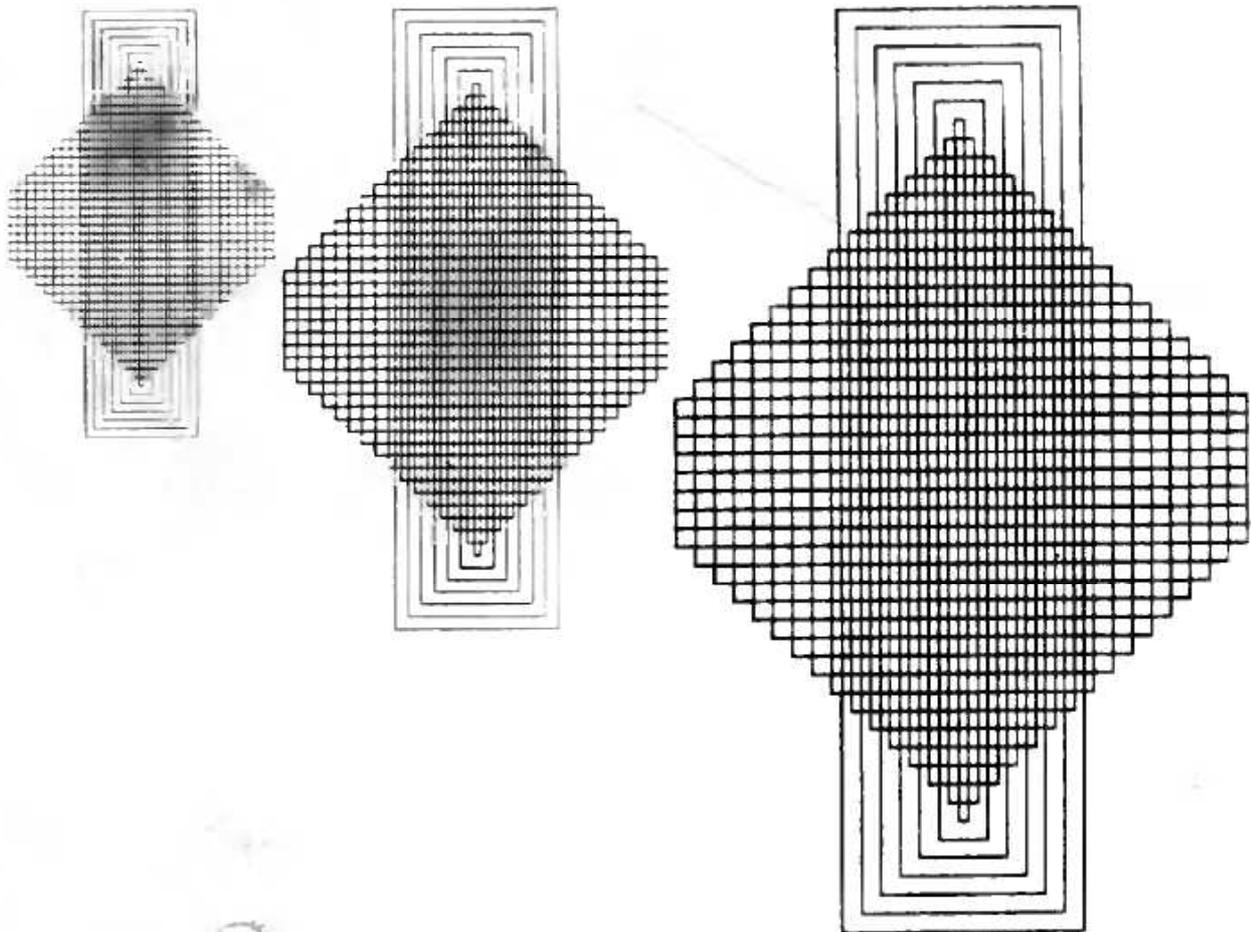


ATARI[®]

ST BASIC[™]

Guide Résumé

Pour le programmeur confirmé



Les informations contenues dans le présent manuel ont fait l'objet d'une vérification minutieuse. Néanmoins, procédant à des développements constants de ses matériels et logiciels, ATARI Corp. ne peut garantir l'exactitude de la présente documentation au-delà de sa date de publication et décline toute responsabilité quant aux modifications, erreurs ou omissions qui pourraient en résulter.

ATARI, ST, ST BASIC et TOS sont des marques déposées d'ATARI Corp.
GEM est une marque déposée de Digital Research Inc.

Ce document ne peut être reproduit, en partie ou en totalité, sans l'autorisation écrite expresse d'ATARI Corp.

Traduction et Composition : M.V.Consultant
Montage : TECHNI-VF

TABLE DES MATIERES

TABLE DES MATIERES	1
BIENVENUE A ST BASIC ETENDU	1
Comment utiliser ce Guide	1
Contenu de la disquette Langage ST	2
Le Manuel de Référence et Tutoriel de ST BASIC	2
PREMIERS CONTACTS AVEC ST BASIC	2
Chargement de ST BASIC	3
CONVERSION DE PROGRAMMES VERS ST BASIC	4
Lecture d'un Programme BASIC	4
Différences entre les divers BASIC	4
ST BASIC	5
Autres BASICs	6
CODES ET MESSAGES D'ERREUR	10
GUIDE DE REFERENCE	13
Caractères de Déclaration	13
Délimiteurs	13
Commandes d'édition	14
Opérateurs	14
Mode Tracé/Dessin	15
Motifs de remplissage	16
Motifs des Lignes	16
Ports	16
Fenêtre	17
Sons	17
LISTE DES COMMANDES	17
LISTE DES INSTRUCTIONS	18

LISTE DES FONCTIONS	19
LISTE DES VARIABLES DU SYSTEME	19
COMMANDES, INSTRUCTIONS, FONCTIONS ET VARIABLES DU SYSTEME	20
SERVICE INFORMATIONS CLIENTS	41

Bienvenue à **ST BASIC étendu**

Cette version étendue de **ST BASIC** remplace celle fournie jusqu'ici avec les ordinateurs ATARI ST. Les deux versions sont similaires au langage BASIC standard mais utilisent en plus les fenêtres, les menus descendants et les éléments graphiques du Bureau GEM. Elles profitent également de la vitesse et des possibilités graphiques des ordinateurs **ATARI ST**.

Le nouveau **ST BASIC** s'exécute environ trois fois plus rapidement que la version précédente. Il comporte plus de fonctions - avec 33 nouveaux mots réservés, un intervalle de valeurs entières étendu et une syntaxe plus efficace. La liste des messages d'erreur a été agrandie et les messages d'erreur affichent une explication claire lors de leur apparition.

Le **ST BASIC** étendu est compatible avec la version précédente de **ST BASIC** et peut ainsi utiliser les programmes écrits avec la première version. Consultez le paragraphe "**Conversion de programmes vers ST BASIC**" pour savoir comment utiliser vos anciens programmes avec cette nouvelle version du langage.

