de positionnement fixées par les séparateurs virgule et point-virgule sont respectées.

Exemple:

```
PRINT "A";TAB(10);"B"
A         B
```

TAN(nombre)
fonction

Donne la tangente du nombre exprimé en radians.

Le résultat est en double précision si l'argument est en double précision en Basic 128.

Exemple:

```
?TAN(.5)
14.1014
```

TRACE ⋆
fonction

Indique si la tortue active laisse une trace de son déplacement (résultat −1) ou non (résultat 0).

Exemple:

```
?TRACE
-1
```

TRACE nombre ⋆
instruction

Fixe ou enlève la trace de la tortue active.
Si l'argument vaut 0 (FAUX), la tortue ne trace pas, sinon elle trace.

Exemple:

```
TRACE 1
```

la tortue laissera une trace de tous ses déplacements

TROFF
instruction

Annule le mode de trace du programme déclenché par TRON et ferme le fichier de trace s'il y a lieu.

TRON descripteur de fichier, n° ligne 1 - n° ligne 2
instruction

Déclenche le mode de trace du programme.
Le mode de trace provoque la visualisation sur l'écran ou l'enregistrement dans le fichier indiqué des numéros de lignes exécutés.
En Basic 1, il ne faut pas indiquer de descripteur de fichier, la visualisation se fait toujours sur l'écran.
Les numéros tracés apparaissent entre crochets.
Si le descripteur de fichier est présent, la trace est écrite sur ce fichier, sinon elle est visualisée sur l'écran.
Seules les lignes exécutées dont le numéro est compris entre les deux numéros indiqués sont tracées. La plage des numéros de lignes est indiquée avec la même syntaxe que LIST.

Exemple:

TRON "LPRT:", 200-400 trace sur imprimante la partie de programme comprise entre les lignes 200 et 400
TRON trace tout le programme à l'écran

TUNE ♦

instruction

Réglage du crayon optique. Le pointage du crayon optique sur l'écran provoque l'apparition d'une barre verticale que l'on fait coïncider avec le crayon à l'aide des touches ← et → . La validation du réglage se fait avec ENTRÉE.

TURTLE n°, colonne, ligne, chaîne de caractères ★

instruction

Définit et place la tortue active.

Le premier paramètre indique le numéro de la tortue active. On peut définir jusqu'à dix tortues différentes, numérotées de 0 à 9.
Les deux paramètres suivants fixent la position de la tortue dans l'écran: colonne et ligne compris entre −32768 et 32767. Ces deux paramètres ne sont pas obligatoires. Quand ils sont absents, la position de la tortue n'est pas modifiée ou, si cette tortue est active pour la première fois, elle est placée au centre de l'écran.
Le quatrième paramètre, facultatif, est une chaîne de caractères qui permet de définir la forme de la tortue.
Cette chaîne est constituée de doublets indiquant chacun une rotation suivie d'un déplacement.

Il existe quatre types de doublets possibles: RaDn, RaUn, LaDn et LaUn, où a indique l'angle dont il faut tourner et n le nombre de pas dont il faut avancer.

Les angles et les nombres de pas sont des nombres compris entre 0 et 255. Un angle de 256 est égal à 360°.
R signifie rotation à droite et L rotation à gauche.
D signifie déplacement avec tracé et U déplacement sans tracé.
Les espaces ne sont pas pris en compte.

Exemple:

TURTLE 1,80,120,"L0D40L64D40" définit la tortue n° 1 en forme d'angle droit, et la place en colonne 80 ligne 120
A$="L0D40L128U20L64D20" : TURTLE 2,,,A$ définit la tortue n° 2 en forme de T et la place au centre de l'écran
TURTLE 0 active la tortue n° 0

UNLOAD n° de lecteur

instruction

Ferme tous les fichiers ouverts sur la disquette située dans le lecteur dont on précise le numéro et recopie toutes les informations utiles sur la disquette.

Par défaut, UNLOAD considère le lecteur n° 0 ou celui défini dans DEVICE.
Cette instruction garantit le retrait de la disquette du lecteur sans aucun risque, en particulier après l'arrêt d'un programme d'écriture de fichiers par CNT-C ou par une erreur.

Exemple:

UNLOAD 1 ferme tout sur la disquette 1

USR(donnée)

fonction

Appel de la fonction utilisateur en langage machine de numéro indiqué (0 par défaut).

La fonction doit avoir été définie au préalable par DEFUSRn.
Pour utiliser cette fonction, voir Annexe 4.

