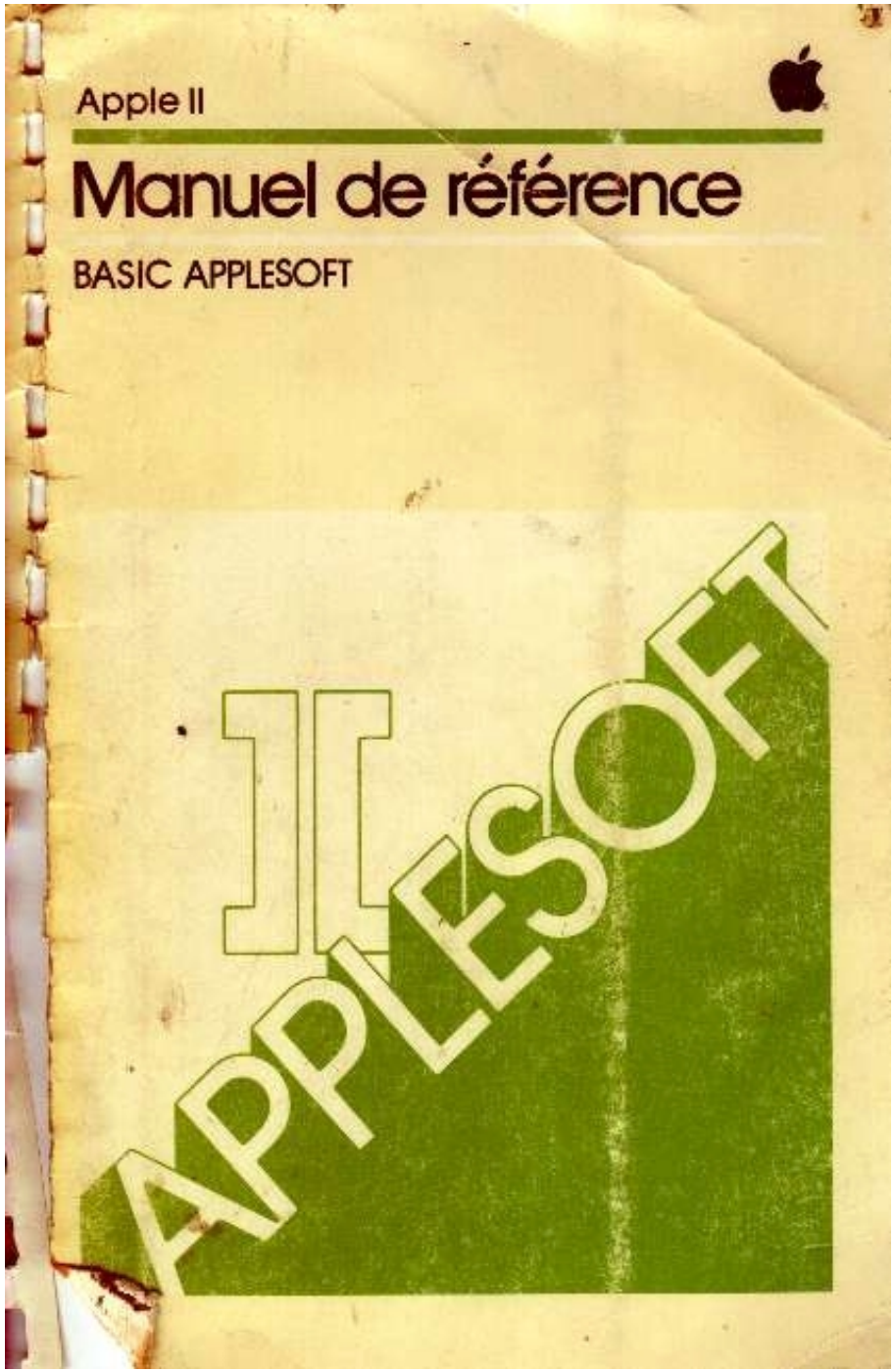


AVERTISSEMENT

La Société Apple Computer Inc se réserve le droit d'apporter des améliorations au produit décrit dans le présent manuel à tout moment et sans préavis

REFUS DE TOUTES GARANTIES ET RESPONSABILITES

Apple Computer Inc n'accorde aucune garantie, explicite ou implicite, en ce qui concerne le présent manuel ou le logiciel décrit dans ce manuel, sa qualité, ses performances, sa commercialisabilite ou son adéquation a un objet quelconque Le logiciel d'Apple Computer Inc est vendu ou licencie "en l'état" La totalité du risque, pour sa qualité et ses performances, est supportée par l acheteur Si les programmes se révèlent défectueux a la suite de leur achat, l'acheteur (et non pas Apple Computer Inc, son distributeur ou son revendeur) supportera la totalité du coût de dépannage, de réparation ou de correction et des dommages indirects En aucun cas, Apple Computer Inc ne sera responsable des dommages directs ou indirects résultant d'un défaut quelconque du logiciel, même si Apple Computer Inc a été avise de l'existence de tels dommages

Apple Computer International
7 rue de Chartres
92200 Neuilly-sur-Seine
FRANCE

" Apple " est une marque déposée de Apple Computer Inc
N° de produit Apple A2L -0006 -F

TABLE DES MATIERES

VUE GENERALE

CHAPITRE 1

Commençons

CHAPITRE 2

Définition

CHAPITRE 3

Instructions du système et instructions utilitaires

CHAPITRE 4

Instructions d'édition et instructions relatives du format

Dans le chapitre 3, voir également CTRL C.

CHAPITRE 5

Tableaux et Chaînes

CHAPITRE 6

Instructions d'entrée/sortie

Dans le chapitre 3, voir également LOAD et SAVE.
Dans le chapitre 5, voir également STORE et RECALL.

CHAPITRE 7

Instructions relatives au déroulement du programme

CHAPITRE 8

Commandes graphiques et commandes de jeux

CHAPITRE 9

Formes haute résolution

CHAPITRE 10

Fonctions mathématiques

APPENDICES

VUE GENERALE

Introduction

Applesoft II BASIC est le langage BASIC très étendu de APPLE. BASIC a été étendu car il y a de nombreuses caractéristiques de l'ordinateur APPLE II qui ne sont pas disponibles sur les autres ordinateurs qui utilisent BASIC. En ajoutant quelques nouveaux mots au langage BASIC, ces caractéristiques sont immédiatement disponibles à quiconque utilise Applesoft. Parmi les caractéristiques reprises par Applesoft il y a les graphiques en couleur de APPLE, les graphiques couleur haute résolution et les entrées analogiques directes (les contrôleurs de jeu).

Une autre caractéristique de Applesoft est ce manuel. Ce n'est pas un manuel autodidacte, puisque APPLE fournit un autre manuel (le manuel de programmation Applesoft II Basic qui vous aide à étudier la programmation même si vous n'avez jamais touché à un ordinateur auparavant. Ce manuel suppose que vous savez comment programmer en BASIC et que vous souhaitez simplement apprendre les caractéristiques supplémentaires offertes par Applesoft. Le chapitre 1 (commençons) est une rapide revue de ce que le langage a à offrir. Le reste du manuel est une description soigneuse et exacte de chaque instruction du langage et de la façon dont elle est exécutée. Pour vous éviter la frustration et l'ennui que certains manuels peuvent engendrer, ce manuel attire l'attention sur les endroits où les erreurs de programmation peuvent vous causer des difficultés. Des symboles spéciaux attirent votre attention sur ces points.

La méthode utilisée pour décrire Applesoft est presque un langage simple en lui-même. Après un certain temps pour vous y habituer, vous trouverez qu'il accélère votre compréhension de ce qui est exactement légal et illégal dans te langage. Vous ne resterez pas avec des doutes qui vous harcèlent au sujet de l'interprétation d'une phase, comme cela peut se produire avec des descriptions uniquement faites en langue courante.

Les programmeurs déjà avancés trouveront ce manuel particulièrement utile. Les programmeurs débutants doivent savoir qu'ils ne resteront pas longtemps débutants et ils apprécieront l'effort supplémentaire fait par APPLE pour leur fournir un manuel complet (ce qui n'est pas habituel). Bien sûr un manuel plus épais semble plus redoutable mais lorsque vous aurez besoin de l'information vous serez heureux que nous ayons pris le temps et la place de l'y mettre.

Pour Utiliser ce manuel

Ce manuel de référence suppose que vous disposez d'une connaissance minima de travail du langage de programmation BASIC. Si vous n'êtes pas familiarisé avec le BASIC, le manuel de programmation Applesoft II BASIC peut servir d'introduction: il recouvre une version du BASIC très voisine de Applesoft (l mais plus simple.

Nous vous recommandons d'avoir le Applesoft II Basic (auquel on se réfère sous le nom de Applesoft) prêt à être utilisé lorsque vous consulter Ce manuel, de façon à pouvoir essayer sur votre ordinateur tout ce que ce manuel décrit ou suggère. Si Applesoft passe sur votre système, le caractère ( ]) sera affiché. Voir appendice A pour une explication sur la façon de charger Applesoft dans votre ordinateur.

Il y a deux termes que vous devez connaître pour lire ce manuel. Le mot "syntaxe" se rapporte à ta structure d'une instruction ordinateur, c'est-à-dire à l'ordre et à la forme correcte des différentes parties de l'instruction. Le mot "analyse" se rapporte à la façon dont l'ordinateur s'efforce d'interpréter ce que vous tapez, en examinant les différentes parties des instructions ordinateurs pour les exécuter. Par exemple la synthèse de Applesoft vous permet de taper:

12x5-4*3^2

Lorsque Applesoft analyse cette entrée, il prend d'abord 12 comme numéro de la ligne de programme, puis interprète X5 comme étant un nom de variable arithmétique. Enfin Applesoft évalue 3/2 comme valant 9, puis multiplie par 4 et affecte la valeur 36 à la variable dont le nom est X5.
Le chapitre 1 donne une vue générale de plusieurs instructions Applesoft pour ceux qui n'ont que peu d'expérience de programmation en BASIC. On introduit de nombreux concepts primaires en utilisant les exemples que vous pouvez taper sur votre ordinateur. L'appendice B attire l'attention sur certains points concernant l'édition des programmes Applesoft.
La notation introduite au début du chapitre 2 s'utilise pour décrire la syntaxe de Applesoft avec concision et sans ambiguïté. Elle vous économisera du temps et des efforts pour comprendre la façon dont les instructions doivent être structurées. Vous n'avez pas besoin d'utiliser cette notation par vous-même mais elle vous aidera à répondre à de nombreuses questions qui ne sont pas spécifiquement discutées dans le texte. Par exemple les crochets ([et]) sont utilisés pour indiquer les portions optionnelles d'une instruction; les accolades ([et]) sont utilisées pour indiquer les portions qui peuvent être répétées. Par exemple

[LET] C = 3

indique le mot LET est optionnel et peut être omis. Et

REM [{character}]

indique que l'instruction REMARK comprend le mot REM optionnellement suivi d'un ou de plusieurs caractères,

Les abréviations et les définitions syntaxiques de la première partie du chapitre 2 sont présentées dans un ordre logique pour ceux qui souhaitent voir comment nous avons bâti notre système de symbole et de définitions. Vous pouvez préférer ignorer ces symboles et définitions jusqu'à ce que vous en rencontriez dans le texte. A ce moment là vous pouvez pour référer au glossaire alphabétique des termes syntaxiques donnés dans l'appendice M.
Les chapitres 3 à 10 présentent des explications détaillées des instructions Applesoft, groupées par sujet. Si vous êtes intéressé de connaître une commande spécifique, l'index alphabétique de la dernière page vous indiquera où il faut regarder. Vous pouvez trouver dans les appendices des renseignements supplémentaires non repris dans les différents chapitres.
A certains endroits vous verrez le symbole:

précédant un paragraphe. Ce symbole indique une caractéristique inhabituelle sur laquelle il faut attirer votre attention.
Le symbole

précède les paragraphes qui décrivent des situations où il peut ne pas être possible de récupérer Applesoft. Vous perdrez votre programme et vous aurez probablement à redémarrer Applesoft.

CHAPITRE 1

Commençons

Instructions à exécution immédiate

Essayez de taper ce qui suit

```
PRINT 10-4
```

puis enfoncer la touche marquée RETURN

APPLESOFT va immédiatement imprimer
6

L'instruction PRINT que vous avez tapée a été exécutée dés que vous avez enfoncé la touche RETURN APPLESOFT a calculé la valeur de ta formule située après le 4 et a tapé sa valeur, dans ce cas 6.

Maintenant essayez de taper ceci:

```
PRINT 1/2,3*10
```

(* signifie multiplier, / signifie diviser)

Lorsque vous enfoncez la touche RETURN APPELSOFT va imprimer

```
.5          30
```

Comme vous pouvez voir, APPLESOFT effectue la division et la multiplication aussi bien que la soustraction. Notez la façon dont on a utilisé une virgule (,) dans l'instruction PRINT pour imprimer deux valeurs au lieu d'une. L'utilisation de la virgule avec l'instruction PRINT divise la ligne de 40 caractères en 3 colonnes ou "champs du tabulateur". Voir la discussion concernant les champs du tabulateur dans le chapitre 6 sous l'instruction PRINT.

Instructions à exécution différée

Les instructions comme l'instruction PRINT que vous venez de taper sont appelées instructions à exécution immédiate. Il y a un autre type d'instructions appelées instructions différées. Chaque instruction à exécution différée commence par un numéro de ligne. Une numéro de ligne est un entier de 0 à 63999.

Essayez de taper les lignes suivantes:

```
10 PRINT 2+3
20 PRINT 2-3
```

(n'oubliez pas que chaque ligne doit se terminer en enfonçant la touche RETURN).

Une suite d'instructions à exécution différée s'appelle un "programme". Au lieu d'exécuter immédiatement les instructions à exécution différée, APPLESOFT BASIt enregistre les instructions à exécution différée en mémoire APPLE. Lorsque vous tapez RUN, APPLESOFT exécute d'abord l'instruction enregistrée en mémoire dont le numéro de ligne est le plus faible, puis l'instruction à numéro de ligne le plus élevé et le plus proche, etc. . . , jusqu'à ce que tout le programme soit exécuté.

Supposons que vous tapiez maintenant RUN (n'oubliez pas d'enfoncer la touche RETURN à la fin de chaque ligne tapée):

```
RUN
APPLESOFT va maintenant afficher sur votre TV:
5
-1
```

Dans l'exemple précédent vous avez tapé la ligne 10 d'abord puis la ligne 20. Toutefois l'ordre dans lequel vous avez tapé les instructions à exécution différée n'a pas d'importance. APPLESOFT les exécute toujours dans l'ordre numérique correct en fonction de leur numéro de ligne.

Pour avoir un listage du programme complet actuellement en mémoire, avec les instructions disposées dans leur ordre correct, tapez

```
LIST
```

APPLESOFT va répondre par

```
10 PRINT 2+3
20 PRINT 2-3
```

On souhaite parfois effacer une ligne d'un programme. Ceci s'obtient en tapant le numéro de la ligne que l'on souhaite effacer, puis en enfonçant la touche RETURN.

Tapez ce qui suit:

```
10
LIST
```

APPLESOFT va répondre par

```
20 PRINT 2-3
```

Vous avez donc effacé la ligne 10 du programme. Il n'est pas possible de la faire revenir pour insérer une nouvelle ligne 10 il suffit de taper 10 suivi de ta nouvelle instruction que vous désirez faire exécuter par APPLESOFT.
Tapez ce qui suit:

```
10 PRINT 2*3
LIST
```

APPLESOFT va répondre par

```
10 PRINT 2*3
20 PRINT 2-3
```

Ceci constitue une façon plus facile de remplacer la ligne 10 que de l'effacer et d'insérer une nouvelle ligne. Vous pouvez le faire en tapant simplement la nouvelle ligne 10 (et en enfonçant bien sûr la touche RETURN). APPLESOFT enlève automatiquement l'ancienne ligne 10 et la remplace par la nouvelle.

Tapez ce qui suit:

```
10 PRINT 3-3
LIST
```

APPLESOFT répond par

```
10 PRINT 3-3
20 PRINT 2-3
```

II n'est pas recommandé de numéroter les lignes de programme à la suite: il peut être nécessaire plus tard d'insérer une nouvelle ligne entre deux lignes existantes. Un de 10 entre les numéros de lignes est généralement suffisant.

Si vous désirez effacer la totalité du programme actuellement enregistré en mémoire, tapez

```
NEW
```

Si vous avez fini de faire passer un programme et si vous en commencez maintenant un autre, commencez par taper NEW. Ceci évitera un mélange de l'ancien et du nouveau programme.
Tapez ce qui suit:

```
NEW
```

APPLESOFT va répondre par le caractère

```
]
```

Maintenant tapez

```
LIST
```

APPLESOFT va répondre par

```
]
```

ce qui indique que votre programme précédent ne se trouve plus en mémoire.

Format des nombres

Nous allons faire une digression pendant un moment pour expliquer le format des nombres imprimés par APPLESOFT BASIC. Les nombres sont enregistrés en mémoire interne avec une précision qui peut dépasser 9 chiffres. Lorsqu'un nombre est imprimé, il n'apparaît que 9 chiffres seulement. Chaque nombre peut comporter un exposant (une puissance de dix)

En APPLESOFT BASIC les nombres en "précision réelle" (également nommés 'à virgule flottante') doivent être compris entre -1 * 10^38 et 1* 10^38, sinon vous risquez d'avoir un message d'erreur. En faisant des additions ou des soustractions vous pouvez parfois engendrer des nombres aussi grands que 1.7* 10 ' 38 sans recevoir de message d'erreur. Un nombre dont la valeur absolue est inférieure à environ 3*10^-39 sera converti en zéro par APPLESOFT. En plus de ces limitations, les valeurs réelles des entiers doivent être entre -32767 et 32767.

Lorsqu'un nombre est imprimé, on utilise les règles suivantes pour déterminer son format exact.
1. Si le nombre est négatif, un signe moins (-) est imprimé
2. Si la valeur absolue du nombre est un entier entre 0 et 999999999, ce nombre est imprimé comme entier.
3. Si la valeur absolue du nombre est supérieure ou égale à .01 et inférieure à 999999999.2, le nombre est imprimé en notation à virgule fixe sans exposant.
4. Si le nombre ne tombe pas dans les catégories 2 ou 3, c'est la notation scientifique qui est utilisée.

La notation scientifique est utilisée pour imprimer les nombres en précision réelle et son format est comme suit:
SX.XXXXXXXESTT
où chaque X est un entier de O à 9.

Le S de tête est le signe du nombre, rien pour un nombre positif et signe moins (-) pour un nombre négatif. Un chiffre autre que 0 est imprimé avant le point décimal, II est suivi du point décimal puis des huit autres chiffres de la mantisse. Un E (pour exposant) s'imprime alors, suivi du signe (S) de l'exposant: puis des deux chiffres (TT) de l'exposant lui-même. Les zéros de tête ne sont jamais imprimés; i.e. le chiffre avant le point décimal n'est jamais zéro. De même les zéros de queue ne sont jamais imprimés.
S'il n'y a qu'un seul chiffre à imprimer après suppression de tous les zéros de queue, on n'imprime pas de point décimal. Le signe de l'exposant sera plus (+) pour positif et moins (-) pour négatif. On imprime toujours deux chiffres de l'exposant, c'est-à-dire que les zéros ne sont pas supprimés dans le champ de l'exposant.
La valeur d'un nombre exprimé sous forme de notation scientifique comme décrit ci-dessus est le nombre qui se trouve à gauche de E multiplié par 10 élevé à ta puissance du nombre qui se trouve à la droite de E.

Ci-dessous quelques exemples de différents nombres et du format de sortie que utilise pour les imprimer:

NOMBRE	FORMAT DE SORTIE
+1	1
-1	-1
6523	6523
-23.460	-23.46
45.72E5	4572000
1*10^20	1E+20
-12.34567896*10^10	-1.2345679E+11
1000000000	1E+09
999999999	999999999

Un nombre tapé au clavier ou une constante numérique utilisée dans un programme APPLESOFT peuvent avoir autant de chiffres que désiré, jusqu'à la longueur maxima de 38 chiffres. Toutefois seuls les dix premiers chiffres sont habituellement significatifs et le dixième chiffre est arrondi:
Par exemple si vous tapez

```
PRINT 1.23456787654321
APPLESOFT répond par
1.23456788
```

Exemples de graphiques couleur

Tapez

```
GR
```

Ceci va effacer et rendre noir les vingt lignes supérieures du texte sur votre écran TC et ne laisser que quatre lignes de texte en bas. Votre APPLE se trouve maintenant en mode "Color Graphics" basse résolution.
Maintenant tapez

```
COLOR =13
```

APPLESOFT ne va répondre que par le caractère

```
]
```

et le clignotement du curseur, mais à l'intérieur il se souvient que vous avez choisi une couleur jaune.

Maintenant tapez

```
PLOT 20, 20
```

APPLESOFT va répondre en traçant un petit carré jaune au centre de l'écran. Si le carré n'est pas jaune, c'est que votre appareil TV n'est pas correctement réglé: il faut régler les commandes de couleur pour obtenir un jaune citron clair.

Maintenant tapez

```
HLIN 0,30 AT 20
```

APPLESOFT va tirer un trait horizontal en travers du quart le plus à gauche de l'écran, à un quart en dessous du haut.

Maintenant tapez COLOR = 6

pour avoir une nouvelle couleur puis tapez la ligne

```
VLIN  10,39 AT 30
```

Vous en apprendrez davantage plus tard sur "Color Graphics". Pour revenir au mode texte, tapez

```
TEXT
```

l'affichage de caractères sur l'écran est la façon dont APPLE montre l'information couleur sous forme de texte.

Lors de l'impression des réponses aux problèmes, il est souvent souhaitable d'inclure du texte avec les réponses, pour expliquer la signification des nombres tapez ce qui suit:

```
PRINT  ONE THIRD IS EQUAL TO",  1/3
```

APPLESOFT va répondre par

```
ONE THIRD IS EQUAL TO .333333333
```


Format d'impression

Comme expliqué précédemment, le fait d'inclure une virgule (,) dans une instruction PRINT oblige à passer jusqu'au champ de tabulateur suivant avant d'imprimer la valeur qui suit la virgule. Si vous utilisez un point virgule (;) à la place d'une virgule, la valeur suivante sera imprimée immédiatement à la suite de la valeur précédente. Essayez,

```
PRINT 1,2,3
1        2        3
```

Essayez les exemples suivants

```
PRINT 1:2;3
123


PRINT-1:2;-3
-12-3
```

Ce qui suit est un exemple d'un programme qui lit une valeur à partir du clavier et utilise cette valeur pour calculer et imprimer un résultat.

```
10 INPUT R
20 PRINT 3.14159*R*R
RUN
?10
314.159
```

Voici ce qui se passe. Lorsque APPLESOFT rencontre l'instruction INPUT, il affiche un point d'interrogation (?) sur l'écran puis attend que vous tapiez un nombre. Lorsque vous opérez, par exemple dans l'exempte précédent, c'est 10 qui a été tapé, la valeur tapée est affectée à la variable qui suit INPUT (dans ce cas on a donné à la variable R la valeur 10). Puis l'exécution se poursuit par l'instruction suivante du programme, qui dans l'exemple précédent est la ligne 20. Pour le calcul de la formule qui vient après l'instruction PRINT, on remplace la variable R par la valeur 10 chaque fois que R apparaît dans la formule. La formule devient donc 3.14159* 10* 10, soit.314.159.

Si vous ne t'aviez pas déjà deviné, le programme ci-dessus calcule la surface d'un cercle de rayon R.

Si nous désirons calculer ta surface de différents cercles, nous pourrions ré exécuter le programme pour chaque cercle successif. Mais il y a une façon plus simple de le faire, simplement en ajoutant une autre ligne au programme comme suit:

```
30 GOTO 10

RUN
?10
314.159
?3
28.27431
?4.7
69.3977231
?
BREAK IN 10
]
```

En plaçant une instruction GOTO à la fin de votre programme, vous l'obligez à revenir à la ligne 10 chaque fois qu'il a imprimé une réponse. Ceci pourrait se poursuivre indéfiniment mais nous avons décidé d'arrêter après avoir calculé la surface de trois cercles. Cet arrêt se fait en tapant un CONTROL C (c'est-à-dire en tapant ta lettre C tout en maintenant enfoncée la touche CTRL) puis en enfonçant la touche RETURN. Ceci provoque une rupture dans l'exécution du programme qui nous permet de l'arrêter. En utilisant CONTROL C on peut arrêter tout programme après avoir exécuté l'instruction en cours. Essayez par vous-même.

Noms de variables

La lettre R dans le programme que nous venons d'exécuter s'appelait une "variable". C'est simplement un emplacement de mémoire dans l'ordinateur, identifié par le nom R. Un nom de variable doit commencer par un caractère alphabétique et peut être suivi de tout caractère alphanumérique. Un caractère alphanumérique est toute lettre de A à Z ou tout chiffre de 0 à 9.

Un nom de variable peut avoir jusque 238 caractères de long, mais APPLESOFT n'utilise que les deux premiers caractères pour distinguer un nom d'un autre. Par conséquent les noms GOOD 4 NOUGHT et GOLDRUSH se réfèrent à la même variable.

Dans un nom de variable tous les caractères alphanumériques qui suivent les deux premiers sont ignorés à moins qu'ils ne contiennent un "mot réservé". Certains mots utilisés dans les instructions APPLESOFT BASIC sont "réservés' ' pour leur usage spécifique. Vous ne pouvez pas utiliser ces mots comme noms de variable ou comme partie d'un nom de variable. Par exemple FEND serait illégal car END est un mot réservé. Les mots réservés de APPLESOFT BASIC sont repris sur une liste et discutés dans l'appendice F.

Les noms de variable qui se terminent par un $ ou un % ont une signification spéciale et on en discute plus loin dans ce chapitre dans le paragraphe variables réelles, entières et chaînes.

Ci-dessous quelques exemples de noms de variable légaux et illégaux.

LEGAL	**ILLEGAL**
TP.	TO (les noms de variable ne peuvent pas être des mots
PDTG $	réservés)
COUNT	RGOTO (les noms de variable ne peuvent pas contenir
N 1%	de mots réservés).

En plus d'affecter une valeur à une variable avec l'instruction INPUT, vous pouvez également définir la valeur d'une variable avec une instruction LET ou instruction d'affectation.

Essayez l'exempte suivant:

```
A = 5
PRINTA, A*2
5      10

LET Z=7
PRINTZ, Z-A
7      2
```

Comme on peut le voir sur ces exemples, le LET est en option dans une instruction d'affectation.

BASIC se souvient des valeurs qui ont été affectées aux variables en utilisant ce type d'instructions. Ce processus de souvenir utilise de la place dans la mémoire de APPLE pour enregistrer les données.

Les valeurs des variables sont enlevées et l'emplacement en mémoire est utilisé pour les enregistrer et libéré lorsqu'il se produit l'un des quatre événements suivants
1. Une nouvelle ligne est tapée dans le programme ou une ancienne ligne est effacée.
2. Une instruction CLEAR est émise.
3. Une instruction RUN est émise.
4. On tape NEW.

Voici un autre fait important: à moins que vous ne leur affectiez une autre valeur, toutes les variables numériques prennent automatiquement la valeur zéro. Essayez cet exemple:

```
PRINT Q, Q+2, Q*2
0              2   0
```

Une autre instruction est l'instruction REM. REM est l'abréviation de remarque. Cette instruction s'utilise pour insérer des commentaires ou des notes dans un programme. Lorsque BASIC rencontre une instruction il ignore le reste de la ligne. Cette instruction sert principalement à aider le programmeur et non pas comme fonction utile du programme pour résoudre un problème particulier.

IF... THEN...

Ecrivons un programme pour vérifier si un nombre qui a été tapé est zéro ou non. Avec les instructions dont nous avons discuté jusqu'ici, ceci ne peut pas se faire. Ce dont nous avons besoin est une instruction qui réalise un branchement conditionnel à une autre instruction. C'est ce que fait l'instruction IF...THEN.

Tapez NEW puis tapez ce programme.

```
10 INPUTB
20 IF B=0THENGOTO50
30 PRINT "NON-ZERO"
40 GOTO10
50 PRINT "ZERO"
60 GOTO10
```

Lorsque ce programme passe, il va imprimer un point d'interrogation et attendre que vous tapiez une valeur pour B.
Tapez la valeur que vous voulez.
L'ordinateur va alors arriver à l'instruction IF. Entre la portion IF et la portion THEN de l'instruction, il y a une relation. Cette relation consiste en deux expressions séparées par l'un des symboles suivants

SYMBOLE	SIGNIFICATION
=	EGAL A
>	SUPERIEUR A
<	INFERIEUR A
<> ou ><	DIFFERENT DE
<=	INFERIEUR OU EGAL A
>=	SUPERIEUR OU EGAL A

L'instruction IF est vraie ou fausse, selon que la relation est vraie ou fausse. Dans notre programme actuel par exemple si on a tapé 0 pour B, la relation B = 0 est vraie. Donc l'instruction IF est vraie et l'exécution du programme se poursuit avec la portion THEN de l'instruction: GOTO. A la suite de cette instruction l'ordinateur va sauter à la ligne 50. Il va imprimer 0 puis l'instruction GOTO de la ligne 60 va renvoyer l'ordinateur à la ligne 10.

Supposons que pour B on ait tapé le chiffre 1. Du fait que la relation B == 0 est maintenant fausse, l'instruction IF est fausse et l'exécution du programme se poursuit par la ligne de numéros suivante, en ignorant la portion THEN de l'instruction et toute autre instruction de cette ligne. C'est donc NON-ZERO qui va être imprimé et l'instruction GOTO de la ligne 40 va renvoyer l'ordinateur à la ligne 10.

Essayez maintenant le programme suivant pour comparer deux nombres (n'oubliez pas de taper NEW d'abord, pour effacer votre dernier programme)

```
10 INPUTA.B
20 IF A <=B THEN GOTO 50
30 PRINT"A IS LARGER"
40 GOT010
50 IF A <B THEN GOTO 80
60 PRINT "THEY ARE THE SAME"
70 GOT010
80 PRINT "B IS LARGER"
90 GOT010
```

Lorsque ce programme passe, la ligne 10 fait imprimer un point d'interrogation et attend que vous tapiez deux nombres, séparés par une virgule. A la ligne 20, si A est supérieur à B, A <=B est faux et THEN GOTO 50 est ignoré. L'exécution du programme saute alors à l'exécution donnée par le numéro de ligne suivant, et imprime A IS LARGER et finalement la ligne 40 renvoie l'ordinateur à la ligne 10 pour recommencer.

A la ligne 20, si A a la même valeur que B, A<=B est vrai alors THEN GOTO est exécuté et envoie l'ordinateur à la ligne 50. A la ligne 50 puisque A a la même valeur que B, A<B est faux. Donc THEN GOTO 80 est ignoré et l'ordinateur passe au numéro de ligne suivant, où on lui demande d'imprimer THEY ARE THE SAME. Finalement la ligne 70 envoie l'ordinateur au début.

A la ligne 20 si A est inférieur à B, A<=B est vrai de sorte que l'exécution du programme se poursuit avec THEN GOTO 50. A la ligne 50, A<B est vrai donc THEN GOTO 80 est exécuté. Finalement on imprime B IS LARGER et on renvoie à nouveau l'ordinateur au début.

Essayez de faire passer les deux derniers programmes plusieurs fois. Puis essayez d'écrire votre propre programme en utilisant l'instruction IF. . .THEN. En fait essayer d'écrire des programmes vous-même est la méthode la plus rapide et la plus facile de comprendre comment travaille APPLESOFT BASIC. N'oubliez pas que pour arrêter ces programmes, il suffit de taper CONTROL C et d'enfoncer RETURN.

Un autre exemple concernant la couleur

Essayons un programme graphique. Notez l'emploi des instructions REM peut donner des explications. Les deux-points (:) s'utilisent pour séparer les instructions multiples sur une des lignes de programme numérotée. Après avoir tapé le programme ci-dessous, demandez à le voir par l'instruction LIST et, vérifiez que vous l'avez tapé correctement. Puis faites-le passer par l'instruction RUN.


```
100 GR : Positionner le mode graphique couleur
110 HOME: Effacer la zone de texte
120 X=0:Y=5: Positionner la position de départ
130 XV = 2: Positionner la vitesse sur l'axe X
140 YV = 1 : Positionner sur l'axe Y
150 REM Calculer ta nouvelle position
160 NX=X+XV:NY=Y+YV
170 REM IF Si la balle dépasse le bord de l'écran, elle rebondit
180 IFNX>39THENNX=39:XV=-XV
190 IFNX<0THENNX=0:XV=-XV
200 IFNY>39THENNY=39:YV=-YV
210 IFNY<0THENNY=0:YV=-YV
220 REM PLOT Tracer la nouvelle position en jaune
230 COLOR= 13: PLOT NX, NY
240 REM Effacer l'ancienne position
250 COLOR= 0: Tracer X,Y
260 Sauvegarder la position actuelle
270 X=NX:Y=NY
280 REM STOP AFTER 250 MOVES
290 l = l + 1 : IF 1 < 250 THEN GOTO 160
300 PRINT pour revenir à votre programme tapez TEXT
```

L'instruction GR demande à APPLE de commuter en mode graphique couleur. Il efface également la zone de tracé de 40 par 40 pour la mettre en noir, il positionne l'emplacement de sortie de texte en une fenêtre de quatre lignes de 40 caractères chacune en bas de l'écran et définit la nouvelle couleur à tracer comme étant le noir.

HOME s'utilise pour effacer la zone de texte et positionner le curseur dans l'angle supérieur gauche de la fenêtre actuellement définie pour le texte. Dans le mode graphique couleurs ceci se fera au début de la ligne de texte 20 puisque les lignes de textes 0 à 19 sont maintenant utilisées comme zones de tracé graphique couleur.

Les instructions COLOR des lignes 230 et 250 donnent à la nouvelle couleur à tracer la valeur de l'expression qui vient à la suite de COLOR=.
L'instruction PLOT NX,NY de la ligne 230 trace un petit carré de couleur jaune définie par la plus récente instruction COLOR, à la nouvelle position spécifiée par les expressions NX et NY. Se souvenir que NX et NY doivent être chacun un nombre allant de 0 à 39, sinon te carré sera en dehors de l'écran et il en résultera un message d'erreur.

De même l'instruction PLOT X,Y de la ligne 250 trace un petit carré à la position spécifiée par les expressions X et Y. Mais X et Y sont simplement les anciennes coordonnées NX et NY, sauvegardées après le tracé du précédent carré jaune. Par conséquent PLOT X,Y repasse sur l'ancien carré jaune en y traçant un carré dont la couleur est définie par COLOR= 0. C'est le noir, la même couleur que le fond, c'est pourquoi l'ancien carré jaune semble avoir été effacé.

Note: pour revenir du mode graphique couleur au mode entièrement texte, tapez TEXT puis enfoncez la touche RETURN.

Tapez TEXT comme on vient de le dire est votre façon d'échapper au mode graphique. Ne faîtes pas attention aux symboles étranges qui se trouvent sur l'écran - - ils résultent de la conversion de l'affichage graphique en caractères de texte. Si vous n'avez pas compris la ligne 290, soyez patient. Elle sera expliquée dans les pages suivantes.

Comme vous le voyez APPLE II peut faire plus que simplement utiliser des nombres. Nous reviendrons au mode graphique couleur lorsque vous en aurez appris davantage sur APPLESOFT BASIC.

FOR... NEXT

Un avantage des ordinateurs est leur aptitude à exécuter des tâches répétitives.
Supposons que nous souhaitions avoir une table des racines carrées pour les entiers de 1 à 10.
La fonction APPLESOFT BASIC pour la racine carrée est SQR; la forme étant SQR (X).
Où X est le nombre dont on souhaite calculer la racine carrée. Nous pourrions écrire le programme comme suit:

```
10   PRINT1,SQR(1)
20   PRINT 2, SQR(2)
30   PRINT 3, SQR(3)
40   PRINT 4, SQR(4)
50   PRINT 5, SQR(5)
60   PRINT 6, SQR(6)
70   PRINT 7, SQR(7)
80   PRINT 8, SQR(8)
90   PRINT 9, SQR(9)
100 PRINT 10, SQR(10)
```

Ce programme va faire le travail; toutefois il est terriblement inefficace. Nous pouvons t'améliorer beaucoup en utilisant l'instruction IF, que nous venons de voir, comme suit:

```
10 N= 1
20 PRINT            N,SQR(N)
30 N = N + 1
40 IFN<= 10 THEN GOTO 20
```

Lorsque l'on fait passer ce programme ces données de sortie sont exactement les mêmes que celles du programme ci-dessus comportant 10 instructions. Voyons comment il travaille

A la ligne 10 il y a une instruction LET qui donne à la variable N la valeur 1. A la ligne 20 l'ordinateur doit imprimer N et la racine carrée de N, en utilisant la valeur actuelle de N. La ligne 20 devient donc

```
20 PRINT 1, SQR (1)
```

et le résultat du calcul est imprimé.

A la ligne 30 il y a ce qui nous apparaît pour la première fois être une instruction LET plutôt inhabituelle. Mathématiquement l'instruction N = N -l- lest un non-sens. Toutefois la chose importante dont il faut se souvenir est que dans une instruction LET, le symbole " = " ne signifie pas l'égalité. Dans ce cas " = " signifie "être remplacé par". Cette instruction saisit simplement la valeur actuelle de N et lui ajoute 1. Donc après le premier passage par la ligne 30, N devient 2.

A la ligne 40 puisque maintenant N est égal à 2, la relation N< = 10 est vraie, donc la portion THEN de l'instruction renvoie l'ordinateur à la ligne 20, N ayant maintenant la valeur 2.

Le résultat global est que les lignes 20 à 40 se répètent, en ajoutant chaque fois 1 à la valeur N. Lorsque finalement N devient égal à 10 à la ligne 20, la ligne suivante va l'incrémenter pour l'amener à 11. Ce résultat est une relation fausse pour la ligne 40, la portion THEN est donc maintenant ignorée et puisqu'il n'y a pas d'autres instructions le programme s'arrête.

On appelle cette technique 'boucle' ou 'itération'. Du fait qu'on l'utilise de façon extensive en programmation, il existe des instructions spéciales BASIC pour l'utiliser. Nous pouvons les montrer avec le programme suivant.


```
10 FOR N = 1 TO 10
20 PRINT N,SQR (N)
30 NEXT N
```

Les données de sortie du programme dont le listage se trouve ci-dessus sont exactement les mêmes que les données de sortie des deux programmes précédents.

A la ligne 10, N est défini comme étant égal à 1. La ligne 20 fait imprimer la valeur de Net de sa racine carrée. A la ligne 30 nous voyons un nouveau type d'instructions. L'instruction NEXT N ajoute une unité à N et si N <= 10, l'exécution du programme revient à l'instruction qui suit le FOR. Il n'y a rien de spécial à dire concernant N dans ce cas. On peut utiliser n'importe quelle variable à condition que ce soit le même nom de variable dans les deux instructions FOR et NEXT. Par exemple on pourrait remplacer N par Z1 partout dans le programme ci-dessus et il fonctionnerait exactement de la même façon.

Supposons que nous désirions imprimer une table des racines carrées pour uniquement les entiers pairs de 1Oà 20. Le programme suivant effectuerait cette tâche.

```
10 N=10
20 PRINTN,SQR (N)
30 N=N+2
40 IFN<=20 THEN GOTO 20
```

Notez la structure similaire entre ce programme et celui pour impression des racines carrées pour les nombres de 1 à 10. Ce programme peut également s'écrire en utilisant la boucle FOR que l'on vient d'introduire.

```
10 FOR N=10 TO 20 STEP 2
20 PRINTN,SQR (N)
30 NEXT N
```

Notez que la principale différence entre ce programme et le précédent qui utilise les boucles FOR est l'addition STEP 2. Ceci demande à APPLESOFT d'ajouter 2 à N chaque fois, au lieu de 1 dans le programme précédent. S'il n'y a pas de STEP indiqué dans une instruction FOR, APPLESOFT suppose que l'on ajoute un chaque fois. Le STEP peut être suivi de n'importe quelle expression.

