Canon

X-07

MANUEL DE REFERENCE BASIC

ORDINATEUR INDIVIDUEL

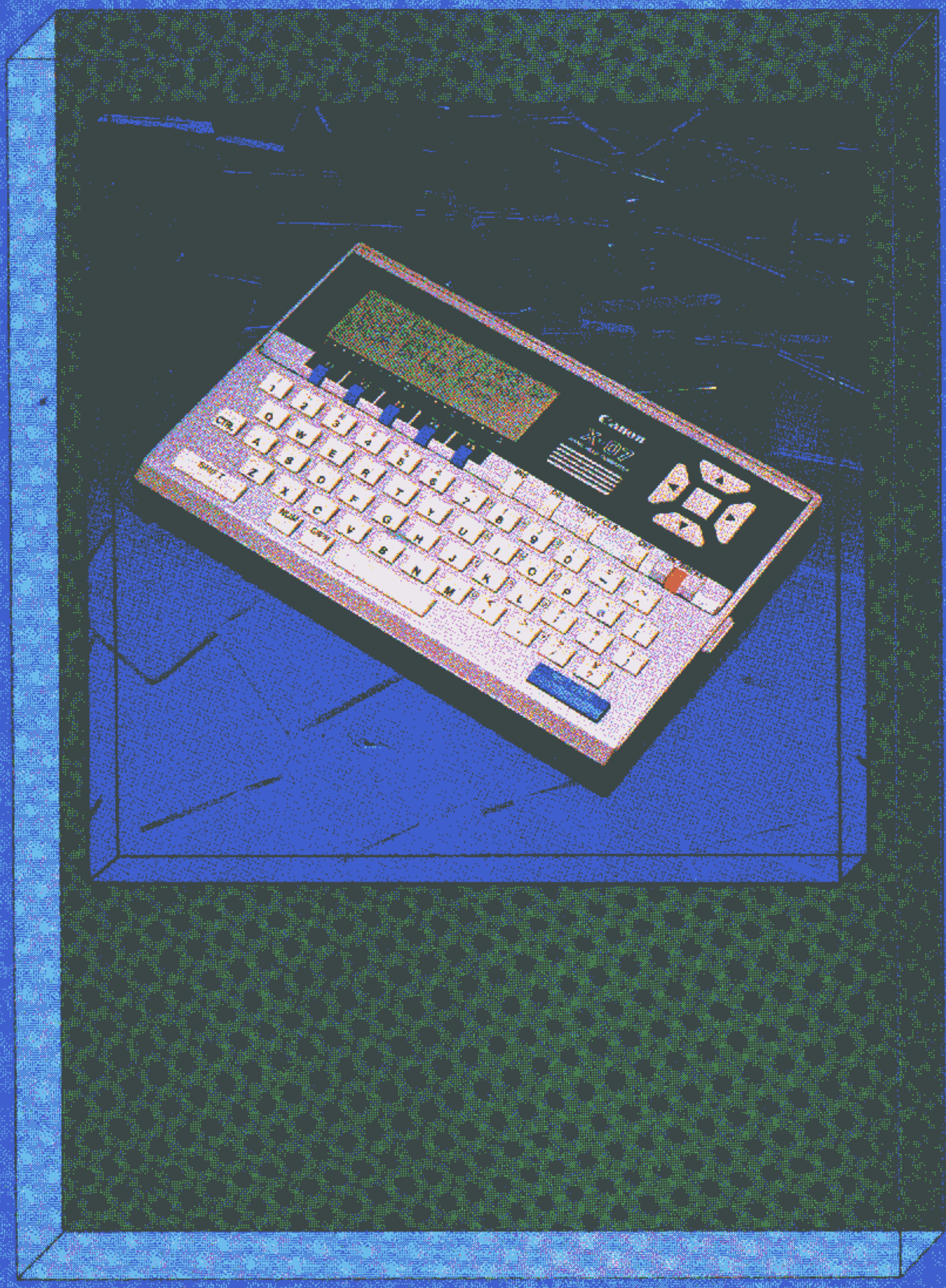


TABLE DES MATIERES

Chapitre I:
Syntaxe fondamentale du BASIC 1

1.1 Langage BASIC et l'interpréteur BASIC 3
1.2 Editeur 4
1.3 Format d'un programme BASIC 4
1.4 Format de ligne 5
1.5 Modes 6
1.6 Messages d'erreur 7
1.7 Editeur d'écran 7
1.8 Jeu de caractères du X07-BASIC 8
1.9 Symboles spéciaux 8
1.10 Constantes 10
1.10.1 Chaines de caractères 10
1.10.2 Constantes numériques 10
1.11 Variables 12
1.11.1 Désignation des variables 12
1.11.2 Caractères de déclaration 12
1.11.3 Variables de tableau 13
1.11.4 Conversion de type 14
1.12 Expressions et opérations 15
1.12.1 Expressions arithmétiques 15
1.12.2 Expressions de relation 16
1.12.3 Expressions logiques 17
1.12.4 Fonctions 18
1.12.5 Expressions en chaîne 18
1.12.6 Priorité de calcul 19
1.13 Précision 19

Chapitre 2:
Syntaxe BASIC 21

2.1 Numéro de fichier et unités 23
2.2 Des cripteur de fichier 23
2.2.1 Désignation des unités 24
2.2.2 Désignation des fichiers 25
2.3 Périphériques 25
2.3.1 Imprimante 25
2.3.2 Enregistreur à cassette 26
2.3.3 Port série 26
2.3.4 Cartes utilitaires 26
2.3.5 Fichier en mémoire RAM 26
2.3.6 Déroulement direct 27
2.4 Graphisme 28
2.4.1 Coordonnées relatives et absolues 28

Chapitre 3:
Ordres et instructions 31

Chapitre 4:
Fonctions incorporées 109
Chapitre 5:

Appendice 167

☞ *INDEX*

1. Syntaxe Fondamentale du BASIC

2. Syntaxe du BASIC

3. Ordres et instructions

4. Fonctions Incorporées

5. Appendices

Comment consulter ce manuel

Ce manuel décrit la syntaxe fondamentale du langage X07-BASIC et les fonctions de l'interpréteur BASIC propres à l'ordinateur individuel CANON X-07.

Ce manuel est composé ainsi:

Chapitre 1: Syntaxe fondamentale du BASIC
Chapitre 2: Syntaxe du BASIC
Chapitre 3: Ordres et instructions
Chapitre 4: Fonctions incorporées
Chapitre 5: Appendices: messages d'erreur, tableau des codes de caractères, indices, etc.

Le chapitre 1 décrit la syntaxe fondamentale du langage X07-BASIC. Il concerne essentiellement les informations fondamentales relatives au X07-BASIC, à l'architecture des programmes, des lignes, aux procédures de calcul, etc.

Le chapitre 2 introduit le concept de fichier, et explique les instructions du X07-BASIC relatives aux périphéiques.

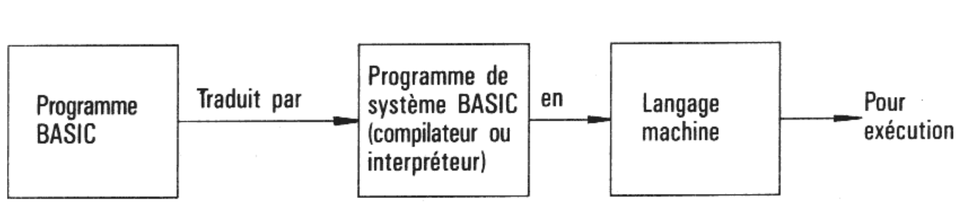
Les chapitres 3 et 4 expliquent de manière détaillée toutes les instructions et fonctions du X07-BASIC.

Le chapitre 5 donne l'explication des messages d'erreur, le tableau des codes de caractères, les indices, etc.

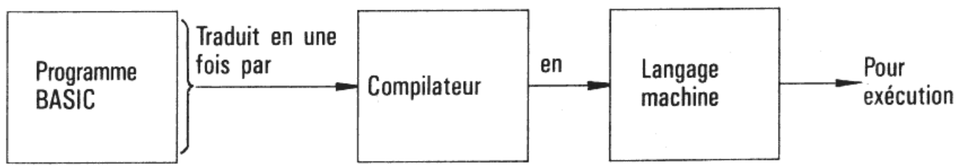
Chapitre 1
Syntaxe Fondamentale du BASIC

1.1 Langage BASIC et interpréteur BASIC

Le BASIC est un langage de programmation destiné à faire exécuter à l'ordinateur des tâches spécifiques. Le langage BASIC possède sa propre syntaxe ou grammaire, qui régit sa structure. Le langage BASIC se compose d'instructions qui commandent l'ordinateur, et qui sont structurées de façon à lui faire exécuter les tâches désirées dans un ordre donné.
Le langage BASIC est d'une compréhension relativement facile pour nous, mais il est dénué de sens pour l'ordinateur, qui n'assimile que le langage machine. Il est possible de rédiger un programme en langage machine, mais celui-ci est d'un accès difficile et sa maîtrise exige beaucoup de temps. La solution à ce problème est un programme dit "programme de système BASIC". Celui-ci traduit un programme rédigé en langage BASIC en langage machine. Il existe deux types de programmes de système BASIC:

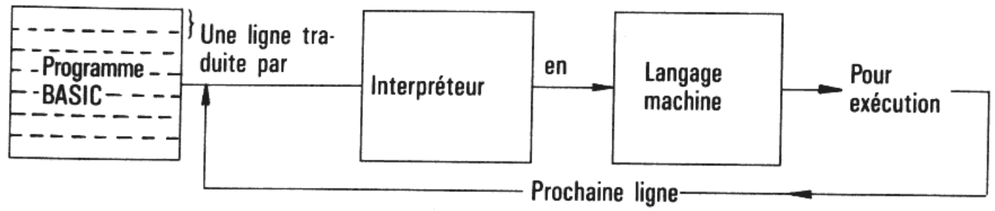


a) Compilateur
Un compilateur convertit un programme BASIC complet en une fois en langage machine pour son exécution.



Les compilateurs sont utilisés dans les grands ordinateurs. Les programmes sont exécutés à grande vitesse, mais ceci nécessite une vaste mémoire.

b) Interpréteur
Un interpréteur convertit ligne par ligne un programme BASIC en langage machine, et l'exécution se déroule à ce rythme.



Avec les interpréteurs, les programmes sont exécutés à une vitesse plus lente qu'avec les compilateurs, mais ils nécessitent une plus petite mémoire. C'est pour cette raison qu'ils sont couramment utilisés dans les ordinateurs individuels.
Le X-07 utilise un interpréteur comme programme de système BASIC.

1.2 Editeur

Afin qu'un ordinateur puisse exécuter un programme en BASIC, celui-ci doit au préalable être introduit dans la machine. C'est la fonction de l'éditeur. En d'autres termes, l'éditeur introduit le programme en BASIC (texte source) dans l'ordinateur. Au stade suivant, l'ordinateur exécute le programme en BASIC au moyen de son interpréteur.
L'éditeur du X-07 est incorporé à l'interpréteur, et il est utilisé pour créer et éditer des programmes. Dans ce manuel, les instructions relatives à l'édition de programmes sont appelées "ordres d'édition".

1.3 Format d'un programme BASIC

Voici un exemple d'un progrmme BASIC:

```
100 INPUT "A=";A
110 FOR N=0 TO 9:PRINT A+1;: NEXT N
120 PRINT " ":GOTO 100
```

Un programme BASIC se compose de ligne d'instruction(s). A chaque ligne est attribuée un numéro, et le programme est traduit et exécuté dans l'ordre des numéros de ligne.

1.4 Format de ligne

Le format des lignes de programme BASIC est le suivant:

⟨Numéro de ligne⟩ ⟨instruction 1⟩ ⟨instruction 2⟩:....: ⟨instruction n⟩

a) Numéro de ligne
Les instructions d'un programme BASIC sont exécutées séquentiellement selon les numéros de ligne. Ces derniers sont également utilisés pour les branchements (saut d'une ligne à l'autre) commandés par les instructions GOTO et GOSUB, et pour l'édition d'un programme. En X07-BASIC, les numéros de ligne doivent être des nombres entiers compris entre 0 et 65 529.

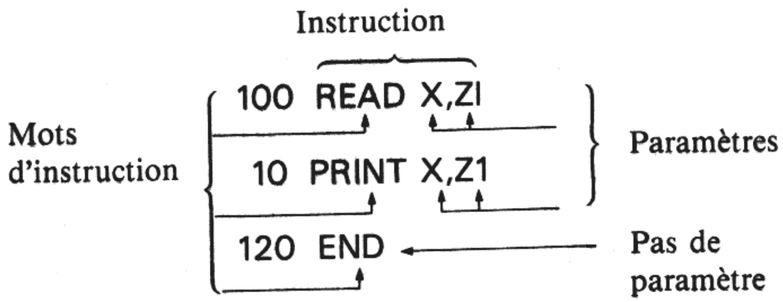
Exemple:
```
100 Instruction(s)
110 Instruction(s)
120 Instruction(s)
 .
 .
 .
```

b) Instructions
Une instruction peut être isolée ou associée à des paramètres. Il peut y avoir plus d'une instruction par ligne, mais elles doivent alors être séparées par deux-points (:). Une ligne comprenant plusieurs instructions est appelée "ligne multi-instructions".
La forme générale d'une instruction est la suivante:

Mot d'instruction ⟨expression 1⟩,⟨expression 2⟩,...,⟨expression n⟩

⟨expression 1⟩,....,⟨expression n⟩ sont appelés les "paramètres" d'une instruction. Cependant, il existe des instructions sans paramètres.

Exemple:



Comme vous pouvez le voir ci-dessus, une ligne se compose d'un numéro de ligne et de une ou plusieurs instructions.

Une ligne ne peut pas comprendre plus de 255 caractères (un espace compte comme un caractère). Cependant, lors de l'utilisation du X07-BASIC tel qu'il se présente, une ligne ne doit pas dépasser 80 caractères. L'affichage à cristaux liquides (LCD) du X-07 et l'éditeur d'écran incorporé permettent uniquement d'afficher 80 caractères au maximum (20 caractères sur 4 lignes) pour rédiger une ligne de programme BASIC.

1.5 Modes

Il existe deux modes d'instruction d'un ordinateur: a) direct, et b) indirect.

a) Mode direct
En mode direct, les instructions sont entrées sans numéro de ligne, et sont immédiatement exécutées lorsque la touche [RETURN] est frappée. Les instructions entrées en mode direct sont appelées ordres.

```
Exemple:  PRINT "ALLO"   ←   Ordre
          [RETURN]       ←   Frapper la touche
          ALLO           ←   Sortie
```

b) Mode indirect
En mode indirect, les instructions sont introduites dans l'ordinateur avec des numéros de ligne. Ces instructions ne sont pas exécutées immédiatement, mais plutôt stockées dans la zone texte de la mémoire sous forme de programme. Les instructions ainsi stockées (formant un programme) sont exécutées lorsqu'un ordre RUN est appliqué. Entrées en mode indirect, ce sont les "instructions".

```
Exemple:  100 LET X=5
          110 PRINT X+5  ←   Instructions
          120 END
          RUN            ←   Ordre
          10             ←   Sortie
```

Certaines instructions peuvent uniquement être introduites sous forme d'ordre ou d'instruction. D'autres peuvent être entrée sous une forme ou l'autre. Par exemple PRINT peut être entré comme ordre (exemple (a)), ou comme instruction de programme (exemple (b)).

1.6 Messages d'erreur

Lorsque le X07-BASIC décèle une erreur durant le déroulement d'un programme, il s'arrête et affiche un message d'erreur. Ensuite il se commute sur le niveau d'attente d'ordres BASIC (affichage du curseur). En mode direct, les messages d'erreur prennent le format général suivant:

```
XX Erreur
```

En mode indirect, les messages d'erreur prennent le format général suivant:

```
XX Erreur en nnn
```

"XX" indique le type d'erreur, et "nnn" le numéro de la ligne dans laquelle l'erreur s'est glissée. Veuillez vous reporter à "Messages d'erreur" à la page 169.

1.7 Editeur d'écran

Un éditeur est utilisé pour créer et corriger les programmes BASIC. Un éditeur d'écran est incorporé à l'interpréteur X07-BASIC. Une ligne du programme BASIC affichée sur l'écran peut être corrigée, mais veuillez noter les points suivants:

```
100  FOR I=0 TO 9:PRI
NT A+I;:NEXT I
```

La ligne de programme entière est affichée et peut donc être corrigée.

```
100 FOR I=0 TO 9:PRI
NT A+I;:NEXT I
```

Seule une partie de la ligne de programme est affichée. "100 FOR I=0 TO 9:PRI" a roulé hors de l'écran; de ce fait, cette ligne ne peut pas être corrigée.

1.8 Jeu de caractères du X07-BASIC

L'alphabet, les chiffres, symboles spéciaux et graphismes propres au X07-BASIC sont collectivement identifiés par "jeu de caractères". Il existe également des caractères de commande qui, bien qu'ils ne soient pas affichés, sont traités de la même manière que le jeu de caractères. Chaque caractère est converti en un code de 8 bit avant d'être transmis pour son exécution. Ces codes sont conformes à ASCII (American Standard Code for Information Exchange). Prière de vous reporter au "tableau des codes de caractères" apparaissant en page xx. De plus, une partie des codes de caractères peuvent être librement définis par l'utilisateur. Se reporter à "FONT$" en page xx.

1.9 Symboles spéciaux

Voici quelques symboles spéciaux utilisés par le X07-BASIC

1) Symboles de calcul

+, −, /, *, etc.

2) Moins (−)

Moins est utilisé pour spécifier la gamme de lignes dans un ordre LIST, etc.

Exemple: `LIST 100−200`

3) Deux-points (:)

Les deux-points sont utilisés sur une ligne multi-instructions pour séparer chaque instruction.

Exemple: `X=Y+Z:PRINT X`

4) Virgule (,)

Les virgules sont utilisées pour séparer les paramètres.

Exemple:
```
INPUT A,B,C
PRINT X,Y,Z
```


5) Point-virgule (;)

Les point-virgules sont utilisés pour séparer les paramètres de l'instruction PRINT.

Exemple: `PRINT A;B;C`

* Les virgules et point-virgules remplissent des fonctions différentes dans l'instruction PRINT. Se reporter à "PRINT"à la page 91.

6) Apostrophe (')

Un apostrophe peut être utilisé à la place du mot "REM" dans une instruction "REM".

Exemple: `'X07-BASIC`

7) Point d'interrogation (?)

Un point d'interrogation peut être utilisé à la place du mot "PRINT" dans une instruction PRINT.

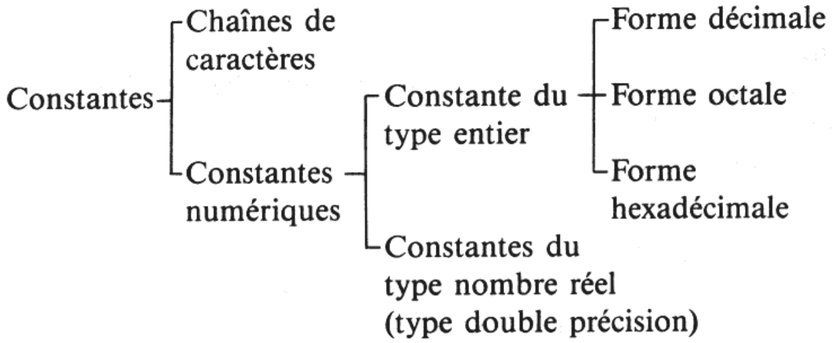
Exemple: `?''A='';A`

8) Guillemets (")

Les guillemets (") sont utilisés pour signifier chaînes de caractères. Un guillement ne peut être inclus dans une chaîne de caractères. Se reporter à "Chaîne de caractères".

1.10 Constantes

Les constantes sont les valeurs actuelles utilisées par le X07-BASIC durant l'exécution. On distingue grossièrement deux genres de constantes: chaînes de caractéres et constantes numériques.



1.10.1 Chaînes de caractères

Une chaîne de caractères peut être tout jeu de caractères (y compris les espaces) enfermés entre guillemets. Ces caractères sont stockés en tant que caractères et non sous forme numérique. Aucun opérateur arithmétique ne peut être appliqué à une chaîne de caractères. Les guillemets et codes de commande ne peuvent être inclus dans une telle constante. Pour utiliser de tels caractères avec l'instruction PRINT ou la fonction START$, ETC., utiliser la fonction CHR$.