Comment utiliser ce Guide

Ce guide est conçu pour des programmeurs avertis qui connaissent le langage BASIC et sont familiarisés avec les procédures du Bureau GEM. Il est architecturé de façon à ce que le programmeur puisse déceler les différences entre cette version de **ST BASIC** et les versions précédentes. Ainsi, les caractéristiques dédiées aux ordinateurs **ATARI ST** sont présentées avec des exemples qui montrent comment des programmes écrits avec d'autres versions de BASIC peuvent être chargés et exécutés avec cette version étendue de **ST BASIC**.

Contenu de la disquette Langage ST

La disquette Langage ST livrée avec votre ordinateur **ATARI ST** contient les fichiers nécessaires à l'exécution de **ST BASIC** étendu.

BASIC.PRG est le programme BASIC.

BASIC.RSC est le fichier ressource pour ce langage.

Note : Ne tenez pas compte des autres fichiers qui peuvent se trouver sur la disquette langage. Ces autres fichiers peuvent être des accessoires de bureau ou des programmes d'application. Cependant, les deux fichiers indiqués ci-dessus sont les seuls nécessaires à la programmation avec **ST BASIC** étendu.

Le Manuel de Référence et Tutoriel de ST BASIC

Le Manuel de référence et Tutoriel de **ST BASIC** est un manuel complet de **ST BASIC**. Il comprend plus de 300 pages qui fournissent un accès aisé à tous les niveaux de programmation. Le programmeur novice y découvre un tutoriel (apprentissage pas à pas) et le programmeur averti a accès à une documentation technique complète.

Si vous désirez programmer en **ST BASIC** étendu, demandez à votre distributeur habituel "Le Manuel de Référence et Tutoriel de **ST BASIC**".

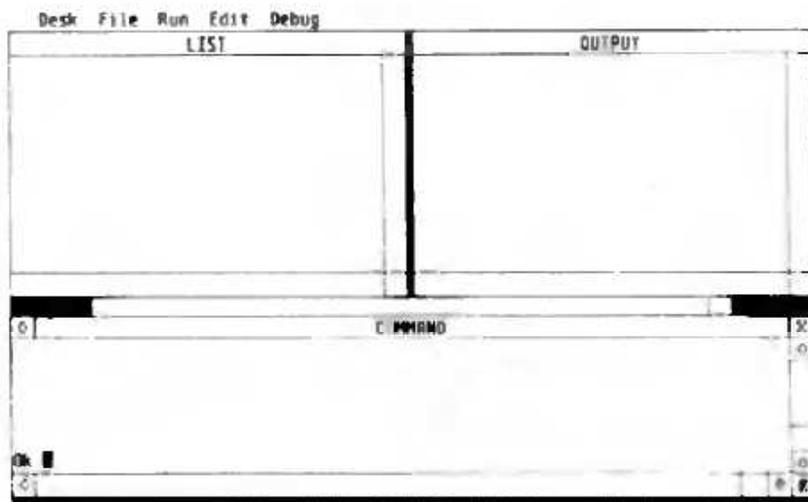
Premiers contacts avec ST BASIC

AVANT de commencer à utiliser **ST BASIC**, faites une copie de sécurité de votre disquette Langage ST. Ainsi, vous serez assuré contre une perte accidentelle de votre disque d'origine. (Consultez votre Manuel d'Utilisation pour savoir comment effectuer cette copie.)

Lorsque cette copie est effectuée, vous êtes prêt à utiliser **ST BASIC**. Commencez par charger le langage en respectant les instructions suivantes.

Chargement de ST BASIC

1. Lorsque l'ordinateur est sous tension et que le Bureau GEM est affiché, double cliquez sur l'icône de l'unité de disque A.
2. Lorsque le répertoire du disque apparaît, double cliquez sur **BASIC.PRG**. Le Bureau **ST BASIC** apparaît.



Ce bureau représente l'environnement de programmation de **ST BASIC**.

Note : **ST BASIC** utilise les procédures d'utilisation standard du Bureau GEM pour accéder aux objets d'un menu, sélectionner des options, manipuler des fenêtres et charger des applications. Ces procédures sont expliquées en détail dans le Manuel d'utilisation correspondant à votre ordinateur.

Conversion de programmes vers ST BASIC

Ce paragraphe décrit les améliorations de **ST BASIC** ajoutées à **ST BASIC** étendu, et les différences principales entre **ST BASIC** et les autres langages BASIC. Utilisez les informations données dans ce paragraphe pour convertir les programmes écrits dans une autre version de BASIC en **ST BASIC**.

Lecture d'un Programme BASIC

ST BASIC ne peut lire que des programmes qui ont été sauvegardés sous forme d'un texte ASCII. Vérifiez bien que les programmes que vous voulez convertir sont sous forme de textes ASCII.

Une ligne d'instructions **ST BASIC** doit commencer par un numéro de ligne, se terminer par un line feed et ne doit pas comporter plus de 255 caractères. Les numéros de lignes autorisés vont de 1 à 65 529 (0 à 65 529 dans d'autres BASICs). **ST BASIC** ne reconnaît pas les caractères de suite de ligne (line feed dans certains BASICs) ; la seconde partie de ligne effectuée de cette manière est perdue.

A part les exceptions ci-dessus, **ST BASIC** conserve le texte de votre programme, même s'il contient des erreurs de syntaxe ; vous pouvez utiliser ensuite l'éditeur de **ST BASIC** pour modifier votre programme. Le moyen le plus simple pour convertir un programme BASIC est de le charger dans **ST BASIC**, de voir les erreurs signalées et d'éditer les lignes erronées.

Différences entre les divers BASIC

Les différences entre **ST BASIC** étendu et les autres BASICs sont décrites ci-dessous.

ST BASIC

ST BASIC étendu est compatible avec les versions précédentes de **ST BASIC**. Les programmes écrits dans une version précédente peuvent être utilisés avec la nouvelle version à condition de prendre en considération les éléments suivants.

DEF SEG a été supprimée et remplacée par des versions spéciales de **PEEK** et **POKE** pour un mot, un octet et un mot long. Ce sont **PEEK_W**, **PEEK_B**, **PEEK_L** et **POKE_W**, **POKE_B**, **POKE_L**. Les programmes utilisant **DEF SEG** dans la version précédente de **ST BASIC** doivent être ré-écrits.

Les adresses de **PEEK** et **POKE** utilisent des nombres entiers ; la précision est suffisante.

Les nombres entiers sont dorénavant des nombres sur 32 bits. Cela permet d'agrandir l'intervalle de -32 768 à 32 767 de -2 147 483 648 à 2 147 483 647.

Une nouvelle syntaxe, plus efficace, a été introduite pour **GEMSYS** et **VDISYS**. L'ancienne fonctionne encore avec une modification : le nombre entre parenthèses doit être placé dans **GEM_CONTROL(0)** pour **GEMSYS** et dans **CONTROL(0)** pour **VDISYS**. Les programmes utilisant **GEMSYS** et **VDISYS** devront être modifiés pour utiliser la nouvelle syntaxe.