Supposons que nous désirions décompter de 1Oà 1. Un programme pour le faire serait le suivant:

```
10 I=10
20 PRINTI
30 I=t-I
40 IFI>= I THEN GOTO 20
```

Notez que nous vérifions actuellement si I est supérieur ou égal à la valeur finale. La raison en est que maintenant nous décomptons. Dans les exemples précédents c'était le contraire et c'est pourquoi nous devions vérifier si une variable était inférieure ou égale à la valeur finale.

L'instruction STEP que l'on vient de voir peut également s'utiliser avec des nombres négatifs dans le même but. Ceci peut se faire en utilisant le même format utilisé dans l'autre programme comme suit:

```
10 FOR I -10 TO I STEP -1
20 PRINTI
30 NEXT I
```

Les boucles FOR peuvent également s'emboîter. Voici un exemple de cette méthode

```
10 FOR I=1 TO 5
20 FOR J = 1 TO 3
30 PRINT I,J
40 NEXT J
50 NEXT I
```

Notez que NEXT J vient avant NEXT l. C'est parce que la boucle J est à l'intérieur de la boucle l. Le programme suivant est incorrect; exécutez le et voyez ce qui arrive.

```
10 FOR I=1 TO 5
20 FOR J = 1 TO 3
30 PRINTI,J
40 NEXT I
50 NEXT J
```

II ne marche pas parce que lorsque l'on rencontre NEXT l on perd tout ce que l'on savait sur la boucle J.

Tableaux

II est souvent commode de pouvoir choisir les éléments dans un tableau de nombres. APPLESOFT le permet grâce à l'emploi des tableaux.

Un tableau est une table de nombres. Le nom de ce tableau est tout nom légal de variable, A, par exemple. Le nom de tableau A est distinct de la simple variable A et vous pouvez utiliser l'un et l'autre dans le même programme.

Pour choisir un élément d'un tableau, nous donnons à A un indice: c'est-à-dire que pour choisir l'élément d'ordre l, nous mettons l entre parenthèses et nous faisons suivre A de cet indice. Par conséquent A(l) est l'élément d'ordre l du tableau A.

Note: dans cette section du manuel nous ne serons concernés que par les tableaux à une dimension; pour une discussion complémentaire des instructions APPLESOFT relatives aux tableaux, voir chapitre 5 "tableaux et chaînes".

A(l) n'est qu'un élément du tableau A. APPLESOFT doit savoir combien de place il faut réserver pour la totalité du tableau; c'est-à-dire quelles sont les dimensions maxima du tableau. Ceci se fait avec l'instruction DIM, en utilisant te format DIM A (15).

Dans ce cas nous avons réservé un emplacement pour permettre au tableau d'indice l d'aller de O ai 5. Les indices de tableau commencent toujours à 0; par conséquent dans l'exemple précédent nous avons réservé la place pour 16 nombres du tableau A.

Si on utilise A(l) dans un programme avant de l'avoir dimensionné à l'aide de l'instruction DIM, APPLESOFT réserve un emplacement pour 11 éléments (indices O à 10).

Comme exemple de la façon d'utiliser les tableaux, essayez te programme suivant qui trie une liste de 8 nombres que vous avez tapés.


```
90 DIM A(8) : DIMENSION ARRAY WITH MAX. 9 ELEMENTS
100 REMASKFOR8NUMBERS
110 FORI=1TO8
120 PRINT"TYPEANUMBER:";
130 INPUTA(I)
140 NEXTI
150 REM PASS THROUGH 8 NUMBERS, TESTING BY PAIRS
160 F=0:REMRESETTHEORDERINDICATOR
170 FORI=1TO7
180 IFA(1)<=A(1+1)THENGOTO140
190 REM INTERCHANGE A(1)ANDA(1+1)
200 T=A(1) 210 A(1)=A(1+1) 220 A(1+1)=T
230 F = 1 : REM ORDER WAS NOT PERFECT
240 NEXTI
250 REM F =0 MEANS ORDER IS PERFECT
260 IF F = 1 THEN GOTO 160 : REM TRY AGAIN
270 PRINT:REMSKIPALINE
280 REM PRINT ORDERED NUMBERS
290 FOR I = 1 TO 8
300 PRINT A(I)
310 NEXTI
```

Lorsque la ligne 90 s'exécute, APPLESOFT réserve la place pour 9 valeurs numériques, A (0) à A (8).Avec les lignes 110à 140, l'utilisateur donne la liste non triée. Le tri lui-même se fait sur les lignes 170 à 240, en explorant la liste des nombres et en inter changeant tout couple de deux nombres qui ne sont pas dans l'ordre. S est l'indicateur d'ordre parfait. F = 1 indique que l'on vient d'inter changer deux nombres. Dans ce cas la ligne 260 demande à l'ordinateur de revenir en arrière et de recommencer à vérifier.

Si un passage complet se fait à travers les huit nombres sans avoir à en inter changer (ce qui signifie qu'ils sont tous en ordre), les lignes 290 à 310 vont imprimer la liste ainsi triée. Notez qu'un indice peut être n'importe quelle expression.

GOSUB . . . RETURN

Un autre couple utile d'instructions est GOSUB et RETURN. Si votre programme exécute la même action en plusieurs endroits différents, vous pouvez utiliser des instructions GOSUB et RETURN pour éviter de dupliquer toutes les mêmes instructions pour cette action à chaque emplacement dans le programme.

Lorsqu'elle rencontre une instruction GOSUB, APPLESOFT exécute un branchement sur ta ligne dont le numéro suit GOSUB. Mais APPLESOFT se souvient de l'endroit où l'on était dans le programme avant ce branchement. Lorsqu'on rencontre l'instruction RETURN, APPLESOFT revient à la première instruction qui suit la dernière instruction GOSUB exécutée. Considérons le programme suivant:

```
20 PRINT "WHAT IS THE FIRST NUMBER";
30 GOSUB 100
40 T = N: REM SAVE INPUT
50 PRINT "WHAT IS THE SECOND NUMBER";
60 GOSUB 100
70 PRINT "THE SUM OF THE TWO NUMBERS IS"; T + N
80 STOP: REM END OF MAIN PROGRAM
100 INPUT N : REM BEGIN INPUT SUBROUTINE
110 IFN=INT(N)THENGOTO140
120 PRINT "SORRY, NUMBER MUSTBE AN INTEGER.TRY AGAIN."
130 GOTO100
140 RETURN: REM END OF SUBROUTINE
```

Ce programme demande deux nombres qui doivent être des entiers plus imprime la somme de ces deux nombres. Le sous-programme de ce programme est constitué par les lignes 100 à 140. Ce sous-programme demande un nombre et si le nombre tapé dans la réponse n'est pas un entier, il en demande un autre. Il continue jusqu'à ce que l'on tape une valeur entière.

Le programme principal imprime WHAT IS THE FIRST NUMBER (quel est le premier nombre) puis appelle le sous-programme pour avoir la valeur du nombre N. Lorsque le sous-programme renvoie (à la ligne 40) le nombre qui a été tapé (N) est sauvegardé sous forme de la variable T. Ceci se fait dans le but que, lorsque le sous-programme est appelé une seconde fois, on ne perde pas la valeur du premier nombre.

WHAT IS THE SECOND NUMBER (quel est le second nombre) est alors imprimé et le sous-programme est à nouveau appelé, le temps de prendre te second nombre.

Lorsque le sous-programme renvoie pour la seconde fois à la ligne 70, on imprime THE SUM OF THE TWO NUMBERS IS fia somme des deux nombres est), suivie de la valeur de leur somme. T contient la valeur du premier nombre tapé et N contient la valeur du second nombre.

L'instruction suivante du programme est une instruction STOP. Ceci oblige le programme à arrêter l'exécution à la ligne 80. Si l'on n'avait pas prévu une instruction STOP à ce point, l'exécution du programme retomberait dans le sous-programme à la ligne 100. Ceci n'est pas souhaitable car on nous demanderait encore de taper un autre nombre. Si nous le faisions, le sous-programme essaierait de donner une instruction RETURN; et comme il n'y avait pas d'instructions GOSUB pour appeler le sous-programme, il se produirait une erreur. Chaque instruction GOSUB dans un programme doit avoir une instruction RETURN correspondante exécutée ensuite et on ne doit rencontrer une instruction RETURN que si elle fait partie d'un sous-programme qui était appelé par une instruction GOSUB.

On peut utiliser soit l'instruction STOP soit l'instruction END pour séparer un programme de ses sous-programmes. L'instruction STOP va imprimer un message disant à quelle ligne on a rencontré le STOP; END va terminer le programme sans donner de message. Les deux instructions renvoient la commande à l'utilisateur en imprimant le caractère APPLESOFT] et un curseur clignotant.

READ... DATA... RESTORE

Supposons que vous vouliez que votre programme utilise les nombres qui ne changent pas chaque fois que l'on exécute le programme mais qu'il soit possible de changer facilement si nécessaire. BASIC contient des instructions spéciales dans ce but appelées des instructions READ et DATA.

Considérons le programme suivant:

```
10 PRINT"GUESS A NUMBER":
20 INPUT G
30 READ D
40 IF D- -999999 THEN GOTO 90
50 IF D<>G THEN GOTO 30
60 PRINT"YOU ARE CORRECT"
70 END
90 PRINT"BAD GUESS, TRY AGAIN."
95 RESTORE
100 GOT010
110 DATA 1,393,-39,28,391,-8,0,3,14,90
120 DATA 89,5,10,15,-34,-999999
```

Voici ce qui se passe lorsqu'on exécute le programme: lorsque l'on rencontre l'instruction READ, l'effet est le même que si c'était l'instruction INPUT, mais au lieu d'entrer un nombre au clavier, on lit un nombre au moyen des instructions DATA.

La première fois que l'on a besoin d'un nombre pour une instruction READ, le premier nombre de la première instruction DATA est renvoyé. La seconde fois c'est le second nombre de la première instruction DATA qui est renvoyé. Lorsque la totalité du contenu de la première instruction DATA a été lue de cette façon, c'est la seconde instruction DATA qui sera utilisée. DATA est toujours lue séquentiellement de cette façon et il peut y avoir un nombre quelconque d'instructions DATA dans votre programme.

Le but de ce programme est de jouer un jeu dans lequel vous essayez de deviner un des nombres contenus dans les instructions DAIA. Pour chaque réponse à la devinette que vous tapez, l'ordinateur lit tous les nombres contenus dans les instructions DATA jusqu'à ce qu'il trouve le nombre qui corresponde à ce que vous avez tapé. Si READ renvoie -999999 c'est que tous les nombres disponibles en DATA ont été utilisés et qu'il faut poser une nouvelle question.

Avant de passer à ta ligne 10 pour une nouvelle question, nous devons faire en sorte que READ commence à nouveau par te premier élément des données. C'est la fonction de RESTORE. Après que l'on a rencontré RESTORE, la première donnée qui sera lue sera à nouveau le premier poste de la première instruction DATA.

Les instructions DATA peuvent se placer à n'importe quel endroit du programme. Seules les instructions READ utilisent les instructions DATA dans un programme et si on rencontre ces instructions DATA à un autre moment pendant l'exécution d'un programme, elles seront ignorées.

Variables réelles entières et chaînes

II y a trois types différents de variables utilisés dans APPLESOFT BASIC. Jusqu'ici nous n'en avons utilisé qu'un seul type - les variables en précision réelle. Les nombres de ce mode sont affichés avec jusque 9 chiffres décimaux de précision et peuvent aller jusqu'à environ 10 à la puissance 38. APPLESOFT convertit vos nombres de l'écriture décimale à l'écriture binaire pour son usage interne puis à nouveau en écriture décimale lorsque vous lui demandez d'imprimer la réponse. Du fait des erreurs par arrondi et autres erreurs imprévues, les routines internes de mathématiques comme racine carrée, division et exposant ne donnent pas toujours le nombre exact que vous attendiez.

Le nombre de places à droite du point décimal peut se définir en arrondissant la valeur avant de l'imprimer. La formule générale pour ce faire est
X=INT(X*10^D+.5)/INT(10^D+.5)

Dans ce cas D est le nombre de places décimales. Une façon plus rapide de définir le nombre de places décimales est de poser P= 10^D et d'utiliser la formule
X = INT(X*P+.5)/P
où P= 10 correspond à une place, P= 100 à deux places, P= 1000 à trois places . ..etc. ..Ce qui précède fonctionne pour X>=1 et X<999999999. Une routine pour limiter le nombre de chiffres après le point décimal est donnée dans la section suivante de ce chapitre.

Le tableau ci-dessous résume les trois types de variables utilisées dans la programmation APPLESOFT BASIC

Description	Symbole à mettre à la suite du nom de la variable	Exemple
Chaînes (0 à 255 caractères)	$	ASALPH A$
Entiers (doivent aller de -32767 ...à +32767....)	%	B% C1%
Précision réelle (exposant -38 à+38 avec 9 chiffres décimaux)	Aucun	C Boy

Une variable entière ou une variable chaîne doit être suivie de % ou $ à chaque emploi de cette variable. Par exemple X,X% et X$ sont des variables différentes.

Les variables entières ne sont pas autorisées dans les instructions FOR ou DEF. Le plus grand avantage des variables entières est leur emploi dans les opérations sur les tableaux, partout où c'est possible, pour économiser de la place en mémoire.

Toutes les opérations arithmétiques se font en précision réelle. Les entiers et les variables entières sont converties en précision réelle avant d'être utilisées dans un calcul. Les fonctions SIN, COS, ATN, TAN, SQR, LOG, EXP et RND convertissent également leurs arguments en précision réelle et donnent les résultats de même.

Lorsqu'un nombre est converti en un entier, il est tronqué (arrondi par le bas).
Par exemple

```
I%=.999          A%=-.01
PRINTI%          PRINTA%
0                -1
```

Si vous affectez un nombre réel à une variable entière puis si vous imprimez la valeur de la variable entière, c'est comme si la fonction INT avait été appliquée. Il n'y a pas de conversion automatique entre les chaînes et les nombres. Le fait d'affecter un nombre à une variable chaîne par exemple se traduit par un message d'erreur. Par contre, il existe des fonctions spéciales pour convertir un type dans l'autre.

Chaînes

On se réfère à une séquence du caractère sous le nom "expression littérale". Une 'chaîne' est une expression littérale entre guillemets. Les expressions suivantes sont toutes des chaînes:
"BILL"
"APPLE"
"THIS IS A TEST"

Comme aux variables numériques, on peut affecter des valeurs spécifiques aux variables chaînes. Les variables chaînes se distinguent des variables numériques par un $ après leur nom variable.

Par exemple, essayez ce qui suit:

```
A$ = "GOODMORNING"
PRINTA$
GOOD MORNING
```

Dans cet exemple nous avons affecté à la variable chaîne A$ la valeur chaîne "GOOD MORNING".

Maintenant que nous avons donné à A$ une valeur chaîne, nous pouvons chercher quelle est la longueur de cette valeur (c'est-à-dire le nombre de caractères qu'elle contient). Nous opérons comme suit:

```
PRINT LEN(A$), LENC'YES")
12       3
```

La fonction LEN fournit un entier égal au nombre de caractères de la chaîne: c'est-à-dire sa LENgth (longueur).
Le nombre de caractères d'une expression chaîne peut aller de 0 à 255. Une chaîne qui contient 0 caractères s'appelle une chaîne 'nulle'. Avant d'attribuer une valeur du programme à une variable chaîne, cette variable est initialisée à la chaîne nulle. Le fait d'imprimer une chaîne nulle sur le terminal ne fait pas imprimer de caractères et le curseur ne s'avance pas à la colonne suivante. Essayez ce qui suit:

```
PRINT LEN(Q$); Q$; 3
03
```

Une autre façon de créer une chaîne nulle est d'utiliser

```
Q$ = ""
```

ou l'instruction équivalente

```
LET Q$ = ""
```

Le fait d'attribuer une chaîne nulle à une variable chaîne peut s'utiliser pour libérer l'espace chaîne utilisé par une variable chaîne non nulle. Mais vous pouvez avoir des ennuis en attribuant la chaîne nulle à une variable chaîne, comme on le discutera au chapitre 7 au titre de l'instruction IF.
Il est souvent souhaitable de retrouver une partie d'une chaîne et de la traiter. Maintenant que nous avons attribué la valeur A$ à ".GOOD MORNING", nous pouvons ne vouloir imprimer que les quatre premiers caractères de A$.
Nous pouvons le faire comme suit:

```
PRINT LEFT$(A$,4)
GOOD
```

LEFT$(A$,N) est une fonction chaîne qui renvoie une sous-chaîne composée des N caractères les plus à gauche de son argument de chaîne, A$ dans ce cas. Voici un autre exemple:

```
FOR N = 1 TO LEN(A$) : PRINT LEFT$(A$,N) : NEXT N
G
GO
GOO
GOOD
GOOD
GOOD M
GOOD MO
GOOD MOR
GOOD MORN
GOOD MORNI
GOOD MORNIN
GOOD MORNING
```

Puisque A$ a douze caractères, cette boucle va s'exécuter avec N=1, 2,3, ...,11,12. Au premier passage le premier caractère sera seul imprimé; au second passage les deux premiers caractères seront imprimés, etc. . . .

Il existe une autre fonction chaîne nommée RIGHT$. RIGHT$(A$,N) renvoie les N caractères les plus à droite à partir de l'expression chaîne A$. Essayez de remplacer RIGHT$ par LEFT$ dans l'exemple précédent et regardez ce qui arrive.

Il existe également une fonction chaîne qui nous permet de prendre les caractères qui se trouvent au milieu de la chaîne. Essayez ce qui suit:

```
FOR N = 1 TO LEN(A$) : PRINT MID$(A$,N) : NEXT N
```

MID$(A$,N) renvoie une sous-chaîne qui commence à la position d'ordre N de A$ et qui va jusqu'à la fin (dernier caractère) de A$. La première position de la chaîne est la position 1 et la dernière position possible de la chaîne est la position 255.

Très souvent il est souhaitable de n'extraire que le caractère d'ordre N dans la chaîne. Ceci peut se faire en appelant MID$ avec trois arguments: MID$(A$,N, 1). Le troisième argument spécifie le nombre de caractères à renvoyer, commençant par le caractère N.
Par exemple:

```
FOR N=1TO LEN(A$):   PRINT MID$(A$,N, 1),MID$(A$,N,2):NEXTN
G                    GO
O                    OO
O                    OD
D                    D
                     M
M                    MO
O                    OR
R                    RN
N                    NI
I                    IN
N                    NG
G                    G
```

Voir te chapitre 5 pour avoir davantage de détails sur le mode de travail de LEFT$, RIGHT$ et MID$. Il est également possible de concaténer des chaînes (c'est-à-dire de les réunir) par l'utilisation de l'opérateur plus (+). Essayez ce qui suit:

```
B$ = A$+" "+ "BILL"
PRINT B$
GOOD MORNING BILL
```

La concaténation est particulièrement utile si vous souhaitez extraire une chaîne et la replacer avec de légères modifications. Par exemple

```
C$ = RIGHT$(B$.3) + "-" + LEFT$(B$,4) + "-" + MID$(B$,6,7)
PRINT C$
BILL-GOOD-MORNING
```

Parfois il est souhaitable de convertir un nombre en sa représentation chaîne et vis-versa. Les fonctions VAL et STR$ effectuent ces tâches. Essayez ce qui suit.

```
STRING$ = "567.8"
PRINT VAL(STRING$)
567.8

STRING$ = STR$(3.1415)
PRINT STRING$, LEFT$(STRING$,5)
3.1415       3.141
```


La fonction STR$ peut s'utiliser pour modifier les nombres pour leur donner un certain format, pour l'entrée ou, la sortie. Vous pouvez convertir un nombre en une chaîne puis utiliser LEFT$, RIGHT$, MID$ et la concaténation pour redonner au nombre le format désiré.
Le court programme qui suit démontre comment on peut utiliser les fonctions chaînes pour modifier le format de sorties numériques.

```
100 INPUT "TYPE ANYNUMBER: "; X
110 PRINT:REM SKIP A LINE
120 PRINT "AFTER CONVERSION TO REAL PRECISION,"
130 INPUT "HOW MANY DIGITS TO RIGHT OF DECIMAL? ": D
140 GOSUB 1000
150 PRINT"***" :"REM SEPARATOR"
160 GOTO 100
1000 X$=STR$(X): REM CONVERT INPUT TOSTRING
1010 REM FIND POSITION OF E, IF IT EXISTS
1020 FORI=1 TO LEN(X$)
1030 IF MID$(X$,1,1) <> "E" THEN NEXT l
1040 REM I IS NOW AT EXPONENT PORTION (OR END)
1050 REM FIND POSITION OF DECIMAL, IF ITEXISTS
1060 FORJ = I TO I-1
1070 IF MID$(X$,J,1)<> "."THEN NEXT J
1080 REM J IS NOW AT DECIMAL (OR END OFNUMBER PORTION)
1090 REM DO D DIGITS EXIST TO RIGHTOF DECIMAL?
1100 IFJ+D<=I-1THEN N=J+D : GOTO 1130:REM YES
1110 N = I-1: REM NO, SO PRINT ALL DIGITS
1120 REM PRINT NUMBER PORTION AND EXPONENT PORTION
1130 PRINT LEFT$(X$,N) + MID$(X$,I)
1140 RETURN
```

Le programme ci-dessus utilise un sous-programme commençant à la ligne 1000 pour imprimer une variable réelle prédéfinie X tronquée, et non arrondie, pour la limiter à D chiffres après le point décimal. Les variables X$, I et J sont utilisées dans le sous-programme comme variables locales.

La ligne 1000 convertit la variable réelle X en la variable chaîne X$. L PS lignes 1020 et 1030 explorent ta chaîne pour voir s'il y a un E (exposant). S'il y a un E, I est placé à la position de ce E, ou s'il n'y en a pas, à la position LEN(X$) +1. Les lignes 1060 et 1070 cherchent s'il y a un point décimal dans la chaîne. S'il y a un point décimal J est placé à sa position où s'il n'y en a pas, à la position I - 1.

La ligne 1100 teste s'il existe au moins D chiffres à droite du point décimal. S'ils existent, la portion nombre de la chaîne doit être tronquée à la longueur J-I-D, qui est D positions à droite de J, la position du point décimal. La variable N est positionnée à cette longueur.
S'il y a moins de D chiffres à droite du point décimal, on peut utiliser la portion entière du nombre. La ligne 1110 positionne ta variable N à cette longueur (I - 1)

Finalement ta ligne 1130 imprime la variable X sous la forme de concaténation de deux sous-chaînes. LEFT$ (X$,N) renvoie tes chiffres significatifs de la portion nombre et MID$(X$,1) renvoie la portion exposant s'il y en a un.

STR$ peut également s'utiliser pour trouver commodément combien de positions d'impression un nombre va prendre. Par exemple

```
PRINT LEN(STR$(33333.157))
```

Si vous avez une application où un utilisateur tape une question comme

```
WHAT IS THE VOLUME OF A CYUNDER OF RADIUS 5.36 FEET
AND HEIGHT 5.1 FEET?
```

(quel est le volume d'un cylindre de rayon 5.36 pieds et de hauteur 5.1 pieds)
vous pouvez utiliser la fonction VAL pour extraire les valeurs numériques 5.36 et 5.1 de cette question. Dans le chapitre 5 se trouve une information supplémentaire concernant ces fonctions ainsi que les fonctions CHR$ et ASC.

Le programme suivant trie une liste de données chaînes et imprime la liste par ordre alphabétique. Ce programme est très semblable à celui donné précédemment pour trier une liste numérique.

```
100 DIM A$(15)
110 FORI=1 TO 15:READA$(1):NEXTI
120 F=0:I=1
130 IFA$(I)<=A$(I+1)THEN GOTO 180
140 T$=A$(I+1)
150 A$(I+1)=A$(t)
160 A$(I)=T$
170 F=1
180 I=I+1:IFI<=15 THEN GOTO 130
190 IF F=1 THEN GOTO 120
200 FOR I = 1 TO 15: PRINT A$(I) : NEXT I
220 DATA APPLE,DOC,CAT,RANDOM,COMPUTER,BASIC
230 DATA MONDAY,"***ANSWER***","FOO:"
240 DATA COMPUTER,FOO,ELP,MILWAUKEE,SEATTLE,ALBUQUERQUE
```

En savoir davantage sur le graphique couleur

Dans les deux précédents exemples, nous avons expliqué comment APPLE II peut faire aussi bien le graphique couleur que le texte. En mode graphique APPLE affiche jusque 16OO petits carrés, en une quelconque des 16 couleurs possibles selon une trame de 40 par 40. On dispose également de 4 lignes de texte en bas de l'écran. L'axe X ou axe horizontal est standard, avec 0 comme position la plus à gauche et 39 la plus à droite. L'axe Y ou vertical est non standard en ce sens qu'il est inversé: 0 est la position la plus haute et 39 la plus basse.

```
10 GR : REM INITIALIZE COLOR GRAPHICS:
   SET 40X40 TO BLACK.
   SET TEXT WINDOW TO 4 LINES AT BOTTOM
20  HOME : REM CLEAR ALL TEXT AT BOTTOM
30  COLOR = 1 : PLOT 0,0 : REM MAGENTA SQUARE AT 0,0
40  LIST30:GOSUB1000
50  COLOR = 2 : PLOT 39,0 : REM BLUE SQUARE AT X=39,Y = 0
60  HOME :LIST 50 :GOSUB 1000
70  COLOR = 12 : PLOT 0,39 : REM GREEN SQUARE AT X=0,Y=39
80  HOME :LIST 70 :GOSUB 1000
90  COLOR = 9: PLOT 39,39: REM ORANGE SQUARE AT X=39,Y=39
100 HOME :LIST 90 :GOSUB 1000
110 COLOR = 13: PLOT 19,19: REM YELLOW SQUARE AT CENTER OF SCREEN
120 HOME :LIST 110 :GOSUB 1000
130 HOME : PRINT "PLOT YOUR OWN POINTS"
140 PRINT "REMEMBER, X & Y MUST BE > =0 & < =39"
150 INPUT "ENTER X,Y:";X,Y
160 COLOR = 8 : PLOT X,Y : REM BROWN SQUARES
170 PRINT "TYPE 'CTRL C' AND PRESS RETURN TO STOP"
180 GOTO150
1000 PRINT "***HITANYKEYTO CONTINUE"*";: GETA$: RETURN
```


Après avoir tapé ce programme demandez-en le listage par l'instruction et vérifier s'il y a des erreurs de frappe. Vous pouvez demander à le sauvegarder sur une bande cassette pour utilisation future. Puis faîtes passer le programme (instructions RUN).

L'instruction GR demande à APPLE de commuter en mode graphique couleur. L'instruction COLOR définit la prochaine couleur à tracer. Cette couleur reste définie jusqu'à ce qu'elle soit modifiée par une nouvelle instruction COLOR. Par exemple la couleur tracée à la ligne 160 demeure la même indépendamment du nombre de points à tracer. La valeur de l'expression qui suit COLOR doit être de 0 à 255 sinon il peut se produire une erreur. Mais il n'y a que 16 couleurs différentes, habituellement numérotées de 0 à 15.
Modifiez le programme en retapant tes lignes 150 et 160 comme suit:

```
150 INPUT "ENTER X, Y, COLOR:"; X, Y, Z
160 COLOR =Z: PLOT X,Y
```

Maintenant faîtes passer le programme (instructions RUN et vous pourrez choisir vos propres couleurs ainsi que les points. Nous allons faire une démonstration de l'étendue de couleurs de APPLE à la suite

L'instruction PLOT X,Y trace un petit carré de couleur définie par, la dernière instruction COLOR à la position spécifiée par les expressions X et Y. N'oubliez pas que X et Y doivent un nombre de O à 39, L'instruction GET de la ligne 1OOO est semblable à une instruction INPUT. Lorsque cette instruction apparaît l'ordinateur attend que vous tapiez un seul caractère au clavier et affecte ce caractère à la variable qui suit l'instruction GET. Il n'est pas nécessaire d'enfoncer la touche RETURN. A la ligne 1OOO, GET/A$ est simplement utilisé pour arrêter le programme jusqu'à ce que vous enfonciez une touche.

N'oubliez pas: pour repasser du mode graphique couleur au mode entièrement texte, tapez TEXT puis enfoncez la touche RETURN. Le caractère APPLESOFT apparaîtra alors.
Tapez le programme suivant et faîtes-le passer (instructions RUN pour afficher la plage de couleurs de APPLE) n'oubliez pas de taper tout d'abord NEW.

```
10 GR: HOME
20 FOR T=0TO31
30 COLOR = I/2
40 VLIN 0.39 AT I
50 NEXT I
60 FOR I= 0 TO 14 STEP 2:PRINT TAB(I*2+1);I;:NEXT I
70 PRINT
80 FORI=1 TO 15 STEP 2:PRINT TAB(I*2+1);I;:NEXTI
90 PRINT: PRINT "STANDARD APPLE COLOR BARS";
```

Les rectangles de couleur sont affichés au double de leur largeur normale. Le rectangle le plus à gauche est noir, étant défini par COLOR = 0; le plus à droite est blanc, défini par COLOR = 15. Selon le réglage de couleurs de votre appareil TV, le second rectangle défini par COLOR = 1 sera magenta (pourpre-rougeâtre) et le troisième (COLOR = 2) sera bleu foncé. Réglez votre appareil TV pour obtenir ces couleurs. En Europe les valeurs des couleurs peuvent être différentes.
Dans te dernier programme on a utilisé ligne 40 une instruction de la forme VLIN Y 1, Y2 AT X. Cette instruction trace une ligne verticale partant de la coordonnée Y spécifiée par l'expression Y1 jusqu'à la coordonnée Y spécifiée par l'expression Y2, à la position horizontale spécifiée par l'expression Y. Y1, Y2 et X doivent prendre des valeurs de 0 à 39. Y2 peut être supérieur ou égal ou inférieur à Y 1. L'instruction HLIN XI, X2 AT Y est semblable à VLIN mais elle trace une ligne horizontale.

Note: APPLE trace une ligne entière aussi facilement qu'un point unique.

Graphique couleur haute résolution

Maintenant que vous êtes familiarisé avec le graphique basse résolution de APPLE, vous allez trouver facile de comprendre le graphique haute résolution. Les instructions ont un aspect semblable: habituellement elles sont formées en ajoutant simplement un H (pour haute résolution) à celles que vous connaissez déjà. Par exemple l'instruction HGR définit le mode graphique haute résolution, efface l'écran haute-résolution pour le mettre entièrement noir et laisse 4 lignes de texte en bas de l'écran. Dans ce mode, vous tracez des points sur une trame de largeur 280 positions X et de hauteur 160 positions Y.

Ceci vous permet de dessiner sur l'écran avec beaucoup plus de détails qu'avec la trame de 40 par 40 du graphique basse résolution. En tapant TEXT vous revenez au mode normal texte. En plus de l'écran HGR il existe également un deuxième écran haute résolution que vous pouvez utiliser si votre APPLE contient au moins 24 K octets de mémoire. Le mode graphique haute résolution pour cette seconde 'page de mémoire' s'obtient par l'instruction

HGR2

Cette instruction efface la totalité de l'écran pour le rendre noir, ce qui vous assure une surface de tracé de largeur 280 positions X et de hauteur 192 positions Y, sans texte en bas. A nouveau il faut taper TEXT pour voir votre programme.

Vraiment merveilleux? En effet; mais pour cette nouvelle possibilité vous devez faire des sacrifices il y a moins de couleurs. La couleur du graphique haute résolution est définie par une instruction de la forme

```
HCOLOR = N
```

où N est un nombre de 0 (noir) à 7 (blanc). Voir au chapitre 8 une liste complète des couleurs disponibles. Du fait du mode de construction des appareils de télévision couleur, ces couleurs varient d'un appareil à l'autre et d'un trait à l'autre du tracé.

Enfin, il existe une instruction facile pour tous les tracés en graphique haute résolution. Pour l'avoir en action, tapez

```
HCOLOR = 3
HGR
HPLOT 130,100
```

La dernière instruction trace un point haute résolution dans la couleur que vous avez définie avec HCOLOR (blanc) au point X = 130, Y = 100. Comme en graphique basse résolution, X = 0 est sur le bord gauche de l'écran, en allant en croissant vers la droite; Y = 0 est en haut de l'écran, en allant en croissant vers le bas. La valeur maxima de X est 279: la valeur maxima de Y est 191 (mais dans le mode mixte graphique plus texte de HGR, les valeurs Y ne sont visibles vers le bas que jusqu'e Y = 159).

Maintenant tapez

```
HPLOT 20.15 TO 145.80
```

Comme par magie, une droite blanche est tirée du point X = 20, Y = 15 au point X = 145, Y = 80.

HPLOT peut tirer des droites entre deux points quelconques de l'écran - horizontalement verticalement ou selon un angle quelconque. Désirez-vous relier une autre droite à l'extrémité de la précédente? Tapez

```
HPLOT T0 12,80
```

Cette forme d'instruction prend son point de départ au dernier point précédemment tracé et prend sa couleur en ce point également (même si vous avez émis une nouvelle instruction HCOLOR depuis que ce point a été tracé). Vous pouvez même 'chaîner' ces instructions en une seule instruction. Essayez ceci

```
HPLOT 0.0 TO 279,0 TO 279,159 TO 0,  159 TO 0,0
```

Vous devriez maintenant avoir une bordure blanche tout autour des quatre cotés de l'écran.

Voici un programme qui dessine de jolis dessins moirés sur votre écran.

```
80 HOME : REM CLEAR THE TEXT AREA
100 VTAB24:REM MOVE CURSOR TO BOTTOM LINE
120 HGR : REM SET HIGH-RESOLUTION GRAPHICS MODE
140 A=RND(1)*279:REMPICKANXFOR"CENTER"
160 B=RND(1)*159:REMPICKAYFOR"CENTER"
180 1% = RND(1) * 4)+2: REM PICK A STEP SIZE
200 HTAB 15 : PRINT "STEPPING BY"; 1 %;
220 FOR X=0TO 278 STEP 1%: REM STEP THRUX VALUES
240 FORS=0TO1:REM2LINES.FROMXANDX+1
260 HCOLOR=3*S:REMFIRSTLINEBLACK,NEXTWHITE
280 REM DRAW UNE THROUGH"CENTER-TO OPPOSITE SIDE
300 HPLOTX+S,0TOA,BTO279-X-S,159 320 NEXTS, X
340 FOR Y =0 TOI 58 STEP 1%: REM STEP THRU Y VALUES
360 FOR S = 0 TOI: REM 2 UNES, FROM Y AND Y+)
380 HCOLOR-3*S:REMFIRSTLINEBLACK,NEXTWHITE
400 REM DRAW LINE THROUGH 'CENTER' TO OPPOSITE SIDE
420 HPLOT279,Y+STOA,BTO0,159-Y-S
440 NEXTS, Y
460 FOR PAUSE = 1 TO 1500 : NEXT PAUSE : REM DELAY
480 GOTO120:REMDRAWANEWPATTERN
```

C'est un programme plutôt long; tapez-le soigneusement et demandez à le voir sur l'écran (instructions LIST par portion)(UST 0,320 par exemple) pour vérifier votre frappe. Nous avons ajouté une ligne blanche entre certaines lignes pour en faciliter la lecture. Votre listage ne fera pas apparaître ces blancs. Lorsque vous êtes certain qu'il est correct, faîtes passer le programme (instructions RUN)

VTAB et HTAB sont des instructions de déplacement du curseur utilisées pour imprimer un caractère à une position prédéterminée sur l'écran en mode texte. VTAB 1 place le curseur sur la ligne du haut; VTAB 24 place sur la ligne du bas. HTAB 1 le place dans la position ta plus à gauche sur la ligne en cours; HTAB 40 le place à la position la plus à droite. Dans une instruction PRINT comme celte de la ligne 200 vous pouvez avoir besoin d'un point et virgule à la fin pour éviter une 'alimentation de ligne' à la suite qui viennent déplacer votre message.

La fonction RND(N) où N est tout nombre positif fournit un nombre aléatoire de 0 à .999999999 (voir chapitre 10 pour la discussion complète de RND). La ligne 181 affecte donc à la variable entière 1% un nombre aléatoire de 2 à 5 (un nombre est toujours arrondi par valeur inférieure lorsqu'il est converti en un entier. La dimension STEP dans la boucle FOR . . . NEXT n'est pas obligatoirement un entier mais il peut être plus facile de prédire les résultats pour un STEP entier.

Comme vous le voyez dans les lignes 320 et 440, une instruction peut prévoir le NEXT pour plus d'une instruction FOR. Faîtes attention de donner la liste des variables NEXT dans l'ordre correct pourtant, pour éviter des boucles croisées.

La ligne 460 est simplement une 'boucle d'attente' qui vous donne un moment pour admirer un dessin avant que le suivant ne commence. Chaque fois que la ligne 480 renvoie l'ordinateur à l'instruction HGR de la ligne 120, HGR efface l'écran pour le dessin suivant. Pour revenir au programme arrêtez le dessin en tapant

```
Ctrl C
```

puis en tapant

```
TEXT
```

Avez-vous des idées sur la façon de modifier le programme? Après avoir sauvegardé cette version sur votre enregistreur de cassette ou sur disquette, essayez de modifier au hasard la valeur de HCOLOR. Essayez de tirer tout d'abord des traits blancs puis des traits noirs ou uniquement des traits blancs.

JOYEUSE PROGRAMMATION! ?

Chapitre 2

Définitions

Définitions et abréviations syntaxiques

(pour une liste alphabétique de ces définitions voir appendice N)

Les définitions suivantes utilisent des métasymboles comme { et \ --, caractères utilisés pour indiquer sans ambiguïté des structures ou des relations dans APPLESOFT. Les métasymboles ne font pas partie de APPLESOFT En plus des métasymboles vrais, le symbole spécial := indique le début d'une définition complète ou partielle du terme qui se trouve à la gauche de :=

| := métasymbole utilisé pour séparer des alternatives
(note: un item peut également se définir séparément pour chaque alternative)
[ ] := métasymboles utilisés pour isoler un élément optionnel
{ } := métasymboles utilisés pour isoler un élément qui peut se répéter
\ := métasymbole utilisé pour isoler un élément dont il faut utiliser la valeur la valeur de X s'écrit \X\
~ := métasymbole qui indique un espace nécessaire

métasymbole
:= ||[|]|{|}|\|
Lettres minuscules
:=a | b | c | d | e | f | g | h | i | j | k | l | m | n | o | p | q | r | s | t | u | v | w | x | y | z

métasymbole
:= lettres minuscules

chiffres
:= 1|2|3|4|5|6|7|8|9|0

métanom
:= {métasymbole} [chiffre]

métasymbole
:= un chiffre unique concaténé à un métanom

symbole spécial utilisé par APPLESOFT II
:= spécial

Spécial
:= ! | # | $ | % | & | ' | (|) * | : | = | - @ | + | ; | ? | / | > . | < | , |] | ^ | "

Les caractères de contrôle (caractères qui sont tapés tout en maintenant enfoncée la touche CTRL) et le caractère nul sont également des caractères spéciaux. APPLESOFT utilise le crochet à droite (|) uniquement pour le caractère ; dans ce document il est utilisé comme métasymbole.

Lettre
:= A|B|C|D|E|F|G|H|I|J|K|L|M|N|O|P|Q|R|S|T|U|V|W|X|Y|Z
Caractère
:= lettre | chiffre | spécial
Caractères alphanumériques
:= lettre I chiffre
nom
:= lettre [{lettre I chiffre }]

Un nom peut avoir une longueur allant jusque 238 caractères. Lorsqu'il faut distinguer un nom d'un autre APPLESOFT ignore tout caractère alphanumérique situé après les deux premiers APPLESOFT ne distingue donc pas entre les noms GOOD4LITTLE et GOLDRUSH. Toutefois même la partie ignorée d'un nom ne doit pas contenir un spécial, un guillemet ou un des mots réservés de APPLESOFT (voir en appendice A la liste de ces mots réservés et des commentaires sur les exceptions à cette règle).