Exemple
```
''CANON''
''1234567''
''ORDINATEUR''
'''' (chaîne nulle: constante en chaîne sans caractères)
```

1.10.2 Constantes numériques

Une constante numérique représente une valeur numérique. On dénombre deux genres de constantes numériques:a) du type entier, et b) du type nombre réel.

(a) Constantes entières

Les constantes entières peuvent être sous forme décimale, octale ou héxadécimale.

1) Forme décimale

Les constantes entières compris sous forme décimale sont des nombres entiers entre −32768 et +32767, avec le symbole entier "%" attaché à la fin. Elles ne comprennent pas de virgule décimale. "+" peut être omis.

Exemple:
```
3333%
−1234%
789%
```

2) Forme octale

Les constantes entières sous forme octale sont des chaînes composées de chiffres 0 à 7 en base 8 (la forme décimale est de base 10), et dotées du préfixe "&O" ou "&". Elles doivent être comprises entre − & 0 et & 177777.

Exemple:
```
&O12345
&177777
```

3) Forme hexadécimale

Les constantes entières de forme hexadècimale sont des chaînes composées de chiffres de 0 à 9 et de caractères de A à F (A à F représentant les chiffres de 10 à 15) de base 16, avec préfixe "&H". Elles doivent être comprises entre −&H0 à &HFFFF.

Exemple:
```
&H12F3
−&H7EDC
```

b) Constantes du type nombre réel

Toutes les constantes du type nombre réel sont traitées comme constantes à double précision. En l'absence de spécification, les calculs sont exécutés avec une précision de 14 chiffres. Il existe 5 formes admissibles de constantes du type nombre réel.

1) A l'intérieur du nombre de chiffres de précision.
2) Avec caractère de déclaration " # ".
3) Elévation à une puissance avec D.
4) Elévation à une puissance avec E.
5) Avec caractère de déclaration "!".

Dans d'autres BASIC, la forme 5) est en général une constante à simple précision (précision de 6 chiffres). Cependant, en X07-BASIC, toutes les constantes du type nombre réel sont traitées en double

précision. En conséquence, il n'existe pas de problème de précision des constantes lors de la conversion d'un programme rédigé dans un autre BASIC. La gamme de constantes du type nombre réel va de -9.9999999999999E62 à 9.9999999999999E62. exprimé sous forme exponentielle, les exposants doivent être compris entre -63 et 62.

Exemple:
```
1.23E-5
1.234!
963472.4418621
1.86793D23
1.367823331 #
```

1.11 Variables

Les variables sont utilisées pour représenter des valeurs chaînes de caractères constantes numériques stockées en des endroits spéciaux de la mémoire. Le nom d'une variable (nom de variable) est utilisé pour référencier la valeur stockée en mémoire sous ce nom. La valeur d'une variable peut être attribuée explicitement par le programmeur, ou peut être attribuée comme résultat d'un calcul ordonné par le programme. Avant qu'une valeur lui soit attribuée, la variable est supposée nulle si elle est de forme numérique, et "" (chaîne nulle) si elle est de forme chaîne.

1.11.1 Désignation des variables

Une variable est désignée par une lettre A à Z suivie de caractères alphanumériques (A à Z et 0 à 9). La désignation peut avoir toute longueur jusqu'à 255 caractères, mais seuls les deux premiers sont significatifs. Par exemple, ABlCDE est identique à AB36H. Une désignation de variable ne peut être un mot réservé (ordre, instruction, etc.), et ne peut commencer par "FN" qui est l'appel d'une fonction définie par l'utilisateur. Il est possible d'utiliser des lettres minuscules ou majuscules.

1.11.2 Caractères de déclaration

Une variable est classifiée selon le type de constante qu'elle représente, et un caractère de déclaration est placé à la fin de sa désignation pour déclarer ce qu'elle représente. Lorsque le caractère de déclaration est

déclarer ce qu'elle représente. Lorsque le caractère de déclaration est omis, la variable est supposée représenter une constante de type nombre réel à double précision. Les variables de même désignation, mais de caractères de déclaration différents sont considérées comme des variables différentes.

Les caractères de déclaration sont les suivants:

% Les variables comportant ce caractère à la fin de leur désignation représentent des constantes de type entier, et sont appelées "variables entières".

Exemple: AB%

! Les variables affectées de ce caractère représentent des constantes de type nombre réel à simple précision, et sont appelées "variables du type nombre réel à simple précision".

Exemple: C2!

Les variables affectées de ce caractère représentent des constantes de type nombre réel à double précision, et sont appelées "variables du type nombre réel à double précision".

Exemple: PT #

$ Les variables affectées de ce caractère représentent des chaînes de caractères, et sont appelés "variables chaînes de caractères".

Exemple: WORD$

Les variables de type entier, à simple ou double précision sont collectivement appelées "variables numériques".

1.11.3 Tableaux de variables

Une désignation de variable ne référencie qu'une seule valeur. Cette méthode convient lorsque le programme ne contient que quelques valeurs, mais est trop lourde lorsqu'il s'agit de traiter de nombreuses valeurs. Ceci est réalisé par l'emploi de variables indicées connues sous le nom de "variables en tableau". Se reporter à instruction "DIM" à la page 49.

1.11.4 Conversion de type

Le X07-BASIC est capable de convertir une constante numéque d'un type à l'autre conformént aux règles suivantes.

1) Lorsqu'une constante numéque d'un type est attribuée à une variable numérique d'un autre type, la constante numérique est convertie au type propre à la variable.

   Exemple: `A% = 12.34` 12 est attribué à la variable A%.

2) Tous les calculs sont effectués en double précision.

   Exemple: `100#/7` Calcul effectué à double précision.(100#/7#)

3) Durant les opérations logiques, toutes les constantes sont converties en constantes entières, et les résultats sont du type entier.

   Exemple: `A = NOT 12.3` −13 est attribué à la variable A.

4) Lorsqu'une constante de type nombre réel est attribuée à une variable du type entier, la partie dècimale est tronquée.

   Exemple: `A% = 24.6` 24 est attribuée à la variable A%.

5) Lorsqu'une constante du type nombre réel double précision est affecté à une variable DV type nombre réel simple précision, la valeur est arrondie à 6 chiffres significatifs.

   Exemple: `A! = 1.23456789` 1.23457 est attribué à la variable A!

Essayez le programme suivant:

```
10 A! = 1.23456789123456
20 A# =1.23456789123456
30 A% = 1.23456789123456
40 PRINT A!,A#,A%
```

Résultat de l'exécution:

```
1.23457
1.23456789123456
1
```


1.12 Expressions

Une expression est tout arrangement significatif de constantes, variables, etc. reliées par des opérateurs. Il existe 5 types d'expression:

1) Expression arithmétique
2) Expression de relation
3) Expression logique
4) Expression de fonction (fonctions)
5) Expression en chaîne

1.12.1 Expressions arithmétiques

1) Les opérateurs arithmétiques sont les suivants:

Priorité ↓

Opérateur BASIC symbole	Opérateur	Exemple en BASIC
∧	Elévation à une puissance	A∧B
−	Signe moins	−A
*,/	Multiplication, division de constantes de type nombre réel	A*B, A/B
+,−	Addition, soustraction	A+B, A−B

L'ordre dans lequel les opérations sont exécutées (priorité) peut être modifié en plaçant les expressions entre parenthèses. Les expressions placées entre parenthèses prennent la priorité sur toutes les autres.

Voici quelques expressions algébriques et leur équivalent en BASIC.

Expression algébrique	Equivalent BASIC
$2\times A+B$	2*A+B
$\frac{A}{B}+C$	A/B+C
$\frac{A+C}{B}$	(A+C)/B
A^3+3A+4	A∧3+3*A+4
A^{B^2}	A∧(B∧2)
$(A^B)^3$	A∧B∧3
$A(-B)$	A*(−B)

2) Division et module d'entiers

La division de l'entier est indiquée par "¥". Les constantes numériques sont tronquées à leur entier avant la division, et le résultat est également tronqué à l'entier.

Exemple: A%=10¥3 (10/3=3....1)
B%=13.5¥3 (13/3=4...1)

3 et 4 sont respectivement attribués aux variables A% et B%.

Le calcul de module est indiqué par l'opérateur MOD. Il donne le reste de la division d'un entier.

Exemple: A%=10MOD3 1 est attribué à la variable A%

3) Division par zéro et débordement

Lorsque la division par zéro est rencontrée durant l'exécution d'une expression, le X07-BASIC cesse l'exécution et affiche un message d'erreur. Lorsque le résultat d'une affectation ou d'un calcul dépasse la gamme des variables traitées, il se produit un débordement, et le X07-BASIC cesse l'exécution en affichant un message d'erreur.

1.12.2 Expressions de relation

Les opérateurs de relation suivants sont utilisés pour comparer deux valeurs. Le résultat est soit vrai (-1) soit faux (0).

Opérateur de relation	Description	Exemple
=	Egal	A=B
<> ou ><	Différent de	A<> B ou A><B
<	Plus petit que	A<B
>	Plus grand que	A>B
<= ou =<	Plus petit ou égal à	A<= B ou A=<B
>= ou =>	Plus grand ou égal à	A>=B ou A=>B

Example: X=A>0

X= −1 lorsque A est plus grand que 0.
X=0 lorsque A est plus petit que 0.

Les expressions de relation peuvent être utilisées pour déterminer la condition d'une instruction IF. (se reporter à l'instruction "IF" à la page xx.)

Exemple: IF X=0 THEN X=X+1

1.12.3 Expressions logiques

Les expressions logiques sont utilisées pour effectuer des comparaisons de bits, des opérations en algèbre de DE Boole et tester des situations complexes. Les valeurs et les résultats traitées en chiffres binaires (bis), (0 ou 1). Les opérateurs logigues sont les suivants:

NOT (négation)

A	A NOT
1	0
0	1

AND (prodiut logique)

A	B	A AND B
1	1	1
1	0	0
0	1	0
0	0	0

OR (somme logique)

A	B	A OR B
1	1	1
1	0	1
0	1	1
0	0	0

XOR (exclusif)

A	B	A XOR B
1	1	0
1	0	1
0	1	1
0	0	0

EQV (équivalent)

A	B	A EQV B
1	1	1
1	0	0
0	1	0
0	0	1

Dans les expressions logiques, les entiers de -32768 à 32767 sont convertis en données binaires, et les entiers sont calculés bit par bit.

Exemple: Dans cet exemple, l'indice 10 dénote des nombres. décimaux, et l'indice 2 des nombres binaires

$A=62_{10}$ AND 9_{10}

1) $62_{10} = 111110_2$ et $9_{10}=001001_2$
2) Se rappeler

A	B	A AND B
1	1	1
1	0	0
0	1	0
0	0	0

3) Ainsi:
62_{10} 111110_2
AND
9_{10} $\underline{001001_2}$
001000_2
4) $001000_2=8_{10}$
5) $A=62_{10}$ AND $9_{10}=8_{10}$

Voici un test de multiples conditions avec une instruction IF:

```
100 IF A<0 OR B<10 THEN 170
```

Lorsque A est négatif ou que B est moins grand que 100, l'exécution du programme est dérivée sur la ligne 170. Dans cet exemple, lorsque les valeurs résultant de l'expression logique sont 0 (faux) ou −1 (vrai), le résultat est aussi 0 ou −1.

1.12.4 Expressions de fonctions (fonctions)

Une fonction est utilisée pour exécuter une opération prédéfinie avec un paramètre donné. le X07-BASIC comporte des fonctions "incorporées" qui sont prédéfinies par celui-ci, telles que SIN (sinus), SQR (racine carrée), appelées fonctions numériques, et CHR$ et LEFT$, appelées fonctions de chaîne. Pour plus de détails, se reporter au chapitre 4. Le X07-BASIC permet également des "fonctions définies par l'utilisateur", qui sont spécifiées par le programmeur. Prière de se reporter à "DEFFN" à la page xx. En général, lorsqu'une constante de type nombre réel est utilisée comme paramètre d'une fonction qui n'accepte qu'un paramètre entier, celle-ci est tronquée à son entier avant l'exécution du calcul. De même, une constante de type nombre réel simple précision est également utilisable comme paramètre d'une fonction n'acceptant qu'un paramètre du type nombre réel, mais le calcul est effectué en double précision, et le résultat est restitué en double précision également.

1.12.5 Expressions en chaîne

Les constantes en chaîne peuvent être liées par "+".

Exemple: Lorsqu "ABC" est attribué à la variable A$ et "DEF" à la variable B$,

```
C$=A$+B$+''GHI''
```

"ABCDEFGHI" est attribué à C$.

Tout comme les constantes numériques, les chaînes de caractères comparées à l'aide d'opérateurs de chaîne sont effectuées caractère par caractère et par comparaison de leur code ASCII.

1) Lorsque tous les codes ASCII sont identiques, les chaînes sont égales.
   Exemple: ''ALLO'' = ''ALLO''

2) Lorsqu'un code ASCII est différent, la chaîne dont le code ASCII est le plus grand est considérée comme la plus grande.
   Exemple: ''ABDEF'' > ''ABCHKLM''
   D > C

3) Dans la comparaison, la chaîne la plus courte est considérée comme la plus petite.
   Exemple: ''ABC'' < ''ABCDEF''

4) Les espaces sont considérés comme étant des caractères.
   Exemple: ''CL '' > ''CL''

1.12.6 Priorité de calcul

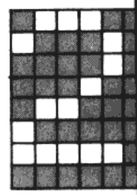
La priorité de calcul est la suivante:

1) Opérations entre parenthèses [(A+B)]
2) Fonctions [SIN(X)]
3) Elévations à une puissance (A∧B)
4) Signes (−A)
5) Multiplications, divisions (A∗B, A/B)
6) Divisions d'entier (A¥B)
7) Calcul de module (10MOD3)
8) Additions, soustractions (A+B, A−B)
9) NOT (NOT A)
10) AND (A AND B)
11) OR (A OR B)
12) XOR (A XOR B)
13) EQV (A EQV B)

1.13 Précision

Tous les calculs +, −, ∗, / sont exécutés en double précision (14 chiffres significatifs). Essentiellement, toutes les fonctions incorporées sont exécutées en double précision. Toutefois, certaines ne réalisent pas ce degré de précision. Prière d'accorder une attention particulière aux chiffres significatifs spéciaux d'une fonction (par exemple TAN (π/2)).

Chapitre 2
Syntaxe BASIC



Le X07-BASIC utilise le concept de fichiers pour contrôler les opérations d'entrée/sortie des périphériques. Un fichier peut être considéré comme une collection d'informations, les programmes et données étant traités en tant que fichiers.

2.1 Numéros de fichier et unités

Un numéro de fichier est utilisé pour définir le fichier (unité) avec lequel l'opération d'entrée/sortie est effectuée. Un numéro de fichier doit être un entier de 1 à 5, et est attribué au fichier (unité) au moyen de l'instruction INIR # .

Exemple: INIT # 3, ''OPT:'',1200,''B''

Dans l'exemple ci-dessus, le numéro de fichier 3 est défini comme le fichier (unité) appelé "OPT:", la cadence de transmission est de 1200 baud, et le mode de transmission en série (ACIA) est "B". Pour plus de détails au sujet des désignations d'unité et modes de transmission en série, se reporter au guide de l'utilisateur. Dans l'exemple ci-dessus, "OPT:" représente coupleur optique: cadence de transmission de 1200 baud veut dire 1200 bit/seconde, et mode de transmis sion "B" signifie que le format des données est de 8 bit avec contrôle de parité impaire. Une unité spécifiée par l'instruction INIT # peut être utilisée avec les instructions INPUT # et PRINT # .

Exemple
```
100 INIT # 3,''OPT:'',1200,''B''
110 INPUT # 3, A$
120 PRINT # 3, B$
```

Le contenu de A$ et B$ est entré/sorti à travers le coupleur optique.

2.2 Descripteur de fichier

Le X07-BASIC utilise également le concept de fichier pour transférer les données allant ou venant d'une unité d'entrée/sortie. Les opérations d'entrée/sortie peuvent être exécutées par une instruction générale, quel que soit le dispositif d'entrée/sortie utilisé. Ceci est accompli au moyen d'un descripteur de fichier, dont la forme générale est la suivante:

''⟨désignation d'unité⟩:[⟨désignation du fichier⟩]''

Le descripteur de fichier doit être inclus entre guillemets.

2.2.1 Désignation des unités

<désignation d'unité> définit l'unité d'entrée/sortie utilisée, et se compose de 3 à 4 caractères à la fin desquels est placé un deux-points (:). Les désignations d'unité en X07-BASIC sont les suivantes:

Désignation des unités

Désignation d'unité	Unité	Entrée	Sortie	1er paramètre	2è paramètre
CON:	Console	o	o	—	—
KBD:	Clavier	o	—	—	—
COM:	E/S série (RS-232C)	o	o	Cadence de transmission: 100 - 8000 baud Par défaut: 4800	Mode ACIA: A - H Par défaut: B
OPT:	Coupleur optique	o	o	Cadence de transmission 100 - 2400 baud Par défaut: 1200	Mode ACIA: A - H Par défaut: B
GPR:	Imprimante graphique	—	o	—	—
LPT:	Imprimante type Centronics	—	o	—	—
PRT:	Imprimante thermique	—	o	— 300 baud, fixe	—
CASI:	Entrée enregistreur à cassette	o	—	— 1200 baud, fixe	— Mode B, fixe
CASO:	Sortie enregistreur à cassette	—	o	— 1200 baud, fixe	— Mode B, fixe
RAM:	Fichier sur carte RAM	o	o	Grandeur: nombre d'octets	Type de données: A - Z Par défaut: D

Tableau des modes ACIA

MODE	EP	PEN	CBL
A	0	0	0
B	0	0	1
C	0	1	0
D	0	1	1
E	1	0	0
F	1	0	1
G	1	1	0
H	1	1	1

EP: parité paire
1: pair
0: impair

PEN: validation de parité
1: validé
0: invalidé

CBL: Longueur en bit des caractères
1: 8 bit
0: 7 bit

Lorsque la <désignation de fichier> est omise, le X07-BASIC désigne automatiquement une unité; celle-ci est appelée "unité par défaut".

2.2.2 Désignations de fichier

<désignation de fichier> définit le fichier devant subir l'opération d'entrée/sortie, et est fixée par le programme. Une <désignation de fichier> est nécessaire pour exécuter . une opération d'entrée/sortie avec une unité de stockage externe, par exemple un enregistreur à cassette ou une carte de mémoire ROM ou RAM. Elle peut être omise lors de l'utilisation d'autres unités. Une <désignation de fichier> se compose de caractères alphanumériques et de symboles, totalisant 6 caractères au maximum; seuls les 6 premiers caractères sont actifs, les suivants étant ignorés.

2.3 Périphériques

Dans le X07-BASIC, chaque unité périphérique est traitée au moyen du concept de fichiers. En conséquence, toute operation d'entrée/sortie peut être effectuée à l'aide d'une instruction généralisée.

2.3.1 Imprimante

Deux types d'imprimantes peuvent être utilisées avec le X07-BASIC. Une imprimante graphique est utilisable lorsque la désignation d'unité "GPR:" est spécifiée. Outre l'instruction générale de sortie, l'imprimante accepte les suivantes:

```
LPRINT, LPRINT USING
LLIST
```

Lors de l'utilisation d'une imprimante graphique, il est possible de spécifier la dimension et la couleur des caractères. Lorsque la désignation d'unité "PRT:" est spécifiée, l'imprimante connectée au port série est utilisable, tandis que lorsque "LPT:" est spécifié, l'imprimante reliée au port parallèle est utilisable.

2.3.2 Enregistreur à cassette

Le X07-BASIC peut utiliser un enregistreur à cassette ordinaire comme unité de stockage externe. En spécifiant la désignation d'unité "CASO:" ou "CASI:", un fichier peut être stocké sur ou chargé à partir de bandes cassettes. Les instructions pour l'enregistreur à cassette sont:

CLOAD, CLOAD?
CSAVE
MOTOR

2.3.3 Port série

Le X07-BASIC peut communiquer avec des unité extérieures, telles qu'un coupleur optique, à travers son port série. Il est possible de spécifier les conditions de transmission, notamment la cadence. Pour plus de détails, se reporter à "INIT #", "LOAD", "LIST" et "SAVE" au chapitre 3.