Les mots-clés suivants ont été rajoutés :

AREA	GEM_ADDROUT	PATTERN
ASK MOUSE	GEM_CONTRL	PEEK_B
ASK RGB	GEM_GLOBAL	PEEK_L
BIOS	GEM_INTIN	PEEK_W
BOX	GEM_INTOUT	POKE_B
CLEAR	GEMDOS	POKE_L
DRAW	GSHAPE	POKE_W
DRAWMODE	LINEPAT	RGB
ED	MAT AREA	SSHAPE
ERR\$	MAT DRAW	STATUS
GEM_ADDRIN	MAT SOUND	XBIOS

Les programmes écrits dans une version précédente de **ST BASIC** et qui utilisent l'un de ces mots sous une autre forme que celle de mot réservé doivent être ré-écrits.

Toute syntaxe qui utilise pour une liste une paire x,y est décrite comme nécessitant un point-virgule (;) entre les palres. Le point-virgule augmente la lisibilité, bien que l'ancienne syntaxe soit toujours valable.

SYSTAB est maintenant un tableau d'entiers sur deux octets. Les accès à SYSTAB sont les éléments d'un tableau dont l'index est la moitié du décalage précédent : par exemple, SYSTAB+6 devient SYSTAB(3).

INP utilisé avec -1 ne donne pas toujours un nombre négatif ; cependant, il retourne toujours un nombre non nul.

Le point (.) n'est plus admis dans les noms de variables et de mots-clés ; il est remplacé par le caractère de soulignement (_).

Autres BASICs

Identificateurs

Le nom des variables et des mots clés dans **ST BASIC** doit commencer obligatoirement par une lettre et peut contenir des lettres (A à Z ou a à z), des chiffres (0 à 9) et le caractère de soulignement (_). Notez que majuscules et minuscules sont équivalentes. D'autres BASICs admettent le point (.), mais pas le caractère de soulignement. , et dans certains cas il y a une différence entre majuscules et minuscules.

Constantes Chaînes de caractères

ST BASIC permet d'insérer un guillemet (") dans une constante chaîne en la terminant par un autre guillemet. Par exemple :

```
PRINT ""Ceci est un citron," dit-il."
```

Imprime

```
"Ceci est un citron," dit-il.
```

D'autres BASICs traitent le guillemet comme la fin d'une chaîne ou le commencement de la suivante. Puisque les séparateurs entre les items ne sont pas obligatoires dans les listes PRINT, vous pouvez trouver une instruction comme

```
PRINT "A"."B"
```

Dans un programme BASIC existant, qui ne donne pas le résultat attendu en **ST BASIC**. **ST BASIC** imprime "A""B" comme A"B, alors que d'autres BASICs l'impriment comme AB.

Arithmétique

ST BASIC reconnaît trois types numériques : entier, flottant simple précision et flottant double précision.

Un nombre entier occupe 4 octets et peut représenter des nombres dans l'intervalle de -2 147 486 648 à 2 147 486 647. Dans de nombreux autres BASICs, les nombres entiers occupent 2 octets, dans l'intervalle de -32 768 à 32 767. Ceci affecte les fonctions MKI\$ et CVI, ainsi que le format des enregistrements contenant des entiers.

ST BASIC évaluant les expressions comportant au moins un terme entier sous forme entière, ce qui peut provoquer un dépassement de capacité dans les valeurs intermédiaires. Par exemple :

```
a% - b% * c% - 170000  
est évaluée sous forme entière.
```

Les valeurs flottantes sont conservées selon le format IEEE. ce qui donne un intervalle et une précision plus importants que d'autres formats. **ST BASIC** ne reconnaît pas les nombres non arithmétiques tel l'infini. Par exemple :

```
PRINT 1E-38
```

Imprime 0, car 1E-38 est plus petit que le plus petit nombre normalisé en simple précision dans le format IEEE.

L'arithmétique multi-mode est effectuée en double précision puisque la précision des entiers (31 bits plus le signe) est plus grande qu'en flottant simple précision (24 bits plus le signe).

Traduction et Interprétation

ST BASIC traduit le programme dans un jeu interne de codes au fur et à mesure que vous le tapez ou le chargez. Lorsqu'un programme est exécuté, c'est le code interne qui est exécuté. Ceci diffère de la plupart des BASICs que l'on trouve sur les micro-ordinateurs qui traduisent et exécutent au fur et à mesure, et provoque ainsi des différences dans les effets des instructions de déclaration.

Les instructions de type DEF (qui définissent les types des variables par défaut) modifient l'action du traducteur. Elles prennent effet dès qu'elles sont tapées, sans tenir compte de l'endroit du programme où elles devraient être. Pour cette raison, le programme suivant n'est pas possible en **ST BASIC** :

```
100 input "type ";a%
110 if a%=1 then gosub 2000: goto 500
130 a=5.0 :goto 600
500 a = "oui"
600 print a
700 end
2000 defstr a
2010 return
```

DEF FN définit une fonction utilisateur quel que soit l'endroit où elle se trouve dans le programme. La fonction DEF FN n'a pas besoin d'être exécutée pour définir la fonction. Il n'est pas possible d'avoir deux instructions définissant la même fonction. Pour cette raison, l'exemple suivant n'est pas possible en **ST BASIC** :

```
100 input "quelle définition ";a%
110 if a% =1 then gosub 2000 else gosub 2100
120 print FNR(1,2)
130 end
2000 def fnr(x,y) = x/y
2010 return
2100 def fnr(x,y) = y/x
2110 return
```

FOR/NEXT : restrictions

ST BASIC nécessite qu'à chaque instruction FOR corresponde une instruction NEXT et qu'à chaque instruction WHILE corresponde une instruction WEND. Le contrôle est effectué avant l'exécution du programme. La plupart des BASICs le font lors de l'exécution du programme, autorisant des constructions comme :

```
100 for i% = 1 to 1000
.
.
.
300 if i% >500 then next
.
.
.
500 next
```

ST BASIC indique une erreur en ligne 300, puisqu'il ne peut pas dire avant d'exécuter le programme si le next de la ligne 300 correspond bien au for de la ligne 100.

ST BASIC ne permet pas un saut à l'intérieur d'une boucle FOR/NEXT ou WHILE/WEND depuis l'extérieur. Par exemple, pour

```
10 goto 200
100 for i% =1 to 1000
.
.
.
200 print "Valeur de i% = ";i%
.
.
.
300 next
```

ST BASIC indique une erreur au moment de lancer l'exécution du programme. D'autres BASICs exécuteront le programme et donneront un message d'erreur comme "NEXT without FOR" en ligne 300.

Environnement des Commandes

ST BASIC fait une distinction entre les instructions (comme **PRINT**) qui sont une partie du programme, mais qui ne peuvent pas modifier le programme lui-même et les commandes qui sont utilisées pour examiner ou modifier un programme et qui ne peuvent pas faire partie du programme. Certains BASICs permettent aux commandes d'être intégrées au programme sur lequel elles agissent. **ST BASIC** indique une erreur lorsqu'il rencontre une commande sur une ligne (numérotée) de programme. Les commandes de **ST BASIC** sont listées dans la suite de ce guide.