Entier:

:= [+ | -] { chiffres }

Les entiers doivent être entre ?32767 et 32767.
Pour convertir des nombres non entiers en entiers, on peut habituellement considérer que APPLESOFT tronque le non?entier pour obtenir l'entier inférieur le plus proche. Toutefois ce n'est pas tout à fait vrai dans la mesure où le non?entier s'approche de l'entier supérieur le plus proche.
Par exemple:

```
A%=123.999 999 959 999      B%=123.999 999 96
PRINT A%                    PRINT A%
      123                   124

C%=12345.999 995 999        D%=12345.999 996
PRINT C%                    PRINT D%
12345                       12346
```

(les lignes blanches ont été ajoutées pour faciliter la lecture)
Un entier élément d'un tableau occupe deux octets (16 bits) en mémoire. Nom d'une variable entière

:=nom%

Une variable réelle peut être enregistrée sous forme de variable entière, mais APPLESOFT convertit d'abord la variable réelle en une variable entière. Nombre réel:
(en remplaçant digit par chiffre)

:= [+ | _] { digit } [. { digit }] [E [+ | ?] digit [digit]]
:= [+ | _] [{ digit }] . [{ digit }] [E [+ | ?] digit [digit]]

La lettre E, utilisée dans la notation du nombre réel (une forme de 'notation scientifique') signifie 'exposant'. C'est l'abréviation de *10^. Dix est élevé à la puissance du nombre qui se trouve à la droite de E et le nombre qui se trouve à la gauche de E est multiplié par le résultat.

En APPLESOFT les nombres réels doivent aller de ?1E38 à 1E38, sinon vous risquez le message OVERFLOW ERROR?
En utilisant l'addition et la soustraction il est possible que vous obteniez des nombres de l'importance de 1.7E38 sans recevoir ce message. Un nombre réel dont la valeur absolue est inférieure à environ 2.9388E?39 sera converti par APPLESOFT en zéro. APPLESOFT reconnaît les nombres suivants comme des nombres réels lorsqu'ils se présentent par eux?mêmes et les évalue pour zéro.

```
.       +.      -.      .E      +.E     -.E
.E+     .E-     +.E-    +.E+    -.E+    -.E-
```

Par conséquent l'élément de tableau M(.) est le même que M(0).
En plus des nombres réels abrégés de la liste ci?dessus, les suivants sont reconnus comme nombre réels et évalués pour zéro lorsqu'ils sont utilisés comme réponse numérique à une instruction INPUT ou comme éléments numériques de l'instruction DATA.

```
+       -       E       +E      -E      espace
E+      E-      +E+     +E-     -E+     -E-
```


L'instruction GET évalue pour zéro tous les nombres réels à simple caractère des listes ci?dessus. Lorsqu'on imprime un nombre réel, indiquera au plus 9 chiffres (voir les sections ci?dessous) non compris l'exposant (s'il y en a un).Tous les chiffres qui viennent à la suite sont arrondis. A la gauche du point décimal, tous les zéros qui précèdent le chiffre le plus à gauche qui ne soit pas un zéro ne sont pas imprimés. A la droite du point décimal tous les zéros qui suivent le chiffre le plus à droite qui ne soit pas un zéro ne sont pas imprimés. S'il n'y a pas à droite du point décimal de chiffres quine soient pas des zéros, le point décimal n'est pas imprimé.

Le résultat de l'arrondi peut être curieux

```
PRINT 99 999 999.9
99 999 999.9

PRINT 99 999 999.90
100 000 000

PRINT 11.111 111 450 00
11.111 111 5

PRINT 11.111 111 451 9
11.111 111 4
```

(les espaces blancs sont ajoutés pour faciliter la lecture)

Si la valeur absolue d'un nombre réel est supérieur ou égal à.0 1 et inférieur à 999 999 999.2, le nombre réel est imprimé en notation à point décimal fixe. C'est?à?dire que l'exposant n'est pas affiché. Sur l'étendue

.0 100 000 000 5 à .0 999 999 999,

les nombres réels sont imprimés avec dix chiffres maximum, y compris le zéro qui se trouve immédiatement à la droite du point décimal. C'est la seule exception à la limite de 9 chiffres imprimés non compris l'exposant.

Si vous essayez d'utiliser un nombre de plus de 38 chiffres comme
211.11111111111111111111111111111111111111
alors le message
?OVERFLOW ERROR
s'imprime, même si ce nombre se trouve certainement entre ?1 E38 et 1 E38. Ceci est vrai même si la plus grande partie des chiffres sont des zéros de queue, comme dans
211.00000000000000000000000000000000000000
Les zéros de tête toutefois sont ignorés. Le premier chiffre est un et si le second chiffre est inférieur ou égal à six, des nombres contenant 39 chiffres peuvent s'utiliser sans donner lieu à message d'erreur.

Un nombre réel occupe 5 octets (40 bits en mémoire)

Nom d'une variable réelle

:=nom

Variable arithmétique

:= avar

avar

:= nom | nom%

Toutes les variables simples occupent 7 octets en mémoire, 2 octets pour le nom et 5 octets pour la valeur réelle ou entière.

Délimiteur
:= ~|(|)|=|-|+|^|>|<|||*|,|;|:
un nom n'a pas à être séparé d'un mot réservé qu'il précède ou qu'il suit par l'un de ces délimiteurs.

Opérateur arithmétique
:= aop

aop
:=+ | * | | | ^

Opérateur logique arithmétique
:= alop

alop
:= AND|OR|=|>|<|<>|><|>=|=>|<=|=<
C'est volontairement que NOT n'est pas compris ici.
Opérateur
:= op

op
:= aop I alop

Expression arithmétique
:= aexpr
aexpr
:= avar | nombre réel | nombre entier
:= (aexpr)

Si des parenthèses sont emboîtées sur plus de 36 niveaux de profondeur, il apparaît le message OUT OF MEMORY ERROR?
:= [+ | - | NOT] aexpr

L'opérateur unaire NOT apparaît ici avec les opérateurs unaires + et ?
:= aexpr op aexpr

Indice
:= (aexpr [{ ,aexpr }])

Le nombre maximum de dimensions est 89, bien qu'en pratique il soit limité par la taille de la mémoire disponible. aexpr doit être positif et, pour l'utiliser, est converti en un entier.

avar
:= indice de avar

aexpr
:= indice de avar

Expression littérale
:= [{ caractère }]

Chaîne
:= " [{ caractère }] "

une chaîne occupe un octet (8 bits) pour sa longueur, 2 octets pour son pointeur d'emplacement et un octet pour chaque caractère de la chaîne.
:= [{ caractère }] return
Cette forme de chaîne ne peut apparaître qu'à la fin d'une ligne
Chaîne nulle
:=""

Nom d'une variable chaîne
:= nom$
Variable chaîne
:= svar

svar
:= nom$ | nom$ indice

Le pointeur d'emplacement et le nom de la variable occupent chacun deux octets en mémoire. La longueur et chaque caractère de la chaîne occupent un octet.
Opérateur de chaîne
:= sop

sop
:=+
Expression chaîne
:= sexpr

sexpr
:= svar | chaîne
:= sexpr sop sexpr

opérateur logique de chaîne
:= slop

slop
:= |=|>|<|<>|><|>=|=>|<=|=<

aexpr
:= sexpr slop sexpr
Variable
:=var

var
:= avar | svar

Expression
:= expr
expr
:= aexpr | sexpr

Caractère souffleur
:=]

Le crochet à droite est affiché lorsque APPLESOFT est prêt à accepter une autre instruction.

reset
:= enfoncer la touche marquée "RESET"

esc
:= enfoncer la touche marquée "ESC"

return
:= enfoncez la touche marquée "RETURN"

ctrl
:= maintenir enfoncée la touche marquée "CTRL" tout en enfonçant la touche dont le nom suit

numéro de la ligne
:= Linenum

Linenum
:= {chiffre }

Les numéros de lignes doivent aller de 0 à 63999 sinon il en résulte un message SYNTAX ERROR?

Ligne
:= Linenum [(instruction : }] instruction return
Une'ligne peut avoir jusque 239 caractères. Ceci comprend tous les espaces blancs tapés par l'utilisateur, mais ne comprend pas les espaces blancs ajoutés par APPLESOFT pour mettre la ligne au format

Règles d'évaluation des expressions

Les opérateurs sont listés verticalement dans l'ordre d'exécution, depuis la plus forte priorité (parenthèses) jusqu'à la plus faible priorité (OR). Les opérateurs listés sur la même ligne ont la même priorité. A priorité égale dans une expression les opérations s'exécutent en allant de gauche à droite.

( )

+ - NOT opérateurs unaires

^

*/
+-
>< >= <= => =< <> >< =
AND
OR

Conversion de types

Lorsqu'un nombre entier et un nombre réel sont tous les deux présents dans un calcul, tous les nombres sont convertis en nombres réels avant que le calcul ne soit effectué. Les résultats sont convertis en le type arithmétique (entier ou réel) de la variable finale à laquelle ils sont affectés. Les fonctions définies sur un type arithmétique donné convertissent les arguments d'un autre type en le type pour lequel elles sont définies. Les types chaînes et arithmétiques ne peuvent pas être mélangés. Chaque type doit être converti dans l'autre type au moyen de fonctions prévues dans ce but.

Modes d'exécution

Imm Certaines instructions peuvent être utilisées en mode exécution immédiate (imm) en APPLESOFT. En mode exécution immédiate une instruction doit être tapée sans numéro de ligne. Dès que l'on enfonce la touche RETURN l'instruction est immédiatement exécutée.

Def Les instructions utilisées en mode exécution différée (clef) doivent apparaître sur une ligne qui commence par un numéro de ligne. Lorsqu'on enfonce la touche RETURN, APPLESOFT enregistre la ligne numérotée en mémoire pour utilisation ultérieure. Les instructions en mode à exécution différée ne sont exécutées que lors du RUN

CHAPITRE 3
Instructions du système et instructions utilitaires

LOAD et SAVE

imm dif sont les abréviations de immédiat et différé &

LOAD imm & dif
SAVE imm & if

Ces instructions chargent LOAD un programme à partir d'une bande sur cassette et sauvegardent SAVE un programme sur cassette, respectivement. II n'y a pas de message indicateur ou autre signal émis par ces instructions; l'utilisateur doit avoir son lecteur enregistreur de cassettes fonctionnant dans le mode correct (lecture ou enregistrement) lorsque l'instruction est exécutée. Les instructions LOAD et SAVE ne vérifient pas que le lecteur de cassette est en mode correct ou même qu'il en existe un. Les deux instructions émettent un bip pour signaler le début et la fin des enregistrements ou d'une lecture.

L'exécution du programme se poursuit après une opération SAVE mais une opération LOAD efface le programme en cours lorsqu'elle commence à lire une nouvelle information à partir d'une bande sur cassette.
Seule l'instruction reset peut interrompre une instruction LOAD ou SAVE.

Si le mot réservé LOAD ou SAVE est utilisé comme premiers caractères d'un nombre de variable,

l'instruction à mot réservé peut être exécutée avant que tout message ?SYNTAX ERROR ne soit donné. L'instruction

SAVERING = 5

fait que APPLESOFT s'efforce de sauvegarder le programme en cours. Vous pouvez attendre après le second bip (et le message ?SYNTAX ERROR) ou enfoncez la touche reset.

L'instruction LOADTOJOY=47 suspend le fonctionnement du système, tandis que APPLESOFT efface le programme en cours et attend indéfiniment après un programme en provenance du lecteur de cassette. Ce n'est qu'en enfonçant la touche dif que vous pouvez reprendre le contrôle de votre ordinateur.

NEW

NEW imm dif
Pas de paramètres. Efface le programme en cours et toutes les variables.

RUN

RUN imm & dif
RUN [linenum]

Cette instruction efface toutes les variables, les pointeurs et les registres et commence l'exécution à la ligne dont le numéro est in diqué par linenum. Si le numéro de ligne linenum n'est pas indiqué, l'exécution commence à la ligne de plus faible numéro du programme, ou bien cette instruction renvoie le contrôle à l'utilisateur s'il n'y a pas de programme en mémoire.

En mode exécution différée, si le numéro de ligne linenum est donné mais si cette ligne n'existe pas dans le programme ou si linenum est négatif, il apparait le message
?UNDEF'D STATEMENT ERROR
Si le numéro de ligne linenum est supérieur à 63999, il apparait le message
?SYNTAX ERROR.
On ne vous indique pas à quelle ligne l'erreur s'est produite.

En mode d'exécution immédiate d'un autre côté ces deux messages deviennent

?UNDEF'D STATEMENT ERROR IN XXXX

et

?SYNTAX ERROR IN XXXX

où XXXX peut être différents numéros de lignes, habituellement supérieurs à 65000
Si l'instruction RUN est utilisée dans un programme d'exécution immédiate, toute ou portion suivante du programme d'exécution immédiate n'est pas exécutée.

STOP, END, ctrl C, reset et CONT

STOP : imm & dif
END : imm & dif
ctrl C : imm
reset : imm
CONT : imm & dif

STOP
END
ctrl C
reset
CONT

STOP amène un programme à cesser son exécution et renvoie le contrôle de l'ordinateur à l'utilisateur. Cette instruction imprime le message BREAK IN linenum où linenum est le numéro de ligne de l'instruction qui a exécuté le STOP.

END amène un programme à cesser son exécution et renvoie le contrôle à l'utilisateur. Aucun message n'est imprimé.

ctrl C a un effet équivalent à l'insertion d'une instruction STOP placée immédiatement après l'instruction en cours d'exécution. ctrl peut s'utiliser pour interrompre un listage (instruction LIST). Elle peut également s'utiliser pour interrompre une instruction INPUT mais seulement si c'est le premier caractère entré. L'instruction INPUT n'est pas interrompue avant que la touche RETURN ne soit enfoncée.

reset arrête immédiatement tout programme ou instruction APPLESOFT inconditionnellement. Le programme n'est pas perdu mais certains pointeurs et certains registres sont effacés. Cette instruction vous laisse dans le langage moniteur du système comme indiqué par le caractère de moniteur ("). Pour revenir à APPLESOFT sans détruire le programme enregistré en cours, tapez ctrl C return.

Si l'exécution du programme a été arrêtée par STOP, END ctrl C, l'instruction CONT fait que l'exécution reprend à l'instruction suivante et non pas au numéro de ligne suivant. Rien n'est effacé. Sil n'y a pas de programme arrêté, alors l'instruction CONT n'a pas d'effet. Après reset ctrl C return le programme peut ne pas poursuivre son exécution correctement par l'instruction CONT car certains pointeurs et supports du programme peuvent avoir été effacés.
Si une instruction INPUT est arrêtée par ctrl C, un essai de continuer l'exécution du programme par l'instruction CONT se traduit par un message ?SYNTAX ERROR IN linenum, où linenum est le numéro de la ligne qui contient l'instruction INPUT.

L'exécution de l'instruction CONT va se traduire par le message
?CAN'T CONTINUE ERROR
si, après l'arrêt de l'exécution du programme, l'utilisateur a) modifie ou efface une ligne quelconque du programme b) essaie toute opération qui se traduit par un message d'erreurs Par contre les variables du programme peuvent être changées à l'aide d'instructions d'exécution immédiate, pour autant que l'on n'a pas subi de message d'erreur.

Si l'on utilise DEL dans une instruction à exécution différée, les lignes spécifiées sont effacées et l'exécution du programme s'arrête. Un essai d'utiliser CONT dans ces circonstances amène le message:

?CAN'T CONTINUE ERROR

Si l'on utilise dans une instruction à exécution différée, l'exécution du programme s'arrête à cette instruction mais le contrôle de l'ordinateur n'estpas renvoyé à l'utilisateur. L'utilisateur peut reprendre le contrôle de l'ordinateur en émettant une instruction ctrl C mais un essai de poursuivre l'exécution du programme en introduisant CONT dans l'instruction suivante n'abouti qu'à dessaisir le contrôle et arrêter a nouveau le programme:

TRACE et NOTRACE

TRACE: imm & dif
NOTRACE: imm & dif

TRACE positionne un mode de mise au point qui affiche le numéro de ligne de chaque instruction lorsqu'elle est exécutée. Si le programme imprime également sur l'écran, TRACE peut y apparaître de façon inattendue ou être surchargé. NOTRACE supprime le mode de mise au point TRACE
Une fois positionné, TRACE n'est pas supprimé par RUN, CLEAR, NEW, DEL ou reset; reset ctrl B supprime TRACE (et élimine tout programme enregistré en mémoire)

PEEK

PEEK: imm & dif
PEEK (aexpr)

donne le contenu, en décimal, de l'octet qui se trouve à l'adresse \aexpr. \ L'appendice J contient des exemples de la façon d'utiliser PEEK.

POKE

POKE: imm & dif
POKE aexpr1, aexpr2

POKE enregistre en mémoire un octet, l'équivalent binaire de la valeur décimale \aexpr2 \ à l'emplacement dont l'adresse est donnée par aexpr 1. L'étendue de \aexpr2\ doit être de 0 à 255; celle de \,aexpr \ doit être de 65535 à 65535. Les nombres réels sont convertis en nombres entiers avant exécution.
Des valeurs qui se trouvent en dehors de l'étendue amènent l'impression du message
? ILLEGAL QUANTITY ERROR
\aexpr2\ ne sera correctement enregistré ue si le matériel informatique correct pour la recevoir (mémoire ou dispositif de sortie convenable est présent à l'adresse spécifiée par \aexpr1. \ \ aexpr2 \ ne sera pas correctement enregistré à des adresses qui ne peuvent pas la recevoir, comme les mémoires ROM du moniteur ou des canaux utilisés entrée/sortie.
En général ceci signifie que \aexpr1 \ sera dans l'étendue 0 à max, où max est déterminé par la taille de la mémoire de l'ordinateur. Par exemple sur un APPLE II de 16 K de mémoire, max est 16384. Si APPLE II a 32 K de mémoire, max est 32768; et si APPLE II a 48 K de mémoire max est 49152.
De nombreux emplacements de mémoire contiennent une information nécessaire au fonctionnement du système ordinateur. Une instruction POKE à ces emplacements peut altérer l'exploitation du système ou de votre programme ou introduire des perturbations dans APPLESOFT.
WAIT: imm & dif

WAIT

WAIT aexprl, aexpr2 f, aexpr3]

permet à l'utilisateur d'insérer une pose conditionnelle dans un programme. Seul reset peut interrompre une instruction WAIT.
\aexpr 1 \ est l'adresse de l'emplacement de mémoire; elle doit être dans l'étendue -65535 à 65535 pour éviter le message ?ILLEGAL QUANTITY ERROR. En pratique, \aexprl \ se limite habituellement à l'étendue des adresses correspondant aux emplacements où il existe des dispositifs à mémoire valide, de 0 à la valeur maxima de HIMEM dans votre ordinateur. Voir HIMEM et POKE pour plus de détails. On peut utiliser des adresses équivalentes positives et négatives.

\aexpr2 \ et \aexpr3 \ doivent être dans l'étendue 0 à 255, en décimal. Lorsque l'instruction WAIT s'exécute, ces valeurs sont converties en nombres binaires dans l'étendue 0 à 111111111.

Si \aexprl \ et \aexpr2 \ sont seules spécifiées, chacun des huit bits du contenu binaire de l'emplacement aexprl est additionné logiquement avec le bit correspondant de l'équivalent binaire de \aexpr2. \ Pour chaque bit, ceci donne un 0 à moins que les deux bits correspondants ne soient de valeur 1. Si le résultat de ce processus est huit zéros, alors le test est répété. Si l'un des résultats est non-zéro (ce qui signifie qu'au moins 1 bit de valeur 1 dans \aexpr2\ a correspondu à un bit correspondant de valeur 1, à l'emplacement \aexpr 1, \ l'instruction WAIT est terminée et le programme APPLESOFT reprend son exécution à l'instruction suivante.
WAIT \aexpl \, 7 fait que le programme s'arrête jusqu'à ce qu'au moins 1 des trois bits les plus à droite à l'emplacement \aexpr 1 \ ait la valeur 1.

WAIT aexpr1,0

fait que le programme s'arrête définitivement.

Si les trois paramètres sont spécifiés, alors l'instruction WAIT agit comme suit: premièrement chaque bit du contenu binaire de l'emplacement \aexprl \ est comparé, dans une opération logique OU exclusive (instruction XOR) au bit correspondant de l'équivalent binaire de \aexpr3. \ Un bit de valeur 1 dans \ aexpr3 \ donne un résultat qui est l'inverse du bit correspondant de l'emplacement aexpr 1 (un 1 devient un zéro, un zéro devient un 1). Un bit de valeur zéro dans \ aexpr 3 \ donne un résultat qui est le même que le bit correspondant à l'emplacement aexpr 1. Si \aexpr 3 \ est justement zéro, la portion XOR n'a pas d'effet.

Deuxièmement, chaque résultat est additionné logiquement (AND) avec le bit correspondant de l'équivalent binaire de aexpr 2. Si le résultat final est huit zéros, le test est répété. S'il y a un résult non zéro, l'instruction WAIT est terminée et l'exécution du programme APPLESOFT se continue à l'instruction suivante.

Une autre façon de regarder l'instruction WAIT: son objet est de tester les contenus des emplacements \aexpr 1 \ pour voir si l'un quelconque de certains bits a la valeur un ou l'un quelconque de certains autres bits a la valeur 0. Chacun des huit bits de l'équivalent binaire de \aexpr 2 \ indique si vous êtes ou non intéressé au bit correspondant à l'emplacement \aexpr 1 \: la valeur 1 signifie que vous êtes intéressé la valeur 0 signifie que vous devez ignorer ce bit. Chacun des huit bits de l'équivalent binaire de \ aexpr 3 \ indique quel état vous attendez (WAIT) pour le bit correspondant à l'emplacement \ aexpr 1 \: la valeur 1 signifie que le bit doit avoir la valeur 0, la valeur 0 signifie que le bit doit avoir la valeur 1. Si l'un quelconque des bits pour lesquels vous avez indiqué être intéressé (par un 1 dans le bit correspondant de \ aexpr 2 \ )correspond à l'état que vous avez spécifié pour ce bit (au moyen du bit correspondant de \aexpr 3 \) l'instruction WAIT est terminée. Si \aexpr 3 \ est omis, sa valeur par défaut est 0.
Par exemple:

WAIT aexpr t, 255,0 — signifie une pause jusqu'à ce qu'au moins un de huit bits qui se trouvent à l'emplacement aexpr 1 ait la valeur 1.

WAIT aexprl, 255 — identique à l'exemple ci-dessus, en exploitation,

WAIT aexpr1,255,255 — signifie une pause jusqu'à ce qu'au moins un des huit bits qui se trouvent à l'emplacement aexpr 1 ait la valeur 0

WAIT aexpr1,1,1 — signifie une pause jusqu'à ce que le bit le plus à droite qui se trouve à l'emplacement aexprl ait la valeur 0, indépendamment de la valeur des autres bits.

WAIT aexpr 1,3,2 — signifie une pause jusqu'à ce que soit le bit le plus à droite à l'emplacement aexpr l ait la valeur 1 soit le bit à gauche du à droite ait la valeur 0, soit que les deux conditions existent.

Le programme suivant s'arrête jusqu'à ce que vous tapiez un caractère quelconque dont le code ASCII (voir appendice K) soit pair

```
100 POKE -16368, 0 : REM KEYBOARD STROBE (HIGH BIT)
105 REM PAUSE UNTIL KEYBOARD STROBE IS SET BY ANY KEY.
110 WAIT -16384, 128: REM WAIT UNTIL HIGH BIT IS ONE.
115 REM PAUSE SOME MORE UNTIL KEY STRUCK IS EVEN.
120 WAIT -16384, 1, 1 : REM WAIT UNTIL LOW BIT IS ZERO.
130 PRINT "EVEN"
140 GOTO 100
```

CALL

CALL: imm & dif
CALL aexpr

amène l'exécution du sous-programme en langage machine qui se trouve l'emplacement mémoire dont l'adresse décimale est spécifiée par \aexpr\.

\ aexpr \ doit être dans l'étendue -65535 à 65535, sinon il y a affichage du message ?ILLEGAL QUANTITY ERROR
En pratique \aexpr\ se limite habituellement à l'étendue des adresses pour lesquelles il existe des dispositifs à mémoire valide, de zéro à la valeur maxima de HIMEM sur votre ordinateur. Voir HIMEM et POKE pour tous détails.

On peut utiliser de façon interchangeable les adresses équivalentes positives et négatives. Par exemple "CALL-936" et "CALL 64600" sont identiques.

L'appendice J contient des exemples de l'utilisation de CALL

HIMEM:

HIMEM: imm& dif
HIMEM: aexpr

Positionne l'adresse de l'emplacement en mémoire le plus élevé disponible pour un programme BASIC, y compris les variables. S'utilise pour protéger la zone de mémoire située au-dessus de cette adresse, pour les données, les graphiques ou les routines en langage machine.

\aexpr \ doit être dans l'étendue -65535 à 65535, les deux limites comprises, pour éviter le message
?ILLEGAL QUANTITY ERROR

Toutefois, les programmes ne peuvent pas s'exécuter de façon fiable à moins qu'il n'y ait une mémoire physique appropriée aux emplacements spécifiés par toutes les adresses jusque et y compris \ aexpr \.

En général la valeur maxima de aexpr est déterminée par l'importance de la mémoire de l'ordinateur. Par exemple sur un APPLE II à 16 K de mémoire, \ aexpr \ serait 16384 ou moins. Si APPLE II a 32 K de mémoire aexpr pourrait aller jusque 32768; et si APPLE II a 48 K de mémoire, \aexpr \ pourrait aller jusque 49152.

Normalement, APPLESOFT positionne automatiquement HIMEM à l'adresse de mémoire la plus élevée disponible dans l'ordinateur de l'utilisateur si APPLESOFT est appelé en premier lieu.
La valeur actuelle de HIMEM est enregistrée aux emplacements de mémoire 116 et 115 (décimal). Pour voir la valeur actuelle de HIMEM, tapez

```
PRINT PEEK (116)*256 + PEEK (115)
```

Si HIMEM positionne une adresse de mémoire la plus élevée qui soit inférieure à celle positionnée par LOMEM, ou qui ne laisse pas suffisamment de mémoire disponible pour l'exécution du programme, on obtient le message

```
?OUT OF MEMORY ERROR
```

\aexpr\ peut être dans l'étendue croissante 0 à 65535, ou dans l'étendue équivalente croissante -65535 à -1. On peut utiliser de façon interchangeable les valeurs équivalentes positives et négatives.
HIMEM n'est pas remis à l'état initial par CLEAR, RUN, NEW, DEL, par le fait de modifier ou d'ajouter une ligne de programme, ou par reset. HIMEM est remis à l'état initial par reset ctrl B return, qui efface également tout programme enregistré.

LOMEM:

LOMEM: imm& dif
LOMEM: aexpr

positionne l'adresse de l'emplacement de mémoire le plus faible disponible pour un programme BASIC. C'est habituellement l'adresse de l'emplacement de mémoire de démarrage pour la première variable BASIC. Normalement APPLESOFT positionne automatiquement LOMEM à la fin du programme en cours, avant d'exécuter le programme. Ces instructions permettent de protéger les variables provenant des graphiques haute résolution dans les ordinateurs à grande capacité de mémoire.

\ aexpr \ doit être dans l'étendue -65535 à 65535, les deux bornes comprises, pour éviter le message ?
ILLEGAL QUANTITY ERROR.
Toutefois si LOMEM est positionné au-dessus de la valeur en cours de HIMEN le message

```
? OUT OF MEMORY ERROR
```

est affiché. Ceci signifie que aexpr doit être inférieur à la valeur maxima qui peut être positionnée par HIMEM (voir HIMEM pour une discussion de sa valeur maxima)
Si LOMEM est positionné à une valeur inférieure à l'adresse de l'emplacement en mémoire le plus élevé occupé par le système d'exploitation en cours (plus tout programme enregistré actuellement), le message

```
? OUT OF MEMORY ERROR
```

est à nouveau affiché. Ceci impose une limite inférieure absolue sur \aexpr\ d'environ 2051 pour APPLESOFT en ROM
LOMEM est remis à l'état initial par NEW, DEL et en ajoutant ou modifiant une ligne de programme.
LOMEM est remis à l'état initial par reset ctrl B qui efface également tout programme enregistré en mémoire. II n'est pas remis à l'état initial par RUN, reset ctrl C return reset 0G return.

La valeur actuelle de LOMEM est enregistrée aux emplacement LOMEM de mémoire 106 et 105 (décimal). Pour voir la valeur actuelle de LOMEM, tapez

```
PRINT PEEK (106)*256 + PEEK (105)
```

Une fois positionné, à moins d'être tout d'abord remis à l'état initial par l'une des instructions ci-dessus, LOMEM peut être positionné à une nouvelle valeur uniquement si la nouvelle valeur est supérieure (en mémoire) à l'ancienne. Un essai de positionner LOMEM à une valeur inférieure à la valeur actuellement en effet donne le message

```
? OUT OF MEMORY ERROR
```

Le fait de changer LOMEM pendant le cours d'un programme peut risquer de rendre indisponible certains supports ou portions du programme, de sorte que ce programme ne continuera pas à s'exécuter correctement.
On peut utiliser de façon interchangeable des adresses équivalentes positives et négatives.

USR

USR: imm& dif
USR (aexpr)

Cette fonction envoie \aexpr \ dans un sous-programme de langage machine.
L'argument aexpr est évalué et placé dans l'accumulateur à point décimal flottant (emplacements $9D à $A3) et il s'établit un JSR décimal flottant (emplacements $9D à $A3) et il s'établit un JSR renvoyant à l'emplacement $0A
Les emplacements $0A à $0C doivent contenir un JMP renvoyant à l'emplacement de début du sousprogramme de langage machine. La valeur obtenue pour la fonction est placée dans l'accumulateur à point décimal flottant.
Pour obtenir un entier de 2 octets à partir de la valeur qui se trouve dans l'accumulateur à point décimal flottant, votre sousprogramme doit établir un JSR renvoyant à $E1 0C. Lorsqu'on l'obtient, la valeur entière se trouvera aux emplacements $A0 (octet d'ordre supérieur) et $A1 (octet d'ordre inférieur).
Pour convertir un résultat entier en son équivalent à point décimal flottant, de façon que la fonction puisse fournir cette valeur, placez l'entier de 2 octets dans les registres A (octet d'ordre supérieur) et Y octet d'ordre inférieur). Puis établissez un JSR renvoyant à $E2F2. Lorsqu'on l'obtient, la valeur à point décimal flottant se trouvera dans l'accumulateur à point décimal flottant.
Pour revenir à APPLESOFT, établissez un RTS.

Voici un programme sans importance, utilisant la fonction USR, simplement pour vous en montrer le format.

```
]reset
* 0A:4C 00 03 return
* 0300: 60 return
* ctrl C return
] PRINT USR(8)*3

 24
```

A l'emplacement $0A nous plaçons un JMP (code 4C) renvoyant à l'emplacement $300 (octet d'ordre inférieur d'abord, puis d'ordre supérieur). A l'emplacement $300, nous plaçons un RTS (code 601. Revenant à APPLESOFT, lorsque l'on a rencontré USR (81, l'argument 8 a été mis en place dans l'accumulateur, le moniteur a établi un JSR renvoyant à l'emplacement S0A où il a trouvé un JMP renvoyant à $300. A l'emplacement $300 il a trouvé un RTS qui l'a renvoyé à APPLESOFT. La valeur obtenue était justement la valeur d'origine 8 qui se trouvait dans l'accumulateur, valeur que APPLESOFT a alors multipliée par 3 pour obtenir 24.

CHAPITRE 4

Instructions d'édition et instructions relatives au format

Dans le chapitre 3. voir également ctrl C.

LIST

LIST: imm & dif

LIST [linenum1 l f- linenum2]
LIST [linenum1] f, linenum2]

S'il n'y a ni linenuml ni linenum2, avec ou sans délimiteur, la totalité du programme est affichée sur l'écran et s'il y a linenuml sans délimiteur ou linenum1 =linenum2, alors c'est seulement la ligne numérotée linenum1 qui est affichée. S'il y a linenuml et un délimiteur, alors le programme est affiché depuis la ligne numérotée linenum1 jusqu'à la fin. S'il y a un délimiteur et linenum2, alors le programme est affiché depuis le début jusqu'à la ligne numérotée linenum2. S'il y a à la fois linenum1, un délimiteur et linenum2, le programme est affiche depuis la ligne numérotée linenum1 jusqu'à la ligne numérotée linenum2, comprise.

Si plus d'une ligne doit être affichée, si la ligne numérotée linenum1 dans l'instruction LIST n'apparaît pas dans le programme, l'instruction LIST utilisera la ligne de numéro immédiatement supérieur qui apparaît dans le programme. Si la ligne est numérotée linenum2 dans l'instruction LIST, n'apparaît pas dans le programme, l'instruction LIST utilisera la ligne de numéro immédiatement inférieur qui apparaît dans le programme.

Les instructions suivantes affichent toutes la totalité du programme.
LIST 0 LIST [,l-] 0 LIST 0 [,l-] 0

L'instruction

LIST linenum, 0 affiche depuis la ligne de numéro linenum jusqu'à la fin du programme

LIST, 0 Affiche la totalité du programme puis donne le message ?SYNTAX ERROR.

APPLESOFT simule vos lignes de programme avant de les enregistrer, en enlevant les espaces non nécessaires dans le processus. Lors de l'affichage, APPLESOFT reconstitue les lignes de programmes simulées, en ajoutant des espaces selon ses propres règles. Par exemple:

10 C =+5/-6:B=-5 devient
10C=+5/-6:B=-5 une fois affiché

LIST utilise une largeur de ligne variable et des reculs sur l'alignement variables. Ceci peut constituer un problème lorsque vous essayez d'éditer ou de copier une instruction affichée. Pour obliger LIST à abandonner son format avec des espaces supplémentaires, effacez l'écran et réduisez la fenêtre de texte à la largeur 33 (maximum).

HOME
POKE 33,33

APPLESOFT ramène une ligne à 239 caractères puis l'instruction LIST ajoute libéralement des espaces. Vous pouvez donc entrer de nombreux caractères supplémentaires en abandonnant les espaces lorsque vous les tapez - LIST les ajoute ensuite. Si vous essayez de copier votre instruction ainsi étendue à partir de l'écran, ceci se traduit par le fait qu'elle est à nouveau ramenée à 239 caractères, y compris les espaces ajoutés par LIST.
L'affichage est abandonné par ctrl C.

DEL

DEL: imm & dif
DEL linenum1, linenum2

DEL efface l'étendue des lignes allant de linenum1 à linenum2, comprises. Si linenum1 n'est pas un numéro existant de ligne du programme, c'est le numéro de ligne immédiatement supérieur existant dans le programme que l'on utilise au lieu de linenum1; si linenum2 n'est pas un numéro de ligne existant dans le programme, c'est le numéro de ligne immédiatement inférieur existant qui est utilisé.
Si vous ne suivez pas le format habituel, l'instruction DEL donne les résultats variables suivants:

Syntaxe	**résultat**
DEL	?SYNTAX ERROR
DEL,	?SYNTAX ERROR
DEL ,b	?SYNTAX ERROR
DEL -a[,b]	?SYNTAX ERROR
DEL 0,b	efface la ligne zéro, indépendamment de la valeur de b.
DEL 1,-b	est ignorée, même si le numéro de ligne le plus faible du programme est zéro.
DEL a,-b	?SYNTAX ERROR si a est supérieur au numéro de ligne le plus faible du programme, à moins que le numéro de ligne le plus faible du programme ne soit zéro et que a ne soit un.
DEL a,-b	est ignoré si a n'est pas zéro et si l'unique ligne de programme est la ligne no zéro.
DEL a,-b	est ignoré si a n'est pas zéro et si a est inférieur ou égal au numéro de ligne le plus faible du programme.
DEL a[,]	est ignoré
DEL a,b	est ignoré si a n'est pas zéro et si a est supérieur à b

Lorsqu'on l'utilise en exécution différée, DEL travaille comme indiqué ci-dessus, puis arrête l'exécution. CONT n'a pas d'action dans cette situation.

REM

REM imm & dif
REM (caractère l")

Cette instruction sert à permettre l'insertion dans un programme d'un texte de toute sorte. On peut y inclure tous les caractères, y compris des séparateurs d'instructions et des blancs. Leur signification habituelle est ignorée. Une instruction REM ne se termine que par return.

Lorsque les instructions REM sont listées, APPLESOFT insère un espace supplémentaire après REM, indépendamment du nombre d'espaces tapés après REM par l'utilisateur.

VTAB

VTAB imm & dif
VTAB aexpr

Déplace le curseur pour l'amener à la ligne qui se trouve à \aexpr \ lignes plus bas sur l'écran. La ligne supérieure est la ligne 1; la ligne inférieure est la ligne 24. Cette instruction peut impliquer le déplacement du curseur vers le haut ou vers le bas, mais jamais vers la droite ou vers la gauche.
Des arguments qui se trouvent en dehors de l'étendue 1 à 24, font apparaître le message ?
ILLEGAL QUANTITY ERROR

VTAB utilise des mouvements absolus, relatifs uniquement à la ligne supérieure et à la ligne inférieure de l'écran elle ignore la fenêtre de texte. En mode graphique, VTAB va déplacer le curseur dans la zone graphique de l'écran. Si VTAB déplace le curseur vers une ligne qui se trouve en-dessous de la fenêtre de texte, toute l'impression ultérieure s'effectue sur cette ligne.

HTAB

HTAB imm & dif
HTAB aexpr

Supposons que la ligne sur laquelle le curseur est situé présente 255 positions, de 1 à 255. Quelle que soit la largeur de fenêtre de texte, que vous pouvez avoir définie, les positions 1 à 40, sont sur la ligne en cours, les positions 41 à 81 sont sur la ligne immédiatement en dessous et ainsi de suite. HTAB déplace le curseur pour l'amener à la position qui se trouve à \aexpr\ positions à partir du bord gauche de la ligne d'écran en cours. Les déplacements définis par VTAB sont relatifs à la marge gauche de la fenêtre de texte, mais indépendants de la largeur de la ligne. HTAB peut déplacer le curseur à l'extérieur de la fenêtre de texte, mais uniquement assez loin pour imprimer (PRINT) un caractère. Pour placer le curseur dans la position la plus à gauche de la ligne en cours, utilisez HTAB 1.

HTAB 0 déplace le curseur à la position 256.
Si \ aexpr \ est négatif ou supérieur à 255, il apparaît le message
?ILLEGAL QUANTITY ERROR
Notez que les structures de HTAB et VTAB ne sont pas parallèles, en ce sens qu'une valeur de HTAB allant au-delà du bord droit de l'écran n'amène pas le message
?ILLEGAL QUANTITY ERROR
mais fait que le curseur saute à la ligne immédiatement inférieure et tabule
(aexpr1)MOD 40)+ 1.

TAB

TAB imm & dif
TAB (aexpr)

TAB doit être utilisé dans une instruction PRINT et \aexpr\ doit être mis entre parenthèses. TAB déplace le curseur pour l'amener à la position qui se trouve \ aexpr \ positions d'impression à partir de la marge de gauche de la fenêtre du texte si aexpr est supérieur à la valeur de la position en cours du curseur par rapport à la marge de gauche. Si \ aexpr \ est inférieur à la valeur de position en cours du curseur, alors le curseur n'est pas déplacé - TAB ne déplace jamais le curseur vers la gauche (utilisez HTAB dans ce but)
Si TAB déplace le curseur au-delà de la limite la plus à droite de la fenêtre du texte, le curseur est envoyé à la limite la plus à gauche -e la ligne immédiatement en-dessous de la fenêtre de texte et l'espacement continue à partir de là

TAB (0) place le curseur à la position 256.
\aexpr \doit être dans l'étendue 0 à 255, sinon il apparaît le message
?ILLEGAL QUANTITY ERROR
TAB n'est analysé comme mot réservé que si le caractère suivant autre qu'un espace est une parenthèse gauche.

POS

POS imm & dif
POS (expr)

donne la position horizontale actuelle du curseur sur l'écran, par rapport au bord gauche de la fenêtre de texte. Si le curseur est sur le bord gauche, on obtient 0. Bien que expr n'ait ici comme rôle que de séparer les parenthèses, elle est néanmoins évaluée, et ne doit donc pas être illégale. Tout ce qui peut s'interpréter comme un nombre, un nom de chaîne ou un nom de variable peut s'utiliser pour expr. Si expr est un ensemble de caractères qui ne peut pas être un nom de variables, les caractères doivent se trouver entre guillemets.
Notez que pour HTAB et TAB les positions sont numérotées à partir de 1, tandis que pour POS et SPC, elles sont numérotées à partir de 0. Donc

```
PRINT TAB (23); POS(0)
```

amène l'impression de 22, alors que

```
PRINT SPC (23); POS(0)
```

amène l'impression de 23.