2.3.4 Cartes utilitaires

En utilisant les cartes de mémoire RAM, ROM ou RAM/ROM, disponibles en option, des programmes peuvent être chargés ou sauvegardés (sur cartes RAM ou RAM/ROM seulement) à grande vitesse. Ces cartes possèdent une alimentation par pile de soutien pour conserver leur contenu lorsqu'elles sont séparées du X07-BASIC.

2.3.5 Fichier en mémoire RAM

En X07-BASIC, la mémoire RAM possède les caractéristiques suivantes:

- Les programmes et données sont retenus même lorsque le X-07 est mis hors tension.

- Les programmes et données sont transmis à grande vitesse.

Pour utiliser le fichier en mémoire RAM, il est nécessaire de réserver une zone en mémoire à cet effet. Les fichiers stockés à cet endroit, ainsi que la place utilisée et restant disponible en nombre d'octets peuvent être vérifiés à l'aide de l'instruction DIR. Se reporter au guide de l'utilisateur pour la manière d'utiliser la mémoire RAM. Les instructions associées à la mémoire RAM sont indiquées ci-dessous. Pour plus de détails se reporter au chapitre 3.

LOAD
SAVE
DELETE
FSET
DIR
INIT #
PRINT #
INPUT #
OUT #

2.3.6 Déroulement direct

Un programme sauvegardé en mémoire RAM peut être exécuté directement sans devoir au préalable le charger dans la zone texte de la mémoire. Ceci est appelé "déroulement direct". Pour exécuter un déroulement direct d'un programme ainsi sauvegardé (PROG1), composer.

RUN''PROG1''

Puis frapper la touche

RETURN key.

Dans cet exemple, le X07-BASIC exécutera "PROGl" directement, sans au préalable le charger en zone texte de la mémoire. Il faut toutefois noter noter les points suivants:

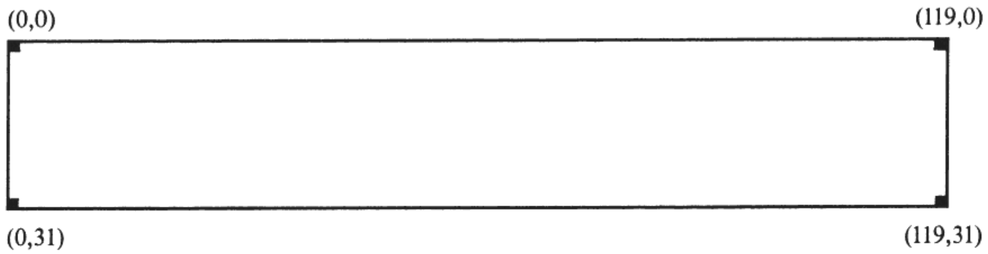
1) Lorsque l'exécution d'un programme est arrêtée par une rupture ou une suspension,l'interpréteur retourne à l'état normal, c'est à dire relié à la zone texte de la mémoire.
2) En conséquence, un programme ne peut pas être édité immédiatement après la fin de son exécution.
3) Les instructions telles que NEW et LOAD ne sont pas exécutées durant le déroulement direct d'un programme. Lorsque le programme atteint de telles instructions, le X07-BASIC suspend son exécution.

2.4 Graphisme

Le X07-BASIC possède les instructions graphiques suivantes:

1) PSET Pour tracer un point
2) PRESET Pour effacer un point
3) LINE Pour tracer un trait
4) CIRCLE Pour tracer un cercle
5) POINT Pour vérifier un point

L'écran à cristaux liquides du X07-BASIC comprend une zone graphique verticale (Y) de 32 points et horizontale (X) de 120 points, l'origine étant au coin supérieur gauche.



2.4.1 Coordonnées relatives et absolues

L'affichage graphique du X07-BASIC peut être spécifié en coordonnées absolues ou relatives. Les coordonnées absolues ont leur origine au point (0,0), soit au coin supérieur gauche de l'écran. Exécutonspar exemple l'ordre suivant.

```
LINE(0,0)—(50,0) RETURN
```

Contrairement aux coordonnées absolues, les coordonnées relatives se rapportent à la derniere position. Exécutons par exemple le programme suivant.

```
 5 CLS
10 PSET(10,10)
20 PSET STEP(10,10)
```

En X07-BASIC, les coordonnées relatives sont spécifiées en prenant le dernier point défini dans les instructions PSET, PRESET, POINT, LINE ou CIRCLE comme point de départ.

Voici quelques exemples d'instructions fixant des coordonnées relatives.

```
PSET STEP (10,10)
PRESET STEP(0,5)
LINE STEP(0,0)—STEP(20,0)
LINE STEP(15,0)—(30,30)
CIRCLE STEP(5,0),10
```

La forme générale des coordonnées relative est la suivante:

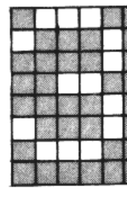
```
STEP (ΔX,ΔY)
```

Voici un autre exemple de programme. Accorder une attention spéciale aux dernières coordonnées (appelées aussi "coordonnées actuelles").

```
10 CLS
20 LINE (0,0)—(50,0)
30 CIRCLE (15,15),10
40 PSET(50,15)
50 LINE STEP(0,0)—STEP(50,0)
60 CIRCLE STEP(15,15),10
70 CIRCLE STEP(-5, -10),10
```

Attention: (0,0) sont les coordonnées actuelles juste après l'exécution de l'instruction

Chapitre 3
Ordres et Instructions



Toutes les instructions, ordres et fonctions admises par le X07-BASIC sont décrits de la manière suivante dans les chapitres 3 et 4.

Fonction: Simple explication de la fonction de l'instruction.

Format: Forme générale de l'instruction. Les règles suivantes sont applicables.

1) Les éléments en lettres majuscules doivent être entrés comme indiqué.
2) Les éléments entre crochets (⟨ ⟩) sont spécifiés par l'utilisateur.
3) Les éléments entre parenthèses carrées ([ ]) prevent être omis. Lorsque c'est le cas, le X07-BASIC les met automatiquement en place (ce sont les valeurs de défaut)
4) Tous les symboles (( ) , ; - = etc.), à l'exception des parenthèses mentionnées ci-dessus, doivent être entrés comme indiqué.
5) Les éléments suivis d'une ellipse (.....) peuvent êter entré répétés un aussi grand nombre de fois que désiré, mais le nombre total de caractères ne doit pas dépasser 255.

   Exemple: INPUT ⟨variable⟩[,⟨variable⟩...]
   ⟨variable⟩peut être répété.
   INPUT A,B$,C,D$

6) L'utilisateur peut choisir l'un des éléments indiqués sur differentes lignes verticales.

   Exemple:

IF	THEN	⟨instruction⟩
		⟨numéro de ligne⟩
	GOTO	⟨numéro de ligne⟩

   On peut ainsi choisir THEN ⟨instruction⟩
   ou
   THEN ⟨numéro de ligne⟩
   ou
   GOTO ⟨numéro de ligne⟩

Exemple: Exemples sur la manière d'utiliser une instruction.

Explication: Explication détaillée, mode d'emploi et points à observer.

Référence: Instructions similaires ou relatées.

BEEP

Fonction: Emet un ''bip'' par le haut-parleur.
Format: BEEP ⟨note⟩, ⟨durée⟩
Exemple: BEEP 100,2

Explication: lorsque la valeur de:

⟨Note⟩ =0 Pas de son
=1 à 48 augmente d'un demi-ton à partir du "do".
=49 à 4095 le son produit est de 19200/⟨note⟩ en Hz.

⟨durée⟩ = 0 à 255 Durée réelle ⟨durée⟩ ×50ms.

Exemple de programme

```
100 DEFINT A-Z
110 C=12:T=3
120 FOR I=1 TO 2
130 RESTORE 190
140 FOR J=1 TO 43
150 READ D,L
160 IF D>=0 THEN BEEP C+D,T*L ELSE BEEP
0,T*L
170 NEXT
180 NEXT
190 END
200 DATA 8,4,5,8,-1,2,5,2,8,2,10,2,3,8,-
1,4
210 DATA 3,2,5,2,6,4,13,4,-1,2
220 DATA 13,2,12,2,8,2,10,2,8,1,6,1,5,8,
-1,2
230 DATA 8,2,10,2,10,4,10,2,15,2,13,1,12
,2,13,1,10,2,8,6,-1,2
240 DATA 1,2,3,2,5,2,10,4,8,4,8,2,6,2,5,
4,0,2,1,10,-1,4
```


CIRCLE

Fonction: Trace un cercle.
Format: CIRCLE [STEP](⟨coordonnée x⟩, ⟨coordonnée y⟩), ⟨rayon⟩
Exemple: CIRCLE (98,16),8

Explication: Trace un cercle dont le centre se trouve aux coordonnées (X,Y) et le rayon est ⟨rayon⟩.
Les coordonnées et le rayon doivent être des entiers de 0 à 127.

Exemple de programme

```
100 CLS
110 FOR I=1 TO 3
120 CIRCLE(10*I,10),7
130 NEXT
140 CIRCLE STEP(-5,8),7
150 CIRCLE STEP(-10,0),7
160 END
```

CLEAR

Fonction: Initialise les variables, et définit la zone de mémoire à utiliser.
Format: CLEAR [⟨zone chaîne⟩[,⟨limite supérieure de la zone utilisateur⟩]]
Exemple: CLEAR 100, 5000

Explication: Lorsque les variables sont initialisées, il est attribué zéro à toutes les variables numériques, et chaîne nulle ("") à toutes les chaînes de caractères.

⟨zone chaîne⟩ fixe la zone en mémoire pour les chaînes de caractéres. Elle est initialement fixée à 50 octets.

⟨limite supérieure de zone utilisateur⟩ fixe l'adresse de la limite supérieure de la mémoire. Elle est initialement fixée à 8177 lorsque X-07 est utilisé seul. Lorsque elle est spécifiée, toute adresse au-delà de celle-ci est utilisée pour les programmes BASIC.

Toute la zone située au-dessus de l'adresse fixée est utilisable pour stocker les données et programmes en langage machine.

Exemple de programme

```
100 A=5
110 PRINT A
120 CLEAR
130 PRINT A
140 END
```


CLOAD

Fonction: Charge un fichier (fichier de programme) situé sur bande cassette.
Format: CLOAD [''⟨désignation de fichie⟩'']
Exemple: CLOAD ''PROG1''
(charge le programme désigné ''PROG1'')
CLOAD
(charge le premier programme rencontré sur la bande cassette.)

Explication: ⟨désignation de fichier⟩ définit le fichier de programme à charger. Tous les autres fichiers se trouvant sur la bande cassette sont ignorés. Lorsque la désignation est omise, le premier fichier rencontré sur la bande cassette est chargé.

La cadence et le mode de transmission sont préfixés à respectivement 1200 et B.

Référence: CSAVE, CLOAD?, MOTOR

CLOAD?

Fonction: Vérifie la présence d'un fichier sur bande cassette.
Format: CLOAD?[''⟨désignation de fichier⟩'']
Exemple: CLOAD?''PROG1''

Explication: Lit le fichier spécifié sur la bande cassette et le compare au programme présent en zone texte de la mémoire. S'ils sont identiques, "≥" est affiché. S'ils ne le sont pas, "Bad"est affiché.

Après avoir sauvegardé un programme par l'ordre CSAVE, utiliser CLOAD? pour s'assurer qu'il a été stocké correctement.

Lorsque ⟨désignation de fichier⟩ est omis, le premier fichier rencontré sur la bande cassette est comparé.

Référence: CSAVE, CLOAD, MOTOR

CLS

Fonction: Efface l'écran.
Format: CLS
Exemple: CLS

Explication: Le curseur est déplacé aux coordonnées (0,0), et ce point devient la position actuelle.

Exemple de programme

```
100 FOR I=1 TO 8
110 PRRINT "CANON";
120 NEXT
130 CLS
140 END
```

L'écran s'est-il effacé après avoir affiché "CANON" 8 fois?

CONSOLE

Fonction: Fixe l'état de la console
Format: CONSOLE[〈ligne de départ du roulement〉]
[,[〈nombre de lignes de roulement〉]
[,[〈commutateur d'affichage〉][,[〈commutateur de son de déclic〉][,[〈commutateur de répétition〉]]]]]
Exemple: CONSOLE 1,2,1
(Les 2è et 3è lignes sont spécifiées pour le roulement, et les touches définies par l'utilisateur sont affichées sur la 4è ligne.)
CONSOLE,,,0,1
(Le déclic émis par les touches du clavier est annulé, et la répétition automatique est mise en service).

Explication: La plage de roulement est fixée en spécifiant la 〈ligne de départ de roulement 〉(0 à 3) et le 〈nombre de lignes de roulement〉 (1 à 4).

Lorsque 〈commutateur d'affichage de touche〉 est fixé à 1, les touches définies par l'utilisateur sont affichées au bas de l'écran. Lorque il est fixé à 0, ces informations ne sont pas affichées. Lorsque ces informations sont présentes, la zone de roulement est de 1 à 3 lignes.

Lorsque 〈commutateur de son de déclic〉 est fixé à 1, les déclics sont émis, et ils ne le sont pas lorsqu'il est fixé à 0.

Lorsque 〈commutateur de répétition〉est fixé sur 1, une entrée est exécutée de manière repetée tant que la touche est maintenue enfoncée, tandis qu'un seul caractère est obtenu à chaque pression lorsque 〈commutateur de répétition〉 est fixé sur 0.

CONSOLE@

Fonction: Met en et hors service le haut-parleur d'alarme, initialise les touches définies par l'utilisateur et fixe le mode du clavier.
Format: CONSOLE@[[〈 commutateur d'alarme〉]
[,[〈définition〉][,〈mode de clavier〉]]]]
Exemple: CONSOLE@,,3
(le clavier est fixé sur mode à 10 touches.)

Explication: Le haut-parleur d'alarme est mis en fonction lorsque 〈commutateur d'alarme〉 est fixé sur 1, et hors fonction lorsqu'il est fixé sur 0.

Les caractères définis par l'utilisateur sont initialisés lorsque 〈définition〉 est fixé sur 1, et les touches définies par l'utilisateur le sont quand il est fixé sur 0.
〈mode de clavier〉 fixe le mode du clavier ainsi:
0 Mode alphanumérique
1 Mode kana (japonais)
2 Mode graphique
3 Mode 10 touches

Référence: FONT$, KEY$

CONT

Fonction: Reprend l'exécution du programme après que son déroulement ait été suspendu.
Format: CONT
Exemple: CONT

Explication: L'ordre CONT est utilisé pour rependre l'exécution du programme après que son déroulement ait été suspendu par un ordre STOP, la touche [ON] ou les touches [CTRL] + [C].
Toutefois, un programme ne peut être redémarré lorsque "Abort" est affiché.
Lors de la mise au point d'un programme, l'ordre CONT est utilisé avec l'ordre STOP, la touche [ON] ou les touches [CTRL] + [C].
Lorsqu'un programme est arrêté en déroulement direct, l'ordre CONT ne permet pas de le redémarrer.

Exemple de programme

```
100 FOR I=1 TO 10
110 PRINT I;"ENTREZ CONT"
120 STOP
130 NEXT
140 END
```

Avez-vous composé "CONT" lorsque "Break in 120" a été affiché?

CSAVE

Fonction: Sauvegarde le programme en zone texte de la mémoire sur bande cassette.
Format: CSAVE ''⟨désignation de fichier⟩''
Exemple: CSAVE ''PROG1''
(Le programme en zone texte de la mémoire est sauvegardé sur bande cassette en tant que ''PROG1''.

Explication: ⟨désignation de fichier⟩ définit le nom sous lequel le programme se trouvant en zone texte de la mémoire est sauvegardé. L'ordre CLOAD? doit être exécuté après l'ordre CSAVE pour s'assurer que le programme a été correctement sauvegardé.

La cadence et le mode de transmission sont respectivement fixés à l'origine à 1200 et "B".

Référence: CLOAD, CLOAD?, MOTOR

DATA

Fonction: Définit les valeurs devant être lues par l'instruction READ.
Format: DATA⟨constante⟩[,⟨constante⟩...]
Exemple:DATA 1, ''X-07'', ''CANON ORDINATEUR'', 3

Explication: ⟨constante⟩ peut être des constantes numériques ou des chaines de caractères. Cependant, le type de chacune doit correspondre au type de la variable qui lui est assortie dans l'instruction READ.

Exemple:

```
100 READ A,XA$,X%
.
.
160 DATA 11.2,''ALLO'',20
```

- `A` : variable du type nombre réel à double précision
- `XA$` : chaîne de caracteres
- `X%` : variable du type entier
- `11.2` : constante du type nombre réel à double précision
- `''ALLO''` : chaîne de caracteres
- `20` : constante type entier

Référence: READ, RESTORE

DEFFN

Fonction:Définit une fonction
Format: DEFFN⟨nom⟩[(⟨argument⟩[,⟨argument⟩...])]=
⟨définition de la fonction⟩
Exemple:DEFFNZ(X,Y)=X*2+Y*3+6
A=FNZ(I,J)

Explication: Le nom de la fonction est fixé par le terme FN⟨nom⟩, et le⟨nom⟩doit être un nom variable autorisé.

L'instruction DEFFN doit être exécutée avant l'utilisation de cette fonction.

⟨argument⟩ peut être identique à un nom variable d'un programme. Ils seront traités distinctement.

Exemple de programme

```
100 DEFFND(X,Y)=SQR(X^2+Y^2)
110 INPUT "X&Y";A,B
120 PRINT FND(A,B)
130 GOTO 110
```

- `DEFFND(X,Y)=SQR(X^2+Y^2)` : définition de la fonction
- `FND(A,B)` : appel de la fonction

DEFINT/SNG/DBL/STR

Fonction: Déclare les types de variable
Format: DEF⟨type⟩⟨gamme(s) de lettres⟩
Exemple: DEFINT I-N,X
(toutes les variables débutant par les lettres I à N, ou X sont définies comme du type entier.)

Explication: Les types de variables sont déclarés de la manière suivante:

DEFINT	Variables entières
DEFSNG	Variables du type nombre réel à simple précision
DEFDBL	Variables du type nombre réel à double précision
DEFSTR	Variables chaînes de caractères

⟨gamme(s) de lettres⟩ peut définir un seul caractère (par exemple A) ou une gamme de lettres (ex. C-S). Dans le dernier cas, les lettres doivent être écrites dans l'ordre logique. Par exemple, S-A n'est pas admis.

Ces déclarations sont annulées au moyen de l'instruction CLEAR.

Référence: Chapitre 1, Section 1.11 Variables

Exemple de programme

```
100 DEFINT A
110 DEFSNG B
120 DEFDBL C
130 DEFSTR D
140 A=1234150 B=1.23457E+56
160 C=1.2345678901234D+56
170 D="ABCDEF"
180 LPRINT A:P=VARPTR(A):GOSUB 230
190 LPRINT B:P=VARPTR(B):GOSUB 230
200 LPRINT C:P=VARPTR(C):GOSUB 230
210 LPRINT D:P=VARPTR(D):GOSUB 230
220 END
230 FOR I=3 TO PEEK(P-3)-1
240 LPRINT I,HEX$(PEEK(P+1))
250 NEXT
260 LPRINT
270 RETURN
```

DELETE

Fonction: Efface un fichier de la mémoire RAM.
Format: DELETE ''⟨désignation de fichier⟩''[,''⟨type de fichier⟩'']
Exemple: DELETE ''MODELE'',''P''

Explication: ⟨désignation de fichier⟩ définit le fichier à effacer.

Quand⟨type de fichier⟩ est omis, le type D (fichier de données) est automatiquement spécifié.

⟨désignation de fichier⟩ et ⟨type de fichier⟩ peuvent être définis par des chaînes de caractères.

DIM

Fonction: Déclare une ou plusieurs listes de variables.
Format: DIM ⟨nom de variable⟩(⟨valeur maximum de l'indice⟩[,⟨valeur maximum de l'indice⟩ ....])[,⟨nom de variable⟩(⟨valeur maximum de l'indice⟩[,⟨valeur maximum de l'indice⟩...])...
Exemple: DIM A$(100),B(18,16)

Explication: ⟨valeur maximum de l'indice fixe⟩la limite de l'indice des variables de la liste.