Codes et Messages d'Erreur

Les codes et messages d'erreur suivants sont utilisés avec **ERL**, **ERR**, **ERR\$** et **ERROR**. (Les explications sur ces mots réservés sont donnés dans le paragraphe **Commandes, Instructions, Fonctions et Variables du Système.**)

Code	Message
0-1	Undefined error
2	Syntax error
3	RETURN without GOSUB
4	Out of data
5	Illegal function call
6	Overflow
7	Out of memory
8	Undefined line number
9	Subscript out of range
10	Duplicate definition
11	Division by zero
12	Illegal in immediate mode
13	Type mismatch
14	Undefined error
15	String too long
16	Expression too complex
17	CONT valid only in BREAK mode
18	Undefined user fonction

19	Undefined error
20	RESUME without error
21	Undefined error
22	Missing operand
23	Program line too long
24-49	Undefined error
50	Field overflow
51	Invalid record length
52	invalid file number
53	File not found
54	invalid file mode
55	File already open
56	Undefined error
57	Device I/O error
58	File exists
59	Unable to create a file
60	Undefined error
61	Disk full
62	End of file
63	Invalid record number
64	Invalid filename
65	Invalid character in program file
66	Direct statement in file
67	KILL failed
68	Device unavailable
69-92	Undefined error
93	Undefined segment
94	Protected file
95	Not a BASIC program
96-98	Undefined error
99	-Break-
100	Undefined error
101	Program too large
102	Undefined error
103	Invalid line number
104	Missing line number
105	Undefined error
106	Statement not found
107	Integer overflow
108	Redo from start
109	Stop
110	GOSUBs nested too deep

111	Invalid BLOAD file
112-200	Undefined error
201	Invalid option
202	Command not allowed here
203	Line number required
204	FOR without NEXT
206	Comma missing
207	Parenthesis missing
208	OPTION BASE must be 0 or 1
209	Undefined error
210	WHILE without WEND
211	WEND without WHILE
212	Undefined error
213	Duplicate DEF FN
214	Invalid jump into loop
215	Duplicate line number
216	Duplicate label
217-220	Undefined error
221	System error #%u
222	Program not run
223	Too many FOR loops
224-230	Undefined error

Guide de Référence

Caractères de Déclaration

Caractère	Fonction
%	Déclaration type Entier
\$	Déclaration type Chaîne de caractères
	Déclaration type Simple précision
#	Déclaration type Double précision

Délimiteurs

Caractère	Délimite les
'	Remarques
"	Chaînes de caractères
;	Messages
,	Arguments
:	Instructions

Commandes d'édition

Touche de Fonction	Option
[F1]	insère un espace
[F2]	Efface un caractère
[F3]	Insère une ligne
[F4]	Efface une ligne
[F5]	Page vers le haut
[F6]	Page vers le bas
[F7]	Charge un texte
[F8]	Sauvegarde un texte
[F9]	Nouveau tampon-mémoire
[F10]	Sortie de la fenêtre d'édition
EDIT[Return]	Début l'édition

Opérateurs

Opérateur arithmétique	Opération
+	Addition; Concaténation de chaînes
-	Soustraction; Négation
*	Multiplication
/	Division
\	Division entière
^ ou **	Exponentiation

Opérateur relationnel	Opération
--------------------------	-----------

=	Egal à
<	Inférieur à
>	Supérieur à
<=	Inférieur ou égal à
>=	Supérieur ou égal à
<>	Différent de

Opérateur logique	Opération
----------------------	-----------

AND	"ET" logique
EQV	Test de l'équivalence
IMP	Test de l'implication
NOT	Négation de l'expression
OR	"OU" logique
XOR	"OU" exclusif

Mode Tracé/Dessin

Numéro	Mode
--------	------

1	Remplace
2	Transparent
3	"OU" exclusif
4	Inverse la transparence

Motifs de remplissage

Numéro	Style
0	Creux
1	Plein
2	Motif
3	Hachure

Motifs des Lignes

Numéro	Style
1	Plein
2	Tiret long
3	Point
4	Tiret Point
5	Tiret
6	Tiret Point Point
7	Défini par l'utilisateur

Ports

Numéro	Port
0	Imprimante
1	Modem
2	Console (Ecran)
3	MIDI

Fenêtre

Numéro	Fenêtre
0	Edition
1	Listage
2	Sortie
3	Commande

Sons

Paramètre	Description	Intervalle
Durée	1/50 ^e de seconde avant son suivant	
Note	Position de la note dans la gamme	1 à 12
Octave	Numéro de l'octave	1 à 8
Voix	Numéro de la voix	1 à 3
Volume	Volume	de 0 (éteint) à 15 (max)

Liste des Commandes

AUTO	LIST	SAVE
BREAK	LLIST	STEP
CONT	LOAD	TRACE
DELETE	MERGE	TROFF
DIR	NEW	TRON
EDIT	RENUM	UNBREAK
ERA	REPLACE	UNFOLLOW
FOLLOW	RUN	UNTRACE

Liste des Instructions

AREA	FILL	OUT
ASK MOUSE	FOR	PATTERN
ASK RGB	FULLW	PCIRCLE
BLOAD	GEMDOS	PELLIPSE
BOX	GET	PRINT[#]
BSAVE	GOSUB	PRINT USING
CALL	GOTO	PUT
CHAIN (MERGE)	GOTOXY	QUIT
CIRCLE	GSHAPE	RANDOMIZE
CLEAR	IF	READ
CLEARW	INPUT[#]	REM
CLOSE	KILL	RESET
CLOSEW	LET	RESTORE
COLOR	LINE INPUT[#]	RESUME
COMMON	LINEF	RETURN
DATA	LINEPAT	RGB
DEF FN	LPRINT	RSET
DEFDBL	LSET	SOUND
DEFINT	MAT AREA	SSHAPE
DEFSNG	MAT DRAW	STOP
DEFSTR	MAT LINEF	SWAP
DIM	MAT SOUND	SYSTEM
DRAW	NAME	WAVE
DRAWMODE	NEXT	WEND
ELLIPSE	ON	WHILE
END	ON ERROR GOTO	WIDTH
ERASE	OPEN	WRITE[#]
ERROR	OPENW	XBIOS
FIELD	OPTION BASE	

Liste des Fonctions

ABS	HEX\$	PEEK_L
ASC	INP	POKE_B
ATN	INPUT\$	POKE_L
BIOS	INSTR	POKE_L
CDBL	INT	POS
CHR\$	LEFT\$	RIGHT\$
CINT	LEN	RND
COS	LOC	SGN
CSNG	LOF	SIN
CVD	LOG	SPACE\$
CVI	LOG10	SPC
CVS	LPOS	SQR
EOF	MID\$	STR\$
ERR\$	MKD\$	STRING\$
EPX	MKI\$	TAB
FIX	MKS\$	TAN
FLOAT	OCT\$	VAL
FRE	PEEK	VARPTR
GEMSYS	PEEK_B	VDISYS