SPC

SPC imm & dif
SPC (aexpr)

doit être utilisé dans une instruction PRINT et aexpr doit se trouver entre parenthèses. Cette instructions introduit des espacements définis par \ aexpr \ entre l'item précédemment imprimé (ou par défaut le bord gauche de la fenêtre de texte) et le nouvel item à imprimer, si l'instruction SPC est concaténée avec les items précédents et suivants par juxtaposition ou par intervention de point et virgule. SPC(0) n'introduit aucun espacement.

\aexpr\ doit être dans l'étendue 0 à 255 compris sinon il apparaît le message
? ILLEGAL QUANTITY ERROR
Toutefois, une instruction SPC(aexpr) peut être concaténée à une autre sous la forme

```
PRINT SPC (250)SPC(139)SPC(255)
```

et ainsi de suite, pour fournir des espacements positifs de grandeur arbitraire.

Notez que tandis que HTAB amène le curseur à une position absolue de l'écran par rapport au bord gauche de la fenêtre de texte, SPC(aexpr) déplace le curseur d'un certain nombre d'espaces à partir de l'item précédemment imprimé. Cette nouvelle position peut être n'importe où dans la fenêtre du texte, selon l'emplacement de l'item précédemment imprimé.
Un déplacement au-delà de la limite la plus à droite de la fenêtre de texte amène le déplacement ou l'impression à reprendre au bord gauche de la ligne immédiatement en-dessous sur la fenêtre du texte.
Dans le cas d'impression dans les champs de tabulateur, cet espacement peut se faire à l'intérieur d'un champ de tabulateur ou en passant dans un autre champ de tabulateur ou peut occuper un champ de tabulateur par lui-même.

Si \aexpr \ est un réel, il est converti en un entier.
SPC est analysé comme mot réservé uniquement si le caractère suivant autre qu'un espacement est une parenthèse gauche.

HOME

HOME imm & dif
HOME

Pas de paramètres. Amène le curseur à la position supérieure gauche de l'écran à l'intérieur de la fenêtre de défilement et efface tout le texte qui se trouve à l'intérieur de la fenêtre. Cette instruction est identique à "CALL-936" et à "esc @ return".

CLEAR

CLEAR imm & dif
CLEAR

Pas de paramètres. Ramène à la valeur zéro toutes les variables, les tableaux et les chaînes. Ramène à l'état initial les pointeurs et les registres.

FRE

FRE imm & dif
FRE (expr)

FRE indique la valeur de la mémoire (en octets) dont l'utilisateur peut encore disposer. Vous pouvez quelquefois disposer de davantage de mémoire que ce que vous pensez, car APPLESOFT n'enregistre qu'une seule fois les chaînes dupliquées. C'est-à-dire que si A$='PIPPIN' et B$="PIPPIN", la chaîne "PIPPIN" ne sera enregistrée en mémoire qu'une seule fois.
Si le nombre d'octets libres en mémoire dépasse 32767, FRE(expr) fournit un nombre négatif. En ajoutant 65536 à ce nombre, vous obtiendrez le nombre effectif d'octets libres en mémoire.
FRE (expr) fournit le nombre d'octets qui subsistent en-dessous de l'emplacement d'enregistrement des chaînes et au-dessus de l'emplacement des tableaux numériques HIMEM: peut être positionné à une valeur aussi élevée que 65535 mais si HIMEM: est positionné au-delà de l'emplacement de mémoire RAM le plus élevé qui se trouve dans votre APPLE, l'instruction FRE peut donner un nombre n'ayant guère de signification et dépassant la capacité de mémoire de l'ordinateur (voir HIMEM: et POKE pour une discussion des limites de la mémoire).
Si les contenus d'une chaîne sont modifiés pendant le cours d'un programme, (par exemple A$ qui était égal à "cat" devient A$="dog"), APPLESOFT n'élimine pas "cat", mais ouvre simplement un nouveau fichier pour "dog '. Il en résulte qu'une quantité d'anciens caractères viennent doucement remplir par le bas, depuis HIMEM: jusqu'à la partie supérieure de l'espace des tableaux. APPLESOFT va automatiquement faire le ménage lorsque ces anciennes données arrivent dans l'espace libre des tableaux, mais si vous utilisez une partie de l'espace libre pour des programmes de langage machine ou comme mémoire intermédiaire de page haute résolution, ils peuvent être démolis.
L'utilisation périodique dans votre programme d'une instruction de la forme :

```
X = FRE (0)
```

oblige le ménage à se faire et évite ces risques.
Bien que expr ne soit utilisé que pour séparer les parenthèses, il est évalué et ne doit donc pas être illégal.

FLASH INVERSE NORMAL

FLASH imm & dif
INVERSE imm & dif
NORMAL imm & dif

Ces trois instructions s'utilisent pour positionner les modes sortie vidéo. Elles n'utilisent pas de paramètres et n'affectent pas l'affichage des caractères que vous tapez sur l'ordinateur ni les caractères qui se trouvent déjà sur l'écran.
FLASH positionne le mode vidéo en "clignotement" qui fait que les informations de sortie provenant de l'ordinateur sont alternativement représentées sur l'écran en blanc sur fond noir puis inversement en noir sur fond blanc.
INVERSE positionne le mode vidéo de façon que les informations de sortie de l'ordinateur s'impriment en lettres noires sur fond blanc.
NORMAL positionne le mode en lettres blanches sur fond noir comme habituellement à la fois pour les entrées et les sorties.

SPEED

SPEED imm & dif
SPEED = aexpr

définit la vitesse à laquelle les caractères doivent être envoyés sur l'écran ou sur d'autres dispositifs d'entrée/ sortie. La vitesse la plus faible est 0. La vitesse la plus rapide est 255. Des valeurs en dehors de cette étendue

?ILLEGAL QUANTITY ERROR du message

ESC

esc A imm seulement (édition seulement)
esc B imm seulement (édition seulement)
esc C imm seulement (édition seulement)
esc D imm seulement (édition seulement)

La touche escape, étiquetée "ESC", peut s'utiliser en liaison avec les touches de lettres A ou B ou C ou D pour déplacer le curseur: pour déplacer le curseur d'un espacement, enfoncez d'abord la touche escape puis relâchez la touche escape et enfoncez la touche lettre appropriée.

Instruction	**déplace le curseur d'un espacement vers**
esc A	la droite
esc B	la gauche
esc C	le bas
esc D	le haut.

Ces instructions escape n'affectent pas les caractères au-dessus desquels passe le curseur: ces caractères subsistent à la fois sur l'écran TV et en mémoire. Par elles-mêmes, les instructions escape n'affectent pas non plus la ligne de programme en cours de frappe.

Pour modifier une ligne de programme, affichez cette ligne sur l'écran (instruction LIST) et utilisez les instructions escape pour déplacer le curseur de façon qu'il se trouve directement sur le véritable premier caractère de la ligne affichée. Puis utilisez la touche de flèche à droite et la touche REPT pour recopier les caractères qui apparaissent sur l'écran, en tapant un caractère différent chaque fois que le curseur se trouve sur un caractère que vous désirez changer. Si vous n'avez pas affiché la ligne à l'aide de l'instruction list, ne copiez pas le caractère (1 1 qui

apparaît au début de cette ligne. Enfin, enfoncez la touche RETURN pour enregistrer en mémoire la ligne ou l'exécuter.
repeat imm seulement (édition seulement)

La touche repeat est la touche étiquetée "REPT". Si vous maintenez enfoncée la touche repeat tout en enfonçant une touche de caractère, ce caractère sera répété. La première fois que vous enfoncez la touche repeat seule, elle répète le dernier caractère tapé.

Flèche

Flèche à droite imm seulement (édition seulement)
Flèche à gauche imm seulement (édition seulement)

La touche flèche à droite déplace le curseur vers la droite. Au fur et à mesure que le curseur se déplace, chaque caractère sur lequel il passe sur l'écran est copié dans la mémoire de APPLE II, exactement comme si vous aviez tapé le caractère. Cette touche s'utilise, avec la touche repeat, pour éviter de devoir retaper une ligne entière qui ne demande que des changements mineurs.
La touche flèche à gauche déplace le curseur vers la gauche. Chaque fois que le curseur se déplace vers la gauche, un caractère est effacé dans la ligne de programme que vous êtes en train de taper quel que soit le caractère sur lequel le curseur se déplace. L'écran est ignoré par cette instruction et rien n'est changé sur l'écran.

A moins que vous ne soyez en train de taper une ligne pour laquelle vous n'avez pas encore enfoncé la touche return, la touche flèche à gauche n'a pas de caractères de la ligne de programme en cours à effacer. Dans ce cas, son utilisation va amener le caractère (1 I à apparaître dans la colonne 0 de la ligne immédiatement inférieure, suivi du curseur. C'est pourquoi le curseur, fréquemment, ne peut pas être amené à la colonne 0 de l'écran TV en utilisant la touche flèche à gauche. Pour chaque déplacement il faut effacer un caractère de la ligne de programme en cours. Pour avoir des déplacements purs, sans effacer ni copier, voir les instructions

ctrl X

ctrl X imm seulement

Demande à APPLE II d'ignorer la ligne actuellement en cours de frappe, sans effacer toute ligne précédente de même numéro de ligne. Une barre oblique inverse (\ 1 est affichée à la fin de la ligne à ignorer et le curseur saute à la colonne 0 de la ligne suivante. Cette instruction peut également s'utiliser durant une réponse à une instruction INPUT.

CHAPITRE 5
Tableaux et chaînes

DIM

DIM imm & dif
DIM var indices (i, var indice)1

Lorsqu'une instruction DIM s'exécute, elle réserve la place d'un tableau du nom de var. Deux octets de mémoire sont utilisés pour enregistrer en mémoire un nom de tableau de variables, deux pour la dimension du tableau, un pour le nombre de dimensions, et deux pour chaque dimension. Comme on le discute ci-dessous, l'importance de la place allouée pour les éléments d'un tableau dépend du type de tableau.

Les indices vont de 0 à \subcript \. Le nombre d'éléments d'un tableau à n dimensions est (\subscript1 \ + 1)*(\subscript2 \ + 1)* . . . *(\subscriptn \ + 11.
Par exemple DIM SHOW (4,5,3) réserve la place de 5*6*4 éléments (120 éléments). Des éléments typiques sont
SHOW (4,4,11
SHOW (0,0,2)

Le nombre maximum de dimensions pour un tableau est 88, même si chaque dimension peut ne contenir qu'un seul élément.
DIM A(0,0, . . .01, où il y a 89 zéros donne un message
? OUT OF MEMORY ERROR
mais
DIM A(0,0, . . . 0) où il y a 88 zéros ne le donne pas

En pratique toutefois, la dimension des tableaux est souvent limitée bien plus par l'importance de la mémoire disponible. Chaque élément entier d'un tableau occupe deux octets (16 bits) en mémoire. Chaque élément réel d'un tableau occupe 5 octets (40 bits) en mémoire. Les variables chaînes d'un tableau utilisent 3 octets pour chaque élément (un pour la longueur, deux pour un pointeur d'emplacement), enregistré en mémoire sous forme d'un tableau d'entiers si le tableau est dimensionné. Du fait que les chaînes elles-mêmes sont enregistrées en mémoire par lé programme, elles occupent encore 1 octet de plus par caractère.

Si un élément sous forme de tableau est utilisé dans un programme avant que cette variable ne soit dimensionnée, APPLESOFT affecte un indice maximum de dix pour chaque dimension dans les indices de l'élément.

Le fait d'utiliser une variable dont l'indice est supérieur au maximum désigné ou qui demande un nombre de dimensions différent de celui spécifié dans une instruction DIM amène l'apparition du message
? BAD SUBSCRIPT ERROR

Si le programme dimensionne un tableau qui au même nom qu'un tableau précédemment dimensionné (même s'il a été dimensionné par défaut), il apparaît le message
? REDIM'D ARRAY ERROR

Les différentes chaînes d'un tableau de chaînes ne sont pas dimensionnées mais croissent et décroissent selon nécessité. L'instruction
WARD $151= "ABCDE"
crée une chaîne de longueur 5. L'instruction
WARD $15) = ""
supprime la place réservée à la chaîne WARD $151. Une chaîne peut contenir au maximum 255 caractères.

Les éléments du tableau sont remis à zéro lors de l'exécution des instructions RUN ou CLEAR.

LEN

LEN imm & dif
LEN (sexpr)

Cette fonction indique le nombre de caractères qui se trouvent dans une chaîne, entre 0 et 255. Si l'argument est une concaténation de chaînes dont la longueur combinée est supérieure à 255, on obtient le message

? STRING T00 LONG ERROR

STR$

STR $ imm & dif
STR $ (aexpr)

Cette fonction convertit \aexpr\ en une chaîne qui représente cette valeur. \aexpr\ est évalué avant d'être converti en une chaîne. STR $1100 000 000 000) donne 1 E+ 11.
Si \aexpr \ dépasse les limites des nombres réels, on obtient l'affichage du message

? OVERFLOW ERROR

VAL

VAL imm & dif
VAL (sexpr)

Cette fonction essaie d'interpréter une chaîne comme réel ou comme entier, en fournissant la valeur de ce nombre.

Le premier caractère de la chaîne doit être un item possible dans un nombre (des- espaces blancs en tête sont acceptables) sinon c'est un zéro qui est donné. Chaque caractère à la suite est également examiné jusqu'à ce que l'on rencontre le premier caractère non numérique bien défini (les espaces blancs intermédiaires), les points décimaux, les signes + et - et E sont tous des caractères numériques possibles dans le contexte correct). Le premier caractère non numérique et tous les caractères suivants sont ignorés et la chaîne jusqu'à ce point est évaluée comme un réel ou un entier.
Si l'argument de VAL est une concaténation de chaînes, comprenant plus de 255 caractères, on reçoit le message

? STRING T00 LONG ERROR

Si la valeur absolue du nombre fourni est supérieure à 1 E38, ou si le numéro contient plus de 38 chiffres (y compris les zéros de queue), on obtient le message
? OVERFLOW ERROR

CHR$

CHR $ imm & dif
CHR $ (aexpr)

C'est une fonction qui donne le caractère ASCII qui correspond à la valeur aexpr. \aexpr\ doit être entre 0 et 255, inclusivement, sinon on obtient le message ? ILLEGAL QUANTITY ERROR
Les nombres réels sont convertis en entiers.

ASC

ASC imm & dif
ASC (sexpr)

Cette fonction fournit un code ASCII (pas nécessairement le nombre le plus faible) pour le premier

caractère de \ sexpr \. Les codes ASCII sur l'étendue 96 à 255 vont engendrer, sur l'ordinateur APPLE, des caractères qui répètent ceux de l'étendue 0 à 95. Par contre, bien que CHR $165) donne un A et que CHR $1193) donne également un A, APPLESOFT ne reconnaît pas les deux comme étant le même caractère lors de l'utilisation d'opérateurs logiques de chaînes.

Si c'est une chaîne qui est l'argument, elle doit être mise entre guillemets et les guillemets ne doivent pas être compris dans la chaîne. Si la chaîne est une chaîne nulle on obtient le message

? ILLEGAL QUANTITY ERROR

Un essai d'utiliser la fonction ASC sur ctrl@ se traduit par le message

? SYNTAX ERROR

LEFT

LEFT $ imm & dif
LEFT $ (sexpr, aexpr)

Cette fonction donne les premiers (les plus à gauche) caractères \aexpr\ de \sexpr \.

```
PRINT LEFT $(-APPLESOFT% 5)
APPLE
```

aucune partie de cette instruction ne peut être omise. Si \aexpr \ < 1 ou \aexpr \ >255, alors on obtient le message

? ILLEGAL QUANTITY ERROR

Si \aexpr \ est un réel, il est converti en un entier.

Si \aexpr\>LEN(sexpr), seuls les caractères qui constituent la chaîne sont fournis. Toute position supplémentaire est ignorée.

Si "$" est omis dans le nom de l'instruction, APPLESOFT traite LEFT comme un nom de variable arithmétique et on obtient le message

? TYPE MISMATCH ERROR

RIGHT

RIGHT $ imm & dif
RIGHT $ (sexpr, aexpr)

Cette fonction fournit les derniers (les plus à droite) caractères \aexpr\ de \sexpr \

```
PRINT RIGHT $("APPLESOFT" + "ARE", 8)
SOFTWARE
```

Aucune partie de cette instruction ne peut être omise. Si \aexpr\> = LEN (sexpr) alors RIGHT $ fournit la totalité de la chaîne. Si \aexpr\ <1 or \aexpr\ >255, on obtient le message

? ILLEGAL QUANTITY ERROR

RIGHT $ (sexpr, aexpr) = MID $(aexpr, LEN(sexpr)+1-\aexpr\)

Si "$" est omis dans le nom de l'instruction, APPLESOFT traite RIGHT comme un nom de variable arithmétique et on obtient le message

? TYPE MISMATCH ERROR

MID

MID $ imm & dif
MID $ (sexpr, aexprl f, aexpr2])

MID $ appelé avec deux arguments fournit la sous-chaîne qui commence au caractère d'ordre \aexprl \ de \sexpr \ et qui va jusqu'au dernier caractère de \sexpr \.

```
PRINT MID $("APPLESOFT", 3)
PLESOFT
```

MID$ (sexpr, aexpr)= RIGHT $(aexpr, LEN(sexpr)+ 1- \aexpr \) MID$ appelé avec trois arguments fournit les \aexpr2\ caractères de \sexpr\, commençant parle caractère d'ordre \aexpr l \ et continuant vers la droite.

```
PRINT MID $("APPLESOFT", 3, 5)
PLESO
```

Si \aexprl \>LEN (sex r) alors MID$ fournit une chaîne nulle. Si \aexprl \+ \aexpr2\ dépasse la longueur de \ sexpr \ (où 255, la longueur maxima d'une chaîne quelconque), tout caractère supplémentaire est ignoré. MID$(A$,255,255) fournit un caractère si LEN(A$)=255, sinon, c'est la chaîne nulle qui est fournie.

Si soit \aexpr l \ soit \aexpr2 \ sont à l'extérieur de l'étendue 1 à 255 inclusivement, on obtient le message

? ILLEGAL QUANTITY ERROR

Si $ est omis dans le nom de l'instruction, APPLESOFT traite MID comme un nom de variable arithmétique et on obtient le message

? TYPE MISMATCH ERROR

STORE et RECALL

STORE imm & dif
RECALL imm & dif

STORE avar
RECALL avar

Ces instructions mémorisent et rappellent les tableaux qui se trouvent sur cassette.

Les noms de tableaux ne sont pas mémorisés avec leurs valeurs de sorte qu'un tableau peut être lu en utilisant un nom différent de celui qui est utilisé avec l'instruction STORE.

Les dimensions du tableau nommé par l'instruction RECALL doivent être identiques aux dimensions du tableau d'origine tel qu'il a été mémorisé avec l'instruction STORE. Par exemple si un tableau dimensionné par DIM A(5,5,5) est mémorisé avec l'instruction STORE, on peut le rappeler dans un tableau dimensionné par DIM 13(5,5,51. Le fait de ne pas en tenir compte se traduit par des nombres brouillés dans le tableau rappelé, par des zéros supplémentaires dans le tableau ou par le message ?OUT OF MEMORY ERROR

En général vous recevrez le message ?OUT OF MEMORY ERROR uniquement si le nombre total d'éléments réservés pour le tableau rappelé est insuffisant pour contenir tous les éléments du tableau qui étaient mémorisés.

```
DIM A (5,5,5)
STORE A
```

Sauvegarde 6*6*6 éléments sur la bande en cassette. DIM B (5,35)

```
RECALL B
```

se traduit par le message

```
ERR
```

et par des nombres brouillés dans le tableau B mais l'exécution du programme va se poursuivre.

Par contre

```
DIM B (5,25)
RECALL B
```

va donner l'affichage du message
? OUT OF MEMORY ERROR
et l'exécution du programme va cesser. Dans ce cas le tableau B contenait 6*26 éléments - trop peu d'éléments pour contenir tous les éléments du tableau A

Si le tableau rappelé au même nombre de dimensions ([DIM A(,5,5,5) spécifie un tableau de trois dimensions, chacune de 6 valeurs d'indice) que le tableau qui était mémorisé, chacune des dimensions du tableau rappelé peut avoir davantage de valeurs d'indice que la dimension correspondante du tableau mémorisé. Toutefois, il va en résulter des nombres brouillés dans le tableau rappelé, à moins que ce soit la dernière dimension du tableau rappelé qui possède davantage de valeurs d'indice que la dernière dimension du tableau mémorisé. Dans tous les cas vous trouverez des zéros supplémentaires mémorisés dans les éléments en excès du tableau rappelé, mais ce n'est que dans le dernier cas que vous trouverez ces zéros là où vous les attendiez. Après avoir mémorisé un tableau avec

```
DIM A (5,5,5)
STORE A
```

vous trouverez que

```
DIM B (10,5,5)
RECALL B
```

ainsi que

```
DIM B (5,10,5)
RECALL B
```

remplissent tous les deux le tableau B avec des nombres mélangés en provenance du tableau A, tandis que

```
DIM B (5,5,10)
RECALL B
```

marche bien avec des zéros dans les éléments supplémentaires du tableau B.

Nous avons discuté deux règles permettant de mémoriser et de rappeler les tableaux d'un môme nombre de dimensions.
1. Seule la dernière dimension du tableau rappelé peut avoir davantage de valeurs d'indice que la dernière dimension du tableau mémorisé.
2. Le nombre total des éléments rappelés doit être au moins égal au nombre des éléments mémorisés.
Si la règle 2 est suivie et si la règle 1 est suivie pour les dimensions qui sont communes aux deux tableaux (ce doit être les premières dimensions) alors on peut rappeler un tableau de plus de dimensions que le tableau qui était mémorisé. On obtient un message d'erreur, mais l'exécution du programme se poursuit.

```
DIM B (5,5,5,5)
RECALL B
```

va travailler mieux dans l'exemple ci-dessus (après le message d'erreur et avec de nombreux zéros supplémentaires dans le tableau B) mais

```
DIM 8(5,5,3,5)
RECALL B
```

va remplir le tableau B avec des nombres brouillés (après avoir donné le message d'erreur) et

```
DIM B (5,5,1,1)
RECALL B
```

va amener le message

```
? OUT OF MEMORY ERROR
```

parce que les (6*6*2*2) éléments du tableau B sont en nombre moindre que les 6*6*6 éléments mémorisés dans le tableau A.

On peut mémoriser uniquement des tableaux de nombres réels et entiers. Les tableaux chaînes doivent être convertis en un tableau d'entiers à l'aide de la fonction ASC pour pouvoir être mémorisés.

Bien que STORE et RECALL se réfèrent à leurs variables sans mention d'indice ou de dimension, on ne peut mémoriser ou rappeler que les tableaux. Le programme

```
100 A(3) = 45
110 A=27
120 STORE A
```

mémorise sur bande les éléments du tableau A(0) à A(10) (par défaut le tableau est dimensionné à 11 éléments), et non la variable A (qui est égale à 27 dans le programme).

L'instruction STORE ne délivre pas de message ou autre signal; l'utilisateur doit avoir son lecteur/enregistreur de cassette en marche en mode enregistrement lorsque l'instruction est en cours d'exécution. Un bip signale le début de l'enregistrement et un autre le bip en signale la fin. Le programme

```
300 DIM B (5,13)
310 B = 4
320 RECALL B
```

lit sur la bande les 84 (6* 14) éléments de tableau 8(0,0) à B(5,13). La valeur de la variable B n'est pas modifiée. Ici non plus il n'y a pas de message; des bips signalent le début et la fin de la lecture.

Si soit STORE soit RECALL contiennent un nom de tableau qui n'a pas été précédemment dimensionné ou utilisé avec un indice, on obtient le message

```
? OUT OF DATA ERROR
```

En mode d'exécution immédiate, si soit STORE soit RECALL se réfèrent à un nom de tableau défini dans une ligne de programme à exécution différée, alors la ligne de programme à exécution différée doit avoir été exécutée avant STORE ou RECALL.

STORE et RECALL ne peuvent être interrompus que par RESET.

Si les mots réservés STORE ou RECALL sont utilisés comme premiers caractères d'un nom de variable quelconque, les instructions peuvent être exécutées avant qu'il apparaisse un message

```
? SYNTAX ERROR
```

L'instruction

```
STOREHOUSE=5
```

amène le message

```
? OUT OF DATA ERROR
```

à moins que l'on ait défini un tableau dont le nom commence par le caractère H0. Dans ce dernier cas APPLESOFT va essayer de mémoriser le tableau: vous allez d'abord entendre un bip puis un second bip, finalement il va apparaître le message ? SYNTAX ERROR du fait que APPLESOFT s'efforce d'analyser le reste de l'instruction "=5". Pour arrêter les bips et le message d'erreur, vous pouvez enfoncer la touche RESET

L'instruction

```
RECALLOUS=234
```

va amener le message

```
? OUT OF DATA ERROR
```

à moins que l'on ait défini un tableau dont le nom commence par les caractères OU. Dans ce cas APPLESOFT va attendre indéfiniment qu'un tableau provienne du lecteur de cassette. La seule façon de reprendre le contrôle de l'ordinateur est d'enfoncer la touche RESET.

CHAPITRE 6

Instructions d'entrée/sorte

Dans le chapitre 3 voir également LOAD et SAVE
Dans le chapitre 5 voir J l ORE et RECALL

INPUT dif

INPUT dif
INPUT [string;] var [{,var}]

Si la chaîne optionnelle est omise, INPUT imprime un point d'interrogation et attend que l'utilisateur tape un nombre (si var est une variable arithmétique) ou des caractères (si var est une variable chaîne). La valeur de ce nombre ou de cette chaîne est placée dans var.

S'il y a une chaîne, elle est imprimée exactement comme spécifié; aucun point d'interrogation espace blanc ou autre signe de ponctuation n'est imprimé après la chaîne. Notez que l'on ne peut utiliser qu'une seule chaîne optionnelle. Elle doit apparaître directement après INPUT et être suivie d'un point virgule.

INPUT va accepter uniquement un réel ou un entier comme entrée numérique et non une expression arithmétique. Les caractères espace blanc, +, -, E, et le point sont des parties légitimes de l'entrée numérique. INPUT va accepter chacun de ces caractères ou toute conténation de ces caractères sous forme acceptable (ar exemple +, -, E est acceptable, +, - ne l'est pas); une telle entrée par elle-même s'évalue comme

Dans une entrée numérique les espaces blancs sont ignorés en n'importe quelle position. Si l'entrée numérique n'est pas un réel, un entier, une virgule ou un point et virgule, il apparaît le message

?REENTER et l'instruction INPUT doit être ré exécutée.

Si on utilise une instruction ONERR, avec un autre GOTO dans la routine de traitement des erreurs pour renvoyer le programme à l'instruction INPUT défaillante, la 86e erreur INPUT peut faire que le programme saute au moniteur. Pour le reprendre, utilisez reset ctrl C return. On peut éviter ce problème en utilisant RESUME pour revenir à l'instruction

De la même façon une réponse assignée à une variable chaîne doit être une chaîne simple ou expression littérale et non une expression chaîne. Les espacements blancs qui précèdent le premier caractère sont ignorés. Si la réponse est une chaîne, des guillemets à un endroit quelconque dans la chaîne vont amener un message

?REENTER

Toutefois, à l'intérieur d'une chaîne tous les caractères, à l'exception des guillemets, de ctrl X et de ctrl M sont acceptés comme caractères pour la chaîne. Ceci comprend les deux points et la virgule. Les espaces blancs qui suivent le guillemet final sont ignorés.

Si la réponse est une expression littérale, alors les guillemets sont acceptés comme caractères en toute partie de l'expression littérale sauf comme premier caractère qui ne soit pas un espace blanc. Les espaces qui suivent le dernier caractère sont acceptés comme partie de l'expression littérale. Toutefois la virgule et les deux points (et ctrl X et ctrl M) ne sont pas acceptés comme caractères dans l'expression littérale.
Si l'utilisateur enfonce simplement la touche RETURN alors que l'on attend une réponse numérique, il ressort le message

?REENTER

et l'instruction INPUT est ré exécutée. Si l'on enfonce seulement la touche RETURN lorsqu'on attend une réponse chaîne, cette réponse est interprétée comme chaîne nulle et l'exécution des programmes continue.
Les variables successives sont affectées des valeurs successivement tapées. Les variables chaînes et les variables arithmétiques peuvent être mélangées dans la même instruction INPUT, mais les réponses de l'utilisateur doivent être chacune du type approprié. Les réponses tapées peuvent être séparées par des virgules ou des return. II en résulte que si un utilisateur tape des virgules dans une réponse qui ne commence pas par des guillemets, tous les caractères tapés à la suite sont ignorés. Après un deux points, les virgules sont également ignorées, de sorte que le début d'une autre réponse doit être signalé par un return.

Si un return est rencontré avant affectation de réponse à toutes variables, deux points d'interrogation sont imprimés pour indiquer que l'on attend une réponse supplémentaire. Lorsque l'on rencontre un return, si la réponse contient davantage de champs de réponse que l'instruction ne l'attendait, ou s'il existe un deux points dans la réponse finale attendue (mais non à l'intérieur d'une chaîne), alors on reçoit le message

? EXTRA IGNORED

et l'exécution du programme continue.

Si un deux points ou une virgule est le premier caractère d'une réponse INPUT, la réponse est évaluée pour zéro ou comme chaîne nulle.

Notez que dans l'instruction INPUT la chaîne optionnelle doit être suivie par un point virgule mais que les variables doivent être séparées par des virgules.

ctrl C peut interrompre une instruction INPUT mais uniquement si c'est le premier caractère tapé. Le programme s'arrête lorsque l'on tape return. Un essai de continuer l'exécution après un tel arrêt se traduit par le message d'erreur

? SYNTAX ERROR ctrl C est traité comme tout autre caractère si ce n'est pas le premier caractère tapé.

En essayant d'utiliser l'instruction INPUT en mode exécution directe on obtient le message

? ILLEGAL DIRECT ERROR

GET

GET dif seulement
GET var

Met en place un unique caractère à partir du clavier, sans l'afficher sur l'écran et sans nécessiter d'enfoncer la touche RETURN.
Le comportement de GET svar réserve parfois des surprises
ctrl C @ amène le caractère nul.

Le résultat de la mise en place d'une flèche gauche ou de ctrl H peut également l'imprimer, comme si le caractère nul était obtenu.
ctrl C est traité comme tout autre caractère; il n'interrompt pas l'exécution du programme.

Bien que APPLESOFT n'ai pas été étudié ou prévu pour prendre (instruction GET) des valeurs arithmétiques, vous pouvez utiliser

GET avar

sous réserve des strictes limitations suivantes:

L'entrée (instruction GET) d'un deux point ou une virgule se traduit par le message

? EXTRA IGNORED
suivi de l'obtention d'un zéro comme valeur tapée.

Le signe plus, le signe moins, ctrl@,E, un espace blanc et le point donnent tous un zéro comme valeur tapée.
En tapant un return ou une entrée non numérique on obtient le message

? SYNTAX ERROR

Avec ONERR GOTO . . . RESUME deux erreurs consécutives GET obligent le système à s'arrêter jusqu'à ce que l'on enfonce RESET. Si l'on remplace RESUME par GOT, tout est bien jusqu'à la 43e erreur GET (dans un ordre quelconque), où le programme saute au moniteur. Pour le récupérer, utilisez reset ctrl C return.

Du fait de ces limitations il est recommandé aux programmeurs sérieux de mettre en place des nombres en utilisant

GET svar

et en convertissant la chaîne résultante en un nombre à l'aide de la fonction VAL.

DATA

DATA dif
DATA [littérale I chaîne I réel I entier] f (, [littérale I chaîne I réel I entier] }]

Cette instruction crée une liste d'éléments que l'on peut utiliser au moyen des instructions READ. Dans l'ordre du numéro de ligne de l'instruction, chaque instruction DATA ajoute ses éléments à la liste des éléments obtenus par les précédentes instructions DATA du programme (numéro de ligne inférieur).
II n'est pas nécessaire que l'instruction DATA précède l'instruction READ dans un programme; les instructions DATA peuvent apparaître à tout endroit d'un programme.

Les éléments DATA qui sont affectés aux variables arithmétiques par l'instruction READ suivent généralement les mêmes règles que pour les réponses assignées par l'instruction INPUT aux variables arithmétiques par l'instruction READ suivent généralement les mêmes règles que pour les réponses assignées par l'instruction INPUT aux variables arithmétiques. Toutefois, le deux points ne peut pas être inclus comme caractère dans un élément numérique DATA.

Si ctrl C est un élément DATA, il n'arrête pas le programme même s'il est le premier caractère d'un élément. Sous réserve de cette exception, les éléments DATA qui sont affectés aux variables chaînes par l'instruction READ suivent les mêmes règles que les réponses affectées par l'instruction INPUT aux variables chaînes:

On peut utiliser soit des chaînes soit des expressions littérales, soit les deux.

Les espaces blancs avant le premier caractère et suivant une chaîne sont toujours ignorés.

Tout guillemet qui apparaît à l'intérieur d'une chaîne amène le message ?SYNTAX ERROR, mais tous les autres caractères sont acceptés comme caractères dans cette chaîne y compris le deux points et la virgule (mais non compris ctrl X et ctrl M1.

Si un élément est une expression littérale, alors le guillemet est accepté comme caractère valide à tout endroit de l'expression littérale sauf comme premier caractère qui ne soit pas un espace blanc; le deux points, la virgule, ctrl X et ctrl M ne sont pas acceptés.

Voir l'instruction INPUT pour autres détails.

Les éléments DATA peuvent être tout mélange de réels, entiers, chaînes et expressions littérales. Si l'instruction READ essaie d'assigner un élément DATA c'est-à-dire une chaîne ou une expression littérale à une variable arithmétique, on obtient le message ?SYNTAX ERROR pour la ligne DATA appropriée.

Si la liste d'éléments d'une instruction DATA contient un élément "non existant", alors on obtient un zéro (numérique) ou la chaîne nulle pour cet élément selon la variable à laquelle on a assigné l'élément. Un élément "non existant" se présente dans une instruction DATA si l'une des conditions suivantes est vraie.

1) II n'y a pas de caractères autres que des espaces blancs entre DATA et return
2) Une virgule est le premier caractère autre qu'un espace blanc à la suite de DATA
3) Entre deux virgules, il n'y a pas de caractères qui ne soient pas un espace blanc
4) La virgule est le premier caractère qui ne soit pas un espace blanc avant return

Par conséquent lorsque cette instruction est lue

100 DATA, ,

elle peut donner jusqu'à trois éléments consistant en zéros ou en chaînes nulles.

Si on l'utilise en mode exécution immédiate, DATA n'amène pas de message SYNTAX ERROR mais ses éléments ne sont pas disponibles pour une instruction READ

READ

READ imm & dif
READ var [{,var}]

Lorsque la première instruction READ est exécutée dans un programme, sa première variable prend la valeur du premier élément de la liste DATA (la liste DATA comprend tous les éléments de toutes les instructions DATA du programme mémorisé). La seconde variable (s'il y en a une seconde) prend la valeur du second élément qui se trouve dans la liste DATA et ainsi de suite. Lorsque l'instruction READ achève son exécution, elle laisse un pointeur de listes de données (DATA) après le dernier élément de données utilisé. L'instruction READ suivante exécutée (s'il y a lieu) commence en utilisant la liste de données à partir de la position du pointeur. Les instructions RUN ou RESTORE ramènent le pointeur au premier élément de la liste DATA.

Un essai de lire (instruction READ) plus de données que la liste de données n'en contient donne le message

? OUT OF DATA ERROR IN linenum

où linenum est le numéro de la ligne d'instruction READ qui demande des données supplémentaires.

En mode immédiat, vous pouvez seulement lire (instruction) des éléments à partir des instructions DATA qui existent comme lignes dans un programme actuellement mémorisé. Les éléments de DATA dans un programme mémorisé peuvent être lus même si le programme mémorisé n'est pas passé. Si aucune instruction DATA n'a été mémorisée, on obtient le message

? OUT OF DATA ERROR IN linenum

L'exécution du programme en mode immédiat ne ramène pas le pointeur de la liste de données au premier élément de la liste DATA.

Les données supplémentaires non lues demeurent inchangées.

RESTORE

RESTORE imm & dif

RESTORE n'a pas de paramètre ni d'option. Cette instruction déplace simplement le pointeur de liste de données (voir instructions READ et DATA) pour le ramener au début de la liste de données.

PRINT

PRINT imm & dif
PRINT [{expr} fil; [f expr}]}]] (,I;]
PRINT {;}
PRINT f , I

Le point d'interrogation (?) peut s'utiliser comme abréviation pour PRINT; il s'affiche comme
Sans option, PRINT se traduit par l'impression d'une ligne et un retour du chariot sur l'écran. Lorsque des options sont exercées, ce sont les valeurs de la liste des expressions spécifiées qui sont imprimées. Si ni une virgule, ni un point et virgule ne termine la liste, une impression de

ligne, un retour de chariot sont exécutés à la suite du dernier item imprimé. Si un item de la liste est suivi par une virgule, alors le premier caractère du item suivant à imprimer va apparaître à la première position du champ de tabulateur suivant disponible.

Le premier champ de tabulateur comprend les 16 positions d'impression les plus à gauche de la fenêtre du texte, positions 1 à 16. Le second champ du tabulateur occupe les 16 positions suivantes (17 à 32) et n'est disponible pour impression en champ de tabulateur que sil n'y a rien imprimé en position 16. Le troisième champ du tabulateur comprend les huit positions d'impression restantes (33 à 40) et n'est disponible que si rien n'est imprimé sur les positions 24 à 32.

On peut modifier la taille de la fenêtre de défilement du texte à l'aide des différentes instructions POKE (voir appendice J).

Le champ d'impression avec tabulateur 3 ne fonctionne pas correctement si la fenêtre de texte est définie à une largeur inférieure à 33 positions; le premier caractère peut être imprimé à l'extérieur de la fenêtre du texte.
HTAB peut également faire que PRINT affiche un premier caractère à l'extérieur de la fenêtre de texte.

Si un item de la liste est suivi par un point et virgule, alors l'item suivant est concaténé: il est imprimé directement à la suite sans espace blanc intermédiaire.

Les items listés sans virgules ou points et virgules intermédiaires sont concaténés si les items peuvent être analysés sans présenter de problèmes de syntaxe. Ceci est mieux illustré par des exemples

```
A=1 : B=2 : C=3 : C141=5: C5=7
PRINT 1/3(2'4)51 ,                  : PRINT 11A)2(B)3C(4)C5
.333333333851                       1122357

PRINT 3.4.5.6. ,                    : PRINT A."B."C.4
3.4.5.60                            10B.3.4
```

PRINT s'efforce de faire apparaître ce que vous souhaitez. S'il ne peut pas interpréter un point comme
point décimal, il le traite comme le nombre zéro, comme illustré sur les exemples ci-dessus.
PRINT suivi d'une liste de points et virgules ne fait rien de plus que PRINT seul, mais est légal.
PRINT suivi d'une liste de virgules défini un champ de tabulateur pour chaque virgule, ceci dans une limite de 239 caractères par instruction.

PRINT A$+ B$

donne le message

? STRING T00 LONG ERROR

Si la longueur des chaînes concaténées est supérieure à 255. Toutefois vous pouvez imprimer la concaténation apparente en utilisant

PRINT A$ B$

sans vous inquiéter de sa longueur.

IN#

IN# imm & dif
IN# aexpr

sélectionne une entrée en provenance du slot \aexpr\. Utilisée pour spécifier quel périphérique fournira une entrée pour les instructions INPUT suivantes. Les périphériques peuvent être dans les slots 1 à 7, comme indiqué par \aexpr\.

IN# 0 indique que l'entrée suivante proviendra du clavier au lieu de provenir du périphérique. Le slot 0
n'est pas adressable au moyen de APPLESOFT pour utilisation avec un appareil périphérique.