Lorsqu'une variable d'une liste est utilisée sans l'instruction DIM, ⟨valeur maximum de l'indice⟩ est fixé à 10.

Référence: ERASE

Exemple de programme

```
100 DEFINT A-Z
110 N=9
120 DIM A(N,N)
130 FOR I=1 TO N
140 FOR J=1 TO N
150 A(I,J)=I*J
160 NEXT
170 '
180 FOR J=1 TO N
190 FOR I=1 TO J
200 PRINT USING "##";A(I,J);
210 NEXT
220 PRINT
230 NEXT
240 END
```

```
1
2 0
3 0 0
4 0 0 0
5 0 0 0 0
6 0 0 0 0 0
7 0 0 0 0 0 0
8 0 0 0 0 0 0 0
9 0 0 0 0 0 0 0 0 0
```


DIR

Fonction: Affiche le contenu de la mémoire RAM.
Format: DIR[#⟨numéro de fichier⟩]
Exemple: DIR
DIR#1

Explication: Les désignations de fichiers et leur type, ainsi que les zones utilisées et disponibles en nombre d'octets sont délivrés à l'unité désignée en ⟨numéro de fichier⟩. ⟨numéro de fichier⟩ est le numéro sous lequel l'unité a été ouverte dans l'instruction INIT#. Lorsqu'il est omis, l'information est délivrée sur l'écran.

Example de programme La dimension de la mémoire RAM pour fichiers et la zone inutilisée sont indiquées en nombre d'octets à la suite des désignations de fichiers.

```
100 INIT #1,"GPR:"
110 DIR #1
120 END

RUN
RAM:
ASCII  P  JIS    P
EBCDIC P  CANON  P
1000/767
```

END

Fonction: Termine l'exécution du programme.
Format: END
Exemple: END

Explication: L'instruction END arrête l'exécution du programme, met le haut-parleur hors fonction et retourne le X07-BASIC au niveau d'attente d'ordres BASIC. ⟨numéro de fichier⟩ défini par l'instruction INIT # est supprimé par l'instruction END.

ERASE

Fonction: Elimine une ou plusieurs listes de variables
Format: ERASE ⟨nom de liste de variables⟩[,⟨nom de liste de variables⟩...]
Exemple: ERASE A, B$

Explication: Lorsqu'une liste de variables est effacée, la surface qu'elle occupait en mémoire est libérée à d'autres fins.
Lorsqu'une liste de variables est redéfinie par l'instruction DIM en utilisant le même nom, elle doit d'abord être effacée au moyen d'une instruction ERASE.

Référence: DIM

Exemple de programme

```
100 PRINT FRE(0)
110 DIM A(100)
120 PRINT FRE(0)
130 ERASE A
140 PRINT FRE(0)
150 END
```

ERROR

Fonction: Simule l'occurence d'une erreur.
Format: ERROR ⟨numéro d'erreur⟩
Exemple: ERROR 200

Explication: L'instruction ERROR est utilisée pour simuler l'occurence d'une erreur et créer un ⟨numéro d'erreur⟩ défini par l'utilisateur.

Référence: ON ERROR GOTO, ERL, ERR, RESUME, Messages d'erreur

EXEC

Fonction: Exécute un programme en langage machine.
Format: EXEC⟨adresse de départ⟩
Exemple: EXEC &H3000

Explication: Un programme en langage machine est exécuté à partir de ⟨l'adresse de départ⟩. Lorsque le programme en langage machine est un sous-programme, l'instruction suivant EXEC est exécutée.

FOR ~ TO ~ STEP ... NEXT

Fonction: Exécution répétée d'une série d'instructions
Format: FOR ⟨variable⟩ = ⟨valeur initiale⟩ TO ⟨valeur finale⟩[STEP ⟨incrémentation⟩]

:

NEXT [⟨variable⟩[,⟨variable⟩...]]
Exemple: FOR N=1 TO 10

:

NEXT N

Explication: La ⟨variable⟩ est utilisée comme un compteur. Elle est d'abord fixée à la ⟨valeur initiale⟩ et les lignes situées entre les instructions FOR et NEXT sont exécutées. Après chaque exécution d'une boucle, la ⟨variable⟩ est incrémentée du degré défini au terme STEP ⟨incrémentation⟩, puis la nouvelle valeur de ⟨variable⟩ est comparée à la ⟨valeur finale⟩. Si la nouvelle valeur est supérieure à ⟨valeur finale⟩, la boucle est terminée, et l'exécution du programme passe à l'instruction suivant l'instruction NEXT.

La boucle FOR ~ NEXT peut être imbriquée (soit une boucle à l'intérieur d'une boucle). Lorsque les boucles FOR - NEXT sont imbriquées, chacune doit avoir une variable exclusive. Lorsque ces boucles ont le même point final, il est possible d'utiliser une seule instruction NEXT.

Lorsque le terme STEP ⟨incrémentation⟩ est omis, le compteur utilise un pas d'incrémentation de 1. ⟨incrémentation⟩ peut également avoir une valeur négative.

La boucle est exécutée au moins une fois, quels que soient la valeur initiale, la valeur finale et le pas d'incrémentation.

Ne manquez pas d'essayer l'exemple de programme.

Exemple de programme

```
100 CLS
110 FOR X=1 TO 119
```



```
120 FOR Y=0 TO X MOD 32
130 PSET(X,Y)
140 NEXT
150 NEXT
160 FOR X=0 TO 119 STEP 3
170 FOR Y=0 TO X MOD 32
180 PSET(X,Y)
190 NEXT Y,X
200 END
```

FSET

Fonction: Fixe la dimension de la mémoire RAM
Format: FSET ⟨dimension de la mémoire pour fichier⟩
Exemple: FSET 1024

Explication: ⟨dimension de la mémoire pour fichier⟩ définit la dimension de la mémoire RAM pour fichier en nombre d'octets. 13 octets au minimum sont nécessaires. La zone texte de la mémoire est réduite du nombre d'octets spécifiés. La dimension et l'état de la mémoire RAM peuvent être vérifiés au moyen d'une instruction DIR.

Référence: DIR

GOSUB ~ RETURN

Fonction: Exécute un sous-programme
Format: GOSUB ⟨numéro de ligne⟩
Exemple: GOSUB 1000

Explication: Lorsqu'il arrive à une instruction GOSUB, le programme est dérivé vers un sous-programme qui débute au ⟨numéro de ligne⟩ spécifié. Lorsque l'instruction RETURN est exécutée, le programme est ramené l'instruction suivant GOSUB.

Un sous-programme peut être appelé à plusieurs reprises au cours du déroulement d'un programme. Un sous-programme peut également en appeler un autre.

Le sous-programme ordonné par une instruction GOSUB doit toujours comprendre une instruction RETURN.

Lorsque l'instruction GOSUB est utilisée en mode direct, seul le sous-programme débutant au ⟨numéro de ligne ⟩spécifié peut être exécuté.

Exemple de programme

```
100 A$="NON"
110 PRINT "CA";
120 GOSUB 160
130 GOSUB 160
140 END
150 '
160 PRINT A$
170 RETURN
```

GOTO

Fonction: Créer une rupture dans l'ordre exécution du programme.
Format: GOTO⟨numéro de ligne⟩
Exemple: GOTO 2000

Explication: Lorsque l'instruction GOTO est exécutée, le programme est branché le ⟨numéro de ligne⟩ spécifié

Lorsque l'instruction GOTO est utilisée en mode direct, le programme peut être débuté à partir du ⟨numéro de ligne⟩ spécifié.

Exemple de programme

```
100 PRINT "CANON"
110 GOTO 100
```


IF ~ THEN ~ ELSE

Fonction: Détermine la condition spécifiée par une expression logique.
Format: IF ⟨expression logique⟩
[THEN ⟨instruction⟩ | ⟨numéro de ligne⟩] [GOTO ⟨numéro de ligne⟩] [ELSE ⟨instruction⟩ | ⟨numéro de ligne⟩]
Exemple: IF A=0 THEN 100 ELSE A=0:GOTO 200

Explication: Si le résultat de ⟨l'expression logique⟩ est vrai, l'instruction THEN ou ⟨numéro de ligne⟩ ou le terme GOTO ⟨numéro de ligne⟩ est exécuté. S'il est faux, ⟨l'instruction⟩ELSE ou le terme ⟨numéro deligne⟩ est exécuté.

Lorsque le terme ELSE est omis et que le résultat est faux, l'exécution du programme passe à l'instruction suivant IF ~ THEN/GOTO.

Exemple de programme

```
100 INPUT "A=";A
110 FLG=0
120 IF A<0 THEN A=ABS(A):FLG=1
130 PRINT SQR(A);
140 IF FLG=1 THEN PRINT"1" ELSE PRINT"0"
150 END
```

INIT

Fonction: Met une unité en état de marche.
Format: INIT # ⟨numéro de fichier⟩, ''⟨descripteur de fichier⟩'' [, ⟨dimension de fichier⟩ | ⟨cadence de transmission⟩ [, ''⟨type de fichier⟩'' | ''⟨mode⟩'']]
Exemple:
```
INIT # 1, ''GPR:''
INIT # 2, ''FILE 1'' ,100, ''D''
INIT # 4, ''OPT:'' ,500, ''H''
```

Explication: Le⟨numéro de fichier⟩ est attribué au fichier, qui est alors ouvert pour le préparer à l'usage.

Une unité doit toujours être ouverte au moyen d'une instruction INIT # avant qu'elle puisse être utilisée. Pour une opération d'entrée/sortie vers/d'une unité, il est nécessaire de définir le⟨numéro de fichier⟩ qu'on a choisi pour celle-ci.

La forme générale de "descripteur de fichier" est la suivante:

"⟨désignation d'unité⟩:⟨désignation de fichier⟩".

Lorsque ⟨désignation d'unité⟩ est omis, c'est la mémoire RAM qui est automatiquement spécifiée.

Lorsqu'on utilise celle-ci, ⟨dimension de fichier⟩ et ⟨type de fichier⟩ doivent être spécifiés. En utilisant une unité telle qu'un coupleur optique ("OPT:"), ⟨cadence de transmission⟩ et ⟨mode⟩ doivent être spécifiés.

Réfénce: PRINT #, LIST #, INPUT #, au Chapitre 2

Exemple de programme
```
100 INIT#1,"GPR:"
110 DIR#1
120 END
```



```
RUN
RAM:
ASCII   P  JIS     P
EBCDIC  P  CANON   P
1000/767
```

Nota: Le résultat de cet exemple de programme diffère selon le contenu des fichiers sauvegardés en mémoire RAM et selon la valeur de FSET.

INPUT

Fonction: Affecte une donnée à une variable.
Format: INPUT[''⟨appel chaîne⟩'';]⟨variable⟩[,⟨variable⟩...]
Exemple: INPUT''VOTRE NOM'';N$

Explanation: "<appel chaîne>" et "?" sont affichés, et le X07-BASIC attend l'introduction sur le clavier. Entrer autant de données qu'il y a de variables spécifiées dans l'instruction INPUT en les séparant par des virgules. Lorsque la touche RETURN est frappée, les données sont affectées aux variables correspondantes.

Ne pas oublier que le type de chaque donnée doit correspondre à la variable à laquelle elle est assortie. Si ce n'est pas le cas, "?Redo from start" (recommencer dès le début) est alors affiché. Dans un tel cas, recomposer les données.

Lorsque le nombre de variables, ? est affiché. Dans ce cas en rajouter.

Lorsqu'il y a davantage de données que de variables, "? Extra ignored" (données superflues ignorées) est affiché. L'exécution du programme se poursuit en ignorant les données superflues.

Lors de l'introduction de chaînes de caractères avec virgule et doubles guillements, ces constantes doit être enfermées dans de doubles guillemets.

Exemple de programme

```
100 INPUT "Entrez votre nom",N$
110 PRINT "Bonjour"
120 END
```


INPUT

Fonction: Lit les données d'un fichier et les attribue à des variables.
Format: INPUT # ⟨numéro de fichier⟩,⟨variable⟩ [,⟨variable⟩...]
Exemple: INPUT # 1,A,B$

Explication: Avec l'instruction INPUT, la donnée est introduite par le clavier. Avec l'instruction INPUT #, la donnée est entrée à partir du fichier spécifié par ⟨numéro de fichier⟩.

⟨numéro de fichier⟩ est le numéro sous lequel le fichier a été ouvert au moyen de l'instruction INIT #.

Référence: INIT #, INPUT

Exemple de programme

```
100 INIT#1,"FICHIER",100,"D"
110 PRINT#1,"CANON"
120 INPUT#1,A$
130 PRINT A$
140 DELETE "FICHIER"
150 END
```

LET

Fonction: Affecte une valeur à une variable.
Format: [LET] ⟨variable⟩ = ⟨expression⟩
Exemple: LET PI=3.1415926545898
LET P2=2*PI

Explication: L'instruction LET affecte la valeur de ⟨expression⟩ à ⟨variable⟩. Le mot l'instruction "LET" peut être omis. Dans les exemples de programme, le mot "LET" est toujours omis.

LINE

Fonction: Trace un trait
Format: LINE [STEP][(⟨coordonnée X⟩,⟨coordonnée Y⟩)]-
[STEP](⟨coordonnée x⟩,⟨coordonnée y⟩)
Exemple: LINE(0,0)-(119,31)
LINE-(60,25)

Explication: L'instruction LINE trace un trait des coordonnées (⟨X,Y⟩) aux coordonnées (⟨x,y⟩)

Lorsque l'instruction "STEP" est incluse, les coordonnées sont considérées comme relatives.

Losque les coordonnées (⟨X,Y⟩) sont omises, le trait est tracé à partir des coordonnées (⟨x,y⟩) de l'instruction LINE placée immédiatement devant.

Exemple de programme

```
100 CLS:PI=3.1415926535898#
110 X=120:Y=31/2
120 LINE(X,Y)-(0,Y)
130 FOR I=1 TO X-1
140 LINE-(I,-Y*SIN(PI*I/20)+Y
150 NEXT
160 END
```

LINE INPUT

Fonction: Introduit une ligne entière.
Format: LINE INPUT [''⟨appel chaîne⟩'';]⟨variable chaîne de caractère⟩
Exemple: LINE INPUT ''QUEL EST VOTRE NOM?'';N$

Explication: Une chaîne de caractères peut être entrée sur le clavier par unité de ligne. Lorsque la touche RETURN est frappée, tous les caractères composés et se trouvant en attente sont affectés et se trouvant en attente sont affectés à ⟨variable chaine de caracteère⟩. Le nombre de caractères pouvant être introduit de la sorte est compris entre 0 et 255.

Contrairement à l'instruction INPUT, LINE INPUT accepte les virgules, ainsi que les doubles guillements ne contenant pas de de caractères. En outre, "?" n'est pas affiché.

Référence: INPUT

Exemple de programme

```
100 CLS
110 LINE INPUT A$
120 PRINT A$
130 GOTO 110
```


LINE INPUT

Fonction: Introduit une ligne entière à partir d'un fichier.
Format: LINE INPUT #⟨numéro de fichier⟩,[''⟨appel chaîne⟩'';]⟨varianle chaîne de caractère⟩
Exemple: LINE INPUT # 1, N$

Explication: Une ligne est reprise du fichier spécifié par ⟨désignation de fichier⟩ et est affectée à ⟨variable de caractères⟩. ⟨numéro de fichier⟩est le numéro sous lequel il a été ouvert dans l'instruction INIT #, et est affectee a ⟨variable chaine de caractères⟩.

Référence: LINE INPUT, INIT #

Exemple de programme

```
100 INIT#1,"KBD:
110 PRINT "ENTREZ A,B,C"
120 LINE INPUT#1,A$
130 PRINT A$
140 PRINT "ENTREZ A,B,C A NOUBEAU"
150 LINE INPUT#1,A$
160 PRINT A$
170 END
```

LIST

Fonction: Affiche la liste (complète ou partielle) d'un programme se trouvant en zone texte de la mémoire.
Format: LIST[⟨numéro de ligne de début⟩][-[⟨numéro de ligne finale⟩]]
Exemple: LIST
(Le contenu complet du programme est affiché.)
LIST 100-200
(Les lignes 100 à 200 sont affichées)
LIST 50-
Toutes les lignes de 50 jusqu'à la fin sont affichées.)
LIST-150
(Toutes les lignes du début jusqu'à la ligne 150 sont affichées.)
LIST 100
(La ligne 100 est affichée.)

Remarque: L'exécution d'un ordre LIST peut être momentanément suspendue en frappant CRTL+S.

LIST@

Fonction: Affiche, ligne par ligne, le contenu du programme se trouvant dans la zone texte de la mémoire.
Format: LIST@[⟨numéro de ligne de début⟩][-[⟨numéro de ligne finale⟩]]
Exemple: LIST@
LIST@100-200

Explication: Après qu'une ligne ait été affichée, le X07-BASIC attend une entrée par le clavier. Frapper toute touche autre que ON ou CTRL + C (par exemple RETURN) pour afficher la ligne suivante.

Lorsque la touche ON est frappée deux fois, le listage est arrêté, et un ordre peut être introduit.

Référence: LIST

LIST

Fonction: Fournit (en totalité ou en partie) le contenu d'un programme se trouvant en zone texte de la mémoire à un fichier.
Format: LIST[@]#⟨numéro de fichier⟩,[⟨numéro de ligne de début⟩][-[⟨numéro de ligne finale⟩]]
Exemple: LIST#1, 100-200
LIST@#1

Explication: L'ordre LIST délivre en sortie le contenu du programme en zone texte de la mémoire à l'affichage, tandis que LIST# délivre ce contenu au fichier défini par ⟨numéro de fichier⟩.

⟨numéro de fichier⟩ est le numéro sous lequel le fichier a été ouvert avec l'instruction INIT#.

Référence: LIST, INIT#

LLIST

Fonction: Fournit(en totalité ou en partie) la liste du programme en zone texte de la mémoire à une imprimante.
Format: LLIST[⟨numéro de ligne de début⟩][-[⟨numéro de ligne finale⟩]]
Exemple: LLIST 100-200

Référence: LIST

LOAD

Fonction: Charge un programme à partir d'un fichier en zone texte de la mémoire.
Format: LOAD''⟨descripteur de fichier⟩''[,
|⟨dimension de fichier⟩ |[,|''⟨type de fichier⟩'' |]]
|⟨cadence de transmission⟩ | |''⟨mode⟩''
Exemple: LOAD ''OPT:''

Explication: ⟨descripteur de fichier⟩ spécifie le fichier à utiliser. Lorsque la désignation d'unité est omise dans ⟨descripteur de fichier⟩, le programme est automatiquement chargé à partir de la mémoire RAM.

Référence: SAVE

LOAD?

Fonction: Vérifie le contenu d'un fichier de programme
Format: LOAD?''⟨descripteur de fichier⟩''[,
|⟨dimension de fichier⟩ |[,|''⟨type de fichier⟩'' |]]
|⟨cadence de transmission⟩ | |''⟨mode⟩''
Exemple: LOAD?''COM:''

Explication: Le fichier de programme spécifié situé dans l'unité spécifiée est comparé au programme de la zone texte de la mémoire. S'ie correspondent, "≥" est affiché. Si ce n'est pas le cas, "Bad" est affiché.

L'ordre LOAD? est utilisé pour s'assurer que le fichier de programme a été sauvegardé sans erreur.

Référence: LOAD

LOCATE

Fonction: Déplace le curseur.
Format: LOCATE⟨position horizontale⟩,⟨position verticale⟩
Exemple: LOCATE 10, 0

Explication: ⟨position horizontale⟩ doit être un entier compris entre 0 à 19, et ⟨position verticale⟩ un entier de 0 à 3. Le coin supérieur gauche est 0,0. Cependant, lorsque la fonction des touches définies par l'utilisateur est affichée, ⟨position verticale⟩ doit être un entier de 0 à 2.