Liste des Variables du Système

CONTRL	GEM_GLOBAL	PI
ERL	GEM_INTIN	PTSIN
ERR	GEM_INTOUT	PTSOUT
GEM_ADDRIN	INTIN	STATUS
GEM_ADDRROUT	INTOUT	SYSTAB
GEM_CONTRL		

Commandes, Instructions, Fonctions et Variables du Système

Nom	Format/Description
=	<i><variable> = <expression numérique></i> <i><variable> = <expression chaîne></i> Assigne une valeur à une variable (voir LET).
ABS	<i>ABS(<expression numérique>)</i> Donne la valeur absolue de l'argument.
AREA	<i>AREA <liste de points></i> Dessine un polygone plein.
ASC	<i>ASC(<expression chaîne>)</i> Donne le code ASCII du caractère spécifié.
ASK MOUSE	<i>ASK MOUSE <x>, <y>, </i> Donne les coordonnées courantes de la souris et l'état du bouton dans les variables indiquées.
ASK RGB	<i>ASK RGB <reg>, <r>, <v>, </i> Place les valeurs courantes de rouge, vert, bleu de la palette spécifiée dans les variables indiquées.
ATN	<i>ATN(<expression numérique>)</i> Donne l'arctangente de l'argument.
AUTO	<i>AUTO[<numéro ligne de départ>]</i> <i>[, <incrément>]</i> Numérotation automatique des lignes entrées.

- BIOS** *BIOS* <expression numérique>, <liste d'arguments>
Effectue un appel de système d'exploitation au BIOS.
- BLOAD** *BLOAD* <spécification de fichier>, <adresse>
Charge le fichier binaire indiqué.
- BOX** *BOX [FILL]* <x1,y1>; <x2,y2>
Dessine un rectangle [plein].
- BREAK** *BREAK* [<liste de numéros de lignes>]
Insère des points d'arrêt entre les lignes du programme.
- BSAVE** *BSAVE* <spécification de fichier>, <adresse>, <longueur>
Sauvegarde une partie de la mémoire dans un fichier binaire.
- CALL** *CALL* <variable numérique> [(liste de paramètres)]
Appelle une routine en langage machine.
- CDBL** *CDBL* (<expression numérique>)
Convertit un argument en un nombre en double précision.
- CHAIN** *CHAIN* <spécification de fichier> [, <description de ligne>] [,ALL]
Remplace le programme courant par le programme spécifié et l'exécute.
- CHAIN MERGE** *CHAIN MERGE* <spécification de fichier> [, DELETE <liste des numéros de lignes>]
Chaîne le programme courant avec le programme spécifié et exécute le programme résultant.

- CHR\$** *CHR\$(<expression numérique>)*
 Convertit les nombres entiers en chaînes d'un caractère en fonction du code ASCII.
- CINT** *CINT(<expression numérique>)*
 Convertit un argument en nombre entier.
- CIRCLE** *CIRCLE <x>, <y>, <rayon> [, <angle de départ, angle de fin>]*
 Trace des cercles et des arcs de cercle.
- CLEAR** *CLEAR*
 Remet à 0 toutes les variables numériques et à chaîne nulle (vide) toutes les chaînes de caractères. Annule l'existence de toutes les variables tableau.
- CLEARW** *CLEARW<expression numérique>*
 Efface les fenêtres de ST BASIC.
- CLOSE** *CLOSE [#]<numéro de fichier>*
 Termine les opérations d'entrée et de sortie et ferme le fichier de données.
- CLOSEW** *CLOSEW<numéro de fenêtre>*
 Fermer les fenêtres BASIC.
- COLOR** *COLOR[<couleur texte, couleur de remplissage, couleur de ligne, index, style>]*
 Définit les couleurs de texte, de remplissage et de tracé, ainsi que le motif de remplissage.
- COMMON** *COMMON<variable>[, <variable>...]*
 Passe les variables spécifiées à un programme appelé, lors d'un chaînage.

- CONT** *CONT*
Reprend l'exécution après un point d'arrêt.
- CONTRL** *CONTRL(<décalage>) = <expression>*
Variable système VDI qui peut agir comme une matrice intégrée.
- COS** *COS(<expression numérique>)*
Donne le cosinus de l'argument.
- CSNG** *CSNG(<expression numérique>)*
Convertit l'argument en un nombre au format simple précision.
- CVD** *CVD(<chaîne sur 8-octets>)*
Convertit une chaîne sur 8 octets en un nombre au format double précision.
- CVI** *CVI(<chaîne sur 4-octets>)*
Convertit une chaîne sur 4 octets en un nombre entier.
- CVS** *CVS(<chaîne sur 4-octets>)*
Convertit une chaîne sur 4 octets en un nombre au format simple précision.
- DATA** *DATA <constante>[, <constante>...]*
Fournit la liste des données qui seront utilisées par l'instruction READ.
- DEF FN** *DEF FN <nom de fonction> [(<liste de variables>)] = <définition>*
Définit une fonction utilisateur.
- DEFDBL** *DEFDBL <lettre>[-<lettre>]*
Définit l'intervalle des initiales pour les nombres en double précision.

- DEFINT** *DEFINT* <lettre> [-<lettre>]
Définit l'intervalle des initiales pour les nombres entiers.
- DEFSNG** *DEFSNG* <lettre> [-<lettre>]
Définit l'intervalle des initiales pour les nombres en simple précision.
- DEFSTR** *DEFSTR* <lettre> [-<lettre>]
Définit l'intervalle des initiales pour les chaînes de caractères.
- DELETE** *DELETE* <numéro de ligne> [-<numéro de ligne>]
Supprime les lignes de programme de la mémoire.
- DIM** *DIM* <nom de matrice> (<indice> [, <indice>]...), <nom de matrice> (<indice>, <indice>...)
Associe le nom de la variable à la matrice (tableau) de dimensions données.
- DIR** *DIR* [<unité disque>:] [[<nom de fichier>], [<extension>]]
Liste le répertoire.
- DRAW** *DRAW* <liste de points>
Trace une ligne reliant les points donnés dans la liste.
- DRAWMODE** *DRAWMODE* <variable entière>
Définit le mode de tracé courant.
- EDIT** *EDIT* [<numéro de ligne>]
Edite le programme courant.
- ELLIPSE** *ELLIPSE* <x>, <y>, <rayon horizontal>, <rayon vertical> [, <angle de départ, angle de fin>]
Trace des ellipses et des arcs d'ellipse.