S'il n'y a pas de périphérique dans le slot \ aexpr \, le système va s'arrêter. Pour le récupérer, utilisez reset ctrl C return.

Si \ aexpr \ est inférieur à 0 ou supérieur à 255 on obtient le message

? ILLEGAL QUANTITY ERROR

Si \ aexpr \ est dans l'étendue 8 à 255, APPLESOFT est modifié de façon imprévisible.
Pour des transferts similaires de résultats de sortie, voir PR#.

PR#

PR# imm & dif
PR# aexpr

PR# transfère les résultats de sortie au slot \aexpr\, où \aexpr\ doit être dans l'étendue 1 à 7 inclusivement.
PR# 0 envoie les résultats de sortie sur l'écran TV et non au siot 0

S'il n'y a pas de périphérique dans le slot spécifié, le système va s'arrêter. Pour le récupérer, utilisez ctrl C return

Si \aexpr\ est inférieur à 255, on obtient le message

? ILLEGAL QUANTITY ERROR

Si \aexpr \ est dans l'étendue 8 à 255, APPLESOFT est modifié de façon imprévisible pour des transferts similaires, de données d'entrées voir IN#

LET

LET imm & dif
[LET] avar[indice] = aexpr
[LET] svar[indice] = sexpr

La variable dont le nom est à gauche reçoit la valeur de la chaîne ou de l'expression qui est à droite. Le LET est optionnel

A=2

et

A=2

sont équivalents

Le message

? TYPE MISMATCH ERROR

est affiché si vous essayez de donner

a) un nom de variable chaîne à une expression arithmétique ou
b) un nom de variable chaîne à une expression littérale ou
c) un nom de variable arithmétique à une expression chaîne

Si vous essayez de donner un nom de variable arithmétique à une expression littérale: APPLESOFT essaie d'analyser une expression littérale comme expression arithmétique.

DEF

DEF dif
FN imm & dif
DEF FN name (real avar) = aexprl
FN name (aexpr2)

permet à l'utilisateur de définir des fonctions dans un programme. D'abord la fonction FN name est définie à l'aide de DEF. Lorsque la ligne de programme définissant la fonction a été exécutée, la fonction peut être utilisée dans la forme FN name argument) où l'argument aexpr2 peut être toute expression arithmétique. L'expression aexprl de définition ne peut avoir qu'une seule ligne de programme de longueur; la fonction FN name ainsi définie peut être utilisée partout où des fonctions arithmétiques peuvent être utilisées dans APPLESOFT.

De telles fonctions peuvent être redéfinies pendant le cours d'un programme. Les règles d'utilisation des variables arithmétiques s'appliquent encore. En particulier les deux premiers caractères du nom doivent être uniques. Lorsque ces lignes

```
10 DEF FN ABC(I)=COS(I)
20 DEF FN ABT(I)=TAN(I)
```

sont exécutées, APPLESOFT reconnaît la définition d'une fonction FN AB à la ligne 10; à la ligne 20, la fonction FN AB est redéfinie.

Dans l'instruction DEF, real avar est une variable factice. Lorsque la fonction FN name, définie par l'utilisateur, est utilisée ensuite, elle est appelée avec un argument aexpr2. Cet argument remplace real avar partout où il apparaît dans l'expression de définition aexprl. aexprl peut contenir tout nombre de variables, mais bien entendu une d'entre elles seulement correspond à la variable factice real avar et donc correspond à la variable argument.

II n'est pas nécessaire que real avar de la définition apparaisse dans aexprl. Dans ce cas lorsqu'on utilise la fonction plus loin dans le programme, l'argument de la fonction est ignoré dans l'évaluation de aexprl. Même dans ce cas toutefois, l'argument de la fonction est évalué lui-même et il doit donc être quelque chose de légal.

Par exemple

```
100 DEF FN A(W) = 2 * W + W
110 PRINT FN A(23)
120 DEF FN B(X) = 4 + 3
130 G = FN B(23)
140 PRINT G
150 DEF FN A(Y) = FN B(Z) + Y
160 PRINT FN A(G)

RUN
69        [ FN A(23)=2*23+23]
7         [ FN B(quelque chose)=7]
14        [ new FN A(7)=7+7]
```

Si une instruction DEF FN name à exécution différée n'est pas exécutée avant d'utiliser FN name, on obtient le message

? UNDEF'D FUNCTION ERROR

Les fonctions chaînes définies par l'utilisateur ne sont pas autorisées. Les fonctions définies à l'aide d'un integer name% (nom de variable entière) pour name ou pour real avar ne sont pas autorisées.

Lorsqu'une nouvelle fonction est définie par une instruction DEF, 6 octets en mémoire sont utilisés pour mémoriser le pointeur relatif à cette définition.

CHAPITRE 7
Instructions relatives au déroulement du programme

GOTO

GOTO Imm & dif
GOTO linenum

effectue le branchement à la ligne dont le numéro de ligne est linenum. Si cette ligne n'existe pas ou s'il n'y a pas linenum dans l'instruction GOTO, on a le message d'erreur

?UN DEF'D STATEMENT ERROR IN linenum

où linenum est le numéro de la ligne de programme qui contient l'instruction GOTO

IF

IF imm & dif
IF expr THEN instruction [(: instruction}]
IF expr THEN [GOTO] linenum
IF expr [THEN] GOTO linenum

Si expr est une expression arithmétique dont la valeur n'est pas zéro let dont la valeur absolue est supérieure à environ 2.93873E-391, expr est considéré être vrai et toutes les instructions qui suivent THEN sont exécutées.

Si expr est une expression arithmétique dont la valeur est zéro (ou dont la valeur absolue est inférieure à environ 2.93873E-391, toutes les instructions qui suivent THEN sont ignorées et l'exécution passe à l'instruction de la ligne de programme numérotée à la suite.

Lorsque l'instruction IF se présente dans un programme à exécution immédiate, si \expr \ est zéro, APPLESOFT va ignorer la totalité du reste du programme.

Si expr est une expression arithmétique impliquant des expressions chaînes et des opérateurs logiques chaînes, expr est évalué par comparaison du rangement alphabétique des expressions chaînes comme déterminé par les codes ASCII pour les caractères impliqués (voir appendice K).

Les instructions de la forme

IF expr THEN

sont valides: on n'obtient pas de message d'erreur. Un THEN sans un IF correspondant ou un IF sans un THEN amène le message.

? SYNTAX ERROR

APPLESOFT n'a pas été étudié ou prévu pour permettre à l'expression de l'instruction IF d'être une expression chaîne, mais les variables chaînes et les chaînes peuvent s'utiliser pour expr sous les strictes conditions suivantes:

Si expr est une expression chaîne d'un type quelconque alors \expr \ est non-zéro, même si expr est une variable chaîne à laquelle on a assigné aucune valeur ou une valeur zéro ou la chaîne nulle, "" Toutefois, la chaîne nulle littérale comme dans

IF""THEN . . .

est évaluée à zéro.

IF chaîne THEN, si on l'exécute plus de deux ou trois fois dans un programme donné, amène le message

? FORMULA T00 COMPLEX ERROR

si expr est une variable chaîne et si l'instruction précédente a assigné la chaîne nulle à toutes les variables chaînes, alors expr est évalué pour zéro. Par exemple le programme:

```
120 IF A$ THEN PRINT "A$"
130 IF B$ THEN PRINT "B$"
140 IF X$ THEN PRINT "X$"
```

lorsqu'on le fait passer, imprime

```
A$
B$
X$
```

parce que les chaînes AS, BS et XS sont évaluées comme non-zéro. Toutefois le fait d'ajouter la ligne

```
100 QS = ""
```

fait que les trois chaînes sont évaluées pour zéro et qu'il n'y a pas de sortie imprimée. Le fait d'effacer la ligne 100 ou d'ajouter une ligne 110 du genre de

```
110 F=3
```

fait à nouveau que les trois chaînes sont évaluées comme non-zéro.

Avant THEN, la lettre A cause des problèmes d'analyse

```
IF BETA THEN 230
```

s'analyse comme

```
IF BET AT HEN230
```

qui amène un message d'erreur

```
? SYNTAX ERROR
```

à l'exécution.
Les instructions suivantes sont équivalentes:

```
IF A=3 THEN 160
IF A=3 GOTO 160
IF A=3 THEN GOTO 160
```

FOR

```
For imm & dif
FOR real avar = aexprl TO aexpr2 [STEP aexpr3]
```

\avar\ est positionné à \aexprl \ et les instructions qui suivent le FOR sont exécutées jusqu'à ce que l'on rencontre une instruction NEXT avar où avar est le même nom que celui qui apparaît dans l'instruction FOR.
Puis \ avar \ est incrémenté de la valeur de \ aexpr3 \ (la valeur par défaut de \ aexpr3 \ est 11. Puis \ avar \ est comparé à aexpr2 et si \ avar \ > \aexpr2, \ l'exécution se poursuit avec l'instruction qui suit le NEXT. Par contre si \avar \ <= \aexpr2, \ l'exécution se poursuit selon l'instruction qui suit le FOR.

Si \aexpr3,\ alors l'opération est légèrement différente après que l'on a ajouté \aexpr3\ à \ avar. \ Si avar < \aexp2, \ l'exécution se poursuit avec l'instruction qui suit le NEXT. Si \avar \ > = \aexpr2, \ l'exécution se poursuit selon l'instruction qui suit le FOR.

Les expressions arithmétiques qui forment les paramètres de la boucle FOR peuvent être des nombres réels, des variables réelles, des entiers ou des variables entières. Toutefois, real avar doit être une variable réelle. Un essai d'utiliser une variable entière pour avar amène le message

```
? SYNTAX ERROR
```


Du fait que \ avar \ n'est incrémenté et comparé à \ aexpr \ qu'à la fin de la boucle FOR . . . NEXT, la partie du programme qui se trouve à l'intérieur de la boucle est donc toujours exécutée au moins une fois. Les boucles FOR . . . NEXT ne doivent pas se chevaucher l'une l'autre. Dans ce cas on aurait le message

```
? NEXT WITHOUT FOR ERROR
```

Si des boucles FOR sont emboîtées l'une dans l'autre avec plus de 10 niveaux de profondeur, on obtient le message

```
? OUT OF MEMORY ERROR
```

Pour exécuter une boucle FOR . . . NEXT en mode exécution immédiate, l'instruction FOR et l'instruction NEXT doivent être toutes les deux sur la même ligne lune ligne peut avoir jusque 239 caractères de long).

Si la lettre A est utilisée immédiatement avant T0, ne laissez pas d'espace entre T et 0. FOR I=BETA TO 56 est bon, mais FOR I=BETA TO 56 s'analyse comme FOR I=BET ATO 56 et donne le message '

```
? SYNTAX ERROR
```

chaque boucle active FOR. . . NEXT utilise 16 octets en mémoire.

NEXT

```
NEXT imm & dif
NEXT [avar]
NEXT avar [{ avar} ]
```

Constitue la fin d'une boucle FOR . . . NEXT. Lorsque l'on rencontre un NEXT le programme soit l'ignore, soit se branche sur l'instruction qui suit le FOR correspondant, selon les conditions que l'on a expliquées dans la discussion concernant l'instruction FOR.

Des avars multiples doivent être spécifiés dans l'ordre correct de façon que les boucles FOR . . . NEXT soient emboîtées l'une dans l'autre et ne se chevauchent pas. Des avars dans un ordre incorrect donnent le message

```
? NEXT WITHOUT FOR ERROR.
```

Une instruction NEXT dans laquelle on ne spécifie pas de nom de variable se réfère par défaut à la boucle FOR la plus récemment entrée qui soit encore en effet. S'il n'y a pas d'instruction FOR du même nom de variable qui soit encore en effet ou s'il n'y a pas d'instruction FOR, d'un nom quelconque, en effet, lorsque l'on rencontre un NEXT sans nom, on obtient le message

```
? NEXT WITHOUT FOR ERROR.
```

NEXT sans avar s'exécute plus rapidement que ne le fait NEXT avar.

En mode d'exécution immédiate, l'instruction FOR et son instruction correspondante NEXT doivent être exécutées toutes les deux sur la même ligne.
Si une instruction FOR à exécution différée est encore en effet, une instruction NEXT à exécution immédiate peut causer un saut au programme à exécution différée s'il y a lieu. Toutefois, l'instruction FOR a été exécutée en exécution immédiate, une instruction NEXT qui se trouve sur une ligne différente à exécution immédiate va amener le message

```
? SYNTAX ERROR
```

à moins qu'il n'y ait pas de lignes intermédiaires et que NEXT soit seul et sans nom.

```
]FOR I = 1 à 5 : PRINT I
1
]NEXT
2
]NEXT
3
]NEXT I
```

?SYNTAX ERROR IN xxxx (XXXX est un numéro de ligne quelconque.)

GOSUB

GOSUB imm & dif
GOSUB linenum

Le programme se branche à la ligne indiquée. Lorsqu'on rencontre l'instruction RETURN, le programme se branche sur l'instruction qui suit immédiatement l'instruction GOSUB exécutée en dernier lieu.

Chaque fois que l'on exécute une instruction GOSUB, l'adresse de l'instruction suivante est mémorisée en tête d'une pile de ces adresses, pour que le programme puisse ensuite retrouver son chemin. Chaque fois que l'on exécute une instruction RETURN ou POP, l'adresse de tête de la pile RETURN est enlevée.

Si le linenum indiqué ne correspond pas à une ligne de programme existante, on obtient le message d'erreur ?UNDEF'D STATEMENT ERROR IN linenum, où linenum indique la ligne de programme qui contient l'instruction. La portion du message

IN linenum

est omise si GOSUB est utilisé en mode exécution directe.

Si les GOSUB sont emboîtés à plus de 25 niveaux de profondeur, on obtient le message

?OUT OF MEMORY ERROR

Chaque instruction active GOSUB (c'est-à-dire qui n'a pas encore été suivie d'une instruction RETURN) utilise 6 octets de mémoire.

RETURN

RETURN Imm & dif

II n'y a pas de paramètre ni d'options dans cette instruction.
C'est un branchement à l'instruction qui suit immédiatement le GOSUB exécuté en dernier lieu. L'adresse de l'instruction à laquelle on se branche est celle qui se trouve en tête de la pile RETURN (voir GOSUB et POP).

Si un programme rencontre des instructions RETURN une fois de plus qu'il n'a rencontré des instructions GOSUB, on obtient le message

?RETURN WITHOUT GOSUB ERROR.

POP

POP imm & dif

II n'y a pas de paramètre ni d'options associés à POP. Un POP a l' effet d'un RETURN sans le branchement. Le prochain RETURN que l'on rencontrera, au lieu d'effectuer un branchement à une instruction qui se trouve au-delà du GOSUB rencontré en dernier lieu, effectuera un branchement sur une instruction qui se trouve au-delà du second GOSUB exécuté en dernier lieu. On l'appelle un POP car il fait sauter (POP) une adresse de la tête de la pile d'adresses RETURN.

Si l'instruction POP est exécutée avant que l'on ne rencontre un GOSUB, on obtient le message?

RETURN WITHOUT GOSUB ERROR.

ON . . . GOTO

ON . . . GOSUB dif

car il n'y a pas d'adresse de retour sur la pile. ON aexpr GOTO linenum 1 [linenum1 i ON aexpr GOSUB linenum (f, linenum1)

ON . . . GOTO effectue un branchement sur la ligne, dont le numéro est spécifié pas l'item \ aexpr \, de la liste des linenums après le GOTO. ON . . . GOSUB travaille de façon semblable, mais on exécute alors un GOSUB plutôt qu'un GOTO.

Si aexpr est zéro ou supérieur au nombre de linenums possibles de la liste, mais inférieur à 256, alors l'exécution du programme se poursuit jusqu'à l'instruction suivante:
aexpr doit être sur l'étendue 0 à 255 pour éviter le message

?ILLEGAL QUANTITY ERROR

ONERR GOTO

ONERR GOTO linenum

Si une erreur se produit on peut utiliser ONERR GOTO pour éviter d'avoir un message d'erreur et un arrêt de l'exécution. Cette instruction positionne un indicateur qui cause un saut inconditionnel (si une erreur se produit ensuite sur le programme) sur la ligne de programme indiquée par linenum. POKE 216,0 ramène à l'origine l'indicateur de détection d'erreur de sorte que les messages d'erreur apparaîtront normalement.
L'instruction ONERR GOTO doit être exécutée avant l'apparition d'une erreur pour éviter l'interruption du programme.
Si une erreur se produit dans le programme, le code du type d'erreur est mémorisé à l'emplacement décimal 222 de la mémoire. Pour voir quelle erreur on a rencontrée, utilisez l'instruction PRINT PEEK (222).

Code	**Message d'erreur**	**Code**	**Message d'erreur**
0	NEXT sans FOR	120	tableau redimensionné
16	Syntaxe	133	division par zéro
22	RETURN sans GOSUB	163	pas de concordance de frappe
42	En dehors des données	176	Chaîne trop longue
53	Quantité illégale	191	Formule trop complexe
69	Dépassement	224	fonction non définie
77	en dehors de la mémoire	254	mauvaise réponse à l'instruction INPUT
90	instruction non définie	255	essai d'interruption par Ctrl C
107	indice erroné		

Il faut faire attention pour le traitement des erreurs qui se produisent à l'intérieur des boucles FOR . . . NEXT ou entre GOSUB et RETURN POP, car les pointeurs et les piles RETURN POP sont perturbés. La routine de traitement des erreurs doit recommencer la boucle, en repartant à l'instruction FOR ou GOSUB et non à l'instruction NEXT ou RETURN POP. Après le traitement de l'erreur, en arrivant à un NEXT ou à un RETURN POP, on aura le message approprié

?NEXT WITHOUT FOR ERROR ON ?RETURN WITHOUT GOSUB ERROR

Si on utilise ONNER GOTO avec RESUME pour traiter les erreurs dans une instruction GET, le programme sera suspendu s'il y a deux erreurs GET consécutives sans un GET réussi intermédiaire. Pour échapper à la suspension de programme, utilisez reset ctrl C return. Si GOTO termine la routine de traitement de l'erreur, tout marche bien (mais voir note suivante).

Lorsqu'on l'utilise en mode TRACE ou dans un programme contenant une instruction PRINT, ONERR cause un saut au moniteur lorsque l'on a rencontré 43 erreurs. Là où ces erreurs sont générées par une instruction INPUT, tout marche bien si on utilise RESUME; mais si GOTO termine la routine de traitement d'erreur, la 87e erreur INPUT cause un saut au moniteur. A nouveau reset ctrl C return vous ramènera à APPLESOFT.

Si vous êtes gêné par l'un des problèmes dont nous venons de discuter, exécutez un CALL pour appeler le sous-programme suivant de langage assembleur faisant partie de votre routine de traitement d'erreur.

Dans le moniteur entrez les données en hexadécimal: 68 A8 68 A6 DIF 9A 48 60 ou en APPLESOFT, entrez les données en décimal: 104 168 104 166 223 154 72 152.

Par exemple en APPLESOFT vous pourriez écrire (instruction POKE) les nombres décimaux aux emplacements 768. 777.
Puis vous utiliseriez CALL 78 dans votre routine de traitement d'erreur.

RESUME

RESUME dif

Lorsqu'on l'utilise à la fin d'une routine de traitement d'erreur, cette instruction fait que le programme reprend l'exécution au début de l'instruction où une erreur s'est produite.

Si RESUME se rencontre avant qu'une erreur ne se produise, le message

?SYNTAX ERROR IN 65278

peut arriver ou tout autre événement étrange. Habituellement votre programme sera arrêté ou il sera suspendu.

Si une erreur se produit dans une routine de traitement d'erreur, l'emploi de RESUME va placer le programme dans une boucle sans fin. Pour en échapper utilisez reset ctrl C return.

En mode d'exécution immédiate, cette instruction peut mettre le système en suspension, peut amener une erreur de syntaxe ou peut commencer l'exécution d'un programme existant ou même d'un programme effacé.

CHAPITRE 8
Commandes graphiques et commandes de jeux

84 TEXT

Graphiques basse résolution
84 GR
85 COLOR
85 PLOT
86 HLIN
86 VLIN
87 SCRN

Graphiques haute résolution
87 HGR
88 HGR2
89 HCOLOR
89 HPLOT

Game Controls
90 PDL

TEXT

TEXT imm & dif
TEXT imm & dif

Pas de paramètres. Positionne l'écran en mode texte habituel, sur toute la largeur de l'écran (40 caractères par ligne, 24 lignes) à partir du mode graphique basse résolution ou de l'un des deux modes graphiques haute résolution. Le caractère U) et le curseur sont amenés à la dernière ligne de l'écran. En édition en mode texte, TEXT est équivalent à VTAB 24.
Une instruction comme

175 TEXTILE = 127

amène l'exécution du mot réservé avant l'apparition du message d'erreur ?SYNTAX ERROR. Si la fenêtre de texte a été positionnée à une largeur autre que la pleine largeur (voir appendix J), TEXT la ramène à la pleine largeur de l'écran.

GR

GR imm & dif
GR

Pas de paramètres. Cette instruction positionne le mode graphique basse résolution (40 par 401 pour l'écran, en laissant quatre lignes pour le texte en bas de l'écran. L'écran est effacé pour devenir noir et le curseur est déplacé dans la fenêtre de texte. Peut être converti en graphique pleine largeur (40 par 48) après exécution de GR avec l'instruction

POKE - 16302,0

où l'instruction équivalente

POKE 49234,0

Si GR suit une instruction POKE pour la totalité de l'écran, c'est le mode mixte graphique plus texte qui est obtenu.
Après une commande GR, la couleur (instruction COLOR) est positionnée à zéro.

Si le mot réservé GR est utilisé comme premiers caractères d'un nom de variable, l'instruction GR peut être exécutée avant de recevoir le message d'erreur

?SYNTAX ERROR

par exemple l'exécution de l'instruction

GRIN = 5

vous laisse avec un écran noir inattendu.

Si elle est exécutée pendant que HGR est en effet, l'instruction GR se comporte normalement. Au contraire si elle est exécutée pendant que c'est HGR2 qui est en effet, GR efface le contenu de l'écran habituellement en mémoire mais vous laisse regarder à la page 2 du graphique basse résolution et du texte. Pour revenir en mode normal, il suffit de taper TEXT. Dans les programmes, utiliser TEXT avant de commuter de HGR2 à GR.

COLOR

COLOR imm & dif
COLOR = aexpr.

Définit la couleur pour le tracé en mode graphique basse résolution. Si /aexpr/ est un nombre réel, il est converti en un entier. L'étendue des valeurs /aexpr/ va de o à 255; elles sont traitées modulo 16.

Les noms des couleurs et leurs nombres associés sont:

0 noir	4 vert foncé	8 brun	12 vert
1 magenta	5 gris	9 orange	13 jaune
2 blue foncé	6 blue moyen	10 gris	14 aqua/turquoise claire
3 pourpre	7 blue clair	11 rose	15 blanc

COLOR est positionné à zéro par l'instruction GR.

Pour connaître la couleur d'un point donné de l'écran, utiliser l'instruction SCRN.

Lorsqu'elle est utilisée en mode TEXT, COLOR est un facteur pour déterminer quel caractère est placé sur l'écran par une instruction PLOT. Lorsqu'elle est utilisée en mode graphique haute résolution, l'instruction COLOR est ignorée.

PLOT

PLOT imm & dif
PLOT aexprl. aexpr2

En mode graphique basse résolution, cette instruction place un point de coordonnée X \ aexpr 1 \ et de coordonnée y \ aexpr2 \ . La couleur de ce point est déterminée par l'instruction exécutée la plus récente (COLOR=0 s'il n'y a pas de spécification précédente).

\aexprl \ doit être sur l'étendue 0 à 39 et \aexpr2 \ doit être sur l'étendue 0 à 47 sinon on reçoit le message
?ILLEGAL QUANTITY ERROR

Une tentative d'utiliser l'instruction PLOT (tracé) pendant que le système est en mode TEXT ou en mode mixte graphique plus texte avec \aexpr \ sur l'étendue 40 à 47 se traduit par le fait qu'un caractère est placé à l'endroit où le point de couleur était apparu (un caractère occupe l'emplacement de deux points graphiques basse résolution empilés verticalement).

Cette instruction n'a pas d'effet visible si on l'utilise en mode graphique haute résolution HGR2, même si elle est précédée d'une instruction GR, du fait que l'écran n'est pas en train de regarder la portion graphique basse résolution (page un) de la mémoire.

L'origine (0,0) de tous les graphiques est l'angle supérieur gauche de l'écran.

HLIN

HLIN imm & dif
HLIN aexprl, aexpr2 AT aexpr3
Utilisé en mode graphique basse résolution, HLIN dessine une ligne droite allant de \aexprl \, \aexpr3 \ à \aexpr2 \, \aexpr3. La couleur est déterminée par l'instruction COLOR exécutée la plus récente.

\aexpr 1 \ et aexpr2\ doivent être sur l'étendue 0 à 39 et \aexpr3\ doit être sur l'étendue 0 à 47 sinon on reçoit le message
?ILLEGAL QUANTITY ERROR
\aexpr 1 \ peut être supérieur, égal ou inférieur à /aexpr2 \.

Si HLIN est utilisé lorsque ce système est en mode TEXT ou en mode mixte graphique plus texte, avec \aexpr3 \ sur l'étendue 40 à 47, alors une ligne de caractères se placera là où la droite de points graphiques aurait dû être tracée. (Un caractère occupe l'emplacement de deux points basse résolution empilés verticalement).

L'instruction n'a pas d'effet visible si on l'utilise en mode graphique haute résolution.

Notez que le 'H' de cette instruction se réfère à "horizontal" et non à "haute résolution". A l'exception de HLIN et HTAB, le préfixe "H" se réfère aux instructions haute résolution.

VLIN

VLIN imm & dif
VLIN aexprl, aexpr2, AT aexpr3

En mode graphique basse résolution, cette instruction dessine une droite verticale allant de (\aexpr1 \ \aexpr3 \) à \aexpr2 \, \aexpr3 \ . La couleur est déterminée par l'instruction COLOR exécutée la plus récente.

\aexpr1 \ et \aexpr2 \ doivent être sur l'étendue 0 à 47, \aexpr3 \ doit être sur la place 0 à 39, sinon on recoit le message
?ILLEGAL QUANTITY ERROR.

\aexprl \ peut être supérieur, égal ou inférieur à \aexpr2.

Si le système est en mode TEXT lorsqu'on utilise VLIN ou en mode mixte graphique plus texte, avec \aexpr2 \ sur l'étendue 40 à 47, la portion de la droite qui se trouve à l'intérieur de la zone de texte apparaîtra sous forme d'une ligne de caractères, placée là où les points du graphique auraient été tracés.

Cette instruction n'a pas d'effet visible si on l'utilise en mode graphique haute résolution.

SCRN

SCRN imm & dif
SCRN (aexpr1, aexpr2)

En mode graphique basse résolution, la fonction SCRN fournit le code couleur du point dont la coordonnée X est \ aexpr 1 \ et dont la coordonnée y est \ aexpr2 \

Bien que le mode graphique basse résolution trace des points se trouvant aux positions de l'écran (x, y) où X est sur l'étendue 0 à 39 et y sur l'étendue 0 à 47, la fonction SCRN accepte a la fois des valeurs de X et y sur l'étendue 0 à 47. Par contre, si SCRN est utilisé avec une valeur X (\aexpr1 \) sur l'étendue 40 à 47, le nombre fourni donne la couleur au point où la coordonnée X est (\ aexpr \-40) et dont la coordonnée y est (\ aexpr2 \ + 16) Si (\aepr2 \ + 16) est sur l'étendue 39 à 47, en mode normal mixte graphique plus texte, le nombre fourni par SCRN est relatif au caractère du texte qui se trouve à la position de la zone de texte située en-

dessous de la portion graphique de l'écran. Si (\ aexpr2 \ + 16) est dans la page 48 à 63, SCRN fournit un nombre qui n'a pas de relation avec l'écran.

En mode TEXT, SCRN fournit le nombre sur l'étendue 0 à 15 dont la valeur est les quatre bits d'ordre supérieur, si aexpr2 est impair; ou les quatre bits d'ordre inférieur, si aexpr2 est pair du caractère qui se trouve à la position de caractère (aexpr 1 + 1, INT «aexpr2+ 1) \ 211. Par conséquent l'expression CHR $(SCRN(X-1, 2*(Y1))+16*SCRN(X-1,2*(Y-1)+1)) va donner le caractère qui se trouve à la position de caractère (X, Y).

En mode graphique haute résolution SCRN à regarder la zone graphique basse résolution et donc le nombre que fournit SCRN n'est pas relatif à l'affichage haute résolution.

SCRN s'analyse comme mot réservé uniquement si le caractère suivant qui ne soit pas un espace blanc est une parenthèse gauche.

HGR

HGR imm & dif

Pas de paramètres. Positionne le mode graphique haute résolution (280 par 160) pour l'écran, en laissant quatre lignes pour le texte en bas de l'écran, L'écran est effacé pour passer au noir et la page 1 de la mémoire (8K-16K) est affichée. HCOLOR n'est pas modifié par cette instruction. La mémoire de la partie de l'écran réservée au texte n'est pas affectée. L'instruction HGR laisse la fenêtre de texte sur la totalité de l'écran mais seules les quatre lignes de texte inférieures sont visibles en-dessous du graphique Le curseur va encore être dans la fenêtre de l'écran mais peut n'être pas visible à moins qu'on ne le déplace vers l'une des quatre lignes inférieures.
On peut convertir l'écran en raphique sur la totalité de l'écran (280 par 192) après exécution de HGR au moyen de l'instruction POKE

POKE - 16302,0

ou en utilisant

POKE 49234,0

qui est équivalent. Si HGR suit l'une ou l'autre des instructions POKE ci-dessus, on repositionne le mode mixte graphique haute résolution plus texte.

Si le mot réservé HGR est utilisé pour constituer les premiers caractères d'un nom de variable, l'instruction HGR peut être exécutée avant que n'apparaisse le message

?SYNTAX ERROR.

C'est ainsi que l'exécution d'instruction

HGRIP=4

se traduit par un déplacement inattendu en mode graphique haute résolution qui peut effacer votre programme.

Un très long programme qui s'étend au-delà de l'emplacement de mémoire 8192 peut être partiellement effacé lorsque vous exécutez HGR ou il peut venir "écrire" sur votre page 1 de l'affichage graphique haute résolution. En particulier, les données en chaîne sont mémorisées en tête de la mémoire; sur les systèmes à faible capacité de mémoire (16K ou 20K) ces données peuvent résider en page 1 du graphique haute résolution. Positionner HIMEM:8192 pour protéger votre programme et la page 1 du graphique haute résolution.

HGR2

HGR2 imm & dif
HGR

Pas de paramètres. Cette instruction positionne le mode graphique haute résolution sur la totalité de l'écran 1280 par 192). L'écran est effacé pour revenir au noir et la page 2 de la mémoire (16K-24K) est affichée. La mémoire de l'écran correspondant au texte n'est pas affectée. Cette page de mémoires (et donc l'instruction HGR2) n'est pas disponible si votre système contient moins de 24K de mémoire.

Sur les systèmes qui le permettent, l'utilisation de HGR2 au lieu de HGR rend maximum l'espace de mémoire disponible pour les programmables.
Sur les systèmes à mémoire de 24K, positionnez HIMEM: 16384 pour protéger la page 2 du graphique haute résolution de votre programme (en particulier les chaînes qui sont mémorisées en tête de la mémoire).

Si l'on utilise le mot réservé HGR2 pour constituer les premiers caractères d'un nom de variable, l'instruction HGR2 peut s'exécuter avant que n'apparaisse le message

?SYNTAX ERROR.

Lorsqu'on l'exécute, une instruction comme

140 IF X > 150 THEN HGR2 PIECES = 12

laisse l'écran soudainement vierge, avec le risque que les parties supérieures du programme ne soient effacées.

L'instruction

POKE - 16301,0

convertit tout mode graphique plein écran en mode mixte graphique plus texte. Si cette instruction est émise après HGR2 toutefois, les quatre lignes de texte sont prises sur la page 2 du texte, qui n'est pas facilement accessible à l'utilisateur.

HCOLOR

HCOLOR imm & dif
HCOLOR = aexpr.

Définit la couleur du graphique haute résolution comme étant celle spécifée parla valeur de HCOLOR, qui
doit être dans l'étendue 0 à 7 inclusivement. Les noms de couleur et leurs valeurs associées sont

0 noir 1	4 noir 2
1 vert (selon votre TV)	5 selon votre TV
2 bleu (selon votre TV)	6 selon votre TV
3 blanc 1	7 blanc 2.

Un point haute résolution tracé avec l'instruction HCOLOR=3 (blanc) sera bleu si la coordonnée X du point est paire, vert, si la coordonnée X est impaire et blanc uniquement si les deux (X, y) et (X+ 1, y) sont traces. Ceci est d6 à la façon dont travaille votre appareil TV.

HCOLOR n'est pas modifié par HGR, HGR2, OU RUN. Jusqu'à ce que la première instruction HCOLOR soit exécutée, la couleur de tracé du graphique haute résolution est indéterminée.
Si on l'utilise pendant que l'on est en mode graphique basse résolution, n'affecte pas la couleur à afficher.

HPLOT

HPLOT imm &, dif'
PLOT aexpr 1, aexpr2
HPLOT TO aexpr3, aexpr4
HPLOT aexpr 1, aexpr2 TO aexpr3, aexpr4 [{ TO aexpr, aexpr 1 }]

HPLOT avec la première option trace un point haute résolution dont la coordonnée X est \aexpr 1 \ et dont la coordonnée y est \aexpr2\ . La couleur du point est déterminée par l'instruction HCOLOR exécutée la plus récente. La valeur de HCOLOR indéterminée si elle n'a pas été précédemment spécifiée.

La seconde option amène le tracé d'une droite partant du dernier point tracé jusque (\aexpr3\, \aexpr4 \). La couleur de cette ligne est déterminée par la couleur du dernier point tracé, même si la valeur de HCOLOR a été modifiée depuis ce tracé précédent. Si aucun point précédent n'a été tracé, aucune droite n'est tirée.

Si l'on utilise la troisième option, une droite est tracée de (\aexpr1 \, \aexpr2 \) à (\aexpr3 \, \aexpr4 \), avec la couleur spécifier par l'instruction HCOLOR la plus récente. La droite ainsi tracée peut se prolonger dans la même instruction presque indéfiniment (sous réserve des limites de l'écran et de la limite d'instruction de 239 caractères) en prolongeant cette instruction avec
to aexpr5, aexpr6 to aexpr7, aexpr8 et ainsi de suite. L'unique instruction
HPLOT, 0,0 to 279, to 279, to 0, 159 to 0,
peut tracer un bord rectangulaire tout autour des quatre côtés de l'écran haute résolution.

HPLOT doit être précédé de HGR ou HGR2 pour éviter de démolir des parties de la mémoire, y compris votre programme et vos variables.
\aexpr1 \ et \aexpr3 \ doivent être sur l'étendue 0 à 279.
\aexpr2 \ et \aexpr4 \ doivent être dans l'étendue 0 à 191.
\aexpr1 \ peut être supérieur, égal ou inférieur à \aexpr3. \aexpr2 \ peut être supérieur, égal ou
inférieur à \aexpr4.

Un essai de tracé à un point dont les coordonnées dépassent ces limites amène le message
?ILLEGAL QUANTITY ERROR.
Si l'écran se trouve en mode mixte graphique haute résolution plus quatre lignes de texte, un essai de tracé des points avec des coordonnées y dans l'étendue 160 à 191 n'a pas d'effet visible.

PDL

PDL imm & dif
PDL (aexpr)
Cette fonction fournit la valeur en cours, de 0 à 255, de la commande de jeux (ou PaDdLe) spécifiée par \ aexpr \ si \ aexpr \ est dans l'étendue 0 à 3. La commande de jeux est une résistance variable de 0 à 150K ohms.

Si deux commandes de jeux sont lues dans des instructions PDL consécutives, la lecture de la seconde commande de jeux peut être affectée par la lecture de la première. Pour avoir des lectures plus précises, il faut laisser plusieurs lignes de programme entre les instructions PDL ou placer une boucle à faible retard (FOR I=1 TO 10:NEXT I) entre les instructions PDL.

Si \ aexpr \ est négative ou inférieure à 225, on obtient le message
?ILLEGAL QUANTITY ERROR.

Si \ aexpr \ est sur l'étendue 4 à 225, la fonction PDL fournit un nombre plutôt imprévisible de 0 à 255 et peut causer différents effets par ailleurs, dont certains peuvent perturber l'exécution du programme.
Par exemple si \aexpr\ est sur l'étendue 204 à 219, l'emploi de la fonction PDL est, de façon fréquente et plutôt aléatoire, accompagnée d'un cliquetis du hautparleur de l'ordinateur.

Si N est sur l'étendue 236 à 239, PDL(N) peut résulter en une instruction
POKE - 16540+N
de sorte que PDL(236) peut positionner le mode graphique, que PDL(237) peut positionner le mode texte, etc. (voir appendix J).

En plus de la lecture des positions de quatre commandes variables de jeux à l'aide de PDL, APPLESOFT peut lire l'état de 3 boutons de commande de jeux (contacteurs on-off), à l'aide des différentes instructions PEEK et peut mettre en circuit et hors circuit 4 sorties de jeux (contacteurs TTL) à l'aide des différentes instructions POKE (voir appendix J).

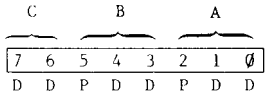
CHAPITRE 9
Formes haute résolution

Comment créer un tableau de forme

APPLESOFT possède cinq instructions spéciales qui vous permettent de manipuler des formes en graphiques haute résolution: DRAW, XDRAW, ROT, SCALE et SHLOAD. Avant de pouvoir utiliser ces instructions APPLESOFT, une forme doit être définie par une "définition de forme". Cette définition de forme consiste en une séquence de vecteurs de tracé qui sont mémorisés en une série d'octets dans la mémoire de APPLE. Une ou plusieurs de ces définitions de forme avec leur index, constituent un "tableau de forme" que l'on peut créer à partir du clavier et sauvegarder sur un disque ou un ruban sur cassette pour réemploi.

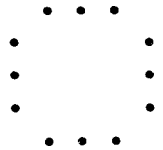
Chaque octet de la définition de forme est divisé en trois sections et chaque section peut spécifier un "vecteur de tracé": s'il y a ou non à tracer un point, ainsi que la direction de dépassement (vers le haut, vers le bas, vers la gauche ou vers la droite). DRAW et XDRAW avancent section par section à travers chaque octet de la définition de forme, depuis le premier octet de la définition jusqu'à son dernier octet.

Lorsque l'on atteint un octet qui ne contient que des zéro, la définition de forme est terminée.
Voici comment les trois sections, A, B et C sont disposées dans l'un des octets qui constituent une définition de forme:
Section

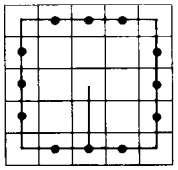


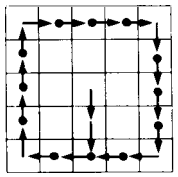
Numéro de bit Spécifie

Chaque paire de bit DD spécifie une direction dans laquelle il faut se déplacer et chaque bit P spécifie s'il faut ou non tracer un point avant de se déplacer, comme suit:
Si DD = 00 déplacement vers le haut
Si DD = H 1 déplacement vers la droite Si P = 0 pas de tracé
Si DD = 10 déplacement vers le bas Si P = 0 1 tracé.
Si DD = 1 1 déplacement vers la gauche
Notez que la dernière section, C (les deux bits de poids fort) ne possède pas de champ P (par défaut P = 01 de sorte que la section C ne peut spécifier qu'un déplacement sans tracé.
Chaque octet peut représenter jusque trois vecteurs de tracé, un dans la section A, un dans la section B, et un troisième (déplacement seulement) dans la section C.
DRAW et XDRAW font progresser les sections de la droite vers la gauche (bit de poids faible vers bit de poids fort: section A, puis B, puis CI. A chaque section de l'octet, si toutes les sections restantes de l'octet ne contiennent que des zéros, alors ces sections sont ignorées. Donc l'octet ne peut se terminer avec un déplacement dans la section C de 00 (c'est-à-dire un déplacement vers le haut sans tracé) car cette section, ne contenant que des zéros, sera ignorée. De même si la section C est 00 (ignorée), la section B ne peut pas être un déplacement de 000 car elle serait également ignorée. Et un déplacement de 000 dans la section A terminera votre définition de forme à moins qu'il n'y ait un bit de valeur 1 quelque part dans la section B ou C.
Supposons que vous voulez dessiner une forme comme celle-ci:



Tout d'abord dessinez-là sur un papier graphique, un point par carré. Puis décidez où il faut commencer à dessiner la forme. Démarrons cette forme au centre. Puis dessinez un chemin passant par chaque point de la forme, en utilisant uniquement des angles à 90° aux virages:





Puis redessinez la forme comme série de vecteurs de tracé, chacun représentant un déplacement d'une place vers le haut, vers le bas, vers la droite et vers la gauche, et distinguez les vecteurs qui tracent un point avant le déplacement la gauche, et distinguez les vecteurs qui tracent un point avant le déplacement (un point rond repère les vecteurs qui tracent les points).