Exemple de programme

```
100 CLS
110 LOCATE INT(RND(1)*20),INT(RND(1)*4)
120 PRINTT "*";
130 GOTO 100
```


LPRINT

Fonction: Délivre des données à l'imprimante graphique.
Format: LPRINT[[[⟨format de caractère⟩][,⟨couleur⟩]][,]
[USING ''⟨chaîne format⟩'';][⟨expression⟩
[,⟨expression⟩...]]
Exemple: LPRINT [N,0]''CANON'';
LPRINT [2], USING ''# # #'';A,B
LPRINT U$

Explication: ⟨format de caractère⟩ peut être compris entre 1 (le plus petit) et 16 (le plus grand), ⟨couleur⟩ peut être 0 (noir), 1 (bleu), 2 (vert) ou 3 (rouge). Remarquer que lorsque le format de caractère et la couleur sont spécifiés, ils doivent être placés entre ([ ]) et séparés par des virgules.

Pour l'utilisation du terme USING "⟨chaîne format⟩", se reporter à l'instruction PRINT USING.

Référence: PRINT, PRINT USING

Avec l'imprimante graphique X-710

La X-710 est une imprimante graphique du type traceuse, possédant 4 stylos couleurs pour l'impression de 160 caractères alphanumériques et graphiques. Pour placer la X-710 en mode graphique, lui envoyer le code ASCII 18 (DC2). Pour la remettre en mode texte, il faut lui envoyer le code ASCII 17 (DCl). On trouvera ci-après une liste des caractères de contrôle du mode graphique.

LP	Spécifie le pas d'un trait pointillé P=0 Trait plein P=1 à 15 Trait pointillé
A	Le style est levé du papier et ramené au coin gauche. Ce point devient le point de départ, et l'imprimante retourne au mode texte.
H	Le stylo est levé du papier et ramené au coin gauche.
I	La position actuelle de la plume devient le point de départ.

Dx$_1$, y$_1$, x$_2$, y$_2$,..,xn, yn A partir de la position actuelle successivement du stylo, des traits sont tracés en reliant chacun des points spécifieés.

RΔx$_1$, Δy$_1$, Δx$_2$, Δy$_2$,...,Δxn, Δyn

Déplace le stylo de (Δx$_1$, Δy$_2$)à partir de sa position actuelle.

Mx, y Le stylo est levé et déplacé à (x,y).

Cn Changement de couleur (n=0 à 3).

Sn Changement de format de caractère (n=0 à 15).

Qn Changement de l'angle des caractères (n=0 à 3).

Pc$_1$ c$_2$...cn Imprime les caractères.

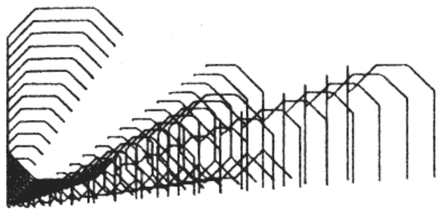
F Exécute le retour du chariot et l'interligne.

J Δx$_1$, Δy$_1$, ...Δxn,Δyn

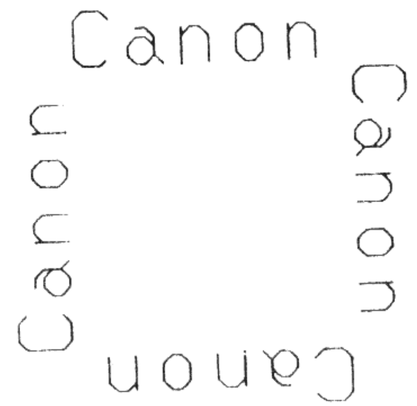
A partir de la position actuelle du stylo, des traits sont un à un tracés jusqu'au point relatif (Δxn, Δyn).

Exemple de programme

```
100 FOR I=1 TO 16
110 LPRINT[I,I MOD 4]"Can";CHR$(13);
120 NEXT
130 END
```




```
100 LPRINT CHR$(17);CHR$(18);
110 LPRINT"S4"
120 LPRINT"P     "
130 LPRINT"Q0"
140 LPRINT"PCanon"
150 LPRINT"Q1"
160 LPRINT"PCanon"
170 LPRINT"Q2"
180 LPRINT"PCanon"
190 LPRINT"Q3"
200 LPRINT"PCanon"
210 LPRINT"A"
220 END
```



```
100 DEFINT A-V
110 N=5:R=20:K=(N-1)/2
120 DIM S(N),C(N)
130 FOR I=1 TO N
140 W=2*3.1415926535898#*I/N
150 S(I)=SIN(W)*R
160 C(I)=-COS(W)*R+R
170 NEXT
180 FOR L=0 TO 3
190 LPRINT[2,L] CHR$(18)
200 FOR I=0 TO N-1
210 O=(K*I)MOD N
220 P=(O+K)MOD N
230 LPRINT "D"+STR$(C(O))+","+STR$(S(O))
+","+STR$(C(P))+","+STR$(S(P))
240 NEXT
250 LPRINT CHR$(17)
260 LPRINT
270 NEXT
280 END
```

MOTOR

Fonction: Met en marche le moteur de l'enregistreur à cassette.

Format: MOTOR [| ON | OFF |]

Exemple: MOTOR ON

Explication: Le moteur d'un enregistreur à cassette est mis en marche et arrêté par sa borne de télécommande. Si ON ou OFF est omis, le moteur est arrêté s'il était en marche, et est mis en marche s'il était arrêté. Durant CLOAD, CLOAD? et SAVE, le moteur est automatiquement mis en marche.

NEW

Fonction: Efface le programme de la zone texte de la mémoire
Format: NEW
Exemple: NEW

Remarque: Toutes les variables sont également effacées.

NEXT

Fonction: Démarque la fin d'une boucle.
Format: NEXT[⟨variable⟩[,⟨variable⟩...]]
Exemple: NEXT
NEXT K
NEXT N, J

Explication: Se reporter à l'instruction FOR. A la dernière ligne de boucles imbriquées, les deux instructions NEXT, NEXT I:NEXT J, peuvent être changées en NEXT I,J.

OFF

Fonction: Met le X-07 hors tension.
Format: OFF[| 1 | 2 |]
Exemple: OFF

Explication: OFF l: Après qu'un programme de départ ait été spécifié, l'alimentation est coupée.

OFF 2: L'alimentation du X-07 est coupée sans spécification d'un programme de départ. Lorsque le X-07 est à nouveau alimenté, aucun programme de départ n'est exécuté.

OFF: L'alimentation du X-07 est coupée sans changer l'existance d'un programme de départ.

Un programme de départ est un programme préalablement enregistré qui est exécuté lorsque le X-07 est mis sous tension. Il est spéfié au moyen de la fonction START$. Lorsque l'alimentation du X-07 est coupé les programmes et fichiers de donné en zone texte de la mémoire et en mémoire RAM sont conservé mais le contenu des variables est perdu.

Référence: START$, SLEEP

ON ERROR GOTO

Fonction: Définit la ligne de destination d'un branchement en l'occurence d'une erreur.
Format: ON ERROR GOTO ⟨numéro de ligne⟩
Exemple: ON ERROR GOTO 1000

Explication: A chaque fois qu'une erreur se produit après l'exécution d'une instruction ON ERROR GOTO, l'exécution du programme est dérivée vers une ligne spécifiée. La définition de cette instruction peut être supprimée en exécutant ON ERROR GOTO 0.

Référence: RESUME

Exemple de programme

```
100 ON ERROR GOTO 190
110 DEFFNX1(A,B,C)=((-B+SQR(B^2-4*A*C))/
2/A)
120 DEFFNX2(A,B,C)=((-B-SQR(B^2-4*A*C))/
2/A)
130 INPUT"A,B,C";A,B,C
140 PRINT A;"X^2+";B;"X+";C;"=0 FOIS"
150 PRINT "X=";FNX1(A,B,C)
160 PRINT "X=";FNX2(A,B,C)
170 GOTO 130
180 END
190 PRINT "INVALIDE"
200 RESUME 130
```

ON ~ GOSUB/ON ~ GOTO

Fonction: Branchement vers une ou plusieurs lignes en fonction de la valeur d'une expression.
Format: ON⟨expression⟩GOSUB⟨numéro de ligne$_1$⟩ [,⟨numéro de ligne$_2$⟩...]
ON⟨expression⟩GOTO⟨numéro de linge$_1$⟩ [,⟨numéro de ligne$_2$⟩...]
Exemple: ON N GOSUB 100, 130, 230, 250
ON 2-K GOTO 10, 20, 30, 40, 50

Explication: Lorsque la valeur de ⟨expression⟩ est n, l'exécution du programme est branchée sur la ligne indiquée par ⟨numéro de ligne$_n$⟩. Dans l'exemple ci-dessus, lorsque N=3, le programme saute à la ligne 230.

Lorsque la valeur de ⟨expression⟩ est supérieure à celle du numéro de ligne ou est égale à zéro, c'est l'instruction suivant ON~GOSUB/GOTO qui est exécutée.

Exemple de programme

```
100 CLS
110 PRINT"MENU"
120 PRINT"CAFE---1"
130 PRINT"THE----2"
140 PRINT"GATEAU-3"
150 INPUT"Entrer nombre";N
160 ON N GOTO 190,200,210
170 GOTO 110
180 '
190 PRINT"$0.50":END
200 PRINT"$0.50":END
210 PRINT"$2.00":END
```


OUT

Fonction: Délivre les données au port de sortie.
Format: OUT ⟨adresse du port⟩,⟨expression⟩
Exemple: OUT &H7F, &H84

Explication: La valeur de ⟨expression⟩ est délivrée au port de sortie spécifié par ⟨adresse du port⟩.

Référence: INIT#, INP

OUT

Fonction: Délivre une valeur vers un fichier.
Format: OUT # ⟨numéro de fichier⟩,⟨expression⟩
Exemple: OUT # 1, 18

Explication: ⟨numéro de fichier⟩doit être le numéro sous lequel le fichier a été ouvert par l'instruction INIT #.

Référence: INIT #,INP

Exemple de programme

```
10 FSET 1024
20 INIT#1,"ABC",30
30 FOR I=1 TO 30
40 OUT#1,I+&H20
50 NEXT I
60 CLS
70 FOR I=1 TO 30
80 PRINT CHR$(INP(#1));
90 NEXT I
100 END
```


POKE

Fonction: Inscrit une valeur numérique en mémoire.
Format: POKE⟨adresse⟩,⟨ expression⟩
Exemple: POKE, &H0C00, &HC9

Explication: La valeur de ⟨expression⟩ est inscrite à l'endroit de la mémoire spécifié par ⟨adresse⟩. L'instruction POKE permet d'inscrire toute donnée dans la zone RAM de la mémoire. Toutefois, cette instruction est à utiliser avec prudence. Appliquée de manière intempestive, la donnée ainsi inscrite pourrait ruiner le programme stocké dans la zone texte de la mémoire, ou causer une défaillance du BASIC.

Référence: PEEK

Exemple de programme

```
100 CLEAR 50,&H1000
110 FOR I=1 TO 5
120 READ A$
130 POKE I+&H1000,ASC(A$)
140 NEXT I
150 DATA C,A,N,O,N
160 FOR J=&H1001 TO &H1005
170 PRINT CHR$(PEEK(J));
180 NEXT J
190 END
```

PRESET

Fonction: Efface un point
Format: PRESET [STEP] (⟨coordonnée x⟩,⟨coordonnée y⟩)
Exemple: PRESET (40,25)

Explication: L'instruction PSET trace un point, et l'instruction PRESET l'efface.

Lorsque STEP est inclus, les coordonnées sont considérées comme relatives.

⟨coordonnée x⟩ doit être un entier compris entre 0 et 119, et ⟨coordonnée y⟩ un entier de 0 à 31.
Le coin supérieur gauche vaut (0,0).

Référence: PSET

Exemple de programme

```
100 CLS
110 FOR I=1 TO 119 STEP 2
120 FOR J=1 TO 31 STEP 2
130 PSET(I,J)
140 NEXT J
150 NEXT I
160 FOR I=1 TO 119 STEP 2
170 FOR J=1 TO 31 STEP 2
180 PRESET(I,J)
190 NEXT J,I
200 END
```


PRINT

Fonction: Délivre les données sur l'écran.
Format: PRINT [⟨expression⟩[, | ; ⟨expression⟩...]]
Exemple: PRINT ''CANON'',3*2
Format: PRINT ''CA'';''NO'';''N''
? ''CANON''

Explication: Des chaînes et des nombres sont délivrés sur l'écran.

Lorsque les ⟨expression⟩ sont séparés par des points-virgules (;), chaque valeur est imprimée en continu. Lorsque des virgules (,) sont utilisées, chaque valeur est imprimée sur une ligne différente. S'il n'y a pas de point-virgule à la fin de la ligne, le retour du curseur et l'interligne sont automatiquement exécutés aprés affichage de la dernière valeur.

Un point d'interrogation (?) peut être utilisé à la place du mot d'instruction "PRINT".

PRINT USING

Fonction: Délivre des chaînes de caractères ou ?des variables numériques selon un format spécifié.
Format: PRINT USING ''⟨chaîne de format⟩'' ;[⟨expression⟩[,⟨expression⟩...]]
Exemple: PRINT USING ''# # #'';A,B
? USING ''$$'';A

Explication: La valeur de ⟨expression⟩ est livrée en sortie selon le format spécifié en ⟨chaîne de format⟩.

⟨chaîne de format⟩ prend la forme suivante:

Pour des chaînes de caractères

! Seul le premier caractère est imprimé.

Exemple: PRINT USING "!"; "CANON"
C

& ⟨n espaces⟩ & n+2 caractères sont imprimé. Si n+2 est plus grand que le nombre de caractères, la chaîne de caractères est imprimée en justification à gauche et des espaces sont ajoutés.

Exemple: PRINT USING"&␣␣&";
"ORDINATEUR"
COMP

Pour des variables numériques

Le nombre de chiffres de la valeur numérique à afficher est spécifié par le nombre de signes #. Lorsque le nombre de # est inférieur au nombre de chiffres, % est ajouté en préfixe.

Exemple: PRINT USING "# # # #"; 1234
1234
PRINT USING "# # # #";123
123
PRINT USING "# # #";12345
%12345

. C'est le point décimal pouvant être utilisé avec #.

Exemple: PRINT USING
"# # # #.# #";123.456
123.46

* * Un double astérisque au début de ⟨chaîne de format⟩ cause l'affichage d'astérisques dans les espaces précédents

Exemple: PRINT USING "* * # # #";1234
*1234

$$ Le signe double dollar au début de ⟨chaîne de format⟩ cause l'affichage du signe dollar à la gauche du nombre.

Exemple: PRINT USING "$$# # # #";1234
$1234

* *$ Le signe double astérisque et dollar au début de ⟨chaîne de format⟩ cause l'affichage du signe dollar à la gauche du nombre et d'astérisques aux espaces précédents.

Exemple: PRINT USING "* *$# # # #";1234
* *$1234

\+ Plus au début (ou à la fin) de ⟨chaîne de format⟩ cause l'affichage du signe de la valeur (+ ou −) devant (ou après) le nombre.

Exemple: PRINT USING "+ # #,# #";1234
+1,234
PRINT USING
"+ # #.# # #"";−2.56
−2.560
PRINT USING "# # # #+";
−4567
4567−

Le signe moins à la fin de ⟨chaîne de format⟩ créer l'affichage de blancs après un nombre positif et d'un signe – après un nombre négatif.

Exemple: PRINT USING "# # # # – ";
1234
1234

^^^^ Quatre signes d'élévation à la puissance à la fin de ⟨chaîne de format⟩ cause l'affichage du nombre sous forme exponentielle.

Exemple: PRINT USING "# # # ^^^^";
1234
12E+02

PRINT

Fonction: Délivre une donnée vers un fichier.
Format: PRINT # ⟨numéro de fichier⟩,[USING "⟨chaîne de format⟩";][⟨expression⟩[| , | ; | ⟨expression⟩...]]

Exemple: PRINT # 1, "CANON"

Explication: Les instructions PRINT et PRINT USING délivrent la donnée sur l'écran, mais l'instruction PRINT # la fournit au fichier désigné par ⟨numéro de fichier⟩.

⟨numéro de fichier⟩ est le numéro sous lequel le fichier a été ouvert à l'aide de l'instruction INIT #.

Exemple de programme

```
100 INIT#2,"CON:"
110 PRINT#2,"CANON"
120 END
```

```
100 INIT#1,"GPR:"
110 PRINT#1,USING "########";1.23457E+06
120 END
```

PSET

Fonction: Trace un point.
Format: PSET [STEP] (⟨coordonnée x⟩,⟨coordonnée y⟩)
Exemple: PSET (40,25)

Remarque: La ⟨coordonnée x⟩ doit être un entier de 0 à 119 et la ⟨coordonnée y⟩ un entier de 0 à 31.
Le coin supérieur gauche vaut (0,0).

Référence: PRESET, POINT

Exemple de programme

```
100 CLS
110 X=120/2:Y=32/2:PI=3.1415926535898#
120 PSET (X,2*Y-1)
130 FOR C=0 TO 2*PI STEP .01
140 LINE-(X*SIN(2*C)+X,Y*COS(7*C)+Y)
150 NEXT
160 END
```


READ

Fonction: Lit les constantes d'une instruction DATA et les affecte à des variables.
Format: READ ⟨variabe⟩[,⟨variable⟩...]
Exemple: READ A, I, M$

Explication: L'instruction READ doit toujours être utilisée associée avec l'instruction DATA.

Les constantes sont affectées une par une aux ⟨variable⟩s. Le type des variables et des constantes doit correspondre.

Référence: RESTORE, DATA

Exemple de programme

```
100 FOR I=1 TO 5
110 READ A
120 PRINT A
130 NEXT I
140 RESTORE 190
150 FOR I=1 TO 6
160 READ A$
170 PRINT A$;
180 NEXT I
190 END
200 DATA 1,2,3,4,5,"CANON"
```

REM

Fonction: Introduit une remarque.
Format: REM [⟨remarque⟩]
Exemple: REM CALCUL DES PRIX
' VITESSE DE CALCUL

Explication: L'instruction REM n'est pas exécutée. Elle est utilisée pour insérer des commentaires afin d'identifier ou apporter des précisions sur une partie de programme. On peut utiliser un apostrophe (') à la place du mot "REM".

RESTORE

Fonction: Réutiliser un jeu de données.
Format: RESTORE [⟨numéro de ligne⟩]
Exemple: RESTORE 1000

Explication: Après exécution de l'instruction RESTORE, la prochaine instruction READ affecte les constantes de l'instruction DATA définie pa ⟨numéro de ligne⟩à ses variables.

Lorsque ⟨numéro de ligne⟩ est omis, la première instruction DATA rencontrée est utilisée.

RESUME

Fonction: Reprend l'exécution du programme après la procédure de rétablissement d'une erreur.
Format: RESUME[| NEXT | ⟨numéro de ligne⟩ |]
Exemple: RESUME 100

Explication: L'instruction RESUME est utilisée avec l'instruction ON ERROR GOTO.

RESUME — L'exécution est reprise à partir de l'instruction ayant causé l'erreur.

RESUME NEXT — L'exécution est reprise à partir de l'instruction suivant celle ayant causé l'erreur.

RESUME ⟨numéro de ligne⟩ — L'exécution est reprise à partir de l'instruction portant le ⟨numéro de ligne⟩ spécifié.

Référence: ON ERROR GOTO

RETURN

Fonction: Indique la fin d'un sous-programme.
Format: RETURN
Exemple: RETURN

Explication: L'instruction RETURN doit être placée à la fin d'un sous-programme. Lorsque l'instruction RETURN est exécutée, le programme reprend à partir de l'instruction suivant GOSUB qui a causé l'appel du sous-programme.

RUN

Fonction: Exécute un programme dans la zone tête de la mémoire.
Format: RUN[⟨numéro de ligne⟩]
Exemple: RUN
(L'exécution commence au plus petit numéro de ligne.)
RUN 1000
(L'exécution commence à la ligne 1000.)

Explication: L'ordre RUN initialise les variables (les variables numériques sont remises à zéro, et les chaînes de caractères sont annulées) avant l'exécution du programme.

RUN "descripteur de fichier"

Fonction: Exécute un programme situé en mémoire RAM.
Format: RUN ''[RAM:]⟨désignation de fichier⟩''[,''⟨type de fichier⟩'']
Exemple: RUN ''RAM: FILE 1''

Explication: L'ordre RUN "descripteur de fichier" exécute le programme spécifié en ⟨désignation de fichier⟩ situé dans la mémoire RAM.

Si le programme exécuté par cet ordre contient ne instruction LOAD ou NEW, une erreur "IR Error" est déclenchée.