- END** *END*
Termine le programme, ferme les fichiers et revient en mode immédiat.
- EOF** *EOF(<numéro de fichier>)*
Détection de la fin d'un fichier.
- ERA** *ERA[<unité disque>:] [<nom de fichier>]*
Supprime un fichier du disque.
- ERASE** *ERASE <nom de matrice>[, <nom de matrice>]...*
Efface les matrices.
- ERL** *ERL = <ligne d'erreur>*
Contient le numéro de la ligne où s'est produit une erreur.
- ERR** *ERR = <code d'erreur>*
Contient le code de l'erreur.
- ERR\$** *ERR\$(<n>)*
Donne le message de l'erreur numéro n.
- ERROR** *ERROR<expression numérique>*
Simule une erreur.
- EXP** *EXP(<expression numérique>)*
Donne e (2,718) à la puissance indiquée.
- FIELD** *FIELD #<numéro fichier>, <largeur champ> AS <variable chaîne> [, <largeur champ> AS <variable chaîne>]...*
Alloue de l'espace mémoire pour les valeurs des variables dans le tampon mémoire du fichier.

- FILL** *FILL* <x>, <y>
Remplit une forme géométrique par une couleur ou un motif.
- FIX** *FIX*(<expression numérique>)
Tronque l'argument en nombre entier.
- FLOAT** *FLOAT*(<expression entière>)
Convertit un nombre entier en un nombre simple précision.
- FOLLOW** *FOLLOW* <variable>, [<variable>...]
Garde la trace des valeurs des variables spécifiées pendant l'exécution d'un programme.
- FOR...TO** *FOR* <variable compteur> = <début>
TO <limite> [*STEP* <Incrément>]
Initialise une boucle de programme.
- FRE** *FRE*[(<expression numérique>)]
Donne le nombre d'octets mémoire non utilisés par **ST BASIC**.
- FULLW** *FULLW* <numéro de fenêtre>
Définit les fenêtres de **ST BASIC** en plein écran.
- GEM_ADDRIN** *GEM_ADDRIN*
(<décalage>) = <expression>
Variable système GEM qui agit comme une matrice intégrée.
- GEM_ADDROUT** *GEM_ADDROUT*
(<décalage>) = <expression>
Variable système GEM qui agit comme une matrice intégrée.
- GEM_CONTRL** *GEM_CONTRL*
(<décalage>) = <expression>
Variable système GEM qui agit comme une matrice intégrée.

GEM_GLOBAL	GEM GLOBAL (<i><décalage></i>) = <i><expression></i> Variable système GEM qui agit comme une matrice Intégrée.
GEM_INTIN	GEM INTIN (<i><décalage></i>) = <i><expression></i> Variable système GEM qui agit comme une matrice Intégrée.
GEM_INTOUT	GEM INTOUT (<i><décalage></i>) = <i><expression></i> Variable système GEM qui agit comme une matrice Intégrée.
GEMDOS	GEMDOS <i><expression numérique></i> , <i><liste d'arguments></i> Fournit un appel du système d'exploitation à GEMDOS.
GEMSYS	GEMSYS <i><Code opération AES></i> Accède à l'Interface AES du Système d'exploitation.
GET	GET [#] <i><numéro fichier></i> [, <i><numéro enregistrement></i>] Lit l'enregistrement du fichier relatif dans le tampon mémoire.
GOSUB	GOSUB <i><numéro de ligne></i> Donne le contrôle du programme à un sous-programme.
GOTO	GOTO <i><numéro de ligne></i> Envoie l'exécution du programme à la ligne indiquée.
GOTOXY	GOTOXY <i><colonne></i> , <i><rangée></i> Place le curseur à la position indiquée (colonne et ligne).

- GSHAPE** *GSHAPE* <x1>, <y1>, <variable>
 Écrit sur l'écran la trame stockée dans la variable.
- HEX\$** *HEX\$*(<expression numérique>)
 Donne l'équivalent hexadécimal de l'argument donné, sous forme de chaîne.
- IF** *IF* <relation> *THEN* <numéro de ligne> | <label>
IF <relation> *GOTO* <numéro de ligne> | <label>
IF <relation> *THEN* <numéro de ligne> [*ELSE* <numéro de ligne> | <label>]
IF <relation> *THEN* <clause> [*ELSE* <clause>]
 Exécute la condition et décide de la direction du flux du programme en fonction du résultat.
- INP** *INP* (<numéro de port>)
 Donne la valeur de l'octet du port indiqué.
- INPUT** *INPUT* [:] [<message><:|.>] <variable> [, <variable>]...
 Entre une donnée au clavier pendant l'exécution du programme.
- INPUT#** *INPUT#* <numéro de fichier>, <variable> [, <variable>]...
 Assigne les données du fichier séquentiel aux variables indiquées.
- INPUT\$** *INPUT\$* (<nombre de caractères>[, [#] <numéro de fichier>])
 Donne une chaîne de longueur spécifiée à partir du clavier ou d'un fichier de données.

- INSTR** *INSTR*([<point de départ>,<chaîne objet>, <format chaîne>)
Recherche une sous-chaîne d'une chaîne et donne sa position.
- INT** *INT*(<expression numérique>)
Arrondit l'argument à l'entier inférieur le plus proche (partie entière).
- INTIN** *INTIN*(<décalage>) = <expression>
Une variable système VDI qui agit comme une matrice intégrée.
- INTOUT** *INTOUT*(<décalage>) = <expression>
Une variable système VDI qui agit comme une matrice intégrée.
- KILL** *KILL* <expression chaîne>
Supprime un fichier.
- LEFT\$** *LEFT\$*(<chaîne objet>, <nombre de caractères>)
Donne la sous-chaîne commençant au début de chaîne objet.
- LEN** *LEN* (<expression chaîne>)
Donne la longueur de la chaîne spécifiée.
- LET** *LET* <variable> = <expression numérique> | "chaîne"
Assigne une valeur à une variable.
- LINE INPUT** *LINE INPUT*[[;]"message";|[:]"message",
] <variable chaîne>
Assigne l'enregistrement d'un fichier relatif à la variable spécifiée.
- LINE INPUT#** *LINE INPUT#* <numéro fichier>,
<variable chaîne>
Lit une ligne du fichier de données et l'assigne à la variable indiquée.

LINEF *LINEF* <liste de points>
Trace une ligne reliant les points donnés dans la liste.

LINEPAT *LINEPAT* <style>[, <motif>]
Définit le motif de ligne.

LIST *LIST* [<liste de numéros de ligne>]
Affiche les lignes indiquées du programme dans la fenêtre LIST.

LLIST *LLIST* [<liste de numéros de ligne>]
Imprime les lignes indiquées du programme sur l'imprimante.

LOAD *LOAD* <nom de fichier>
Charge le fichier spécifié.

LOC *LOC*(<numéro de fichier> ou <numéro enregistrement>)
Donne le numéro d'enregistrement ou le nombre de caractères lus ou écrits.

LOF *LOF* (<numéro de fichier>)
Donne la longueur du fichier.

LOG *LOG*(<expression numérique>)
Calcule et donne le logarithme naturel (népérien) de l'argument.

LOG10 *LOG10*(<expression numérique>)
Calcule et donne le logarithme décimal de l'argument.

LPOS *LPOS*[(argument muet)]
Donne la position de la tête d'imprimante.