Puis "déroulez" ces vecteurs et écrivez-les sur une ligne droite:

Puis dessinez un Tableau comme celui de la Figure 1 ci-dessous:

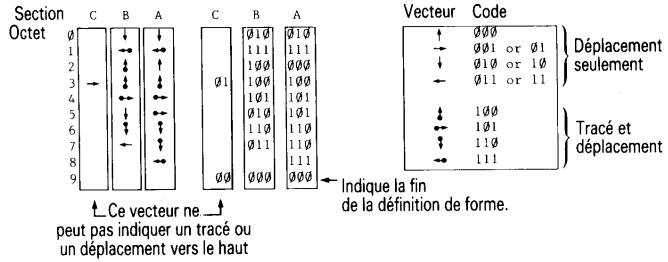


Figure 1

Pour chaque vecteur de la ligne, déterminez le code en bits et placez le dans la section suivante disponible du tableau. Si le code ne s'adapte pas (par exemple le vecteur de la section C ne peut pas tracer un point) ou s'il y a un 00 (ou 000) à la fin d'un octet, alors sauter cette section et passer à la suivante. Lorsque vous avez terminé de coder tous vos vecteurs, vérifier votre travail pour voir si c'est bien précis.

Maintenant faites un autre tableau comme indiqué sur la figure 2, ci-dessous, et recopier les codes de vecteurs à partir du premier tableau. Recodez l'information concernant les vecteurs en une série d'octets hexadécimaux, à l'aide du code hexadécimal de la figure 3.

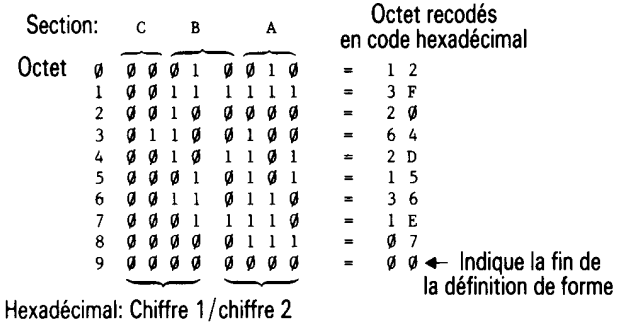


Figure 2

Binaire		Hexadécimal.
ØØØØ	=	Ø
ØØØ1	=	1
ØØ1Ø	=	2
ØØ11	=	3
Ø1ØØ	=	4
Ø1Ø1	=	5
Ø11Ø	=	6
Ø111	=	7
1ØØØ	=	8
1ØØ1	=	9
1Ø1Ø	=	A
1Ø11	=	B
11ØØ	=	C
11Ø1	=	D
111Ø	=	E
1111	=	F

Figure 3

La série d'octets hexadécimaux à laquelle vous arrivez figure 2 constitue la définition de forme. Il y a encore un peu d'information supplémentaire que vous devez donner avant d'avoir un tableau de forme complet. La figure 4 de la page suivante représente le format d'un tableau de forme, complet avec son index.
Pour cet exemple, votre index est facile: il n'y a qu'une seule définition de forme. L'emplacement de départ du tableau de forme dont nous avons appelé l'adresse S doit contenir le nombre de définitions de forme (entre 0 et 255) en hexadécimal. Dans ce cas, ce nombre est justement un. Nous allons placer notre définition de forme immédiatement en-dessous de l'index pour simplifier. Ceci signifie dans ce cas que la définition de forme va commencer à l'octet S+4: l'adresse de la définition de forme n-1 relative à S, est 4 (00 04, en hexadécimal). Par conséquent, l'octet S+2 de l'index doit contenir la valeur 04 et l'oc tel S+3 de l'index doit contenir la valeur 00. La figure 5 de la page suivante représente le tableau de forme complété pour cet exemple.

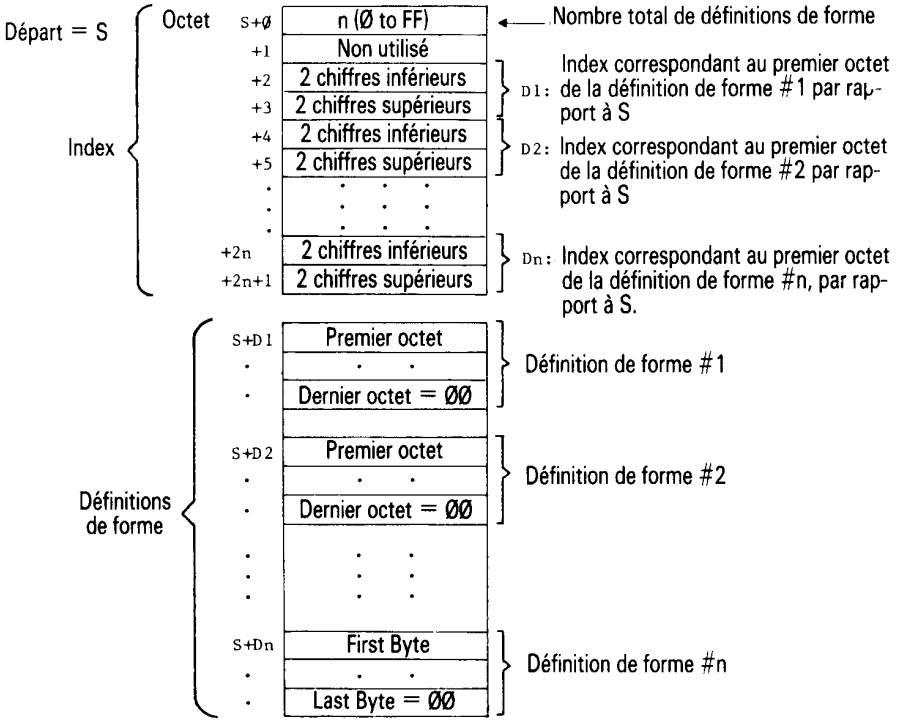


Figure 4

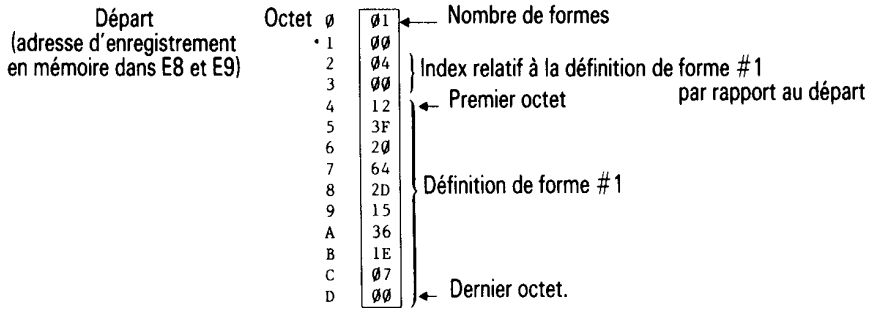


Figure 5

Vous êtes prêt maintenant à taper le tableau de forme dans la mémoire de APPLE. Premièrement choisissez une adresse de départ. Pour cet exemple nous utilisons l'adresse hexadécimale IDFC. (Note: cette adresse doit être inférieure à l'adresse de mémoires la plus élevée disponible dans votre système et ne doit pas se trouver dans une zone qui sera effacée lorsque vous utiliserez HGR ou HGR2. L'emplacement 1 DFC se trouve juste en-dessous du graphique haute résolution page 1 utilisé par HGR. Enfoncez la touche RESET pour entrer le programme Moniteur et tapez l'adresse de départ de votre tableau de forme:

1DFC

si vous enfoncez maintenant la touche RETURN, APPLE vous montrera l'adresse et le contenu de cette adresse. C'est-à-dire que vous examinez une adresse pour voir si vous y avez placé le nombre correct. Si à la place vous tapez un deux points (:1 suivi d'un nombre hexadécimal de deux chiffres, ce nombre sera enregistré en mémoire à l'adresse spécifiée lorsque vous enfoncerez la touche RETURN. Essayez ceci:

1DFC return.

Que donne APPLE comme contenu de l'emplacement 1 DFC ? Maintenant essayez ceci:

1DFC:01 return
1DFC return
1DFC-01

APPLE dit maintenant que la valeur 01 (hexadécimale) est mémorisée à l'emplacement dont l'adresse est 1 DFC. Pour mémoriser plus de nombres hexadécimaux de deux chiffres dans les octets successifs en mémoire, il suffit de déclarer la première adresse

1DFC

puis de taper les nombres, séparés par des espaces blancs:

1DFC:01 00 04 0012 3F 20 64 2D 15 36 1 E 07 00 return

Vous venez de taper votre premier tableau de forme complet . . . ce n'est pas si mal, n'est ce pas? Pour vérifier l'information qui se trouve dans votre tableau de forme, vous pouvez examiner chaque octet séparément ou simplement enfoncer la touche RETURN de façon répétitive jusqu'à ce que tous les octets intéressants let quelques uns supplémentaires, sans doute) aient été affichés:

```
1DFC return
1DFC-01
* return
00 04 00
* return
1E00- 12 3F 20 64 2D 15 36 1 E
* return
1E08- 07 00 DF 1 E 23 00 00 FF
```

Si votre tableau de forme vous semble correct, tout ce qui reste à faire est d'enregistrer l'adresse de départ du tableau de forme à un endroit où APPLESOFT peut la trouver (ceci se fait automatiquement si vous utilisez SHLOAD pour avoir APPLESOFT cherche les quatre chiffres hexadécimaux de l'adresse de départ du tableau aux emplacements hexadécimaux E8 (deux chiffres inférieurs) et E9 (deux chiffres supérieurs). Pour notre adresse de départ de tableau de ID FC, ceci va donner:

E8: FC 1 D return.

Pour protéger votre tableau de forme d'un effacement accidentel par votre programme APPLESOFT, cela pourrait être également une bonne idée de positionner HIMEM (dans les emplacements hexadécimaux 73 et 74) correspondant à l'adresse de départ du tableau:

73: FC 1 D

Ceci se fait également automatiquement lorsque vous utilisez SHLOAD pour avoir le tableau à partir de la bande sur cassette.

Pour sauvegarder un tableau de forme

Pour sauvegarder votre tableau de forme sur une bande, vous avez besoin de connaître trois choses:

1) adresse de départ du tableau 1DFC, dans votre exemple)
2) dernière adresse du tableau (1E09, dans votre exemple)
3) différence entre 2) et 1) (000D, dans votre exemple).

L'item 3, différence entre la dernière adresse et la première adresse du tableau doit être enregistré aux emplacements hexadécimaux 0 (deux chiffres inférieurs) et t (deux chiffres supérieurs):

0: 0D 00 return.

Maintenant vous pouvez écrire (enregistrer sur cassette) d'abord la longueur du tableau qui est enregistré aux emplacements 0 et 1, puis le tableau de forme lui-même qui est enregistré aux emplacements adresse de départ à dernière adresse:

0.1 W 1DFC. 1E09W.

N'enfoncez pas la touche return avant d'avoir placé une cassette dans votre enregistreur de cassette, l'avoir rembobinée et avoir commencé l'enregistrement (enfoncer PLAY et RECORD simultanément). Maintenant enfoncez la touche RETURN de l'ordinateur.
Pour utiliser la bande, rembobinez 1à, commencez à l'écouter (enfoncez) et (en APPLESOFT maintenant) tapez:

SHLOAD return.

Vous devriez entendre un bip lorsque la longueur du tableau a été lue avec succès et un autre bip lorsque le tableau lui même a été lu.

Pour utiliser un tableau de forme

Vous êtes prêt maintenant à écrire un programme APPLESOFT en utilisant les instructions de tableau de forme DRAW, XDRAW, ROT et SCALE.

Voici un programme échantillon APPLESOFT qui va imprimer notre forme définie, la faire tourner de 16° et la répéter, APPLESOFT répétition étant plus grande que la précédente.

```
10 HGR
15 POKE - 16302,0 ou 49234.0 20 HCOLOR = 3
30 FOR R = 1 TO 50
40 ROT = R
50 SCALE = R
60 DRAW 1 AT 139, 79
70 NEXT R
```

Pour voir un unique carré, ajouter une ligne

```
65 END.
```

Pour arrêter puis effacer chaque carré après qu'il ait été dessiné, ajoutez ces lignes

```
63 FOR I=0 TO 1000: NEXT I
65 XDRAW 1 AT 139, 79
```


DRAW

DRAW imm & dif -
DRAW aexpr l AT aexpr2, aexpr3 DRAW aexpr l

DRAW avec la première option dessine une forme en graphique haute résolution en commençant au point dont la coordonnée X est \aexpr2 \ et dont la coordonnée y est \aexpr3 \ . La forme dessinée est la définition de forme d'ordre \aexpr 1 \ du tableau de forme précédemment chargé à l'aide de l'instruction SHLOAD (ou un tableau de forme peut être dans la mémoire de APPLE en code hexadécimal en utilisant le programme Moniteur).

\ aexpr 1 \ doit être sur l'étendu 0 à n, où est le nombre (de 0 à 255) de définitions de forme donné dans l'octet 0 du tableau de forme. \aexpr2 \ doit être sur l'étendue 0 à 278. \aexpr3 \ doit être sur l'étendue 0 à 191. Si l'une ou l'autre de ces étendues est dépassée, on obtient le message ?

ILLEGAL QUANTITY ERROR.

La couleur, la rotation et l'échelle de la forme à dessiner doivent avoir été spécifiées avant exécution de l'instruction DRAW.

La seconde option est semblable à la première mais dessine la forme spécifiée en commençant au dernier point tracé par l'instruction HPLOT, DRAW ou XDRAW exécutée en dernier lieu.

Si l'instruction DRAW est émise quand il n'y a pas de tableau de forme dans l'ordinateur, elle peut amener le système en suspens. Pour le récupérer, utilisez reset ctrl C return. Cette instruction peut également dessiner des formes aléatoires sur toute la zone de graphique haute résolution de la mémoire, en risquant de détruire votre programme même si vous n'êtes pas en mode graphique.

XDRAW

XDRAW imm & dif
XDRAW aexpr l [AT aexpr2, aexpr3]

Cette instruction est la même que DRAW à l'exception que la couleur utilisée pour dessiner la forme est lle complément de la couleur déjà existante à chaque point tracé. Les couples de couleur suivants sont complémentaires :

noir et blanc
bleu et vert.

Le but de XDRAW est de fournir un moyen simple pour effacer: si vous appliquez l'instruction XDRAW à une forme vous effacerez la forme sans effacer le fond.

Voir les remarques de précautions à prendre pour DRAW.

ROT

ROT imm & dif
ROT = aexpr

Définit une position angulaire pour le forme à dessiner par les instructions DRAW ou XDRAW. La valeur de la rotation est spécifiée par \aexpr \, qui doit être entre 0 et 255.

ROT=0 fait que la forme à dessiner est orientée juste comme elle a été définie, ROT= 16 fait que la forme à dessiner tourne de 90° sens d'horloge, ROT=32 fait que la forme à dessiner tourne de 180° sens d'horloge, etc. Le processus se répète à partir de ROT= 64. Pour SCALE= 1 quatre valeurs de rotation seulement sont reconnues (0, 16, 32, 481; pour SCALE= 2, huit rotations sont reconnues, etc. Des valeurs de rotation non reconnues donnent à la forme à dessiner l'orientation de la dernière rotation reconnue la plus faible (habituellement).
ROT s'analyse comme mot réservé uniquement si le caractère suivant autre qu'un espace blanc est le signe de remplacement (=).

SCALE

SCALE imm & dif
SCALE = aexpr

Définit la dimension de l'échelle pour la forme à dessiner par les instructions DRAW ou XDRAW avec un facteur qui va de 1 (reproduction point pour point de la définition de formel à 255 (chaque vecteur est agrandi 255 fois) comme spécifié par \aexpr \ . NOTE: SCALE =0 est la taille maxima et non pas un simple point.
SCALE s'analyse comme mot réservé uniquement si le caractère suivant qui ne soit pas un espace blanc est le signe de remplacement (=I.

SHLOAD

SHLOAD imm & dif
SHLOAD

Charge à tableau de forme à partir d'une bande sur cassette. Le tableau de forme est chargé juste en-dessous de HIMEM: et HIMEM: est repositionné juste en-dessous du tableau de forme pour le protéger. L'adresse de départ du tableau de forme est donnée automatiquement aux routines de dessin de forme de APPLESOFT. Si on charge un second tableau de forme, qui remplace le premier, HIMEM doit être repositionné avant le chargement pour éviter de gâcher de la mémoire. Les bandes du tableau de forme sont préparées à l'aide des instructions données au début de ce chapitre.

Sur le système à 16K de mémoire, HGR efface les 8K supérieurs de la mémoire, de l'emplacement 8192 à l'emplacement 16383. Pour obliger SHLOAD à placer le tableau de forme en-dessous de la page 1 du graphique haute résolution, positionnez HIMEM: 8192 avant d'exécuter l'instruction SHLOAD. Sur le système à 24K de mémoire, n'utilisez pas HGR2 (qui efface la mémoire de l'emplacement 16384 à l'emplacement 245751, ou positionnez HIMEM: 16384 avant d'exécuter l'instruction SHLOAD et n'utilisez pas HGR. Si vous êtes certain qu'il y a encore suffisamment de mémoires sûre au-dessus de l'emplacement 24575 pour contenir votre tableau de forme, il n'y a pas à vous inquiéter.

On peut interrompre SHLOAD qu'avec reset. Si le mot réservé SHLOAD commence à un nom de variable, l'instruction relative au mot réservé peut être exécutée avant d'avoir un message d'erreur

?SYNTAX ERROR.

L'instruction

SHLOADER=59

suspend le système pendant que APPLESOFT attend indéfiniment le programme en provenance du lecteur de cassette. Pour reprendre le contrôle de l'ordinateur, utilisez reset ctrl C.

CHAPITRE 10
Fonctions mathématiques

Fonctions incorporées

Toutes les fonctions peuvent s'utiliser partout où une expression du même type peut s'utiliser. Elles peuvent s'utiliser en exécution soit immédiate, soit différée. Voici une brève description de certaines des -fonctions arithmétiques de APPLESOFT. D'autres fonctions sont décrites dans les sections qui prêtent des instructions similaires.

SIN (aexpr)
Donne le sinus de \aexpr\ radians.

COS (aexpr)
Donne le cosinus de \aexpr\ radians. -

TAN (aexpr)
Donne la tangente de \aexpr\ radians.

ATN (aexpr)
Donne I arc tangente radians de \aexpr \. L'angle donné est sur l'étendue -pi/2 à +pi/2 radians.

INT (aexpr)
Donne le plus grand entier inférieur ou égal à \aexpr\.

RND (aexpr)
Donne un nombre réel aléatoire supérieur ou égal à zéro et inférieur à 1. Si \aexpr\ est supérieur à zéro, RND (aexpr) donne un nouveau nombre aléatoire à chaque nouvelle utilisation.

Si \aexpr\ est inférieur à zéro, -donne le même nombre aléatoire chaque fois qu'on l'utilise avec le même \aexpr\, comme s'il provenait d'un tableau permanent de nombres RDN (aexpr) incorporé dans APPLE. Si on utilise un argument négatif particulier pour générer un nombre aléatoire, alors les nombres aléatoires successifs engendrés avec des arguments positifs vont suivre le même ordre chaque fois. Un ordre aléatoire différent est initialisé par chaque nouvel argument négatif différent. La raison principale d'utiliser un argument négatif pour RND est donc d'initialiser une séquence répétitive de nombres aléatoires. Ceci est particulièrement utile pour la mise au point des programmes qui utilisent RND.

Si \aexpr \ est zéro, RND donne le nombre aléatoire précédent le plus récent généré (les instructions CLEAR et NEW n'affectent pas ceci). Parfois ceci est plus facile que d'assigner le dernier aléatoire à une ,i variable pour le sauvegarder.

SGN (aexpr)
Donne - 1 si \ aexpr \ <0, donne 0 si \ aexpr \ =0 et donne 1 si \ aexpr \ >0

ABS (aexpr)
Donne la valeur absolue de \aexpr\, c'est-à-dire \aexpr\ si \aexpr\>=0 et -\aexpr\ si \aexpr \ <0.

SQR (aexpr)
Donne la racine carrée positive. C'est une exécution spéciale qui opère plus rapidement que ^.5.

EXP (aexgr)
Élève (avec décimales, e = 2.718289) à la puissance indiquée, \aexpr\.

LOG (aexpr)
Donne le logarithme naturel de \aexpr\ .

Fonctions dérivées

Les fonctions suivantes, bien que ne faisant pas intrinsèquement partie de APPLESOFT BASIC, peuvent se calculer à l'aide des fonctions BASIC existantes et peuvent s'utiliser facilement à l'aide de la fonction DEF FN.

SÉCANTE:
SEC (X) = 1 /COS(X)

COSÉCANTE:
CSC(X) = 1 /SIN(X)

COTANGENTE:
COT(X) = 1 /TAN(X)

ARC SINUS:
ARCSIN(X) = ATN(X/SQR(-X*X+1))

ARC COSINUS:
ARCCOS(X) = ATN(X/SQR(-X*X+ 111+ 1.5708

ARC SÉCANTE:
ARSEC(X) = ATN(SQR(X*X-1))+(SGN(X)-1)*1.5708

ARC COSÉCANTE:
ARCCSC(X) = ATN(1/SQR(X*X-1))+(SGN(X)-1)'1.5708

ARC COTANGENTE
ARC COT(X) _ -ATNIXI+ 1.5708

SINUS HYPERBOLIQUE
(SINH(X) = (EXP(X)-EXP(-X))/2

COSINUS HYPERBOLIQUE:
COSH(X) = (EXP(X)+EXP(-X))/2

TANGENTE HYPERBOLIQUE:
TANH(X) _ -EXPI-XI/IEXPIXI+EXPI-X11`2+1

SÉCANTE HYPERBOLIQUE
SECH(X) = 2/(EXP(X)+EXP(-X))

COSÉCANTE HYPERBOLIQUE
CSCH(X) = 2/(EXP(X)-EXP(-X))

COTANGENTE HYPERBOLIQUE:
COTH(X) = EXP(-X)-EXP(-X)*2+ 1

ARC SINUS HYPERBOLIQUE
ARGSINH(X) = LOG(X+SQR(X*X+1))

ARC COSINUS HYPERBOLIQUE:
ARGCOSH(X) = LOG(X+SQR(X*X-1))

ARC TANGENTE HYPERBOLIQUE
ARGTANH(X) = LOG(1+X)/(1-X))/2

ARC SÉCANTE HYPERBOLIQUE:
ARGSECH(X) = LOG((SQR(-X*X+1)+1)/X

ARC COSÉCANTE HYPERBOLIQUE
ARGCSCH(X) = LOG(SGN(X)*SQR(X*X+1)+1)/X

ARC COTANGENTE HYPERBOLIQUE
ARGCOTH(X) = LOG(X+1)/(X-1))/2

A MOD B
MOD(A) = INT((A/B)*B+.05)*SGN(A/B)

APPENDICES

APPENDICE A

Pour avoir APPLESOFT/BASIC prêt à passer

APPLE Computer Inc. offre deux versions du langage de programmation BASIC. Integer BASIC, décrit dans le manuel de programmation APPLE II BASIC, est un BASIC très rapide convenant pour de nombreuses applications, particulièrement pour l'enseignement des jeux et des graphiques, l'autre version de BASIC s'appelle "APPLESOFT" et convient mieux pour de nombreuses applications d'affaires et scientifiques.

APPLESOFT BASIC est disponible en deux versions. La programmation APPLESOFT vient, avec APPLESOFT en mémoire ROM, sur une carte de circuit imprimé (n de pièce APPLE A2B0009X) qui s'enfiche directement dans APPLE II. Avec cette option il suffit de faire commuter un commutateur et d'enfoncer deux touches pour que APPLE II commence à fonctionner en APPLESOFT. En plus de cette commodité, le fait d'avoir APPLESOFT en mémoire ROM vous économise environ 10 K de mémoires et vous économise le temps de chargement du langage à chaque utilisation à partir de la cassette. L'essentiel de ce manuel suppose que vous disposez de la carte de programmation APPLESOFT. Si vous utilisez la version cassette APPLESOFT, voyez *la partie 2 de cet appendice* pour les instructions spéciales et notez où votre APPLESOFT diffère de celui décrit dans le reste de manuel.

Note: dans ce manuel le mot reset (mettre à l'état initial) signifie enfoncer la touche marquée RESET, 1 e mot RETURN signifie enfoncer la touche marquée RETURN et ctrl B signifie taper B tout en maintenant enfoncée la touche marquée CTRL.

Note importante:
Une des fonctions du caractère indicateur, en plus de vous indiquer d'avoir à entrer une donnée dans l'ordinateur est d'identifier d'un coup d'oeil quel est le langage pour lequel l'ordinateur est programmé à répondre à tout moment. Par exemple jusqu'ici vous avez vu deux caractères indicateurs.

* pour le programme moniteur (lorsque vous enfoncez RESET)

> pour Integer BASIC (le integer BASIC normal)

Nous introduisons maintenant un troisième caractère indicateur:

] pour APPLESOFT à virgule flottante.

En regardant simplement ce caractère vous pouvez facilement savoir (si vous l'avez oublié) dans quel langage travaille l'ordinateur.

PARTIE 1: programmation

Pour installer la carte APPLESOFT

La carte de programmation APPLESOFT s'enfiche simplement dans un support qui se trouve dans APPLE II.
Ill vous faut prendre soin toutefois de suivre exactement ces instructions:

1. Mettre APPLE hors-circuit: très important pour éviter d'endommager l'ordinateur

2. Enlever le couvercle de APPLE II. Ceci se fait en tirant sur le couvercle au bord arrière (le bord le plus éloigné du clavier) jusqu'à ce que les deux agrafes d'angle se séparent. A ce moment là ne continuez pas à soulever le bord arrière mais faites glisser le couvercle vers l'arrière jusqu'à qu'il soit libre.

3. A l'intérieur de APPLE I I, à l'arrière du panneau de circuit, il y a une file de huit supports étroits nommés "Slots". Le plus à gauche (en regardant l'ordinateur depuis l'extrémité du clavier) est le slot #0; le plus à droite est le slot #7. Tenir la carte APPLESOFT de façon que ces contacts se trouvent dirigés vers l'arrière de l'ordinateur; insérer la portion de la carte présentant des contacts dans le slot le plus à gauche slot #0. Les contacts pénètrent dans le slot en frottant un peu puis prennent bien leur place. La carte APPLESOFT doit être placée dans le slot #0.

4. Les contacts qui se trouvent â l'arrière de la carte APPLESOFT doivent dépasser partiellement par la fente à l'arrière de APPLE II.

5. Remettre le couvercle de APPLE: faire glisser d'abord le bord avant en place puis appuyer sur les deux angles arrières jusqu'à ce qu'ils viennent se clipser en place.

6. Puis remettre APPLE II en circuit.

Pour utiliser le panneau de programmation APPLESOFT

L'interrupteur de la carte APPLESOFT étant en position vers le bas, APPLE II va commencer à marcher en Integer BASIC lorsque vous effectuez l'opération reset ctrl B (c'est la façon de ce manuel de dire: enfoncez la touche marquée RESET puis maintenez enfoncée la touche marquée CTRL tout en tapant B.) Vous verrez le caractère > qui indique Integer BASIC.

Lorsque le contact de la carte APPLESOFT est en position vers le haut, en effectuant l'opération reset ctrl B vous amenez APPLESOFT BASIC au lieu de Integer BASIC. Le caractère] vous dit que vous êtes en APPLESOFT.

Lorsque vous utilisez le système d'exploitation à disque, l'ordinateur choisira automatiquement Integer ou BASIC APPLESOFT selon nécessité. Indépendamment de la position de l'interrupteur.

Vous pouvez également passer de Integer BASIC à APPLESOFT ou vis-versa sans manœuvrer contact de la carte-de programmation. Pour mettre l'ordinateur en APPLESOFT, utilisez

reset C080 return ctrl B return

et pour mettre l'ordinateur en Integer BASIC, utilisez

reset C081 return
ctrl B return.

AUTRE NOTE IMPORTANTE

II peut arriver d'appuyer accidentellement la touche RESET et de vous trouver dans le moniteur, comme indiqué par le caractère'. Vous pouvez alors revenir à APPLESOFT BASIC, avec APPLESOFT et votre programme intact, en tapant ctrl C return.

PARTIE 2: bande cassette APPLESOFT

APPLE II BASIC est livré sur cassette, sans frais, avec chaque APPLE II. APPLESOFT BASIC, chargé à partir de la cassette, occupe environ 10 K octets de mémoire il faut donc un ordinateur de 16 K octets ou plus de mémoires pour utiliser la version cassette de APPLE 11 BASIC.

Pour mettre en route APPLESOFT sur cassette

Utilisez la procédure suivante pour charger APPLESOFT à partir de votre lecteur de cassettes.

1. Mettez en route Integer BASIC en tapant reset ctrl B. Si vous n'êtes pas familiarisé avec cette procédure, voyez votre manuel de programmation Integer BASIC APPLE. Vous saurez que vous êtes en Integer BASIC lorsque vous verrez le caractère souffleur > affiché sur votre écran TV, suivi du carré clignotant "curseur".

2. Place le ruban APPLESOFT (numéro de pièce A2T0004 sur votre lecteur de cassette et rembobinez le ruban jusqu'au début.

3. Tapez LOAD.

4. Enfoncez la touche "play" de votre lecteur de cassette pour commencer la lecture du ruban.

5. Enfoncez la touche marquée RETURN sur le clavier de APPLE II. Lorsque vous le faites, le curseur clignotant va disparaître. Après 5 à 20 secondes APPLE II va émettre un bip, pour signaler que l'information qui se trouve sur le ruban a commencé à entrer dans l'ordinateur. Après environ 1 /2 minutes il y aura un autre bip et le caractère souffleur > suivi d'un curseur va réapparaître.

6. Arrêtez le lecteur de cassette et rembobinez le ruban. APPLESOFT se trouve maintenant dans l'ordinateur.

7. Tapez RUN et enfoncez la touche marquée RETURN. L'écran va afficher la notice copyright pour APPLESOFT II et le caractère souffleur de APPLESOFT].

Il peut vous arriver de toucher accidentellement la touche et de vous retrouver sur le programme moniteur, comme indiqué par le caractère souffleur Integer BASIC. Vous pouvez revenir alors à APPLESOFT, votre programme et APPLESOFT eux-mêmes restant intacts, en tapant

0G return.

Si ceci n'agit pas, il vous faudra recharger APPLESOFT à partir du ruban sur cassette.
Le fait de taper ctrl C ou ctrl B à partir du programme moniteur vous transfère sur Integer BASIC APPLE; ceci va effacer APPLESOFT.

Dans ce manuel reset (mettre à l'état initial) signifie enfoncer la touche marquée RESET, return signifie enfoncer la touche marquée RETURN et ctrl B signifie taper B tout en maintenant enfoncée la touche marquée CTRL.

Différences entre APPLESOFT en programmation et APPLESOFT sur cassette

APPLESOFT sur ruban en cassette (n de pièce A2T6604) ne travaille pas exactement de la même façon que la version programmation de APPLESOFT. Les commentaires suivants attirent l'attention sur la façon dont APPLESOFT cassette diffère de APPLESOFT en programmation.

Du fait que APPLESOFT sur cassette occupe environ 10 K de mémoire (et que l'ordinateur utilise par ailleurs 2 K), il n'est pas possible d'utiliser APPLESOFT sur cassette sur des ordinateurs APPLE de moins de 16 k de mémoire. APPLESOFT sur cassette étant chargé, le plus faible emplacement de mémoire disponible pour l'utilisateur est d'environ 12.300. APPLESOFT en programmation ne réside pas en mémoire RAM, de sorte qu'il est possible de l'utiliser (sans graphiques haute résolution), dans des systèmes plus petits.

HGR/le graphique haute résolution n'est pas disponible dans APPLESOFT sur cassette. L'instruction HGR efface "la page 1" de la mémoire de graphique (8 K sur 16 K) en faveur du graphique haute résolution. Du fait que APPLESOFT sur cassette occupe partiellement cette portion de mémoire, le fait d'essayer d'utiliser HGR va effacer APPLESOFT et peut effacer votre programme. L'instruction HGR 2 peut être utilisée à la fois en mémoire ROM et dans les versions sur cassette de APPLESOFT mais n'est disponible que si votre APPLE contient au moins 24 K de mémoire. Donc sur un système de moins de 24 K de mémoire, APPLESOFT sur cassette n'offre pas la possibilité du graphique haute résolution.

L'instruction

POKE - 16301,0

convertit tout mode graphique sur la totalité de l'écran en un mode mixte graphique plus texte. Lorsque cette instruction est émise après HGR 2, toutefois, les quatre lignes de texte sont prises sur la page 2 de la mémoire de texte. Dans la version sur cassette de APPLESOFT, APPLESOFT lui-même occupe la page 2 de la mémoire de texte, de sorte que l'on ne peut pas disposer de la possibilité graphique haute.

Avec Integer Basic et avec Applesoft sur la carte programmation, vous pouvez revenir à votre programme après avoir enfoncé accidentellement ou intentionnellement la touche RESET, en utilisant ctrl C return. Pour accomplir la même chose avec APPLESOFT sur cassette, vous devez utiliser 0G return (taper 0, puis taper G et enfoncer la touche RETURN). Si vous utilisez APPLESOFT sur cassette, reset ctrl C return (enfoncer la touche RESET puis enfoncée la touche C tout en maintenant enfoncée la touche, puis enfoncer la touche), va réinstaller Integer BASIC comme votre langage de programmation; ceci va effacer APPLESOFT.

En bref, chaque fois que ce manuel ok dit d'utiliser

reset ctrl C return

les utilisateurs de APPLESOFT sur cassette doivent utiliser à la place reset 0G return (enfoncer la touche RESET, puis la touche 0, puis la touche G, puis la touche RETURN.

Lorsque le manuel dit d'utiliser reset ctrl B return vous pouvez le faire mais vous aurez à recharger APPLESOFT à partir de la bande sur cassette.

Avec APPLESOFT sur cassette, il vous faut utiliser l'instruction CALL 11246 (à la place de l'instruction CALL 62450) pour effacer l'écran après une instruction HGR2 et le ramener au noir. II faut utiliser l'instruction CALL 11250 (à la place de l'instruction CALL 62454) pour effacer l'écran après une instruction HGR2 et le ramener à la couleur définie par la dernière instruction HPLOT. Si ces différentes instructions sont exécutées avant la première émission de l'instruction HGR2 elles peuvent effacer APPLESOFT.

APPENDICE B

Pour éditer un programme

II arrive généralement aux hommes de faire des erreurs à l'occasion . . . particulièrement en écrivant des programmes d'ordinateur. Pour faciliter la correction de ces bévues, APPLE a incorporé un jeu inédit de caractéristiques d'édition dans APPLESOFT BASIC.

Pour les utiliser, il vous faut d'abord familiariser avec les fonctions de quatre touches spéciales du clavier APPLE II. Ce sont la touche escape, la touche marquée ESC, la touche repeat, la touche marquée REPT et les touches flèche à gauche, flèche à droite qui sont marquées d'une flèche à gauche et d'une flèche à droite.

ESC

La touche escape (ESC) est la touche la plus à gauche de la seconde rangée en partant du haut. Elle s'utilise toujours avec une autre touche (par exemple comme les touches A, B, C ou D) de cette façon: enfoncer et relâcher ESC puis enfoncer et relâcher A, par exemple . . . alternativement.

Cette opération ou séquence de la touche ESC et d'une autre touche s'écrit sous la forme "escape A". II existe quatre fonctions escape utilisées pour l'édition:

escape A déplace le curseur vers la droite,
escape B déplace le curseur vers la gauche,
escape C déplace le curseur vers la bas,
escape D déplace le curseur vers le haut.

En utilisant la touche escape et la touche désirée, on peut déplacer le curseur en n'importe quel endroit de l'écran sans affecter ce qui est déjà affiché et sans affecter ce qui se trouve en mémoire.

TOUCHE FLÈCHE A DROITE

La touche flèche à droite déplace le curseur vers la droite. C'est la touche du clavier qui économise le plus de temps car elle ne fait pas que déplacer le curseur mais elle recopie dans la mémoire de APPLE II tous les caractères et symboles sur lesquels passe le curseur, exactement comme si vous les aviez tapé vous-même au clavier. L'affichage TV n'est pas modifié lorsque vous utilisez la touche flèche à droite.

TOUCHE FLÈCHE A GAUCHE

La touche flèche à gauche déplace le curseur vers la gauche. Chaque fois que le curseur se déplace vers la gauche, un caractère est effacé dans la ligne de programme que vous être en train de taper, indépendamment du caractère sur lequel passe le curseur. L'affichage TV n'est pas modifié lorsque vous utilisez la touche flèche à gauche. Habituellement la touche flèche à gauche ne peut pas s'utiliser pour déplacer le curseur dans la colonne la plus à gauche: il faut pour cela utiliser escape B.

REPT

La touche REPT s'utilise avec une autre touche de caractère du clavier. Elle fait que ce caractère se répète aussi longtemps que l'on maintient enfoncées la touche de caractère et la touche REPT.

Maintenant vous êtes prêt à utiliser ces fonctions d'édition pour vous économiser du temps pour faire des modifications ou des corrections à votre programme. Voici quelques exemples de la façon de les utiliser:

Exemple 1 - Pour corriger les erreurs de frappe:

Supposons que vous ayez entré un programme en le tapant et lorsque vous voulez le faire passer l'ordinateur imprime SYNTAX ERR et s'arrête en vous présentant le caractère indicateur 1 et le curseur clignotant.

Entrez le programme suivant et faites le passer. Notez que c'est volontairement que l'on a mal orthographie "PRINT" et "REGRAM". Voici approximativement ce qui va apparaître sur votre écran TV:

```
]10 PRIMT "THIS IS A PREGRAM"
]20 GOTO 10
]RUN
?SYNTAX ERR IN 10
]
```

Maintenant tapez le mot LIST et enfoncez return.

```
]LIST

10 PRIMT "THIS IS A PREGRAM"
20 GOTO 10
]
```

Pour déplacer le curseur vers le haut au début de la ligne 10, tapez escape D trois fois puis tapez escape B. Note: II est important d'utiliser escape B pour placer le curseur exactement sur le premier chiffre du numéro de la ligne. L'écran TV va apparaître comme suit:

```
]LIST

10 PRIMT "THIS IS A PREGRAM"
20 GOTO 10
```

Maintenant enfoncez 6 fois la touche flèche à droite pour déplacer le curseur sur la lettre M de 'PRIMT'.
N'oubliez pas que lorsque la touche flèche à droite déplace le curseur en passant sur un caractère sur l'écran, ce caractère est copié dans la mémoire de APPLE exactement comme si vous l'aviez tapé au clavier.
Votre écran TV apparaît comme suit:

```
]LIST

10 PRIMT "THIS IS A PREGRAM"
20 GOTO 10
]
```

Maintenant tapez la lettre N pour corriger l'orthographe de "PRIMT", puis copiez (à l'aide de la touche flèche à droite et de la touche REPEAT) en allant jusqu'à la lettre E de "PREGRAM". Votre écran TV va donner ce qui suit:

```
]LIST
10 PRINT "THIS IS A PREGRAM"
20 GOTO 10
]
```


Si vous tapez la touche flèche à droite trop souvent en maintenant enfoncée la touche repeat trop longtemps, utilisez la touche flèche à gauche pour revenir en arrière sur la lettre E. Maintenant tapez la lettre 0 pour corriger 'PR EGRAM' et copiez à l'aide de la touche flèche à droite jusqu'à la fin de la ligne 10. Enfin, enregistrez en mémoire du programme, la nouvelle ligne en enfonçant la touche RETURN.