De même, s'il comprend une instruction DELETE, le programme à supprimer doit avoir été sauvegardé après son exécution.

Lorsque le programme est terminé ou suspendu, le X07-BASIC retourne à l'état normal de référence à la zone texte de la mémoire.

SAVE

Fonction: Sauvegarde un programme présent en zone texte de la mémoire sur un fichier.
Format: SAVE ''⟨descripteur de fichier⟩''
[,|⟨dimension de fichier⟩ |⟨cadence de transmission⟩|[,|"type de fichier" |''⟨mode⟩''|]]
Exemple: SAVE ''FILE 1''
SAVE ''OPT:'' ,

Explication: Le programme est transféré au fichier spécifié en ⟨descripteur de fichier⟩.

Lorsque ⟨désignation d'unité⟩ est omise, le programme est transféré à la mémoire RAM.

Si ⟨type de fichier⟩ est omis lorsque le programme est transféré en mémoire RAM, il est automatiquement fixé à "P".

Référence: LOAD

SLEEP

Fonction: Met le X-07 hors tension en conservant l'affichage, les variables, etc.
Format: SLEEP
Exemple: SLEEP

Explication: L'instruction SLEEP conserve les variables, l'affichage, l'état des fichiers, qui sont normalement supprimés lorsque l'alimentation du X-07 est coupée de manière ordinaire.

Référence: ALM$

Exemple de programme

```
10 ALM$="/////1"
20 TIME$=":0.55"
30 SLEEP:PRINT "BONJOUR "
40 AL0$="1901"
50 END
```

STOP

Fonction: Termine l'exécution du programme.
Format: STOP
Exemple: STOP

Explication: Lorsque l'instruction STOP est exécutée, l'affichage suivant est obtenu:

Break in nnn (arrêt sur nnn)

"nnn" indique la ligne sur laquelle l'exécution du programme a été arrêtée.

L'exécution du programme peut être reprise au moyen de l'ordre CONT.

Référence: CONT

Exemple de programme

```
100 FOR I=1 TO 100
110 PRINT I
120 IF(I MOD 5)=0 THEN PRINT "ENTREZ
CONT":STOP
130 NEXT I
140 END
```


TROFF

Fonction: ARRETE le mode d'exécution pas à pas.
Format: TROFF
Exemple: TROFF

Explication: L'ordre TROFF est utilisé pour ARRETER le mode d'exécution pas à pas qui avait été amorcé par l'ordre TRON. L'ordre NEW est également utilisable pour terminer ce mode.

TROFF est utilisé avec l'ordre TRON pour la mise au point d'un programme.

Référence: TRON

TRON

Fonction: Affiche la ligne exécutée.
Format: TRON [#⟨numéro de fichier⟩]
Exemple: TRON
TRON#1

Explication: Les numéros de ligne d'un programme en cours d'exécution sont fournis entre ([ ])aux unités spécifiées.

Lorsque ⟨ numéro de fichier⟩ est omis, les numéros de ligne sont affichés sur l'écran.

Ce type d'état est appelé "mode d'exécution pas à pas".

Référence: TROFF

Chapitre 4
Fonctions Incorporées

ABS

Fonction: Fournit la valeur absolue d'une expression numérique.
Format: ABS(⟨expression numérique⟩)
Exemple: PRINT ABS(– 1.8)
A=ABS(– 123)

ALM$

Fonction: Règle l'alarme.
Format: ALM$=''[⟨année⟩], [⟨mois⟩],[⟨ jour⟩],[⟨ jour de la semaine⟩],[⟨heures⟩],[⟨minutes⟩]''
(''/'' ou '':''sont utilisables à la place de '' , '').
Exemple: CONSOLE @1
(met l'alarme en service)
ALM$=''1983//, &H13,8:15''
(L'alarme est réglée à 8:15 du matin les jeudi, vendredi et samedi de l'an 1983.)
A$=''/////0'':ALM$=A$
(L'alarme est réglée pour sonner à toutes les heures.)
PRINT ALM$

Explication: Lorsque les paramètres sont omis, l'alarme n'est pas réglée, "*" est affiché et l'instruction PRINT ALM$ est exécutée.

⟨année⟩ doit être un nombre de 4 chiffres (ex. 1983).

⟨heures⟩ doit être compris entre 0 et 23 (la fonction d'alarme compte les heures sur 24 h.)

⟨minutes⟩doit être compris entre 0 et 59.

⟨jour de la semaine⟩ est réglé comme suit:

1) 1 est introduit pour le bit correspondant au jour de la semaine, et 0 pour les autres bits.

Exemple:

MSB							LSB
0	1	1	0	0	0	1	0
	DIM	LUN	MAR	MER	JEU	VEN	SAM

2) Dans l'exemple ci-dessus, dimanche, lundi et vendredi sous forme binaire sont: 01100010, c'est-à-dire &H62 en forme hexadécimale, 98 en forme décimale et &142 en forme octale.

Si l'alimentation du X-07 est coupée, elle sera automatiquement enclenchée.
L'alarme retentit pendant 1 minute, et peut être arrêtée en frappant la touche CTRL + S.

Pour utiliser l'alarme, il faut d'abord la mettre en service par l'instruction CONSOLE@.

L'heure réglée pour l'alarme peut être vérifiée par l'instruction PRINT ALM$.

Référence: CONSOLE @.

ASC

Fonction: Produit le code du premier caractère d'une expression en chaîne.
Format: ASC (⟨expression en chaîne⟩)
Exemple: A=ASC(''A'')
PRINT ASC(CHR$(70))

Explication: Un message d'erreur est affiché lorsque la valeur de l'⟨expression en chaîne⟩ est nulle.

ATN

Fonction: Produit la valeur d'arctangente d'une expression numérique en radians.
Format: ATN (⟨expression numérique⟩)
Exemple: A=ATN(0.5)

CDBL

Fonction: Convertit la valeur d'une ⟨expression numérique⟩ en constante de type nombre réel double précision.
Format: CDBL (⟨expression numérique⟩)
Exemple: A # =CDBL(B!)

Explication: Les valeurs numérique de type entier et à simple précision sont converties au type à double précision.

CHR$

Fonction: Produit le caractère spécifié par un code caractère.
Format CHR$ (⟨code caractère⟩)
Exemple: PRINT CHR$(&H41)
("A" est affiché.)

Explication: La fonction CHR$ est couramment utilisée pour les caractères spéciaux. Par exemple, le code retour du curseur peut être produit en utilisant PRINT CHR$ (&H0D).

Se reporter au "tableau des codes de caractères" en page 174.

CINT

Fonction: Convertit la valeur d'une ⟨expression numérique⟩ en constante entière.
Format: CINT (⟨expression numérique⟩)
Exemple: A%=CINT(B#)

Explication: Les constantes de type nombre réel sont tronquées en constantes entières.

Référence: FIX, INT

Exemple de programme

```
100 LPRINT "     X CINT FIN INT"
110 FOR X=-1 TO 1.2 STEP .2
120 LPRINT USING "###.#  ### ### ###";X,
CINT(X),FIX(X),INT(X)
130 NEXT
140 END
```

```
RUN
    X CINT FIN INT
-1.0   -1  -1  -1
-0.8    0   0  -1
-0.6    0   0  -1
-0.4    0   0  -1
-0.2    0   0  -1
 0.0    0   0   0
 0.2    0   0   0
 0.4    0   0   0
 0.6    0   0   0
 0.8    0   0   0
 1.0    1   1   1
 1.2    1   1   1
```


COS

Fonction: Produit la valeur du cosinus d'une ⟨expression numérique⟩ en radians
Format: COS (⟨expression numérique⟩)
Exemple: A=COS(3.1415926535898/3)

CSNG

Fonction: Convertit la valeur d'une expression numérique en une constante de type nombre réel à simple précision.
Format: CSNG (⟨expression numérique⟩)
Exemple: A!=CSNG(B#)

Explication: Les valeurs entières et les valeurs type nombre réel à simple précision sont converties en valeur type nombre réel à double précision.

CSRLIN

Fonction: Rappelle la position verticale du curseur sur l'écran.
Format: CSRLIN
Exemple: Y=CSRLIN

Remarque: La ligne supérieure est le zéro.

DATE$

Fonction: Règle le calendrier.
Format: DATE$=''[⟨année⟩]/[⟨mois⟩]/[⟨jour⟩]''
Exemple: DATE$=''1983/8/22''
DATE$=''//31''
PRINT DATE$

Explication: Règle l'année, le mois et le jour du calendrier. Lorsque les paramètres ne sont pas précisés, la date réglée précédemment est conservée. L'année doit être formée de 4 chiffres (par exemple 1983). L'instruction PRINT DATE $ est utilisée pour afficher l'année, le mois et le jour de la semaine. La plage d'entrée va du ler janvier 1901 au 31 décembre 2099. Lorsque le X-07 est remis à zéro au moyen du commutateur Reset, la date est fixée au ler janvier 2000.

ERL

Fonction: Retourne au numéro de ligne du programme où une erreur s'est produite.
Format: ERL
Exemple: L=ERL

Remarque: Lorsqu'une erreur se produit en mode direct, "65535" est produit.

Référence: ON ERROR GOTO, ERR

Exemple de programme:

```
100 ON ERROR GOTO 140
110 PPRINT"X-07"
120 PRINT"CANON"
130 END
140 PRINT"ERL=";ERL
150 RESUME NEXT
```

ERR

Fonction: Produit le code d'erreur explicitant l'erreur qui est apparue.
Format: ERR
Exemple: C=ERR

Explication: La fonction ERR est utilisée avec l'instruction ON ERROR GOTO et la fonction ERL lors du traitement d'erreurs.

Référence: ON ERROR GOTO, ERL

Exemple de programme

```
100 ON ERROR GOTO 140
110 INPUT"ERREUR NO.";N
120 ERROR N
130 END
140 PRINT"EN LINE";ERL
150 PRINT"UNE ERREUR S'EST PRODUITE."
160 PRINT"DONT LE CODE EST";ERR
170 RESUME 110
```


EXP

Fonction: Produit la valeur de l'exponenrielle d'une expression numérique
Format: EXP(⟨expression numérique⟩)
Exemple: E=EXP(1)

FIX

Fonction: Produit la partie entière d'une expression numérique
Format: FIX (⟨expression numérique⟩)
Exemple: A=FIX(B)

Remarques: La partie décimale de ⟨expression numérique⟩ est tronquée.

Référence: INT

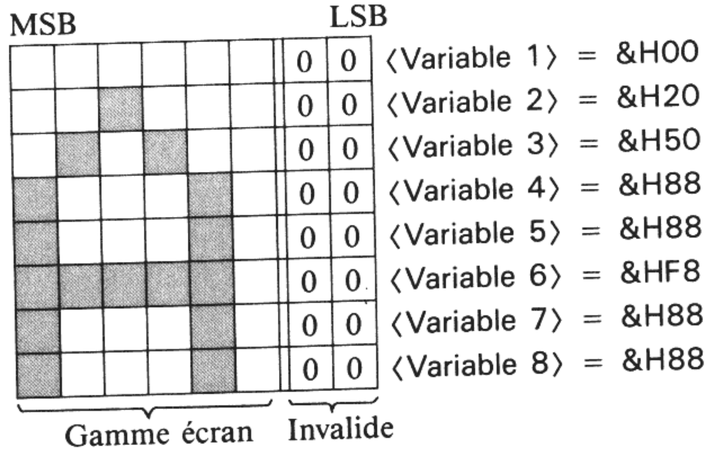
FONT$

Fonction: Spécifie un caractère défini par l'utilisateur.
Format: FONT$(⟨code de caractère⟩) = ''⟨variable$_1$⟩, ⟨variable$_2$⟩,⟨variable$_3$⟩,⟨variable$_4$⟩,⟨variable$_5$⟩, ⟨variable$_6$⟩,⟨variable$_7$⟩,⟨variable$_8$⟩''
Exemple:
```
FONT$(128)=''56, 56, 16, 124, 144, 40, 68,
132''
PRINT CHR$(128)
PRINT FONT$(128)
```

Explication: Les caractères dont les codes sont compris entre &H80 et &H9F, et entre &HE0 et &HFF sont appelés "caractères définis par l'utilisateur". Ils peuvent être librement définis par l'utilisateur.

Un caractère est composé de⟨variable$_1$⟩à⟨variable$_8$⟩. Se reporter à l'exemple ci-dessous.

Le contenu d'une fonction FONT$ peut être vérifié au moyen d'une instruction PRINT FONT$(code de caractère).



Exemple de programme

```
100 FONT$(128)="56,56,16,124,144,40,68,1
32"
110 PRINT FONT$(128)
```

```
120 PRINT CHR$(128)
130 CONSOLE@,1
140 PRINT CHR$(128)
150 END
```


FRE

Fonction: Donne la zone de mémoire disponible en nombre d'octets
Format: FRE(⟨expression numérique⟩)
FRE(⟨expression en chaîne⟩)
Exemple: PRINT FRE (A$)

Explication: FRE (⟨expression numérique⟩) affiche la grandeur de la zone texte disponible en octets, tandis que FRE (⟨expression en chaîne⟩) affiche la grandeur de la zone chaîne disponible en octets. ⟨Expression numérique⟩ et ⟨expression en chaîne⟩ peuvent prendre n'importe quelle forme.

HEX$

Fonction: Produit la chaîne représentant la valeur hexadécimale d'une expression numérique.
Format: HEX$(⟨expression numérique⟩)
Exemple: PRINT HEX$ (255)

INKEY$

Fonction: Produit le caractère de la touche frappée.
Format: INKEY$
Exemple: A$=INKEY$

Explication: Lorsqu'aucune touche n'est frappée, une chaine nulle est produite.

INP

Fonction: Produit une donnée EXTRAITE d'un fichier ou issue d'une adresse de port.
Format: INP (#⟨numéro de fichier⟩)
INP(⟨adresse port⟩)
Exemple: A=INP(&H84)
B=INP(#2)

Référence: SNS, OUT

INSTR

Fonction: Recherche la présence de la chaîne 2 dans la chaîne 1, et si elle est découverte, produit la position de l'occurence.
Format: INSTR ([⟨expression numérique⟩,]⟨chaîne 1⟩,⟨chaîne 2⟩)
Exemple: N=INSTR(B$,"N")

Explication: Si la ⟨chaîne 2⟩n'est pas découverte dans la ⟨chaîne 1⟩, zéro est produit.

L' ⟨expression numérique⟩ fixe la position du début de la recherche. Si elle est omise, la recherche commence au premier caractère de la ⟨chaîne 1⟩.

INT

Fonction: Produit le plus grand entier relatif immédiatement inférieur à l'expression numérique.
Format: INT (⟨expression numérique⟩)
Exemple: PRINT INT(56.41)
56
(56<56.41)
PRINT INT (−1.2)
−2
(−2<−1.2)

Explication: Lorsque la valeur de ⟨expression numérique⟩est négative, le résultat est différent de la fonction FIX. Par exemple, si la valeur de l'⟨expression numérique⟩est −1.2, celle de la fonction est de −2, et celle de la fonction FIX de −1.

Référence: CINT, FIX

KEY$

Fonction: Définit le contenu d'une touche fonction par l'utilisateur.
Format: KEY$(⟨numéro de touche⟩) = ''⟨expression en chaîne⟩''
Exemple: KEY$(3)=''RUNRUN''+CHR$(13)

Explication: ⟨numéro de touche⟩ définit la touche de fonction à définir, qui doit être de 1 à 12.

Les 3 premiers caractères de la chaîne étant les caractères qui seront affichés au bas de l'écran lorsque cela est spécifié par une instruction CONSOLE, ils ne font pas partie de la fonction ainsi définie. Ainsi, ces trois caractères doivent toujours être définis. Le reste de la chaîne est entré lorsque la touche de fonction spécifiée est frappée.

Les touches définies par l'utilisateur 1 à 5 et 7 à 11 possèdent 3 caractères d'affichage et jusqu'à 38 caractères propres. Pour les touches 6 et 12, bien qu'ils ne soient pas affichés, les 3 caractères doivent être spécifiés en tant que caractères fictifs dans le format, de la même manière que pour les autres touches définies par l'utilisateur.

La fonction CHR$ peut faire partie de la fonction KEY$.

Le contenu d'une touche définie par l'utilisateur peut être vérifié au moyen de l'instruction PRINT KEY$ (⟨numéro de touche⟩)

Référence: CONSOLE

Exemple de programme:

```
100 CONSOLE ,,1
110 DEFSTR C
120 CR=CHR$(13)
130 C="run"+"RUN"+CR
140 KEY$(1)=C
150 FOR I=1 TO 12
160 PRINT KEY$(I)
170 NEXT I
180 CONSOLE 0,4,0
190 END
```

LEFT$

Fonction: Produit une chaîne composée des caractères situés à extrême gauche d'une chaîne.
Format: LEFT$(⟨chaîne⟩,⟨longueur⟩)
Exemple: PRINT LEFT$(''CANON'', 3)

Explication: ⟨longueur⟩définit le nombre de caractères à produire. Lorsque ⟨chaîne⟩ est plus court que ⟨longueur⟩, la chaîne entière est produite. Lorsque ⟨longueur⟩est 0, une chaîne nulle est produite.

Exemple de programme

```
100 DEFSTR S
110 S=":"
120 A$="ABCDEFGH"
130 FOR I=1 TO 8
140 PRINT MID$(A$,I,3);S;RIGHT$(A$,I);S;
LEFT$(A$,I)
150 NEXT
160 END

 RUN

ABC:H:A
BCD:GH:AB
CDE:FGH:ABC
DEF:EFGH:ABCD
EFG:DEFGH:ABCDE
FGH:CDEFGH:ABCDEF
GH:BCDEFGH:ABCDEFG
H:ABCDEFGH:ABCDEFGH
```


LEN

Fonction: Produit la longueur d'une chaîne.
Format: LEN (⟨chaîne⟩)
Exemple: PRINT LEN (''CANON'')
5
(CANON comporte 5 caractères)

LOG

Fonction: Produit le logarithme naturel d'une expression numérique
Format: LOG (⟨expression⟩)
Exemple: A=LOG(B)

Remarque: La base du logarithme naturel est e.

MID$

Fonction: Produit une chaine composée d'un nombre spécifié de caractères à partir d'une chaîne.
Format: MID$(⟨chaîne⟩,⟨expression 1⟩[,⟨expression 2⟩])
Exemple:
```
100 A$="CANON"
110 B$="X07-BASIC EST AMUSEMENT."
120 PRINT A$;MID$(B$,10,15)
```
("CANON EST AMUSEMENT." est imprimé.)

Explication: Le nombre de caractères de ⟨expression 2⟩ à partir de la position ⟨expression 1⟩ de ⟨chaîn⟩ est produit. Lorsque ⟨expression 2⟩ est omis, tous les caractères situés à droite de ⟨expression l⟩ deviennent la valeur de MID$.

PEEK

Fonction: Produit le contenu de la mémoire.
Format: PEEK (⟨adresse⟩)
Exemple: A=PEEK(&H0)

Explication: Produit le contenu de l'⟨adresse⟩ spécifiée.

Référence: POKE

POINT

Fonction: Vérifie l'existance d'un point.
Format: POINT[STEP](⟨coordonnée X⟩,⟨coordonnée Y⟩)
Exemple: A=POINT(110,10)

Explication: −1 est produit s'il existe un point aux ⟨coordonnée X⟩,⟨coordonnée Y⟩. 0 est produit s'il n'existe pas de point à ces coordonnées. Lorsque le mot d'instruction "STEP" est inclu, les coordonnées sont considérées comme des coordonnées relatives.

Référence: PSET, PRESET

POS

Fonction: Ramène la position du curseur.
Format: POS (⟨expression⟩)
Exemple: POS(0)

Explication: ⟨expression⟩est une expression fictive, qui peut être n'importe quoi.

Le coin supérieur gauche correspond à 0.

Référence: CSRLIN

RIGHT$

Fonction: Produit une chaîne de longueur spécifiée, composée des caractères extrême gauches d'une chaîne.
Format: RIGHT$(⟨chaîne⟩,⟨longueur⟩)
Exemple: PRINT RIGHT$(''CANON'',3)
(''NON'' est produit.)

Explication: Produit le nombre ⟨longueur⟩ de caractères à partir de la droite d'une⟨chaîne⟩. Est sinon identique à la fonction LEFT$.