Exemple:

```
A=USR2(F)
```
appel de la fonction utilisateur n° 2 avec la variable F comme argument

VAL (chaîne)
fonction

Donne la valeur numérique d'une chaîne de caractères représentant un nombre.

Si le premier caractère n'est pas un caractère décimal, ou +, ou -, ou, un point ou un espace, le résultat de la fonction est nul.
La fonction inverse de VAL est STR$.

Exemple:

```
PRINT VAL("15 Rue Victor Hugo")
15
```

VARPTR (variable)
fonction

Retourne l'adresse du premier octet de la variable indiquée.

Le résultat est un nombre entier compris entre −32768 et 32767. Il faut lui ajouter 65536 pour avoir l'adresse réelle quand il est négatif.
En Basic 128, après l'exécution de cette fonction, la banque courante est devenue celle où se trouve la variable cherchée. On peut obtenir ce numéro de banque par la fonction BANK.
Cette fonction est utilisée le plus souvent pour des modules en langage machine.

Pour les informations concernant l'organisation des variables en mémoire, voir Annexe 3.

Exemple:

```
PRINT VARPTR(A)
```
donne l'adresse de la variable A

VERIFYON
VERIFYOFF
instructions

Avec l'option ON, toute écriture sur disquette sera suivie d'une vérification. L'option OFF supprime cette vérification.

Exemple:

```
VERIFY ON
```

*WAIT adresse, masque 1, masque 2 **
instruction

Suspend l'exécution du programme jusqu'à ce qu'une configuration de bits apparaisse à l'adresse indiquée.

L'adresse est un nombre compris entre −65536 et 65535.
Les deux masques sont des nombres compris entre 0 et 255.
Le fonctionnement de l'instruction est le suivant:
— l'expression booléenne suivante est calculée:
(contenu de l'adresse XOR masque 2) AND masque 1
— l'exécution ne se poursuit que lorsque cette expression n'est pas nulle.
Le masque 2 est facultatif. Dans ce cas l'opérateur XOR n'est pas appliqué.
Cette instruction est utilisée le plus souvent pour attendre un événement sur un circuit périphérique.

On ne peut sortir de l'attente par CNT-C.
L'utilisation de cette instruction nécessite une très bonne connaissance de la structure interne du micro-ordinateur.

WINDOW (colonne 1, ligne 1) - (colonne 2, ligne 2) ★

instruction

Définit la fenêtre de visualisation des instructions graphiques.

La fenêtre est un rectangle défini par deux sommets opposés, le premier étant en haut à gauche et le second en bas à droite.
Les coordonnées en colonne sont des nombres compris entre 0 et 319 (ou 639) et en ligne entre 0 et 199.
Après l'exécution de cette instruction, les instructions de tracé graphique BOX, BOXF, LINE, PSET, CIRCLE, CIRCLEF et PAINT n'auront d'effet visible qu'à l'intérieur de la fenêtre.

Exemple:
WINDOW (100,100)-(200,180)

WRITE # n° de canal, liste de données

instruction

Ecrit dans le fichier, dont on précise le n° de canal, la liste des données.

Les chaînes de caractères sont entourées de guillemets et les différentes données sont séparées par des virgules. Après la dernière donnée, WRITE # écrit deux caractères: le saut de ligne et le Retour Chariot.

Dans le cas d'un fichier à accès direct, WRITE # écrit dans le tampon d'enregistrement. Le transfert de l'enregistrement dans le fichier doit alors se faire par PUT #.

Exemple:
WRITE #2, NOM$, TAUX

ZOOM ★

fonction

Donne la taille de la tortue active.

Le résultat est un nombre compris entre 0 et 255.
La taille normale d'une tortue au moment de sa définition par TURTLE est 16.

Exemple:
?ZOOM
32

ZOOM TO nombre ★

instruction

Fixe ou modifie la taille de la tortue active.

Si TO est présent, la taille est fixée de façon absolue à la valeur indiquée, sinon la taille est augmentée de cette valeur.
La taille d'une tortue peut varier de 0 à 255, mais la longueur d'un segment de la tortue ne doit pas dépasser 255.
Suivant la valeur du 2e paramètre de SHOW, la modification de la taille est immédiate (tortue libre) ou bien elle n'est effective qu'à l'exécution de FWD ou TURTLE.

Exemple:
TURTLE 3: ZOOM TO 32 fixe la taille de la tortue n° 3 à 32
ZOOM 10 augmente de 10 la taille de la tortue active