Tapez LIST pour voir votre programme corrigé:

```
]LIST
10 PRINT "THIS IS A PROGRAM"
20 GOTO 10
]
```

Maintenant faites passer (instruction RUN) le programme (utilisez ctrl C pour arrêter le programme)

```
]RUN
THIS IS A PROGRAM
THIS IS A PROGRAM
THIS IS A PROGRAM
THIS IS A PROGRAM
THIS IS A PROGRAM
THIS IS A PROGRAM
THIS IS A PROGRAM
THIS IS A PROGRAM
BREAK IN 10
]
```

Exemple 2 - Pour insérer un texte dans une ligne existante

Dans l'exemple précédent, supposons que vous souhaitiez insérer une instruction TAB(10) après le PRINT de la ligne 10. Voici comment vous pouvez le faire. D'abord affichez la ligne à modifier (instruction LIST):

```
]LIST 10

10 PRINT "THIS IS A PROGRAM"
]
```

Tapez D escape D et un escape B jusqu'à ce que le curseur soit exactement sur le premier caractère de la ligne à modifier; puis utilisez les touches flèche à droite et repeat pour copier jusqu'au premier guillemet. (N'oubliez pas, un caractère n'est pas copié en mémoire avant que vous n'utilisiez la touche flèche à droite pour faire passer le curseur de ce caractère sur le suivant). Votre affichage TV doit donner ceci:

```
]LIST 10

10 PRINT "THIS IS A PROGRAM"
]
```

Maintenant tapez un autre ESCAPE D pour déplacer-le curseur jusqu'à la ligne vide qui se trouve juste au-dessus de la ligne en cours et l'affichage va donner ce qui suit:

```
]LIST 10

10 PRINT "THIS IS A PROGRAM"
]
```

Tapez les caractères à insérer, dans ce cas, TAB(10). Votre affichage TV va donner ce qui suit:

```
]LIST 10
         TAB(10);
10 PRINT "THIS IS A PROGRAM"
```

Tapez un escape C pour abaisser le curseur d'une ligne de façon que l'affichage donne ce qui suit:

```
]LIST 10
         TAB(10);
10 PRINT "THIS IS A PROGRAM"
]
```

Maintenant revenez en arrière jusqu'au premier guillemet à l'aide de escape B (si vous utilisiez ici la touche flèche à gauche, vous effaceriez les caractères que vous venez de taper). L'affichage TV va donner ce qui suit:

```
]LIST 10
         TAB(10);
10 PRINT "THIS IS A PROGRAM"
]
```

A partir d'ici, copiez le reste de la ligne à l'aide de la touche flèche à droite et repeat jusqu'à ce que l'affichage donne ce qui suit:

```
]LIST 10
         TAB(10);
10 PRINT "THIS IS A PROGRAM"
]
```

Enfoncez la touche RETURN et tapez LIST pour obtenir ce qui suit:

```
]LIST

10 PRINT TAB(10);"THIS IS A PROGRAM"
20 GOTO 10

]
```

Lorsque vous désirez éviter de copier des espaces blancs supplémentaires que le format LIST introduit en cours de ligne (comme ceux qui se trouvent entre le R et le 0 de PROGRAM, dans l'exemple ci-dessus), utilisez escape A. escape A déplace le curseur vers la droite sans copier les caractères. Ceci peut être particulièrement utile pour recopier des instructions PRINT, INPUT et REM, où APPLESOFT n'ignore pas les espaces blancs supplémentaires.

Souvenez-vous qu'en utilisant les touches escape on peut copier et éditer un texte affiché à n'importe quel endroit de l'écran TV.

APPENDIX C

Messages d erreur

Lorsqu'une erreur se produit, BASIC revient au niveau instruction comme indiqué parle caractère indicateur] et un curseur clignotant. Les valeurs variables et le texte du programme demeurent intacts, mais le programme ne peut pas se poursuivre et tous les compteurs de boucle GOSUB et FOR se sont positionnés à zéro.

Pour éviter cette interruption dans un programme qui passe, on peut utiliser l'instruction ONERR GOTO, en liaison avec une routine de traitement d'erreur.

Lorsqu'une erreur se produit dans une instruction à exécution immédiate, aucun numéro de ligne n'est imprimé.

Format des messages d'erreur:

Instruction à exécution immédiate	?XX ERROR.
Instruction à exécution différée	?XX ERROR IN YY.

Dans les deux exemples ci-dessus 'XX' est le nom d'une erreur spécifique; 'YY' est le numéro de la ligne de l'instruction à exécution différée où l'erreur s'est produite. Des erreurs dans une instruction à exécution différée ne sont pas détectées avant exécution de cette instruction.

Ci-dessous les codes des erreurs possibles et leur signification.

CAN'T CONTINUE

Essaie de poursuivre un programme alors qu'il n'y en a pas ou lorsqu'une erreur s'est produite ou lorsqu'une ligne a été effacée du programme ou a été ajoutée au programme.

DIVISION BY ZERO

La division par zéro constitue une erreur.

ILLEGAL DIRECT

Vous ne pouvez pas utiliser une instruction INPUT, DEF FN, GET ou DATA comme instruction à exécution immédiate.

ILLEGAL QUANTITY

Le paramètre affecté à une fonction mathématique ou à une fonction chaîne était hors de l'étendue. Des erreurs ILLEGAL QUANTITY peuvent se produire par suite de:

a) un SUBSCRIPT (indice) de tableau négatif (par exemple LET A(-1)=0);
b) utilisation de LOG avec un argument négatif ou nul;
c) utilisation de SQR avec un argument négatif;
d) A^ B avec A négatif et B non entier;
e) utilisation de MID$, LEFT$, RIGHT$, WAIT, PEEK, POKE, TAB, SPC, ON...GOTO ou de l'une quelconque des fonctions graphiques avec un argument incorrect.

NEXT WITHOUT FOR

La variable dans une instruction NEXT ne correspond pas à la variable de l'instruction FOR encore en effet ou une instruction NEXT sans nom correspond à l'une ou l'autre des instructions FOR qui était encore à l'effet.

OUT OF DATA

Une instruction READ était en cours d'exécution mais toutes les instructions DATA du programme avaient déjà été lues. Le programme essayait de lire trop de données ou bien il y avait insuffisamment de données dans le programme.

OUT OF MEMORY

L'une des causes suivantes peuvent causer cette erreur: programme trop important; trop de variables; des boucles FOR emboîtées à plus de 10 niveaux de profondeur; les instructions GOSUB emboîtées à plus de 24 niveaux de profondeur; une expression trop compliquée; des parenthèses emboîtées à plus de 36 niveaux de profondeur; un essai de positionner LOMEM trop haut; un essai de positionner LOMEM en-dessous de la valeur actuelle; un essai de positionner LOMEM trop bas.

FORMULA TOO COMPLEX

Plus de deux instructions de la forme IF "XX" THEN ont été exécutées.

OVERFLOW

Le résultat d'un calcul était trop important pour être représenté dans le format de nombre du BASIC. Si, inversement, on a un nombre trop faible (très proche de zéro), c'est zéro qui est donné comme résultat et l'exécution du programme se poursuit sans qu'il apparaisse de message d'erreur.

REDIM'D ARRAY

Après que l'on ait dimensionné un tableau, une autre instruction de dimension se rencontre pour le même tableau. Cette erreur arrive souvent si un tableau a reçu la dimension par défaut 10 du fait qu'une instruction comme A(I)=3 est suivie d'un programme par une instruction DIM A (1001. Ce message d'erreur peut se prouver utile si vous souhaitez découvrir sur quelle ligne de programme un certain tableau a été dimensionné: il suffit d'insérer une instruction de dimension pour ce tableau à la première ligne, de faire passer le programme et APPLESOFT va vous dire où se trouve l'instruction de dimension d'origine.

RETURN WITHOUT GOSUB

On a rencontré une instruction RETURN sans qu'une instruction correspondante GOSUB ne soit exécutée.

STRING TOO LONG

En utilisant un opérateur de concaténation, on a essayé de créer une chaîne de plus de 255 caractères de longueur.

BAD SUBSCRIPT

On a essayé de se référer à un élément de tableau qui se trouve en dehors des dimensions du tableau. Cette erreur peut se produire si on utilise un nombre erroné de dimensions dans une

référence de tableau; par exemple si on utilise LET A (1,1,1)=Z alors que le tableau A a été dimensionné à l'aide de DIM A(2,2).

SYNTAX ERROR

Oubli d'une parenthèse dans une expression, caractère illégal dans une ligne, ponctuation incorrecte, etc.

TYPE MISMATCH

Le côté gauche d'une instruction d'affectation était une variable numérique et le côté droit était une chaîne ou vice versa; ou bien une fonction qui attendait un argument chaîne reçoit un argument numérique ou vice versa.

UNDEF'D STATEMENT

On a essayé, par les instructions GOTO, GOSUB ou THEN, de renvoyer à un numéro d'instruction qui n'existe pas.

UNDEF'D FUNCTION

On a fait référence à une fonction définie par l'utilisateur mais que l'utilisateur n'a pas défini.

APPENDICE D
Economiseurs d'espace mémoire

Des conseils pour économiser l'espace mémoire

Pour que votre programme puisse s'adapter dans un espace de mémoire plus petit, on peut suivre les conseils suivants. Toutefois les deux premiers conseils ne doivent être pris en compte que si l'on se trouve en face de sérieuses limitations concernant l'espace. Les programmeurs sérieux donnent souvent deux versions de leur programme: une version étendue largement documentée (avec des REM) et l'autre réduite pour utiliser le minimum d'espace en mémoire.

1) Utilisez des instructions multiples par ligne. II n'y a qu'un petit en-tête (5 octets) associé à chaque ligne de programme. Deux de ces 5 octets contiennent le numéro de ligne de la ligne en binaire. Ceci signifie que quel que soit le nombre de chiffres que vous avez dans votre numéro de ligne (le numéro de ligne minimum est 0, le maximum est 65529), il prend le même nombre d'octets (deux). En plaçant le plus d'instructions possible sur chaque ligne, vous réduisez le nombre d'octets utilisés par votre programme (une seule ligne peut comprendre jusque 239 caractères).

Note: le fait de combiner plusieurs instructions sur une ligne rend très difficile les changements d'édition et autres changements. II rend également très difficile la lecture et la compréhension d'un programme non seulement pour d'autres personnes mais même pour vous si vous revenez plus tard à ce programme.

2) Effacez toutes les instructions REM. Chaque instruction REM utilise au moins un octet plus le nombre d'octets du texte proprement dit. Par exemple l'instruction

```
130 REM THIS IS A COMMENT
```

utilise 24 octets de mémoire. Dans l'instruction

```
140 X=X+Y: REM UPDATE SUM
```

le REM utilise 12 octets de mémoire y compris les deux points avant le REM.

Note: de même que les programmes à lignes multiples, un programme sans instructions détaillées REM est très difficile à lire et à comprendre non seulement pour les autres mais également pour vous si vous revenez plus tard à ce programme.

3) Utilisez des tableaux d'entiers au lieu de tableaux de réels chaque fois que possible (voir l'information concernant l'affectation des mémoires plus loin dans cet appendice).

4) Utilisez des variables au lieu de constantes. Supposons que vous utilisiez la constante 3.14159 dix fois dans votre programme. Si vous insérez une instruction

```
10 P=3.14159
```

dans le programme et utilisez P au lieu de 3.14159 chaque qu'il est nécessaire, vous économiserez 40 octets. Ceci se traduit également par une amélioration de la vitesse.

5) Un programme n'a pas besoin de se terminer avec END; on peut donc supprimer l'instruction END à la fin du programme.

6) Réutilisez les mêmes variables. Si vous avez une variable T utilisée pour conserver un résultat provisoire dans une partie du programme et si vous avez besoin plus loin d'une variable provisoire dans votre programme, utilisez-là à nouveau. Ou si vous demandez à l'utilisateur de l'ordinateur de donner une réponse YES ou NO à deux questions différentes, à deux instants différents, pendant l'exécution du programme, utilisez la même variable provisoire A$ pour enregistrer la réponse

7) Utilisez des instructions GOSUB pour exécuter les sections des instructions de programme qui exécutent des actions identiques.

8) Utilisez les éléments zéros des matrices; par exemple A(01, B(0,X)

9) Si A$="CAT" est réassigné en A$="DOG", l'ancienne chaîne "CAT" n'est pas effacée de la mémoire. En utilisant périodiquement une instruction de la forme
X = FRE (0)
dans votre programme, vous ferez en sorte que APPLESOFT "fasse le ménage" pour enlever les anciennes chaînes du haut de la mémoire.

Information concernant l'affectation des mémoires

Les variables simples (autres que des tableaux) réelles, entières ou chaînes comme V, V%, ou V$ utilisent 7 octets. Les variables réelles utilisent 2 octets pour le nom de la variable et 5 octets pour leur valeur (1 pour l'exposant et 4 pour la mantisse). Les variables entières utilisent 2 octets pour le nom de la variable, 2 octets pour sa valeur et ont des 0 dans les 3 autres octets. Les variables chaînes utilisent 2 octets pour le nom de la variable, 1 octet pour la longueur de la chaîne, 2 octets pour un pointeur à l'emplacement de la chaîne en mémoire et ont des 0 dans les deux autres octets.

Les variables réelles en tableaux utilisent au minimum 12 octets: 2 octets pour le nom de la variable, 2 octets pour les valeurs d'indice, 1 octet pour le nombre de dimensions, 2 pour les valeurs d'indice de chaque dimension et 5 octets pour chaque élément du tableau. Les variables entières en tableau n'utilisent que 2 octets pour chaque élément du tableau. Les variables chaînes en tableau utilisent 3 octets pour chaque élément du tableau: un pour la longueur, deux pour un pointeur.

Les variables chaînes, soit simples soit en tableaux, utilisent un octet de mémoire pour chaque caractère de la chaîne. Les chaînes elles-mêmes sont situées dans l'ordre de leur arrivée dans le programme en commençant à HIMEM:.

Lorsqu'on définit une nouvelle fonction par une instruction DEF, on utilise 6 octets pour enregistrer en mémoire le pointeur correspondant à la définition.

Les mots réservés comme FOR, GOTO ou NOT et les noms de fonctions intrinsèques comme COS, INT et STR $ ne prennent qu'un octet de mémoire de programme. Tous les autres caractères en programme utilisent un octet de mémoire de programme chacun.
Lorsqu'un programme est en cours d'exécution, l'espace est attribué dynamiquement sur les piles comme suit:

1) Chaque boucle active FOR . . . NEXT utilise 16 octets.
2) Chaque instruction GOSUB active (c'est-à-dire qui n'a pas encore fait l'objet de l'instruction RETURN) utilise 6 octets.
3) Chaque parenthèse rencontrée dans une expression utilise 4 octets et chaque résultat provisoire calculé dans une expression utilise 12 octets.

APPENDICE E

Pour accélérer votre programme

Les conseils ci-dessous doivent améliorer le temps d'exécution de vos programmes BASIC. Notez que certains de ces conseils sont les mêmes que ceux utilisés pour diminuer l'espace utilisé par vos programmes. Ceci signifie que dans de nombreux cas vous pouvez à la fois améliorer la vitesse de vos programmes et améliorer l'efficacité de l'utilisation de la mémoire.

1) Ceci est probablement le conseil le plus important, d'un facteur de 10, concernant la vitesse: utilisez des variables au lieu des constantes. II faut plus de temps pour convertir une constante en sa représentation flottante (nombre réel) qu'il n'en faut pour prendre en charge la valeur d'une variable simple ou en tableau. Ceci est spécialement important dans les boucles FOR . . . NEXT ou autres codes à exécuter de façon répétitif.

2) Les variables qui sont rencontrées les premières pendant l'exécution d'un programme BASIC sont affectées au début du tableau de variable. Ceci signifie que l'instruction comme

```
5 A = 0 : B = A : C = A
```

va placer A en premier, B en second et C en troisième dans le tableau de variables (en supposant que la ligne 5 est la première instruction exécutée par le programme). Plus loin dans le programme, lorsque BASIC trouve une référence à la variable A, il ne cherche qu'une entrée dans le tableau des variable pour trouver A, deux entrées pour trouver B et trois entrées pour trouver C, etc. . . .

3) Utilisez les instructions NEXT sans variables indicées. NEXT est un peu plus rapide que NEXT I parce qu'il n'y a pas de vérification à faire pour voir si la variable spécifiée dans le NEXT est la même que la variable qui se trouvait dans l'instruction FOR encore active la plus récente.

4) Pendant l'exécution du programme, lorsque APPLESOFT rencontre la référence à une nouvelle ligne comme "GOTO 1000", il explore la totalité du programme utilisateur en commençant par les lignes les plus basses jusqu'à ce qu'il trouve le numéro de ligne référencé (1000 dans cet exemple). II faut donc placer les lignes à référence fréquente aussitôt que possible dans le programme.

APPENDICE F

Signes décimaux pour les mots-clés

Signe décimal	Mot-clé	Signe décimal	Mot-clé	Signe décimal	Mot-clé
128	END	164	LOMEM:	200	+
129	FOR	165	ONERR	201	-
130	NEXT	166	RESUME	202	*
131	DATA	167	RECALL	203	/
132	INPUT	168	STORE	204	^
133	DEL	169	SPEED=	205	AND
134	DIM	170	LET	206	OR
135	READ	171	GOTO	207	>
136	GR	172	RUN	208	=
137	TEXT	173	IF	209	<
138	PR#	174	RESTORE	210	SGN
139	IN#	175	&	211	INT
140	CALL	176	GOSUB	212	ABS
141	PLOT	177	RETURN	213	USR
142	HLIN	178	REM	214	FRE
143	VLIN	179	STOP	215	SCRN(
144	HGR2	180	ON	216	PDL
145	HGR	181	WATT	217	POS
146	HCOLOR =	182	LOAD	218	SQR
147	HPLOT	183	SAVE	219	RND
148	DRAW	184	DEF	220	LOG
149	DRAW	185	POKE	221	EXP
150	HTAB	186	PRINT	222	COS
151	HOME	187	CONT	223	SIN
152	ROT=	188	LIST	224	TAN
153	SCALE=	189	CLEAR	225	ATN
154	SHLOAD	190	GET	226	PEEK
155	TRACE	191	NEW	227	LEN
156	NOTRACE	192	TAB(	228	STR$
157	NORMAL	193	TO	229	VAL
158	INVERSE	194	FN	230	ASC
159	FLASH	195	SPC(	231	CHR $
160	COLOR=	196	THEN	232	LEFT$
161	POP	197	AT	233	RIGHT$
162	VTAB	198	NOT	234	MID$
163	HIMEM:	199	STEP		

APPENDICE G
Mots réservés dans APPLESOFT

&							
ABS	AND	ASC	AT	ATN			
CALL	CHR $	CLEAR	COLOR=	CONT	COS		
DATA	DEF	DEL	DIM	DRAW			
END	EXP						
FLASH	FN	FOR	FRE				
GET	GOSUB	GOTO	GR				
HCOLOR=	HGR	HGR2	HIMEM:	HLIN	HOME	HPLOT	HTAB
IF	IN#	INPUT	INT	INVERSE			
LEFT$	LEN	LET	LIST	LOAD	LOG	LOMEM:	
MID$							
NEW	NEXT	NORMAL	NOT	NOTRACE			
ON	ONERR	OR					
PDL	PEEK	PLOT	POKE	POP	POS	PRINT	PR#
READ	RECALL	REM	RESTORE	RESUME	RETURN	RIGHT$	
	RND	ROT=	RUN				
SAVE	SCALE=	SCRN(	SGN	SHLOAD	SIN	SPC(	
	SPEED =	SQR	STEP	STOP	STORE	STR$	
TAB(	TAN	TEXT	THEN	TO	TRACE		
USR							
VAL	VLIN	VTAB					
WAIT							
XPLOT	DRAW						

Le signe d'abréviation de et (&) est prévu uniquement pour usage interne de l'ordinateur; il n'est pas une instruction correcte APPLESOFT. Ce symbole, lorsqu'on l'exécute comme instruction, amène un saut inconditionnel à l'emplacement $3F5. Utilisez reset ctrl C pour reprendre le contrôle du programme.

XPLOT est un mot réservé qui ne correspond pas à une instruction APPLESOFT que dans un certain contexte.

XPLOT est un mot réservé qui ne correspond pas à une instruction APPLESOFT en cours.

COLOR, HCOLOR, SCALE, SPEED et ROT
ne s'analysent comme mots réservés que si le caractère suivant autre qu'un espace est le signe de remplacement, =. Ceci ne présente que peu d'intérêt dans le cas de COLOR et HCOLOR, car le mot réservé inclus OR interdit leur usage de toute façon dans des noms de variables.

SCRN, SPC et TAB
ne s'analysent comme mots réservés que si le caractère suivant qui ne soit pas un espace est une parenthèse à gauche,(.

HIMEM:
doit avoir le signe des deux points (:) pour s'analyser comme mot réservé.

LOMEM:
nécessite également le signe des deux points (: 1 pour être analysé comme mot réservé.

ATN
ne s'analyse comme mot réservé que s'il n'y a pas d'espace entre le T et le N. S'il y a un espace entre le T et le N, c'est le mot réservé AT qui est analysé au lieu de ATN.

TO
s'analyse comme mot réservé à moins qu'il ne soit précédé d'un A et qu'il n'y ait un espace entre le T et le 0. S'il y a un espace entre le T et le O, c'est le mot réservé AT qui est analysé au lieu de TO.

On peut parfois utiliser des parenthèses pour tourner les mots réservés:

```
100 FOR A = LOFT OR CAT TO 15
```

donne le listage

```
100 FOR A = LOF TO RC AT TO 15
```

mais

```
100 FOR A = (LOFT) OR (CAT) TO 15
```

donne le listage

```
100 FOR A = (LOFT) OR (C AT) TO 15
```

APPENDICE H

Pour convertir les programmes BASIC en APPLESOFT

Bien que les réalisations de BASIC des différents ordinateurs soient en de nombreux points similaires, il existe certaines incompatibilités qu'il faut connaître si vous envisagez de convertir des programmes BASIC en APPLESOFT.

1) Indices de tableau (de matrice). Certains BASIC utilisent "[" et "]" pour noter les indices de tableau. APPLESOFT utilise "(" et ")".

2) Chaînes. Un certain nombre de BASIC vous obligent à dimensionner (à déclarer) la longueur des chaînes avant les utiliser. Vous devez enlever toutes les instructions de dimension de ce type de votre programme. Dans certains de ces BASIC, une déclaration de la forme DIM A$ (I, J) déclare un tableau de chaînes de J éléments dont chacun a une longueur I. Convertissez les instructions DIM de ce type en leur équivalent en APPLESOFT: DIM A$(J).

APPLESOFT utilise "+" pour la concaténation des chaînes et non pas "," ou "&".

BASIC utilise LEFT $, RIGHT $ et MID $ pour prendre les indices des chaînes. D'autres BASIC utilisent A$(I) pour accéder au caractère d'ordre I des chaînes A$ et A$(I,J) pour prendre une sous-chaîne de la chaîne A$ depuis la position de caractère I jusqu'à la position de caractère J.

Convertissez comme suit:

Ancien	**Nouveau**
`A$(I)`	`MID$(A$,I,1)`
`A$(I,J)`	`MID$(A$,I,J -I + 1 I`

Ceci suppose que la référence à une sous-chaîne de la chaîne A$ est dans une expression ou est côté droit de l'affectation. Si la référence A$ se trouve côté gauche de l'affectation et si X$ est l'expression chaîne utilisée pour remplacer les caractères dans A$

convertissez comme suit:

Ancien	**Nouveau**
`A$(I)=X$`	`A$=LEFT$(A$,I-1)+X$+MID$(A$,I+ 1)`
`A$(I,J)=X$`	`A$=LEFT$(A$,I-1)+X$+MID$(A$,J+ 1)`

3) Certains BASIC permettent des instructions "d'affectation multiple" de la forme:

```
500 LET B=C=0
```

Cette instruction devrait positionner les deux variables B et C à zéro.

Dans APPLESOFT, ceci a un effet entièrement différent. Tous les signes d'égalité qui se trouvent à la droite du premier doivent s'interpréter comme des opérateurs de comparaison logique. Ceci positionnerait la variable B à - 1 si C est égal à 0. Si C n'est pas égal à 0, B serait positionné à 0. La façon la plus simple de converser des instructions comme celle-ci est de la réécrire comme suit:

```
500 C=0:B=C
```


4) Certains BASIC utilisent "/" au lieu de ":" pour délimiter des instructions multiples par ligne. Changez chaque "/" en ":" dans le programme.

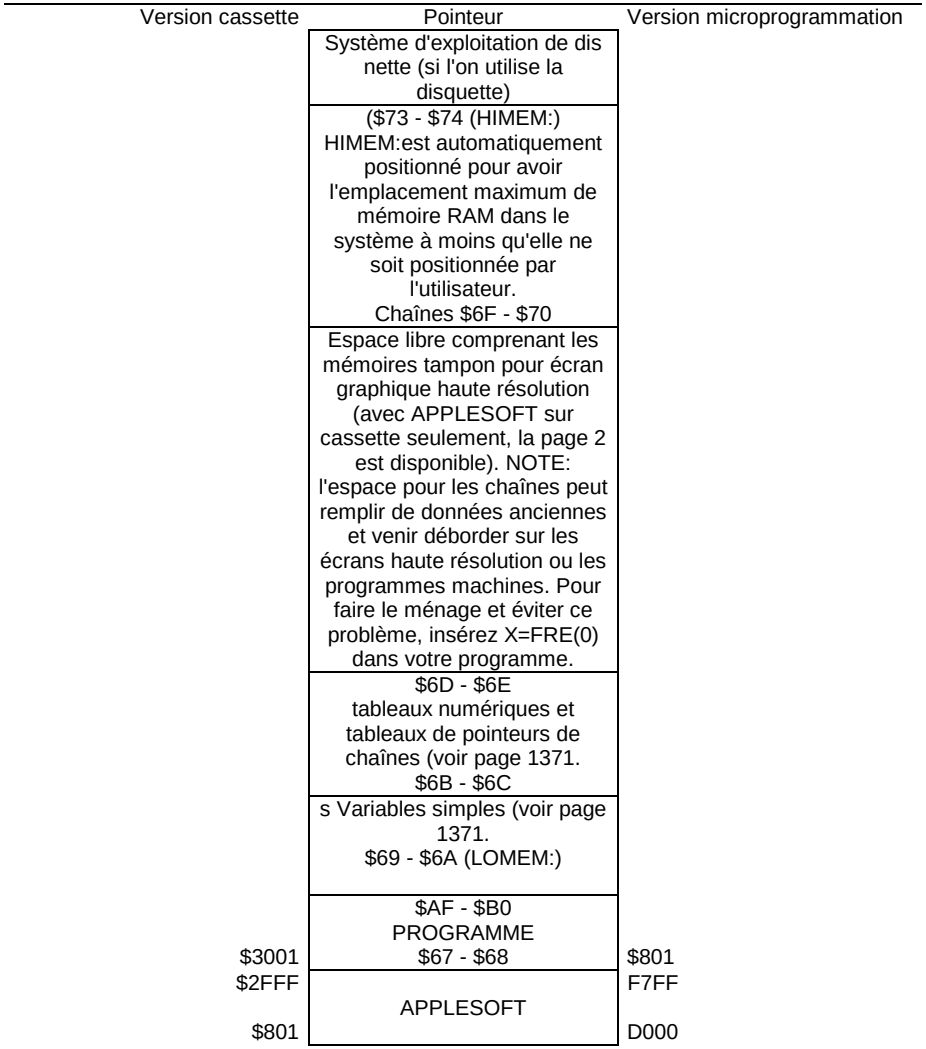
5) Les programmes qui utilisent les fonctions MAT dans certains BASIC devront être réécrits en utilisant les boucles FOR . . . NEXT pour effectuer les opérations appropriées.

APPENDICE I

Carte de la mémoire

Etendue de mémoire	Description
0.1FF	Espace de travail du programme; non disponible pour l'utilisateur.
200.2FF	Mémoire tampon de caractères de clavier.
300.3FF	Disponible pour l'utilisateur pour de courts programmes en langage machine
400.7FF	Zone d'affichage sur écran pour le texte ou les graphiques couleurs page 1.
800.2FFF	Dans la version cassette, zone de l'interpréteur APPLESOFT BASIC.
800.XXX	Si la carte APPLESOFT est mise en place (numéro de pièce A2B0009X) espace réservé au programme et aux variables de l'utilisateur, où XXX est le maximum de mémoires RAM à utiliser par APPLESOFT. C'est soit la totalité de la mémoire RAM du système soit moins si l'utilisateur réserve une partie de la mémoire supérieure pour les routines de langage machine des mémoires tampon de l'écran haute résolution.
2000.3FFF	Microprogrammation APPLESOFT seulement: affichage graphique haute résolution page 1.
3000.XXX	APPLESOFT II sur cassette; réservé au programme et aux variables utilisateurs où XXX est le maximum de mémoires RAM disponible à utiliser par APPLESOFT. C'est soit la totalité de mémoires RAM du système soit moins si l'utilisateur réserve une partie de la mémoire supérieure pour des routines de langage machine où pour le graphique haute résolution page 2.
4000.5FFF	Affichage graphique haute résolution page 2.
C000.CFFF	Adresses d'entrée/sortie du matériel périphérique.
D000.DFFF	Futur extension de la mémoire ROM.
D000.F7FF	Version APPLESOFT II sur carte avec sélecteur position ON (en haut).
E000.F7FF	Integer BASIC APPLE.
F800.FFFF	Moniteur de système APPLE.

Diagramme de la carte de mémoire de programme APPLESOFT



APPENDICE J

PEEK, POKE, CALL

Voici quelques caractéristiques spéciales de APPLESOFT que vous pouvez utiliser au moyen des instructions PEEK, POKE et CALL. Notez que certaines viennent doubler les effets d'autres instructions de APPLESOFT.

Les actions de simple commutation dépendent habituellement d'une adresse: n'importe quelle instruction impliquant cette adresse aura le même effet sur la commutation. Ainsi donc l'exemple peut être

```
POKE - 16304,0
```

mais vous obtiendrez le même effet en assignant à cette adresse, par l'instruction POKE, tout nombre de 0 à 255 ou en regardant cette adresse par l'instruction PEEK

```
X = PEEK (-16304)
```

Ceci ne s'applique pas aux instructions où vous devez assigner à l'adresse spécifiée une valeur *spécifique* qui positionne une marge ou déplace le curseur pour l'amener à une place spécifique.

Pour positionner la fenêtre de texte

Les quatre premières instructions POKE, avec les numéros de ligne par exemple 10, 20, 30 et 40 sont utilisées pour positionner la taille de la fenêtre par laquelle vous voyez et vous déroulez le texte sur votre écran TV. Ces instructions positionnent respectivement la marge de gauche, la largeur de la ligne, la marge du haut et la marge du bas de la fenêtre.

Le fait de positionner la fenêtre de texte n'efface pas le reste de l'écran et ne déplace pas le curseur dans la fenêtre de texte (il faut utiliser pour cela HOME, ou HTAB et VTAB). L'instruction VTAB ignore complètement la fenêtre de texte: le texte imprimé au-dessus de la fenêtre apparaît normalement tandis que le texte imprimé en dessous de la fenêtre apparaît sur toute la ligne. HTAB peut également déplacer le curseur à l'extérieur de la fenêtre mais seulement assez loin pour y imprimer un caractère.

Une modification de la largeur de la ligne prend effet immédiatement mais une modification de la marge de gauche n'est pas détectée avant que le curseur tente de revenir à la marge de gauche.

Le texte qui apparaît sur votre écran TV n'est qu'une représentation d'une portion particulière de la mémoire APPLE (page 1 du texte). L'écran TV regarde toujours le texte qui se trouve dans la même portion de la mémoire et voit ce que APPLE y a écrit. Si vous modifiez la fenêtre de texte, cela revient à dire à APPLE où il doit écrire son texte en mémoire. Ceci marche bien aussi longtemps que vous spécifiez une portion de mémoire quine trouve à *l'intérieur* de la zone habituelle de texte. Mais si vous positionnez la marge de gauche disons à 255 (le maximum devra être 40 puisque l'écran a 40 positions de caractère en largeur), cela revient à dire à APPLE d'écrire le texte bien au-delà de la zone de mémoires habituelle réservée pour le texte. Cette mémoire n'est pas vue sur l'écran et peut contenir des parties de votre programme ou même des informations nécessaires pour APPLESOFT lui-même. Pour sauvegarder votre programme et APPLESOFT, il faut se garder de positionner la fenêtre sur le texte *au-delà* des limites de l'écran de 40 caractères par 24 lignes.


```
10 POKE 32, L
```

Positionne la marge de gauche de l'affichage TV à la valeur spécifiée par L, sur l'étendue allant de O à 39, où 0 est la position la plus à gauche. Cette modification ne s'effectue pas avant que le curseur n'arrive à un retour à la marge de gauche (retour de chariot).

Cette instruction ne modifie pas l a largeur de la fenêtre: ceci signifie que la marge de droite va se déplacer de la même valeur que celle dont vous dépassez la marge de gauche. Pour sauvegarder votre programme et APPLESOFT, il faut d'abord réduire la largeur de la fenêtre de façon appropriée puis modifier la marge de gauche.

```
20 POKE 33, W
```

Positionne la largeur (nombre de caractères par ligne) de l'affichage TV à la valeur spécifiée par W, sur l'étendue allant de 1 à 40.

Ne positionnez pas W à zéro: POKE 33, 0 détruit APPLESOFT.

Si W est inférieur à 33, un PRINT au troisième champ de tabulateur peut imprimer des caractères en dehors de la fenêtre.

```
30 POKE 34, T
```

Positionne la marge de l'affichage TV à la valeur spécifiée par T sur l'étendue allant de 0 à 23 où est la ligne supérieure de l'écran. Une instruction POKE 34,4 permet de ne pas imprimer le texte sur les quatre premières lignes de l'écran. Ne positionnez pas la marge supérieure de la fenêtre (T) en-dessous de la marge inférieure (B) ci-dessous.

```
40 POKE 35, B
```

Positionne la mare inférieure de l'affichage TV à la valeur spécifiée par B, sur l'étendue allant de 0 à 24, où 24 est la ligne inférieure de l'écran. Ne positionnez pas la marge inférieure de la fenêtre (B) au-dessus de la marge supérieure (T, ci-dessus).

Autres instructions affectant le texte, la fenêtre de texte et le clavier

```
45 CALL -936
```

Efface tous les caractères qui se trouvent à l'intérieur de la fenêtre de texte et déplace le curseur pour l'amener à la position d'impression supérieure gauche de la fenêtre. C'est la même instruction que esc @ return (Escape @) et que l'instruction HOME.

```
50 CALL –958
```

Efface tous les caractères qui se trouvent à l'intérieur de la fenêtre de texte depuis la position actuelle du curseur jusqu'à la marge inférieure. Les caractères qui se trouvent au-dessus du curseur et les caractères qui se trouvent à gauche du curseur dans sa ligne d'impression ne sont pas affectés. Cette instruction est la même

esc F (Escape F).

Si le curseur se trouve au-dessus de la fenêtre de texte, cette instruction efface depuis le curseur jusqu'aux marges de droite, de gauche et en bas comme si la marge supérieure était au-dessus du curseur. Il n'est pas habituellement pas souhaitable d'utiliser cette instruction si le curseur se trouve en-dessous de la marge inférieure de la fenêtre de texte: habituellement la ligne inférieure de la fenêtre de texte est effacée ainsi qu'une ligne de la largeur de la fenêtre de texte et la position du curseur.

```
60 CALL -868
```

Efface la ligne en cours depuis le curseur jusqu'à la marge de droite. Cette instruction est la même que esc E (Escape E).

```
70 CALL -922
```

Déclenche un saut de ligne. C'est la même instruction que ctrl J (control J).

```
80 CALL -912
```

Déroule une ligne de texte, c'est-à-dire déplace chaque ligne de texte vers le haut d'une position à l'intérieur de la fenêtre définie. L'ancienne ligne supérieure est perdue: la ligne qui était auparavant la seconde devient la première; la ligne inférieure devient vierge. Les caractères qui se trouvent à l'extérieur de la fenêtre définie ne sont pas affectés.

```
90 X = PEEK(-16384)
```

Lit le clavier. Si X> 127, on a enfoncé une touche et X est la valeur ASCII de la touche enfoncée avec le bit 7 positionné (à un). Cette instruction est utile pour des longs programmes où l'ordinateur vérifié pour voir si l'utilisateur désire interrompre avec de nouvelles données sans arrêter l'exécution du programme.

```
100 POKE -163680
```

Repositionne le signal de sélection du clavier de façon que le caractère suivant puisse être lu; ceci doit se faire immédiatement après avoir lu le clavier.

Instructions qui sont en rapport avec le curseur

```
110 CH = PEEK(36)
```

Lit la position horizontale actuelle de curseur et donne la variable CH égale à cette position. CH va être sur l'étendue 0 à 39 et est la position du curseur par rapport à la marge de gauche et la fenêtre de texte comme on l'a défi ni par POKE 32,L. Par conséquent si la marge de gauche a été défini POKE 32,5 alors le caractère le plus à gauche qui se trouve dans la fenêtre se trouve à la sixième position d'impression à partir du bord gauche de l'écran et si PEEK (36) fournit une valeur de 5 c'est que le curseur était à la onzième position d'impression à partir du bord gauche de l'écran et donc à la sixième position d'impression à partir de la marge de gauche de la fenêtre de texte (cela ne parait pas clair tout d'abord, parce que la position la plus à gauche est

la position zéro et non pas la position 11. Cette instruction est identique à la fonction POS (X). (Voir exemple suivant).

```
120 POKE 36,CH
```

Déplace le curseur pour l'amener à une position qui est la position d'impression d'ordre CH + 1 à partir de la marge de gauche de la fenêtre de texte. (Exemple: POKE 36,0 va faire que le caractère suivant sera imprimé à la marge gauche de la fenêtre). Si la marge de gauche de la fenêtre a été définie à 6 (POKE 32,6) et si vous désirez mettre un caractère à trois positions à partir du bord gauche de l'écran, il faut d'abord modifier la marge gauche de la fenêtre avant d'exécuter l'instruction PRINT. CH doit être inférieur ou égal à la largeur de la fenêtre définie par POKE 22,W et doit être supérieur ou égal à zéro. Comme HTAB cette instruction peut déplacer le curseur audelà de la marge de droite de la fenêtre de texte mais seulement assez loin pour permettre d'imprimer un caractère.

```
130CV = PEEK(37)
```

Lit la position verticale actuelle du curseur et fournit une valeur CV égale à cette position. CV est la position verticale absolue du curseur et ne se rapporte pas aux marges supérieures ou inférieures de la fenêtre de texte. Donc CV=0 est la ligne supérieure de l'écran et CV=23 en est la ligne inférieure.

```
140 POKE 37,CV
```

Déplace le curseur pour l'amener à la position verticale absolue spécifiée par CV. 0 est la ligne supérieure de l'écran et 23 est la ligne inférieure.

Instructions affectant les graphiques

Dans le but de l'affichage du texte et des graphiques, la mémoire de APPLE est divisée en quatre zones: les pages 1 et 2 du texte et les pages 1 et 2 haute résolution.

1) La page 1 de texte est la zone de mémoire habituelle pour tout le texte et pour les graphiques basse résolution utilisée par les instructions TEXT et GR.
2) La page 2 réside juste au-dessus de la page de texte 1 en mémoire. Elle n'est pas facilement lisible à l'utilisateur. Comme pour la page 1 de texte, l'information enregistrée sur la page 2 de texte peut s'interpréter soit comme texte soit comme graphique basse résolution ou les deux.
3) La page 1 graphique haute résolution réside dans la mémoire de APPLE entre 8 K et 16 K. C'est la zone utilisée par l'instruction HGR. Si un texte est affiché avec cette page, ce texte provient de la page de texte 1.
4) La page 2 graphique haute résolution réside dans la mémoire de APPLE entre 16 K et 24 K. C'est la zone utilisée par l'instruction HGR2. Si du texte est affiché avec cette page, ce texte provient de la page 2 de texte.