Référence: LEFT$, MID$

RND

Fonction: Produit des nombres aléatoires
Format: RND (⟨expression numérique⟩)
Exemple: A=RND(0)

Explication: Produit des nombres aléatoires entre 0 et l. Les nombres aléatoires produits sont différents en fonction de la valeur de ⟨expression numérique⟩. Lorsque la valeur de ⟨expression numériqu⟩est:

négative ... la valeur de l'⟨expression numérique⟩est prise comme "semence" et une nouvelle séquence de nombres aléatoires est créée.

0 ..... la "semence" est automatiquement produite, et une nouvelle séquence de nombres aléatoires est créée.

positive .... le prochain nombre aléatoire est produit.

Exemple de programme

```
100 DEFINT I,K,N:CLS
110 DIM N(119)
120 FOR I=1 TO 1200
130 K=240*(RND(1)-.5)*(RND(1)-.5)+60
140 IF N(K)<32 THEN N(K)=N(K)+1
150 PSET(K,32-N(K))
160 NEXT
170 END
```


SCREEN

Fonction: Produit le code d'un caractère et l'affiche sur l'écran.
Format: SCREEN (⟨position horizontale⟩,⟨position verticale⟩)
Exemple: PRINT SCREEN (2,10)

Explication: Produit le code d'un caractère situe aux ⟨position horizontale⟩,⟨position verticale⟩spécifiées de l'écran. Le code de caractère du coin supérieur gauche de l'écran peut être produit par SCREEN (0,0).

SGN

Fonction: Produit −1, 0 ou 1 suivant la valeur d'une expression numérique.
Format: SGN(⟨expression numérique⟩)
Exemple: PRINT SGN(−1.2)

Explication: Lorsque la valeur de expression numérique est:
<0 −1 est produit
=0 0 est produit
>0 1 est produit.

SIN

Fonction: Produit la valeur du sinus d'une expression numérique en radians
Format: SIN (⟨expression numérique⟩)
Exemple: A=SIN(3.14592635898/3)

SNS

Fonction: Introduit des données à partir d'une unité d'entrée
Format: SNS(#⟨numéro de fichier⟩[,⟨expression⟩])
Exemple: A=SNS(#1,&HFF)

Explication: #⟨numéro de fichier⟩ définit l'unité.

Lorsque les données ne sont pas entrées à partir du fichier, la valeur de ⟨expression⟩ est produite. Lorsque ⟨expression⟩ est omise, 0 est produit.

Référence: INP, OUT

Exemple de programme

```
10 INIT#1,"COM:"
20 INIT#2,"CON:"
30 A=SNS(#1)
40 IF A THEN OUT#2,A
50 B=SNS(#2)
60 IF B THEN OUT#1,B
70 GOTO 30
80 END
```


SQR

Fonction: Produit la valeur de la racine carrée d'une expression numérique.
Format: SQR (⟨expression numérique⟩)
Exemple: A=SQR(2)

START$

Fonction: Définit le programme de départ.
Format: START$=[+]⟨expression en chaîne⟩
Exemple: START$="RUN"+CHR$(13)
(le programme se trouvant en zone texte de la mémoire est exécuté lorsque le X-07 est mis sous tension)
PRINT START$

Explication: Le programme défini en tant que programme de départ est exécuté lorsque le X-07 est mis sous tension. Toutefois, le programme de départ doit être disponible avant que l'alimentation du X-07 ne soit coupée.

L'instruction PRINT START$ peut être utilisée pour afficher le contenu du programme de départ.

Avec START$=+⟨expression en chaîne⟩,⟨expression⟩ est accolé avec le START$ précédent. Grâce à cette méthode, il est possible de définir 551 caractères. Avec l'instruction PRINT START$, 255 caractères au maximum peuvent être affichés.

Référence: OFF

STICK

Fonction: Produit l'état du curseur.
Format: STICK (⟨expression⟩)
Exemple: A=STICK(0)

Explication: Donne les valeurs suivantes:

0 Lorsqu'aucune touche n'est enfoncée.
1 Lorsque ↑ est enfoncé
2 Lorsque ↑→ sont enfoncés simultanément
3 Lorsque→ est enfoncé
4 Lorsque↓→ sont enfoncés simultanément
5 Lorsque ↓ est enfoncé
6 Lorsque↓← sont enfoncés simultanément
7 Lorsque← est enfoncé
8 Lorsque ↑← sont enfoncés simultanément

⟨expression⟩est fictif, et peut être tout entier compris entre 0 et 255.

La mémoire tampon du clavier est effacée lorsque cette fonction est utilisée.

STR$

Fonction: Convertit une valeur numérique en une chaîne de caractères.
Format: STR$ (⟨expression numérique⟩)
Exemple: A$=STR$(123)

Remarques: La valeur numérique de ⟨expression numérique⟩ est convertie en une chaîne de caractères

STRIG

Fonction: Produit l'état de la touche d'espacement et de la touche définie par l'utilisateur
Format: STRIG (⟨expression⟩)
Exemple: A=STRIG(0)

Explication: Lorsque la valeur de ⟨expression⟩ est fixée à 0, l'état de la touche d'espacement est produit. Lorsque la valeur de ⟨expression⟩ est fixée à 1, l'état de la touche 6 définie par l'utilisateur est produit.

– 1 est produit lorsque la touche spécifiée est enfoncée, et 0 lorsque celle-ci n'est pas enfoncée. La mémoire tampon du clavier est effacée lorsque cette fonction est utilisée.

STRING$

Fonction: Produit une chaîne de caractères d'une longueur choisie
Format: STRING$(⟨expression numérique$_1$⟩, | ⟨expression en chaîne⟩ | ⟨expression numérique$_2$⟩ |)
Exemple: PRINT STRING$ (5, ''A'')

Explication: ⟨expression numérique⟩ fixe la longueur de la chaîne. Une chaîne se compose d'une suite de mêmes caractères choisis par l'utilisateur.

Lorsque⟨ expression en chaîne⟩est fixé, le premier caractère de la chaîne est utilisé pour la former. Lorsque⟨expression numérique⟩est fixé, la valeur de ⟨expression numérique⟩est considérée comme un code de caractère, et le caractère déterminé par ce code est utilisé pour former la chaîne.

TAB

Fonction: Fixe la tabulation
Format: TAB(⟨expression numérique⟩)
Exemple: PRINT TAB(8); ''CANON''

Explication: L'impression se fait à la position spécifiée par ⟨expression numérique⟩en délivrant des espaces en sortie. Toutefois, en ce qui concerne l'écran, le curseur est déplacé sans formation d'espaces.

Les positions de l'écran sont classees a partir de la gauche elles valent 0, 1, 2, 3, ...

TAN

Fonction: Produit la valeur de la tangente d'une expression numérique en radians.
Format: TAN (⟨expression numérique⟩)
Exemple: A=TAN(3.1415926535898)

TIME$

Fonction: Règle les heures, minutes et secondes de l'horloge.
Format: TIME$=''[⟨heures⟩]:[⟨minutes⟩]:[⟨secondes⟩]''
Exemple: TIME$=''23:53:25''
TIME$=''':5''
PRINT TIME$

Explication: L'horloge est réglée en mode de 24 heures.

Lorsqu'un paramètre est omis, la valeur réglée précédemment est conservée.

L'heure réglée peut être vérifiée au moyen de l'instruction PRINT TIME$.

TKEY

Fonction: Donne l'état d'une touche.
Format: TKEY (⟨chaîne⟩)
Exemple: A=TKEY (''A'')

Remarques: −1 est produit lorsque la touche spécifiée par la valeur de la⟨chaîne⟩ est enfoncée. Si elle n'est pas enfoncée, 0 est produit.
Spécifier un ⟨caractère⟩ ne nécessitant pas la pression de la touche de majuscule en mode alphanumérique. La mémoire tampon du clavier est annulée lorsque cette fonction est utilisée.

USR

Fonction: Appelle un sous-programme en langage machine.
Format: USR (⟨adresse de départ⟩,⟨argument⟩)
Exemple: A=USR(&H0C00,B)
A$=USR(N, A$)

Explication: Le contrôle est transféré au sous-programme en langage machine qui commence par l'⟨adresse de départ⟩et possède l'⟨argument⟩ spécifié, et les valeurs de ce sous-programme sont produites.

Le transfert de l'⟨argument⟩ est opéré avec les registres HL, A et DE. Les types d'argument peuvent être fixés avec le registre A.

Le sous-programme à appeler doit d'abord avoir été rédigé en langage machine à partir de l'⟨adresse de départ⟩.

Lorsque ⟨argument⟩ est une variable de type entier:
Régistre A=2

Contenu de la mémoire

Adresse indiquée par le registre HL→		
+1		
+2	Octet inférieur	Valeurs numériques
+3	Octet supérieur	

Lorsque ⟨argument⟩ est une variable de type nombre réel à simple précision:
Registre A=4

Contenu de la mémoire

Adresse indiquée par le registre HL→		
	Exposant	
+1	Chiffre supérieur	
+2		
+3		Chiffre inférieur

Valeurs numériques

Lorsque ⟨argument⟩ est une variable de type nombre réel à double précision:
Registre A = 8

Adresse indiquée par le registre HL→		
	Exposant	
+1	Chiffre supérieur	
+2		
+3		
+4		
+5		
+6		
+7		Chiffre inférieur

} Valeurs numériques

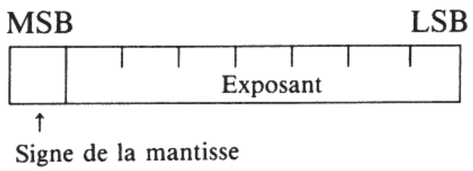
Lorsque ⟨argument⟩ est une variable caractère:
Registre A = 3

Contenu de la mémoire

Adresse indiquée par le registre DE→	
	Longueur de chaîne
+1	Adresse inférieure
+2	Adresse supérieure

} Indicateur de la position de la chaîne

L'exposant d'une variable à simple et à double précision est défini par le signe de la mantisse et l'exposant. L'exposant débute à 40H, et sa valeur à 40H est de −1.



Les résultats sont transmis selon ⟨argument⟩.

Exemple de programme

```
100 CLEAR 50,&HBFF
110 POKE &HC00,&HC9
120 A=USR(&HC00,34)
130 PRINT A
140 END
```



```
100 CLEAR 50,&H1000
110 FOR I=&H1100 TO &H1108
120 READ A$
130 POKE I,VAL("&H"+A$)
140 NEXT
150 DATA 11,00,12     :'LD  DE,1200H
160 DATA 01,08,00     :'LD  BC,8H
170 DATA ED,B0        :'LDIR
180 DATA C9           :'RET
190 LPRINT USR(&H1100,1234)
200 GOSUB 240
210 LPRINT USR(&H1100,1.23456789012345#)

220 GOSUB 240
230 END
240 FOR I=&H1200 TO  &H1207
250 LPRINT HEX$(I)+":"+HEX$(PEEK(I))
260 NEXT
270 LPRINT
280 RETURN
```

VAL

Fonction: Produit la valeur numérique d'une expression en chaîne
Format: VAL (⟨expression en chaîne⟩)
Exemple: A=VAL(''123'')
PRINT VAL(''&HFF'')

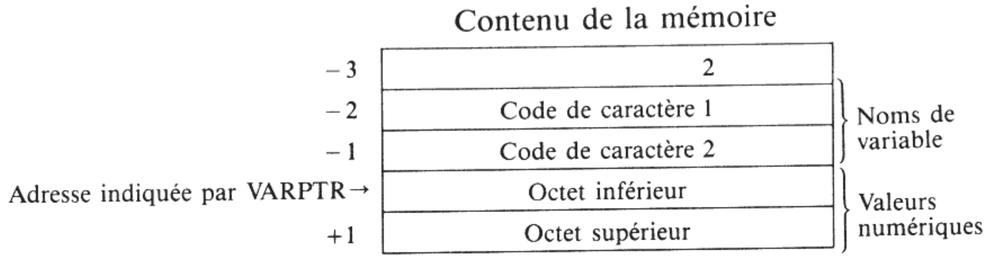
Explication: Si la chaîne ne commence pas par +, −, & ou un nombre, 0 est produit.

Lorsqu'un caractère qui ne représente pas une valeur numérique fait partie de la chaîne, celui-ci et les caractères suivants sont ignorés.

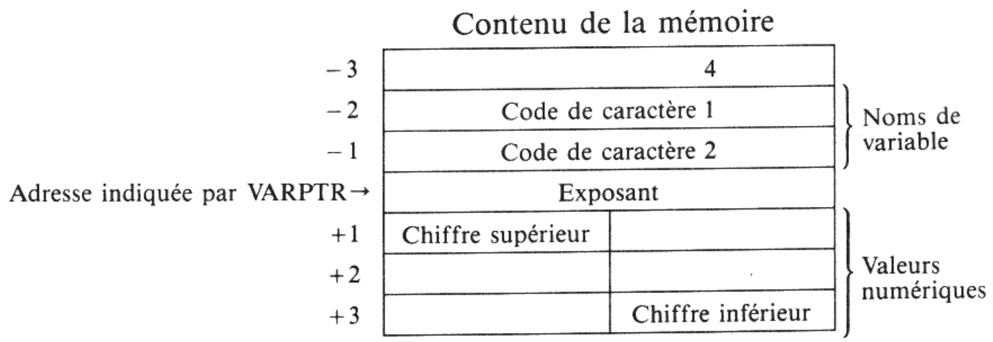
VARPTR

Fonction: Donne l'adresse d'une variable
Format: VARPTR (⟨variable⟩)
Exemple: A=VARPTR (B)

Explication: Le contenu de variables peut être échangé entre un programme en langage machine et un programme en BASIC.
Lorsque ⟨variable⟩est une variable du type entier:



Lorsque ⟨variable⟩ est une variable de type nombre réel à simple précision:



Lorsque ⟨variable⟩ est une variable de type nombre réel à simple précision:

Contenu de la mémoire

−1	8		
−2	Code de caractère 1		Valeurs numériques
−3	Code de caractère 2		
Adresse indiquée par VARPTR→	Exposant		
+1	Chiffre supérieur		Noms de variables
+2			
+3			
+4			
+5			
+6			
+7		Chiffre inférieur	

Lorsque ⟨variable⟩ est une variable caractère (chaîne):

Contenu de la mémoire

−3	3	
−2	Code de caractère 1	Valeurs numériques
−1	Code de caractère 2	
Adresse indiquée par VARPTR→	Longueur de chaîne	
+1	Adresse supérieure	Noms de variables
+2	Adresse inférieure	

Exemple de programme

```
100 DEFINT A
110 DEFSNG B
120 DEFDBL C
130 DEFSTR D
140 A=1234
150 B=1.23457E+56
160 C=1.2345678901234D+56
170 D="ABCDEF"
180 PRINT A:P=VARPTR(A):GOSUB 230
190 PRINT B:P=VARPTR(B):GOSUB 230
200 PRINT C:P=VARPTR(C):GOSUB 230
```



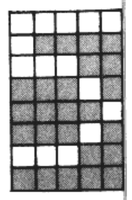
```
210 PRINT D:P=VARPTR(D):GOSUB 230
220 END
230 FOR I=-3 TO PEEK(P-3)-1
240 PRINT I;TAB(7);HEX$(PEEK(P+I))
250 NEXT
260 PRINT
270 RETURN
```

```
RUN
 1234
-3      2
-2      41
-1      0
 0      D2
 1      4

 1.23457E+56
-3      4
-2      42
-1      0
 0      79
 1      12
 2      34
 3      57

 1.2345678901234E+56
-3      8
-2      43
-1      0
 0      79
 1      12
 2      34
 3      56
```

4	78
5	90
6	12
7	34

ABCDEF	
-3	3
-2	44
-1	0
0	6
1	B1
2	5

Chapitre 5
Appendices



Messages d'erreur

- /0 (Division par zéro): code 11
  Une division par zéro est exécutée.
  •Un dénominateur est zéro
  •Une variable non spécifiée est utilisée pour une division

- BF (Erreur de mode fichier): code 25
  La structure d'un fichier est incorrecte
  •Un fichier est ouvert au moyen d'une instruction INIT # dans un mode invalide
  •Une donnée est délivrée à un fichier qui se trouve en mode d'entrée, et vice versa

- BS (erreur d'indice): code 9
  L'indice de la variable d'une liste se trouve au-dessus de la gamme admise
  •La valeur de l'indice est trop grande.
  •L'indice de la variable d'une liste non définie est plus grand que 10.

- CN (impossible de poursuivre): code 17
  L'exécution du programme ne peut pas être reprise:
  •Programme arrêté par suite d'une erreur.
  •Programme modifié aprés avoir été arrêté, ou
  •Le programme a été arrêté par "abort" et non par un "Break". Arrêt d'un programme en déroulement (RUN)

- DD (Définition dédoublée): code 10
  Deux instructions DIM sont exécutées pour la même variable d'une liste, ou deux instructions DEFFN sont exécutées pour un même nom de fonction.
  •Le même nom a été utilisé pour une liste sans avoir exécuté d'instruction ERASE.
  •Une variable de liste est utilisée sans avoir été déclarée, puis est déclarée par la suite.
  •Une instruction DIM ou DEFFN d'une boucle est exécutée deux fois ou plus.

- FC (Appel d'une fonction non admise): code 5
  Une fonction ou une instruction est spécifiée de manière erronée.
  •Les paramètres d'une fonction ou d'une instruction sortent de la gamme admise.
  •L'indice d'une variable de liste est un nombre négatif.

- ID (Exécution en mode direct non admise): code 12
  Une instruction est entrée en mode direct alors qu'elle n'est pas admise.
  •En mode direct, DEFFN et les instructions ne sont pas admises.

- IO (Erreur d'E/S d'unité): code 22
  Une erreur se produit durant l'entrée ou la sortie vers une unité en raison de:
  •Problème avec la bande cassette
  •Niveau d'enregistrement incorrectement réglé
  •Réglage d'état de RS-232C incorrect.

- IR (Exécution non admise): code 27
  Tentative de déroulement direct d'un programme stocké dans un fichier contenant une instruction telle que LOAD ou NEW.
  •Les instructions telles que LOAD et NEW peuvent uniquement être exécutées dans la zone texte de la mémoire.

- LS (Chaîne trop longue): code 15
  La longueur d'une chaîne est trop longue.
  •Une chaîne ne peut comporter plus de 255 caractères.

- MO (Opérande absent): code 23
  Un paramètre nécessaire est absent.
  •Le nombre de paramètres est incorrect.
  •Les valeurs numériques sont séparés par des points au lieu de virgules.

- NE (Il n'existe pas de fichier): code 24
  Le fichier spécifié est absent.
  •La désignation de fichier a été composée avec une erreur.
  •Le type de fichier est incorrect.
  •La carte ROM ou RAM utilisée ne contient pas le fichier désiré.

- NF (NEXT utilisé sans FOR): code 1
  L'instruction FOR assortie est absente.
  •L'imbriquage d'une boucle est erroné.
  •Il est effectué un branchement dans une boucle.

- NO (Fichier non ouvert): code 26
  On utilise un numéro de fichier qui n'a pas été défini.
  •Numéro de fichier erroné
  •L'unité spécifiée par l'instruction TRON # n'a pas été initialisée par l'instruction INIT # .

- NR (Pas de RESUME): code 19
  Il n'y a pas d'instruction RESUME dans un sous-programme de traitement d'erreur.
  •Un sous-programme de traitement d'erreur doit se terminer par une instruction END, RESUME, ou ON ERROR GOTO
  •L'instruction GOTO est utilisée pour le retour d'un sous-programme de traitement d'erreur.

- OD (Manque de données): code 4
  Les données à lire par l'instruction READ ne sont pas fournies.
  •Il n'y a pas suffisamment de données.
  •L'instruction RESTORE est utilisée de manière incorrecte.
  •Des symboles autres que des virgules sont utilisés pour séparer les données.

- OM (manque de mémoire): code 7
  Capacité de mémoire insuffisante
  •Le programme est trop long.
  •Il y a trop de variables.
  •Il y a trop d'eléments de variable dans une liste.
  •L'imbriquage d'une boucle FOR ~ NEXT ou GOSUB ~ RETURN excède la capacité de la mémoire.
  •Une zone trop grande est réservée au moyen de l'instruction CLEAR.
  •Une zone trop grande est réservée pour la mémoire RAM de fichier au moyen de l'instruction FSET.