- LPRINT** *LPRINT* [*<liste d'expressions>*]
Imprime les données sur l'imprimante.
- LPRINT USING* *<expression chaîne format>* ; [*<liste d'expressions>*]
Imprime les données sur l'imprimante en respectant le format indiqué.
- LSET** *LSET* *<variable chaîne>* = *<expression chaîne>*
Déplace la donnée dans la chaîne et la justifie à gauche.
- MAT AREA** *MAT AREA* *<compte>*, *matrice*
Trace un polygone plein, en utilisant les éléments de la matrice comme coordonnées du polygone.
- MAT DRAW** *MAT DRAW* *<compte>*, *matrice*
Trace une ligne, en utilisant les éléments de la matrice comme coordonnées de la ligne.
- MAT LINEF** *MAT LINEF* *<compte>*, *matrice*
Trace une ligne, en utilisant les éléments de la matrice comme coordonnées de la ligne.
- MAT SOUND** *MAT SOUND* *<matrice>*
Emet un son, en utilisant les éléments de la matrice.
- MERGE** *MERGE* *<nom de fichier>*
Chaîne le programme spécifié au programme résident.

MID\$ *MID\$*("chaîne objet", <début> [, <longueur>])
Donne une sous-chaîne de chaîne objet.

MID\$(<variable chaîne>, <début> [, <longueur>]) = "chaîne"
Substitue une sous-chaîne à une autre dans une chaîne de caractères.

MKD\$ *MKD\$*(<expression numérique>)
Convertit un nombre double précision en chaîne de caractères.

MKI\$ *MKI\$*(<expression numérique>)
Convertit un nombre entier en chaîne de caractères.

MKS\$ *MKS\$*(<expression numérique>)
Convertit un nombre simple précision en chaîne de caractères.

NAME *NAME* <ancien nom de fichier> AS <nouveau nom>
Change le nom d'un fichier.

NEW *NEW* [<nom de fichier>]
Efface la mémoire pour le nouveau programme, et en option, lui donne un nom.

NEXT *NEXT* [<compteur> [, <compteur>]]...
Définit la fin d'une boucle.

OCT\$ *OCT\$*(<expression numérique>)
Donne dans une chaîne de caractères l'équivalent en octal de l'argument.

ON *ON* <expression> GOSUB <numéro de ligne> [, <numéro de ligne>]...
Définit des branchements multiples à des sous-programmes.

ON <expression> GOTO <numéro de ligne> [, <numéro de ligne>]...
Définit des branchements multiples.

ON ERROR GOTO *ON ERROR GOTO* 0 | <numéro de ligne>
Définit la ligne de départ de la routine de gestion d'erreurs.

OPEN *OPEN* "mode", #<numéro de fichier>, "nom de fichier" [, <longueur enregistrement>]
Ouvre le fichier de données spécifié.

OPENW *OPENW* <numéro de fenêtre>
Ouvre les fenêtres de ST BASIC.

OPTION BASE *OPTION BASE* <0 | 1>
Définit la base des éléments des matrices.

OUT *OUT* <numéro de port>, <octet>
Emet un octet sur un port de sortie.

PATTERN *PATTERN* <plan>, <matrice>
Définit le motif du style.

PCIRCLE *PCIRCLE* <x>, <y>, <rayon> [, <angle de départ>, <angle de fin>]
Trace des cercles pleins et des "camemberts".

PEEK *PEEK* B <adresse>
Donne la valeur sur 8 bits du contenu de l'adresse mémoire.

PEEK <adresse>
Donne la valeur sur 16 bits du contenu de l'adresse mémoire.

PEEK_L <adresse>
Donne la valeur sur 32 bits du contenu de l'adresse mémoire.

- ON <expression> GOTO <numéro de ligne> [, <numéro de ligne>]...*
Définit des branchements multiples.
- ON ERROR GOTO** *ON ERROR GOTO 0 | <numéro de ligne>*
Définit la ligne de départ de la routine de gestion d'erreurs.
- OPEN** *OPEN "mode", #<numéro de fichier>, "nom de fichier" [, <longueur enregistrement>]*
Ouvre le fichier de données spécifié.
- OPENW** *OPENW <numéro de fenêtre>*
Ouvre les fenêtres de **ST BASIC**.
- OPTION BASE** *OPTION BASE <0 | 1>*
Définit la base des éléments des matrices.
- OUT** *OUT <numéro de port>, <octet>*
Emet un octet sur un port de sortie.
- PATTERN** *PATTERN <plan>, <matrice>*
Définit le motif du style.
- PCIRCLE** *PCIRCLE <x>, <y>, <rayon> [, <angle de départ>, <angle de fin>]*
Trace des cercles pleins et des "camemberts".
- PEEK** *PEEK_B <adresse>*
Donne la valeur sur 8 bits du contenu de l'adresse mémoire.
- PEEK <adresse>*
Donne la valeur sur 16 bits du contenu de l'adresse mémoire.
- PEEK_L <adresse>*
Donne la valeur sur 32 bits du contenu de l'adresse mémoire.

- PELLIPSE** *PELLIPSE* <x>, <y>, <rayon horizontal>, <rayon vertical> [, <angle de départ>, <angle de fin>]
Trace des ellipses pleines et des "camemberts" elliptiques.
- PI** <variable> = *PI*
Contient la valeur de .
- POKE** *POKE_B* (<adresse>, <donnée>)
Insère la donnée sur 8 bits à l'adresse mémoire.
POKE (<adresse>, <donnée>)
Insère la donnée sur 16 bits à l'adresse mémoire.
POKE_L (<adresse>, <donnée>)
Insère la donnée sur 32 bits à l'adresse mémoire.
- POS** *POS* (<numéro de fichier>)
Donne le nombre de caractères imprimés sur la dernière ligne, ou la position du curseur sur l'écran ou l'imprimante.
- PRINT** *PRINT* [<item à imprimer>]<;|,>
[<item à imprimer>[<;|,>]...]]
? [<item à imprimer>]<;|,> [<item à imprimer>[<;|,>]...]]
Affiche les données sur l'écran.
- PRINT USING** *PRINT USING* <"format">; <liste des variables>
Imprime les données sur l'écran en respectant le format spécifié.
PRINT# <numéro de fichier>, *USING* <"format">; <expression>
[, <expression>...]
Imprime les données dans un fichier en respectant le format indiqué.

- PRINT#** *PRINT#* <numéro de fichier>, <item à imprimer> [*item à imprimer* > ...]
Sort les données dans un fichier.
- PTSIN** *PTSIN* (<décalage>) = <expression>
Variable système VDI qui agit comme une matrice intégrée.
- PTSOUT** *PTSOUT* (<décalage>) = <expression>
Variable système VDI qui agit comme une matrice intégrée.
- PUT** *PUT*[#]<numéro de fichier>[, <numéro enregistrement>]
Ecrit un enregistrement dans le fichier relatif.
- QUIT** *QUIT*
Quitte **ST BASIC** et retourne sous contrôle du Bureau GEM.
- RANDOMIZE** *RANDOMIZE* [<expression numérique>]
Positionne l'origine de la table des nombres aléatoires.
- READ** *READ* <variable>, <variable>
Assigne les données des instructions **DATA** aux variables spécifiées.
- REM** *REM* <remarque>
' <remarque>
Insère des remarques dans le programme.
- RENUM** *RENUM* [<nouvelle première ligne>]
[, <incrément>]
Renumérote les lignes du programme courant.