- OS (manque de place pour chaîne): code 14
  La place pour chaîne est insuffisante.
  •La zone chaînes réservée au moyen de l'instruction CLEAR est trop petite.

- OV (Débordement): code 6

  Les valeurs numériques dépassent la capacité.

  •Le résultat du calcul d'entiers dépasse la gamme de −32768 à 32767.

  •Le résultat d'un calcul de nombres réels dépasse la gamme de −9.99...99E62 à 9.99...99E62
  14 chiffres 14 chiffres

  •L'adresse formant le paramètre d'une instruction dépasse la gamme d'adresses.

- RG (RETURN sans GOSUB): code 3

  L'instruction RETURN est exécutée avant que l'instruction GOSUB ne le soit.

  •Branchement à un sous-programme par l'instruction GOTO.

  •L'instruction END ayant été oubliée à la fin du programme principal, le sous-programme venant ensuite est exécuté.

- RW (RESUME sans error): code 20

  L'instruction RESUME est exécutée sans occurence d'une erreur.

  •Branchement à un sous-programme par l'instruction GOTO ou GOSUB.

  •L'instruction END ayant été oubliée à la fin du programme principal, le sous-programme venant ensuite est exécuté.

- SN (Erreur de syntaxe): code 2

  Syntaxe incorrecte.

  •Mot d'instruction non défini.

  •Utilisation erronée des parenthèses

  •Utilisation de symboles incorrects (,,.,:,;, etc.) pour séparer des éléments.

  •Nom de variable ne commençant pas par une lettre de l'alphabet.

  •Nom de variable comprenant un mot réservé.

  •Utilisation de paramètres incorrects dans une fonction, instruction ou ordre.

  •Erreur dactylographique.

- ST (Forme de chaîne trop complexe): code 16

  Une expression en chaîne est trop complexe.

  •Calcul trop complexe.

  •Imbriquage de parenthèses trop important.

- TM (Types non adaptés): code 13

  Le type de variable est incorrect.

  •Affectation d'une chaîne de caractères à une variable numérique.

  •Affectation d'une constante numérique à une chaîne de caractères.

  •Paramètres d'une fonction de type incorrect.

- UF (Fonction utilisateur indéfinie): code 18

  Appel d'une fonction qui n'a pas été définie.

  •Nom de variable commençant par FN.

  •Nom de fonction défini de manière incorrecte.

  •Inexécution de l'instruction DEFFN.

- UL (Numéro de ligne indéfini): code 8

  Un numéro de ligne est défini de manière incorrecte.

  •Il n'existe pas de ligne correspondant au numéro de ligne spécifié.

  •Le numéro de ligne spécifié par l'instruction GOTO, GOSUB ou RESTORE, ou par l'ordre RUN, n'existe pas.

- UE (Erreur indéfinie): code

  Une erreur comportant un code d'erreur autre que les codes d'erreur ci-dessus se produit.

Tableau des codes de caractères

	0	I	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0				0	@	P	`	p	♠	ì	√	—	タ	ミ	ô	ε
I			!	1	A	Q	a	q	♥	î	。	ア	チ	ム	ó	σ
2			"	2	B	R	b	r	♣	í	「	イ	ツ	メ	o̲	θ
3			#	3	C	S	c	s	◆	Ü	」	ウ	テ	モ	ÿ	x
4			$	4	D	T	d	t	○	ü	、	エ	ト	ヤ	Ç	λ
5			%	5	E	U	e	u	●	ù	。	オ	ナ	ユ	ç	μ
6			&	6	F	V	f	v	Ä	û	ヲ	カ	ニ	ヨ	Ñ	ρ
7	BELL		'	7	G	W	g	w	Å	ú	ァ	キ	ヌ	ラ	ñ	π
8			(	8	H	X	h	x	ä	É	ィ	ク	ネ	リ	Γ	τ
9			)	9	I	Y	i	y	à	ë	ゥ	ケ	ノ	ル	Σ	φ
A	LF		*	:	J	Z	j	z	â	è	ェ	コ	ハ	レ	Π	χ
B	HOME		+	;	K	[	k	｛	á	ê	ォ	サ	ヒ	ロ	Ω	ω
C	CLS	→	,	<	L	¥	l	|	å	é	ャ	シ	フ	ワ	α	ν
D	CR	←	−	=	M	]	m	｝	a̲	Ö	ュ	ス	ヘ	ン	β	£
E		↑	·	>	N	^	n	~	Ï	ö	ョ	セ	ホ	゛	γ	¢
F		↓	／	?	O	_	o	¿	ï	ò	ッ	ソ	マ	°	δ	÷

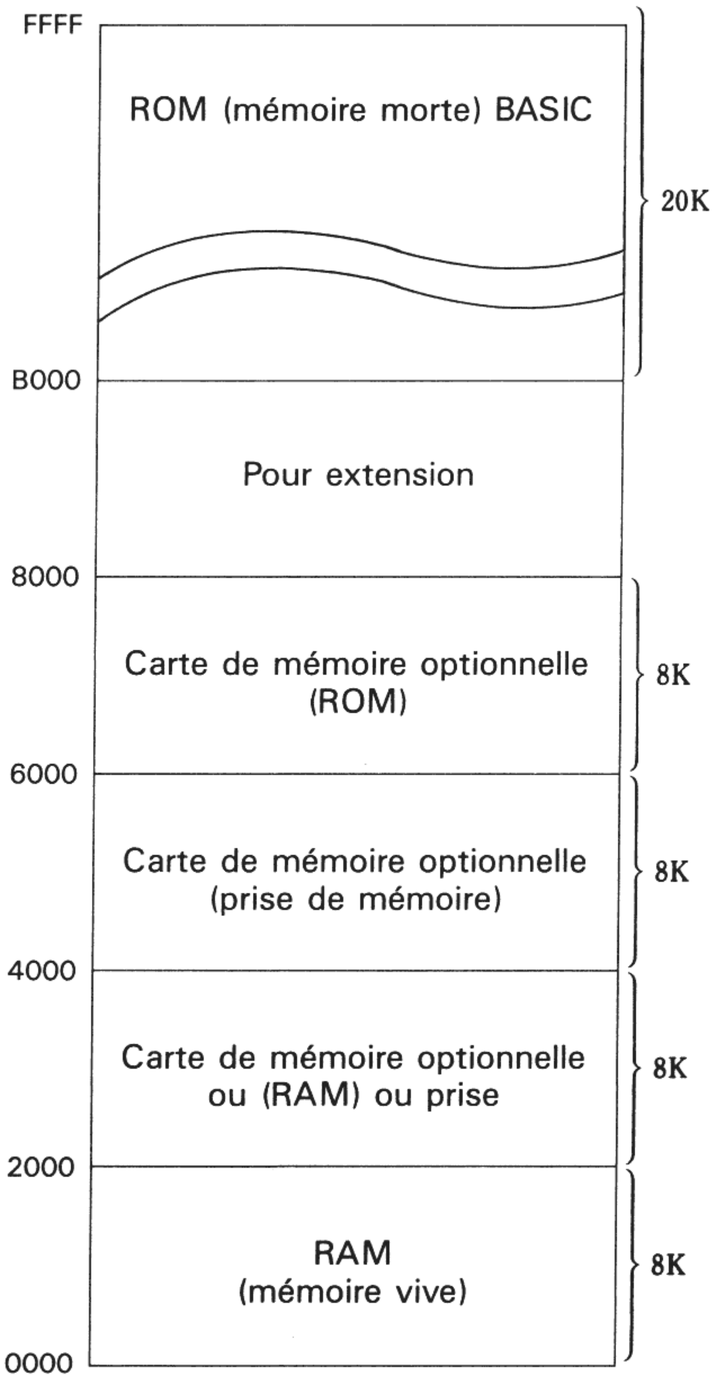
Peut être défini par l'utilisateur

Peut être défini par l'utilisateur

Tableau des codes de contrôle

Code de caractèter (hexa-décimal)	Touche correspondante	Fonction
02	CTRL+B	Amene le cursteur au debut du mot précédent
03	CTRL+C	Arrête l'exécution
05	CTRL+E	Efface la ligne suivante
06	CTRL+F	Déplace le curseur au début du pro+chain mot
08	CTRL+H	Efface le caractère précédent et déplace le reste des caractères d'un pas vers la gauche.
09	CTRL+I	Tabulation
0A	CTRL+J	Interligne
0B	CTRL+K	Ramène le curseur au coin supérieur gauche de l'écran
0C	CTRL+L	Efface l'écran
0D	CTRL+M	Retour du curseur
10	CTRL+P	Efface tout ce qui se trouve après le curseur
12	CTRL+R	Insère un caractère où se trouve le curseur.
13	CTRL+S	Arrête momentanément l'exécution.
15	CTRL+U	Efface la ligne sur laquelle le cur+seur se trouve.
16	CTRL+V	Efface le caractère où se trouve le curseur et déplace les suivants vers la gauche.
18	CTRL+X	Amene le curseur à la fin de la ligne. Carte de mémoire

Carte de mémoire



Mots réservés

ABS
ALM$
AND
ASC
ATN
BEEP
CDBL
CHR$
CINT
CIRCLE
CLEAR
CLOAD
CLS
COLOR
CONSOLE
CONT
COS
CSAVE
CSNG
CSRLIN
DATA
DATE$
DEFDBL
DEFFN
DEFINT
DEFSNG
DEFSTR
DELETE
DIM
DIR
ELSE
END
EQV
ERASE
ERL
ERR
ERROR
EXEC
EXP
FIX
FN
FONT$
FOR
FRE
FSET
GOSUB
GOTO
HEX$
IF
INIT
INKEY$
INP
INPUT
INSTR
INT
KEY$
LEFT$
LEN
LET
LINE
LIST
LLIST
LOAD
LOCATE
LOG
LPRINT
MID$
MOD
MOTOR
NEW
NEXT
NOT
OFF
ON
OR
OUT
PAINT
PEEK
POINT
POKE
POS
PRESET
PRINT
PSET
READ
REM
RESTORE
RESUME
RETURN
RIGHT$
RND
RUN
SAVE
SCREEN
SGN
SIN
SLEEP
SNS
SQR
START$
STEP
STICK
STOP
STR$
STRIG
STRING$
TAB
TAN
THEN
TIME$
TKEY
TO
TR
USING
USR
VAL
VARPTR
XOR

Index

C

Caractères 8
Caractères de contrôle 175
Cartes ROM/RAM 24
Chaînes 10
Chaînes nulles 10, 12
Comparaison de bits 17
Constantes 10
Constantes de type nombre réel 11
Constantes entières 10
Conversion de type 14
Coordonnées actuelles 29
Coordonnées relatives 28

D

Débordement 16
Déclaration du type de variable 12
Déroulement direct 27, 102
Division par 0 16

E

Editeur 4,7
Edition de programmes 7
Erreurs 7, 169
Expressions de relation 16
Expressions logiques 17

F

Fichiers 23, 26
Fichier RAM 26
Fonctions 18, 45

I

Indices 13
Instructions 5
Interpréteur 3

L

Lignes 5
Listes, tableaux 13

M

Mémoire 3
Modes 6
Mots réservés 177

N

Numéros de fichier 23

O

Opérateurs 15
Operations boolèennes 17
Ordres 6
Ordres graphiques 28

P

Paramètres 5
Port série 26
Précision 19
Priorité de calcul 19
Programmes 3, 15

S

Symboles spéciaux 8

U

Unités 23, 24

V

Valeuras par défaut 33
variables 12

Z

Zone fichiers 26
Zone texte de la mémoire 6

Index des instruction et fonctions par groupes

Instructions graphiques

CIRCLE Trace un cercle 35
CLS Efface l'écran 39
LINE Trace un trait 67
PSET Trace un point 94
PRESET Supprime un point 90

Instructions associées à des variables

CLEAR Initialise les variables et définit une zone en mémoire 36
DEFINT Déclare les variables de type entier 46
DEFSNG Déclare les variables de type nombre entier à simple précision 46
DEFDBL Déclare les variables de type nombre entier à double précision 46
DEFSTR Déclare les variables de caractères (chaînes) 46
DEFFN Etablit une fonction définie par l'utilisateur 45
DIM Déclare les listes (tableaux) 49
ERASE Efface les listes (tableaux) 53
LET Affecte des valeurs aux variables 66

Instructions associées aux fichiers

CLOAD Charge un programme à partir d'une bande cassette 37
CLOAD? Vérifie la présente d'un programme sur une bande cassette 38
CSAVE Sauvegarde un programme sur une bande cassette 43
DELETE Supprime un fichier 48
DIR Affiche le contenu de la mémoire RAM sur l'écran 51
DIR # Délivre le contenu de la mémoire RAM à # numéro de fichier 51
FSET Fixe la zone de la mémoire RAM 58
LOAD Charge un programme en zone texte de la mémoire à partir d'un fichier 74
LOAD? Vérifie le contenu d'un fichier programme 75
SAVE Sauvegarde le programme en zone texte de la mémoire sur un fichier. 104

Traitement des erreurs, instructions d'exécution pas à pas

ERROR Produit une erreur 54
ON ERROR GOTO Spécifie le branchement à un sous-programme de traitement d'erreur 85
RESUME Retour d'un sous-programme de traitement d'erreur 100
TRON Indique le numéro de ligne actuellement exécuté 108
TROFF Termine le mode d'exéction pas à pas 107

Instructions associées au langage machine

EXEC Exécute un programme en langage machine 55
OUT Transmet les données au port spécifié 87
POKE Inscrit une valeur en un endroit de la mémoire 89

Instructions de contrôle de programme

CONT Reprend l'exécution du programme 42
END Termine le programme 52
FOR TO STEP-NEXT Etablit une boucle 56
GOTO Branchement au numéro de ligne spécifié 60
GOSUB-RETURN Exécute un sous-programme 59
IF THEN ELSE Branchement conditionnel 61
ON GOTO Branchement conditionnel 86
ON GOSUB Branchement aux sous-programmes spécifiés en fonction des conditions réalisées 86
RUN Exécute un programme 102
STOP Arrête un programme 106

Instructions d'entrée/sortie

INIT # Définit un numéro de fichier 62
INPUT Affecte des valeurs entrées sur le clavier aux variables 64
INPUT # Affecte des valeurs entrées de #⟨numéro de fichier⟩ aux variables..... 64
LINE INPUT Affecte une ligne entrée sur le clavier à une variable de caractères 68
LINE INPUT # Affecte une ligne entrée de #⟨numéro de fichier⟩à une variable de caractères..... 69
LPRINT Délivre vers une imprimante 77
OUT # Délivre vers #⟨numéro de fichier⟩ 88
PRINT(?) Délivre sur l'écran..... 91
PRINT USING Délivre selon un format spécifié 92
PRINT#USING Délivre vers #⟨numéro de fichier⟩ selon un format spécifié. 95
PRINT # Délivre vers # numérode fichier 95

Instructions associées aux données

DATA Spécifie les données 44
READ Affecte les données de l'instruction DATA aux variables 97
RESTORE Spécifie les données à lire 99

Instrucitons d'édition

LIST Délivre un programme sur l'écran 70
LLIST Délivre un programme vers l'imprimante 73
LIST # Délivre un programme vers # numéro de fichier 72
LIST@ Délivre un programme ligne par ligne sur l'écran..... 71
NEW Efface un programme en zone texte de la mémoire..... 82

Instructions associées à l'interrupteur d'alimentation

OFF Coupe l'alimentation du X-07..... 84
SLEEP Conserve divers états puis coupe l'alimentation du X-07 . 105

Instructions associées à console

CONSOLE Contrôle de roulement des lignes, le cliquetis des touches et la répétition 40
CONSOLE@ Contrôle l'alarme, le jeu de caractères et le mode du clavier 41

Autres instructions

BEEP Produit un son à travers le haut-parleur..... 34
REM(') Insère une remarque 98
LOCATE Contrôle le curseur 76
MOTOR Contrôle le moteur d'un enregistreur à cassette..... 81

Fonctions

ABS Valeur absolue..... 111
ALM$ Règle et affiche le contenu de l'alarme..... 112
ASC Conversion de caractère à code..... 114
ATN Arctangente..... 115
CDBL Conversion à double précision 116
CHR$ Conversion de code à caractère 117
CINT Conversion d'entier..... 118
COS Cosinus 119
CSNG Conversion à simple précision..... 120
CSRLIN Position verticale du curseur..... 121
DATE$ Règle et affiche le calendrier..... 122
ERL Numéro de la ligne sur laquelle l'erreur se produit 123
ERR Code de l'erreur qui s'est produite 124
EXP Fonction exponentielle 125
FIX Produit la partie entière..... 124
FONT$ Fixe et affiche un caractère 127
FRE Zone inutilisée de la mémoire en nombre d'octets..... 129
HEX$ Conversion décimal à hexadécimal 130
INKEY$ Produit le caractère de la touche enfoncée 131
INP Valeur d'un port..... 132
INSTR Recherche d'une chaîne 133
INT Produit le plus grand entier..... 134
KEY$ Etablit et affiche une touche de fonction..... 135
LEFT$ Produit les caractèrs situes a l'extrême gauche d'une chaîne 136

LEN Longueur d'une chaîne 137
LOG Fonction logarithmique 138
MID$ Produit une partie de chaîne 139
PEEK Contenu de la mémoire 140
POINT Vérifie la présence d'un point sur l'écran 141
POS Position horizontale du curseur 142
RIGHT$ Produit lescaractères extrême droits d'une chaîne ... 143
RND Produit des nombres aléatoires 144
SCREEN Code d'un caractère présent sur l'écran 145
SGN Produit une valeur fonction d'une expression numérique 146
SIN Sinus 147
SNS Vérifie l'entrée vers une unité 148
SQR Racine carrée 149
START$ Etablit et affiche un programme de départ 150
STICK Etat des touches de curseur 151
STR$ Conversion de valeur numérique à chaîne 152
STRIG Etat de la touche d'espacement et de la touche de fonction 6 153
STRING$ Produit une chaîne de longueur spécifiée 154
TAB Fixe la tabulation 155
TAN Tangente 156
TIME$ Règle et affiche le contenu de l'alarme 157
TKEY Etat d'une touche 158
USR Fonction utilisateur en langage machine 159
VAL Conversion de chaîne à valeur numérique 162
VARPTR Adresse d'une variable 163

Canon **CANON INC.**
7-1, 2-chome, Nishi-shinjuku, Shinjuku-ku, Tokyo 160, Japan
P.O. Box 5050, Shinjuku Dai-ichi Seimei Building, Tokyo 160, Japan

CANON U.S.A., INC.
HEAD OFFICE One Canon Plaza, Lake Success, N.Y. 11042, U.S.A.
CHICAGO 140 Industrial Drive, Elmhurst, Illinois 60126, U.S.A.
LOS ANGELES 123 Paularino Avenue East, Costa Mesa, California 92626, U.S.A.
ATLANTA 6380 Peachtree Industrial Blvd., Norcross, Georgia 30071, U.S.A.
DALLAS 2035 Royal Lane, Suite 290, Dallas, Texas 75229, U.S.A.

CANON CANADA INC.
HEAD OFFICE 6930 Dixie Road Mississauga, Ontario, L5T 1P7, Canada
CALGARY 2828, 16th Street, N.E. Calgary, Alberta, T2E 7K7, Canada

CANON EUROPA N.V.
P.O. Box 7907, 1008 AC Amsterdam, The Netherlands

CANON FRANCE S.A.
DIVISION Micro Informatique 93154 Le Blanc Mesnil, Cedex, France

CANON RECHNER DEUTSCHLAND GmbH.
Fraunhoferstrasse 14, Postfach 8033, München-Martinsried, West Germany

CANON UK LTD.
Waddon House, Stafford Road, Croydon CR9 4DD, England

CANON LATIN AMERICA, INC.
SALES DEPARTMENT P.O. Box 7022, Panama 5, Rep. of Panama
REPAIR SERVICE CENTER P.O. Box 2019, Colon Free Zone, Rep. of Panama

CANON HONG KONG TRADING CO., LTD.
Golden Bear Industrial Centre, 7th Floor, 66-82 Chai Wan Kok Street, Tsuen Wan, New Territories, Hong Kong

CANON AUSTRALIA PTY. LTD.
1 Hall Street, Hawthorn East, Victoria 3123, Australia