- REPLACE** *REPLACE* [*<nom de fichier>*] [, *liste des numéros de ligne*]
Remplace le programme existant avec la nouvelle version.
- RESET** *RESET*
Place le contenu de la fenêtre Sortie dans le tampon mémoire graphique.
- RESTORE** *RESTORE* [*<numéro de ligne>*]
Restaure le pointeur à l'instruction DATA spécifiée.
- RESUME** *RESUME* [*NEXT* | *0* | *<numéro de ligne>*]
Définit le point de retour après un gestion d'erreur.
- RETURN** *RETURN*
Marque la fin d'un sous-programme.
- RGB** *RGB* *<reg>*, *<r>*, *v>*, **
Définit la palette de couleur pour les proportions de rouge, vert et bleu spécifiées.
- RIGHT\$** *RIGHT\$* (*<chaîne objet>*, *<entier>*)
Donne une sous-chaîne située à la fin de la chaîne objet.
- RND** *RND*[(*<expression numérique>*)]
Donne un nombre aléatoire.
- RSET** *RSET* *<variable chaîne>* = *<expression chaîne>*
Déplace la donnée dans la chaîne et la justifie à droite.
- RUN** *RUN* [*<nom de fichier>*] [, *<numéro de ligne>*]
Exécute le programme.

- SAVE** *SAVE* [*<nom de fichier>*] [, *<liste des numéros de ligne>*]
Sauvegarde le programme courant sous forme de programme source.
- SGN** *SGN*(*<expression numérique>*)
Donne le signe du nombre.
- SIN** *SIN*(*<expression numérique>*)
Donne le sinus de l'argument.
- SOUND** *SOUND* *<voix>*, *<volume>*, *<note>*,
<octave>, *<durée>*
Joue une note de musique.
- SPACE\$** *SPACE\$* (*<expression numérique>*)
Donne une chaîne d'espaces de la longueur spécifiée.
- SPC** *PRINT SPC* (*<expression numérique>*)
Insère le nombre d'espaces spécifié dans la chaîne d'impression.
- SQR** *SQR* (*<expression numérique>*)
Donne la racine carrée de l'argument.
- SSHAPE** *SSHAPE* *<x1,y1>*; *<x2,y2>*, *<matrice>*
Sauvegarde la trame dans une matrice donnée.
- STATUS** *STATUS* = *<variable>*
Contient le résultat de l'appel à TOS, GEM, VDI ou AES.
- STEP** *STEP* [*<nom de fichier>*] [, *<numéro de ligne>*]
Exécute le programme ligne par ligne.
- STOP** *STOP*
Arrête l'exécution du programme.

STR\$	<i>STR\$(<expression numérique>)</i> Convertit l'argument numérique en chaîne de caractères.
STRING\$	<i>STRING\$(<expression numérique>, <expression numérique> expression chaîne>)</i> Donne la chaîne de longueur spécifiée composée du caractère spécifié.
SWAP	<i>SWAP <1ère variable>, <2ème variable></i> Echange la valeur des deux variables.
SYSTAB	<i>SYSTAB (<décalage>) = <expression></i> Fournit une table de pointeurs système.
SYSTEM	<i>SYSTEM</i> Sort de ST BASIC et revient sous contrôle du Bureau GEM.
TAB	<i>PRINT TAB(<position de tabulation>)</i> Insère une tabulation dans la ligne d'impression.
TAN	<i>TAN(<expression numérique>)</i> Donne le cosinus de l'argument.
TRACE	<i>TRACE[(<numéro de ligne>-<numéro de ligne>]</i> Exécution pas à pas avec impression des lignes indiquées.
TROFF	<i>TROFF[<numéro de ligne>-<numéro de ligne>]</i> Désactive le mode TRON .

- TRON** *TRON* [*<numéro de ligne>-<numéro de ligne>*]
Exécution pas à pas avec Impression des lignes indiquées et des valeurs des variables.
- UNBREAK** *UNBREAK* [*<numéro de ligne>-<numéro de ligne>*]
Désactive **BREAK**.
- UNFOLLOW** *UNFOLLOW* [*<variable>* [, *<variable>*]...]
Désactive **FOLLOW**.
- UNTRACE** *UNTRACE*[*<numéro de ligne>-<numéro de ligne>*]
Désactive **TRACE**.
- VAL** *VAL*(*<expression chaîne>*)
Donne la valeur numérique de la chaîne spécifiée.
- VARPTR** *VARPTR* (*<variable>* | #*<numéro fichier>*)
Donne le décalage du paramètre dans le segment.
- VDISYS** *VDISYS*{(*argument muet*)}
Permet à l'utilisateur d'accéder à l'interface VDI du système d'exploitation.
- WAVE** *WAVE* *<autorisation>*, *<enveloppe>*, *<forme>*, *<période>*, *<retard>*
Contrôle des ondes utilisées dans l'instruction **SOUND**.
- WEND** *WEND*
Définit la fin de la boucle **WHILE**.

WHILE	WHILE <expression logique> Définit le début et la condition d'une boucle indéfinie.
WIDTH	WIDTH [#<numéro de fichier>,<largeur> Définit la largeur de la sortie sur écran. WIDTH LPRINT <largeur> Définit la largeur de la sortie sur imprimante.
WRITE	WRITE [<expression> , <expression>] Affiche sur l'écran.
WRITE#	WRITE# <numéro de fichier>,<expression> [, <expression>]... Envoie la sortie dans un fichier séquentiel.
XBIOS	XBIOS <fonction> [, <liste d'arguments>] Fournit un appel du système d'exploitation au BIOS.

Note : <relation> est une expression numérique évaluée sous forme entière. Si la valeur est zéro, elle est considérée comme fausse. Si elle est non nulle, elle est considérée comme vraie.

SERVICE INFORMATIONS CLIENTS

Nous nous ferons un plaisir de répondre à toutes vos questions concernant les ordinateurs **ATARI ST, MEGA ST** et leur environnement matériel et logiciel. N'hésitez pas à nous écrire :

ATARI
Service Informations Clients
9, rue Sentou
92150 SURESNES

Avez-vous pensé à contacter un des Clubs d'Utilisateurs d'ATARI ? Ces lieux d'échange stimulants sont de véritables mines d'informations. Vous y apprendrez à exploiter toutes les ressources de votre ordinateur ATARI. Pour en obtenir la liste, écrivez-nous en joignant une enveloppe timbrée, libellée à votre adresse.

N'oubliez pas de nous préciser sur l'enveloppe l'objet de votre lettre !

(C) 1987 ATARI Corporation Tous droits réservés

C100536-003
C033007-0A4
Printed in Taiwan
K. I. 2. 1988

 **ATARI FRANCE**
9 rue Sentou
92150 SURESNES