Pour utiliser les différents modes graphiques et texte, vous pouvez utiliser des instructions texte et graphique de APPLESOFT ou vous pouvez manœuvrer ces quatre sélecteurs différents. Comme pour beaucoup des sélecteurs discutés ici, une instruction PEEK ou POKE à une adresse place le sélecteur d'une façon et la même instruction ou POKE à une deuxième adresse place le sélecteur d'une autre façon.

En bref, ces quatre sélecteurs permettent de choisir entre

1)	Affichage texte	(POKE -16303,0)
2)	Page 1 du texte ou haute résolution	(POKE -16300,0)
	et page 2 du texte ou haute résolution	(POKE -16299,0)
3)	Page 1 ou 2 du texte pour les graphiques	(POKE -16298,0)
	et page 1 ou 2 haute résolution pour les graphiques	(POKE -16297,0)
4)	Graphique haute ou basse résolution sur la totalité de l'écran	(POKE -16302,0)
	et mode mixte graphique haute ou basse résolution + texte	(POKE -16301,0)

150 POKE -16304,0

Cette instruction fait commuter le mode d'affichage pour passer du texte au graphique couleur sans effacer l'écran graphique pour le rendre noir. Selon les positions des trois autres sélecteurs, le mode graphique sur lequel on est ainsi passé peut être basse ou haute résolution, en provenant de la page 1 ou et peut être en mode mixte graphique + texte ou graphique sur la totalité de l'écran.

Instructions APPLESOFT semblables: l'instruction GR fait passer sur page 1 basse résolution, mode mixte graphique + texte et efface l'écran graphique pour le rendre noir. L'instruction HGR fait passer sur page 1 haute résolution, mode mixte graphique + texte et efface l'écran graphique pour le rendre noir. L'instruction HGR2 fait passer sur page 2 haute résolution, graphique sur la totalité de l'écran et efface la totalité de l'écran pour le rendre noir.

160 POKE -16303,0

Commute le mode d'affichage pour passer d'un affichage graphique couleur quelconque au mode entièrement texte sans ramener à l'origine le défilement de la fenêtre. Selon la position du sélecteur page 1 /page 2, la page de texte sur laquelle on est commuté peut être soit la page 1 soit la page 2 de texte.

L'instruction TEXT fait passer en mode entièrement texte, mais en plus choisit la page 1 de texte, ramène au maximum à l'origine le défilement de la fenêtre et positionne le curseur dans l'angle inférieur gauche de l'écran TV.

170 POKE -16302,0

Commute pour passer du mode mixte graphique + texte au mode graphique sur la totalité de l'écran. Selon les positions de autres sélecteurs peut donner un texte, un graphique basse résolution sur la trame 40 par 48 ou un graphique haute résolution sur la trame 278 par 192.

180 POKE -16304,0

Commute pour passer du mode graphique sur la totalité de l'écran au mode mixte graphique + texte, avec quatre lignes de 40 caractères de texte en bas de l'écran.

Selon les positions des autres sélecteurs, la portion supérieure de l'écran peut montrer du texte, du graphique basse résolution sur trame de 40 par 40 ou du graphique haute résolution sur trame de 278 par 160. Les deux portions de l'affichage sur l'écran vont provenir de la page du même numéro (1 ou 2).

184 POKE -16300,0

Commute pour passer de la page 2 à la page 1 sans effacer l'écran ni déplacer le curseur. Cette instruction est nécessaire pour passer en Integer BASIC en venant de APPLESOFT; à défaut, vous pourriez encore regarder à la page 2 de la mémoire.

Selon les positions des autres secteurs, cette instruction peut faire passer l'affichage depuis graphique haute résolution page 2 pour devenir graphique haute résolution page 1, depuis graphique basse résolution page 2 pour devenir graphique basse résolution page 1 ou depuis page de texte page 2 pour devenir page de texte page 1.

186 POKE -16299,0

Commute pour passer de page 1 à page 2 sans effacer l'écran ni déplacer le curseur.

Selon les positions des autres sélecteurs, cette instruction peut modifier l'affichage depuis graphique haute résolution page 1 pour devenir graphique haute résolution page 2, depuis graphique basse résolution page 1 pour devenir graphique basse résolution page 2, ou depuis page de texte 1 pour devenir page de texte 2.

190 POKE -16298,0

Commute la page pour graphique pour passer de la page graphique haute résolution à la même page de texte, sans effacer l'écran. Cette instruction est nécessaire lorsque vous passez en Integer BASIC en venant de APPLESOFT; à défaut, l'instruction Integer BASIC GR peut vous montrer incorrectement la page haute résolution.

Selon les positions des autres sélecteurs, cette instructions peut modifier l'affichage depuis graphique haute résolution page 1 pour devenir graphique basse résolution page 1, depuis graphique haute résolution page 2 pour devenir graphique basse résolution page 2 ou (en mode texte) peut ne pas modifier l'affichage.

195 POKE -16297,0

Commute la page pour graphique depuis une page de texte pour devenir la page correspondante haute résolution, sans effacer l'écran.

Selon les positions des autres sélecteurs, cette instruction peut modifier l'affichage depuis graphique basse résolution page 1 pour devenir graphiques haute résolution page 1, depuis graphique basse résolution page 2 pour devenir graphique haute résolution page 2 ou (en mode texte) peut ne pas modifier l'affichage.

200 CALL –1994

Efface les 20 lignes supérieures de la page texte 1 pour donner les signes inversés @.. Si vous êtes en mode page 1 graphique basse résolution, cette instruction efface 40 lignes supérieures de l'écran graphique pour le rendre noir. Elle n'a pas d'effet sur la page texte 2 ou sur le graphique haute résolution.

205 CALL –1998

Efface la totalité de la page texte 1 pour la donner en signes inversés @. Si vous êtes en mode graphique basse résolution page 1 sur la totalité de l'écran, cette instruction efface la totalité de l'écran pour le rendre noir. Elle n'a pas d'effet sur la page texte 2 ou sur le graphique haute résolution.

200 CALL –62450

Efface l'écran haute résolution en cours (APPLESOFT se souvient de l'écran que vous avez utilisé en dernier lieu, indépendamment des positions des sélecteurs) pour le rendre noir.

210 CALL –62454

Efface l'écran haute résolution en cours (APPLESOFT se souvient de l'écran que vous avez utilisé en dernier lieu, indépendamment des positions des sélecteurs) pour la donner dans la couleur correspondant à l'instruction HCOLOR utilisée la plus récemment dans un tracé. Elle doit être précédé d'un tracé.

Instructions en rapport avec les commandes de jeu et les haut-parleur

220 X = PEEK (-16336)
Bascule une fois l'interrupteur du haut-parleur: produit un "click" en provenance du haut parleur.

225 X = PEEK (-16352)
Bascule une fois l'interrupteur de sortie cassette: produit un "click" en provenance du lecteur de cassette.

230 X = PEEK (-16287)
Lit la position du contact à bouton-poussoir de la commande de jeu #0. Si X> 127 c'est que ce bouton-poussoir est enfoncé.

240 X = PEEK (-16286)
Comme-ci dessus mais pour le bouton-poussoir de la commande de jeu # 1.

250 X = PEEK (-16285)
Comme-ci dessus mais pour le bouton-poussoir de la commande de jeu #2.

260 POKE -16296,1
Met la sortie #0 de l'annonciateur de commande de ,eu (connecteur de jeu I/0 broche 15) au niveau TTL collecteur ouvert haut (3,5 V). C'est la condition off /hors circuit.

270 POKE -16295,0
Met la sortie #0 de commande de jeu au niveau TLL bas (0,3 V). C'est la condition on/en circuit : intensité maxima 1,6 milliampères.

280 POKE -16294,1
Met la sortie de commande de jeu # 1 (connecteur de jeu I/0 broche 14) au niveau TTL haut (3,5 V).

290 POKE -16293,0
Met la sortie de commande de jeu # 1 au niveau TTL bas (0,3 V1.

300 POKE -16292,1
Met la sortie de commande de jeu #2 (connecteur de jeu I/0 broche 13) au niveau TTL haut (3,5 V1.

310 POKE -16291,0
Met la sortie de commande de jeu #2 au niveau TTL bas (0,3 V).

320 POKE -16290,1
Met la sortie de commande de jeu #3 (connecteur de jeu I/0 broche 12) au niveau TTL haut (3,5 V).

330 POKE -16289,0
Met la sortie de commande de jeu au niveau TTL bas (0,3 V1.

Instructions relatives aux erreurs

340 X = PEEK (218) + PEEK (219) * 256
Cette instruction fournit X égal au numéro de ligne de l'instruction où il s'est produit une erreur si une instruction ONERRGOTO a été exécutée.

350 IF PEEK (216) > 127 THEN GOTO 2000
Si le bit 7 à l'emplacement de mémoire 222 (ERRFLG) a été positionné correctement, une instruction ONERRGOTO a été rencontrée.

360 POKE 216,0 -
Efface ERRFLG de sorte que les messages normaux d'erreur sont ceux produits.

370 Y = PEEK (222)

Donne la variable Y selon un code qui décrit le type d'erreur qui était à l'origine de la survenance d'un débranchement ONERRGOTO. Les types d'erreur sont décrits ci-dessous:

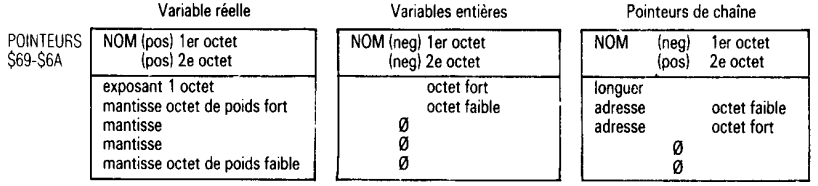
Valeur Y	Type d'erreur rencontré
0	NEXT sans FOR
16	Syntaxe
22	RETURN sans GOSUB
42	En dehors des données/DATA
53	Quantité illégale
69	Déplacement
77	En dehors de la mémoire
90	Instruction non définie
107	Mauvais indice
120	Tableau redimensionné
133	Division par zéro
163	Erreur de frappe
176	Chaîne trop longue -
191	Formule trop complète
224	Fonction non définie
254	Mauvais réponse à une instruction INPUT
255	Tentative de Ctrl C Interrupt

380 POKE 768, 104 : POKE 769,168 : POKE 770,104 : POKE 771,166 : POKE 772, 223 : POKE 773, 154 : POKE 774, 72 : POKE 775, 152 : POKE 776, 72 : POKE 777, 96

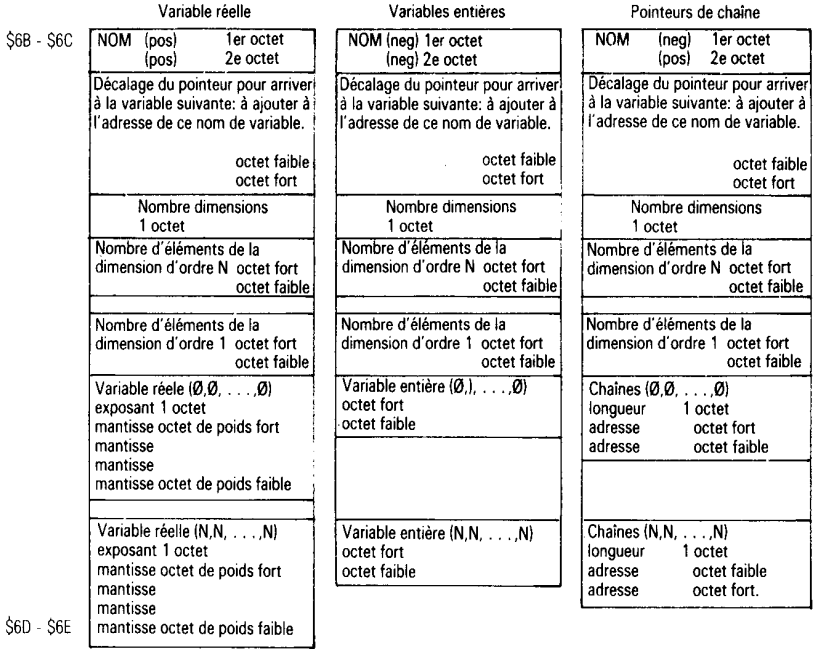
Établit un sous-programme en langage machine à l'emplacement 768, qui peut s'utiliser dans une routine de traitement des erreurs. Dissipe certains des problèmes ONERR/GOTO avec l'instruction PRINT et les messages ?OUT OF MEMORY ERROR. Utilisez l'instruction CALL 768 dans la routine de traitement des erreurs.

Carte des variables APPLESOFT

Variables simples



Variable réelle



Les chaînes sont mémorisées dans leur ordre d'entrée, à partir de HIMEM et en allant vers le bas. Le tableau de chaîne a son pointeur correspondant au premier caractère de chaque chaîne, à la base de la chaîne en mémoire. Lorsque les chaînes sont modifiées, de nouvelles adresses de pointage sont écrites. Lorsque la mémoire disponible est entièrement utilisée, un nettoyage efface toutes les chaînes abandonnées (le nettoyage est déclenché par une instruction FRE (X).

Tous les tableaux sont enregistrés en mémoire avec l'indice le plus à droite progressant le plus lentement; par exemple, on trouve dans l'ordre suivant les nombres qui se trouvent dans le tableau A%11,11 où A%(0,0)=0, A%11,01=1, A%10,11=2, A%11,11=3.

APPENDICE K

Codes de caractère ASCII

DEC = code décimal ASCII
HEX = code hexadécimal ASCII
CHAR = nom du caractère ASCII
n/a = non accessible directement à partir du clavier APPLE II.

DEC	HEX	CHAR	Ce qu'il faut taper	DEC	HEX	CHAR	Ce qu'il faut taper
0	00	NULL	ctrl @	32	20	SPACE	barre espace
1	01	SOH	ctrl A	33	21	!	!
2	02	STX	ctrl B	34	22	"	"
3	03	ETX	ctrl C	35	23	#	#
4	04	ET	ctrl D	36	24	$	$
5	05	ENQ	ctrl E	37	25	%	%
6	06	ACK	ctrl F	38	26	&	&
7	07	BEL	ctrl G	39	27	'	'
8	08	BS	ctrl H ou ←	40	28	(	(
9	09	HT	ctrl I	41	29	)	)
10	0A	LF	ctrl J	42	2A	*	*
11	OB	VT	ctrl K	43	2B	+	+
12	OC	FF	ctrl L	44	2C	,	,
13	0D	CR	ctrl M ou RETURN	45	2D	-	-
14	0E	SO	ctrl N	46	2E	.	.
15	OF	SI	ctrl 0	47	2F	/	/
16	10	DLE	ctrl P	48	30	0	0
17	11	DC1	ctrl 0	49	31	1	1
18	12	DC2	ctrl R	50	32	2	2
19	13	DC3	ctrl S	51	33	3	3
20	14	DC4	ctrl T	52	34	4	4
21	15	NAK	ctrl ou →	53	35	5	5
22	16	SYN	ctrl V	54	36	6	6
23	17	ETB	ctrl W	55	37	7	7
34	18	CAN	ctrl X	56	38	8	8
35	19	EM	ctrl Y	57	39	9	9
26	1A	SUB	ctrl Z	58	3A	:	:
27	1B	ESCAPE	ESC	59	3B	;	;
28	1C	FS	n/a	60	3C	<	<
29	1D	GS	ctrl shift-M	61	3D	=	=
30	1E	RS	ctrl ^	62	3E	>	>
31	1F	US	n/a	63	3F	?	?

DEC	HEX	CHAR	Ce qu'il faut taper
64	40	@	@
65	41	A	A
66	42	B	B
67	43	C	C
68	44	D	D
69	45	E	E
70	46	F	F
71	47	G	G
72	48	H	H
73	49	I	I
74	4A	J	J
75	4B	K	K
76	4C	L	L
77	4D	M	M
78	4E	N	N
79	4F	O	O
80	50	P	P
81	51	Q	Q
82	52	R	R
83	53	S	S
84	54	T	T
85	55	U	U
86	56	V	V
87	57	W	W
88	58	X	X
89	59	Y	Y
90	5A	Z	Z
91	58	[	n/a
92	5C	\	n/a
93	5D	]	] (shift-M)
94	5^E	^	^
95	5F	-	n/a

Les codes ASCII sur l'étendue 96 à 255 vont engendrer des caractères sur APPLE qui répètent ceux de la liste ci-dessus (d'abord ceux de la colonne 2, puis toute la sériel. Bien que CHR $165) donne un A et que CHR$(193) donne également un A, APPLE II ne reconnaît pas les deux comme étant le même caractère lors de leur utilisation comme opérateurs logiques de chaîne et une imprimante reliée à votre ordinateur APPLE les imprimera différemment.

APPENDICE L

Utilisation de la page zéro de la mémoire APPLESOFT

EMPLACEMENT (en hexadécimal)	UTILISATION
$0 - $5	Instructions de branchement à poursuivre dans APPLESOFT (reset 0G return pour APPLESOFT équivalent à reset ctrl C return pour Integer BASIC).
$A - $C	Emplacement pour instruction de branchement de la fonction USR voir description de la fonction USR.
$D - $17	Compteurs/flags d'usage général pour APPLESOFT.
$20 - $4F	Emplacements réservés au moniteur dé système APPLE II.
$50 - $61	Pointeurs d'usage général pour APPLESOFT.
$62 - $66	Résultat de la multiplication /division effectuée en dernier lieu.
$61 - $68	Pointeur indiquant le début du programme. Normalement positionné à $0801 pour la version ROM ou $3001 pour la version RAM (bande cassette).
$69 - $6A	Pointeur indiquant le début de l'emplacement réservé aux variables simples. Indique également la fin du programme plus 1 ou 2, sauf modification demandée par l'instruction LOMEM. -
$6B - $6C	Pointeur indiquant le début de l'emplacement réservé aux tableaux.
$6D - $6^E	Pointeur indiquant la fin de l'emplacement réservé à un enregistrement des valeurs numériques en cours.
$6F - $70	Pointeur indiquant le début de l'emplacement réservé à l'enregistrement des chaînes. Les chaînes sont enregistrées à partir d'ici jusqu'à la fin de la mémoire.
$71 - $72	Pointeur général.
$73 - $74	Emplacement le plus élevé en mémoire disponible pour APPLESOFT plus un. Lors de l'entrée initiale en APPLESOFT, cet emplacement est positionné à l'emplacement le plus élevé disponible de mémoire RAM.
$75 - $76	Numéro de la ligne en cours d'exécution.
$77 - $78	Numéro de l'ancienne ligne. Mis en place par une instruction ctrl C, STOP ou END. Donne le numéro de la ligne où l'exécution a été interrompue.

EMPLACEMENT (en hexadécimal)	UTILISATION
$79 - $7A	Pointeur de l'ancien texte. Indique l'emplacement en mémoire pour la prochaine instruction à exécuter.
$7B - $7C	Numéro de la ligne en cours où on lit actuellement les données.
$7D - $7E	Indique l'emplacement absolu en mémoire où on lit les données.
$7F - $80	Indique la source actuelle des données d'entrée. Positionné à $201 pendant une instruction INPUT. Pendant une instruction READ est position née sur les données lues dans le programme.
$81 - $82	Retient le nom de la variable utilisée en dernier lieu.
$83 - $84	Indique la valeur de la variable utilisée en dernier lieu.
$85 - $9C	Usage général.
$9D - $A3	Accumulateur principal à virgule flottante.
$A4	Usage général dans les programmes mathématiques à virgule flottante.
$A5 - $AB	Accumulateur secondaire à virgule flottante.
$AC - $AE	Flags/pointeurs d'usage général.
$AF - B0	Indique la fin du programme (non modifiée par LOMEM: 1.
$B1 -$C8	Routine CHRGET. APPLESOFT appelle ici chaque fois qu'il désire un autre caractère.
$B8 - $B9	Indique le dernier caractère obtenu au moyen de la routine CHRGET.
$C9 - $CD	Nombre aléatoire.
$D0 - $D5	Pointeurs d'effacement des graphiques haute résolution.
$D8 - $DF	Pointeurs ONERR /effacement.
$E0 - $E2	Coordonnées X et Y en graphique haute résolution.
$E4	Octet de couleur graphique haute résolution.
$E5 - $E7	Usage général pour graphique haute résolution.
$E8 - $E9	Indique le début du tableau de forme.
$EA	Compteur de collisions pour graphique haute résolution.
$F0 - $F3	Signaleurs d'usage général.
$F4-$F8	Pointeurs ONERR.

APPENDICE M

Différences entre APPLESOFT et Integer BASIC

Différences entre les instructions

Les instructions suivantes existent dans APPLESOFT mais non: en Integer BASIC.

ATN
CHR$ COS
DATA DEF FN DRAW
EXP
FLASH FN FRE
GET
HCOLOR HGR HGR2 HIMEM: HOME HPLOT
INT INVERSE
LEFT$ LOG LOMEM:
MID$
NORMAL
ON GO ON...GOTO ONERR GOTO
SUB
POS
READ RECALL RESTORE RESUME RIGHT$ ROT
SCALE SHLOAD SIN SPC SPEED SQR STOP
STORE STR$
TAN
USR
VAL
WAIT
XDRAW

Les instructions suivantes existent dans Integer BASIC mais non dans: APPLESOFT

AUTO
DSP
MAN MOD

Les instructions suivantes ont des noms différents dans les deux langages:

Integer BASIC	APPLESOFT
CLR	CLEAR
CON	CONT
TAB	HTAB (note: APPLESOFT a aussi une instruction TAB)
GOTO X* 10+ 100	ON X GOTO 100, 110, 120
GOSUB X* 100+ 1000	ON X GOSUB 1000, 1100, 1200
CALL-936	HOME (ou CALL -936)
POKE 50,127	INVERSE
POKE 50,255	NORMAL
X	X% (% indique une variable entière)
#	<>or><

Autre différences

En Integer BASIC on vérifie si la syntaxe d'une instruction est correcte lorsque cette instruction est enregistrée dans la mémoire de l'ordinateur (lorsque vous enfoncez la touche RETURN. Dans APPLESOFT, cette vérification s'effectue lorsque l'instruction est exécutée.

Dans APPLESOFT, GOTO et GOSUB doivent être suivis d'un numéro de ligne; Integer BASIC met une variable ou une expression arithmétique.

Les variables et les constantes réelles (nombres à virgule flottante avec virgule décimale et/ou exposant) sont autorisés dans APPLESOFT mais non dans Integer BASIC.

Dans APPLESOFT, seuls les deux premiers caractères d'un nom de variable sont significatifs (par exemple APPLESOFT reconnaît comme étant la même variable les noms GOOD et GOUGE. Dans Integer BASIC tous les caractères d'un nom de variable sont significatifs.
Les opérations sur les chaînes se définissent différemment dans les deux langages. Les chaînes et les tableaux doivent être l'un et l'autre dimensionnés en Integer BASIC; en APPLESOFT, seuls les tableaux doivent être dimensionnés.

En APPLESOFT les tableaux peuvent être multidimensionnels; en Integer BASIC, les tableaux sont limités à une seule dimension.

APPLESOFT positionne tous les éléments d'un tableau à zéro en exécutant RUN, CLEAR, ou RESET. En Inter BASIC le programme utilisateur doit positionner tous les éléments du tableau à zéro.
En Integer BASIC, si la condition posée dans l'instruction IF . . . THEN . . . ne se réalise pas, seule la portion THEN de cette instruction est ignorée. En APPLESOFT, toutes les instructions qui suivent un THEN et se trouvent sur la même ligne sont ignorées si la condition posée par l'instruction IF . . . THEN . . . ne se réalise pas; l'exécution du programme saute à la ligne de programme de numéro suivant.

Dans APPLESOFT, l'instruction TRACE affiche le numéro de ligne de chaque instruction individuelle se trouvant sur une ligne de programme à instruction multiple, et non pas simplement la première instruction comme en Integer BASIC.

Dans APPLESOFT, les instructions CALL, PEEK et POKE peuvent utiliser la totalité de l'étendue des adresses d'emplacement de mémoire (de 0 à 65535) En Integer BASIC, il faut se référer aux emplacements à adresse supérieur à 32767 par leur valeur négative complémentaire (à l'emplacement 32768 s'appelle -32767 -1; l'emplacement 32769 s'appelle -32767; l'emplacement 32770 s'appelle -32766; etc.).

END dans un programme qui s'arrête sur le numéro de ligne le plus élevé est optionnel dans APPLESOFT mais nécessaire dans tous les cas pour éviter un message d'erreur dans Integer BASIC.

NEXT doit être suivi d'un nom de variable dans Integer BASIC; un nom de variable est optionnel dans APPLESOFT.

En Integer BASIC la syntaxe de l'instruction INPUT est
INPUT [string,]{var,}
Si var est une avar, alors INPUT imprime un ? avec ou sans la chaîne optionnelle. Si var est une avar, alors il n'y a pas de ? imprimé, qu'il y ait ou non une chaîne optionnelle. En APPLESOFT, la syntaxe de l'instruction INPUT est INPUT [string;] {ar}
Si la chaîne optionnelle est omise, APPLESOFT imprime un ?; si la chaîne optionnelle est présente, il n'y a pas de ? imprimé.

APPENDICE N

Glossaire alphabétiques des définitions syntaxiques et des abréviations

Voir le chapitre 2 pour une présentation logique (par opposition à alphabétique) de ces définitions. Le symbole := signifie "est au moins partiellement défini comme".

Caractère alphanumérique
:= lettre/chiffre -

alop
:= opérateur logique arithmétique
:= AND IORI=I>I<I><I<>I<=I=<I>=I=>
C'est volontaire que NOT n'est pas compris ici.

aop
:= opérateur arithmétique

avar
:= variable arithmétique '
:= nom I nom%.
Toutes les variables simples occupent 7 octets en mémoire, 2 octets pour le nom et 5 octets pour la valeur réelle ou entière.
:= indice avar -
Dans les tableaux, les nombres réels occupent 5 octets, les nombres entiers 2 octets.

aexpr
:= expression arithmétique -
:= avar I réel I entier
:= indice avar
:= (aexpr)
Si les parenthèses sont emboîtées sur plus de 36 niveaux de profondeur, on obtient le message ? OUT OF MEMORY ERROR
:= [+ I - I NOT] aexpr.
Un NOT unaire apparaît ici, en même temps qu'un + et un - unaire.
:= aexpr slop aexpr
:= sexpr op sexpr

caractère
:= lettre I chiffre I spécial

ctrl
:= maintenir enfoncée la touche marquée "CTRL" tout en enfonçant la touche dont le nom suit.

def
:= mode-exécution différée.

délimiteur
:= ~|(|)|=|-|+|*|^|<|>|/I,I;I:
Un nom n'a pas à être séparé d'un mot réservé qu'il précède ou qu'il suit par l'un de ces délimiteurs.

chiffre
:= 1|2|3|4|5|6|7|8|9|0

esc
:= enfoncer la touche marquée "ESC"

expr
:= expression
:= aexpr | sexpr

imm
:= mode-exécution immédiate

entier
:= [+ |-] {digit} Les entiers doivent être entre -32767 et 32767.
Pour convertir des nombres non entiers en entiers on peut habituellement considérer que APPLESOFT tronque le non-entier pour obtenir l'entier inférieur le plus proche. Toute fois, ce n'est pas tout à fait vrai dans la mesure où le non-entier s'approche de l'entier supérieur le plus proche. Par exemple:

```
A% = 123.999 999 959 999      B% = 123.999 999 96
PRINT A%                      PRINT B%
123                           124

C% = 12345.999 995 999        D% = 12345.999 996
PRINT C%                      PRINT D%
12345                         12346
```

(Les lignes blanches ont été ajoutées pour faciliter la lecture). Un entier élément d'un tableau occupe 2 octets (16 bits) en mémoire.

nom d'une variable entière
:= nom %
Une variable réelle peut être enregistrée sous forme de variable entière, mais APPLE SOFT convertit d'abord la variable réelle en une variable entière.

lettre
:=A|B|C|D|E|F|G|H|I|J|K|L|M|N|O|P|Q|R|S|T|U|V|W|X|Y|Z

ligne
:= linenum [{ instruction: }] instruction return.

linenum
:= numéro de ligne
:= chiffre [{chiffre}]
Les numéros de ligne doivent aller de 0 à 63999 sinon il en résulte un message d'erreur ? SYNTAX ERROR

expression littérale
. = [{caractère}]

lettre minuscule
:= a|b|c|d|e|f|g|h|i|j|k|l|m|n|o|p|q|r|s|t|u|v|w|x|y|z

métanom
:= {métasymbole} [chiffre]

métasymbole
:= caractères utilisés dans ce document pour indiquer qu'ils sont en structures ou relations dans APPLESOFT, mais qui ne font pas partie du langage lui-même.
:= ||[|]|{|}|/|~
:= lettre minuscule
:= un chiffre unique concaténé à un métanom

nom
: = lettre [{ lettre / chiffre }]
Un nom peut avoir une longueur allant jusque 238 caractères. Lorsqu'il faut distinguer un nom d'un autre, APPLESOFT ignore tout caractère alphanumérique situé après les deux premiers. APPLESOFT ne distingue donc pas entre les noms GOOD, 4 LITTLE et GOLDRUSH. Toutefois même la partie ignorée d'un nom ne doit pas contenir un spécial ou un des mots réservés de APPLESOFT.

nom
:= nom d'une variable réelle.

nom%
:= nom d'une variable entière.

nom $
:= nom d'une variable chaîne. chaîne nulle

op
:= opérateur
:= aop | alopl

caractère souffleur
: =] Le crochet à droite (]) est affiché lorsque APPLESOFT est prêt à accepter une instruction.

nombre réel
:= [+ | -] {chiffre} [{chiffre}] [E + | -]chiffre[chiffre]]
:= [+ | -] [{chiffre}].[{chiffre}] [E f+ I -]chiffre [chiffre]]
La lettre E, utilisée dans la notation du nombre réel (une forme de "notation scientifique') signifie "exposant". C'est l'abréviation de 10^. Dix est élevé à la puissance du nombre qui se trouve à la droite de E et le nombre qui se trouve à la gauche de E est multiplié par le résultat.
En APPLESOFT les nombres réels doivent aller de -1E38 à 1E38, sinon vous risquez le message ? OVERFLOW ERROR
En utilisant l'addition et la soustraction il est possible que vous obteniez des nombres de l'importance de 1.7E38 sans recevoir ce message. Un nombre réel dont la valeur absolue est inférieure à environ 2.9388E-39 sera converti par APPLESOFT en zéro.

APPLESOFT reconnaît les nombres suivants comme des nombres réels lorsqu'ils se présentent par eux-mêmes et les évalue pour zéro.

```
+.      -.      .E      +.E     -.E
.E      .E-     +.E-    +.E+    -.E+    -.E-
```

Par conséquent l'élément de tableau M (.) est le même que M (01. En plus des nombres réels abrégés de la liste ci-dessus, les suivants sont reconnus comme nombre réels et évalués pour zéro lorsqu'ils sont utilisés comme réponse numérique à une instruction INPUT ou comme éléments numériques de l'instruction DATA.

```
+       -       E       +E      -E      espace
E+      E-      +E+     +E-     -E+     -E-
```

L'instruction GET évalue tous les nombres réels à simple caractère des listes ci-dessus.

Lorsqu'on imprime un nombre réel, APPLESOFT indiquera au plus 9 chiffres (voir les sections ci-dessous) non compris l'exposant (s'il y en a un). Tous les chiffres qui viennent à la suite sont arrondis. A la gauche du point décimal, tous les zéros qui précèdent le chiffre le plus à gauche qui ne soit pas un zéro ne soient pas imprimés. A la droite du point décimal, tous les zéros qui suivent le chiffre le plus à droite qui ne soient pas un zéro ne soient pas imprimés. S'il n'y a pas à droite du point décimal de chiffres qui ne soient pas des zéros, le point décimal de chiffres qui ne soient pas des zéros, le point décimal n'est pas imprimé.

Le résultat de l'arrondi peut être curieux.

```
PRINT 99 999 999.9
99 999 999.9

PRINT 99 999 999.90
100 000 000

PRINT 11.111 111 451 9
11.111 111 4

PRINT 11.111 111450 00
11.111 111 5
```

(Les espaces blancs sont ajoutés pour faciliter la lecture).

Si la valeur absolue d'un nombre réel est supérieur ou égal à 01 et inférieur 999 999 999.2, le nombre réel est imprimé en notation à point décimal fixe. C'est à dire que l'exposant n'est pas affiché. Sur l'étendue .0 100 000 000 5 à .0 999 899 999, les nombres réels sont imprimés avec dix chiffres maximum, y compris le zéro qui se trouve immédiatement à la droite du point décimal. C'est la seule exception à la limite de 9 chiffres imprimés non compris l'exposant.

Si vous essayez d'utiliser un nombre de plus de 38 chiffres comme 211.11111111111111111111111111111111111

alors le message

```
? OVERFLOW ERROR
```

s'imprime, même si ce nombre se trouve certainement entre -1E38 et 1E38.

Ceci est vrai même si la plus grande partie des chiffres sont des zéros de queue, comme dans

211.00000000000000000 00000000000000000

Les zéros de tête toutefois sont ignorés. Le premier chiffre est un un et si le second chiffre est inférieur ou égal à six, des nombres contenant 39 chiffres peuvent s'utiliser sans donner lieu à message d'erreur.

Un nombre réel occupe 5 octets (40 bits en mémoire).

nom de variable réelle
:= nom

mot réservé
:= certains groupes de caractères utilisés par APPLESOFT pour spécifier dans instructions ou des portions d'instructions. Un nom ne doit pas comprendre un mot réservé. Se reporter à l'appendice G en ce qui concerne une liste des mots réservés de APPLESOFT.

reset
:= enfoncer la touche marquée "RESET".

return
:= enfoncer la touche marquée "RETURN".

sexpr
:= expression 1 chaîne
:= svar chaîne
:= sexpr sop sexpr

slop
:= opérateur logique chaîne
:= =|>|>=|=><|<=|=<|<>|><

sop
:= opérateur chaîne
:= +

special
:= symbole spécial utilisé par APPLESOFT II
:= !|#|$|%|&|'|"|(|)|*|/|^|<|>|=|+|-|:|;|,|.|@|?|]
Les caractères de contrôle (caractères qui sont tapés tout en maintenant enfoncée la touche CTRL et le caractère nul sont également des caractères spéciaux. On peut taper le crochet à droite h) sur le clavier APPLE mais APPLESOFT ne l'utilise que comme caractère indicateur. Dans ce document, il est utilisé comme métasymbole.

indice
:= (aexpr [{,aexpr}]
Le nombre maximum de dimensions (aexpr) est 89, bien qu'en pratique soit limité par la taille de la mémoire disponible. aexpr doit être positif et, pour l'utiliser, est converti en un entier.

chaîne

:= [{caractère}] Une chaîne occupe 2 octets (16 bites) en mémoire pour son pointeur d'emplacement, plus un octet (8 bits) pour chaque caractère de la chaîne.

:= [{ caractère }] return

Cette forme de chaîne ne peut apparaître qu'à la fin d'une ligne.

nom d'une variable chaîne

:= nom$

svar

:= variable chaîne

:= name$ name$ indice Le pointeur d'emplacement et le nom de la variable occupent chacun 2 octets en mémoi re. La longueur et chaque caractère de la chaîne occupent 1 octet.

var

:= variable

:= avar | svar

|

:= métasymbole utilisé pour séparer des alternatives

(note: un item peut également se définir séparément pour chaque alternative)

[]

:= métasymbole utilisés pour isoler un élément optionnel

{}

:= métasymbole utilisés pour isoler un élément qui peut se répéter

:= métasymbole utilisés pour isoler un élément dont il faut utiliser la valeur : la valeur de x s'écrit x.

~

:= métasymbole qui indique un espace nécessaire.

APPENDICE O

Résumé des instructions APPLESOFT

La dernière page de ce manuel contient un index alphabétique qui vous renvoie aux descriptions plus détaillées des instructions APPLESOFT contenues dans les chapitres 3 à 10.

ABS (-3.451)

donne la valeur absolue de l'argument. L'exemple donne 3.451.

Touches à flèche

Les touches repérées d'une flèche à droite ou à gauche sont utilisées pour éditer les programmes APPLESOFT. La touche flèche à droite déplace le curseur vers la droite. Au fur et à mesure qu'il le fait, chaque caractère sur lequel il passe sur l'écran est entré vers la gauche; chaque fois qu'il se déplace, un caractère est effacé dans la ligne de programme que vous êtes en train de taper, quel que soit le caractère sur lequel le curseur se déplace.

ASC ("QUEST")

donne le code décimal ASCII correspondant au premier caractère de l'argument. Dans cet exemple, donne 81 (correspondant à Q dans ASCII).

ATN (2)

donne l'arc tangente, en radians, de l'argument. Dans cet exemple, on obtiendra 1.10714872 (radians).

CALL -922

amène l'exécution de sous-programme en langage-machine qui se trouve à l'emplacement mémoire dont l'adresse décimale est spécifiée. L'exemple va amener un saut de ligne.

CHR $ (65)

donne le caractère ASCII qui correspond à la valeur de l'argument, qui doit être 0 et 255. L'exemple donne la lettre A.

CLEAR

Positionne toutes les variables à zéro et toutes les chaînes à une valeur nulle.

COLOR = 12

définit la couleur pour le tracé en mode graphique basse-résolution. Dans l'exemple la couleur définie est le vert. La couleur est annulée par l'instruction GR. Les noms des couleurs et leur nombres associés sont

0 noir	8 brun
1 magenta	9 orange
2 bleu foncé	10 gris
3 pourpre	11 rose
4 vert foncé	12 vert
5 gris	13 jaune
6 bleu moyen	14 aqua/turquoise clair
7 bleu clair	15 blanc

Pour connaître la couleur d'un point donné de l'écran, utilisez l'instruction SCRN.

CONT

si l'exécution du programme a été arrêtée par STOP, END ctrl C ou reset 0G, l'instruction CONT fait que l'exécution reprend à l'instruction suivante (comme avec GOSUB) et non au numéro de ligne suivant.

Rien n'est effacé. Après reset 0G return, le programme ne peut pas continuer correctement car certains pointeurs et supports du programme sont effacés. L'instruction CONT ne peut pas s'utiliser si vous avez

a) modifié, ajouté ou effacé une ligne du programme ou
b) b) reçu un message d'erreur depuis l'arrêt de l'exécution

COS (2)

donne le cosinus de l'argument, qui doit être en radians. Dans l'exemple on obtient -.415146836

ctrl C

peut s'utiliser pour interrompre un programme en train de passer ou un listage. Peut également s'utiliser pour interrompre une instruction INPUT mais seulement si c'est le premier caractère entré. L'instruction INPUT n'est pas interrompue avant que la touche RETURN ne soit enfoncée.

ctrl X

demande à APPLE II d'ignorer la ligne actuellement en cours de frappe, sans effacer toute ligne précédente de même numéro de ligne. Une barre oblique inverse (\ 1 est affichée à la fin de la ligne à ignorer.

DATA JOHN SMITH, "CODE 32", 23.45, -6

crée une liste d'éléments que l'on peut utiliser au moyen des instructions READ. Dans l'exemple le premier élément est l'expression littérale JOHN SMITH; le deuxième élément est la chaîne "CODE 32"; le troisième élément est le nombre réel 23.45; le quatrième élément est l'entier -6.

DEF FN A(W)=2*W+W

Permet à l'utilisateur de définir dans un programme des fonctions de une ligne. II faut tout d'abord définir la fonction à l'aide de DEF; puis par la suite dans le programme on peut utiliser la fonction précédemment définie à l'aide de l'instruction DEF. L'exemple illustre comment définir une fonction FN A(W); elle peut s'utiliser par la suite dans le programme sous la forme FN A (23) ou FNA (-7*O+ 1) et ainsi de suite. FN A123) va faire que W soit remplacé par READ dans 2*W+W; la fonction sera alors évaluée à 2*23+23 ou 69. Supposons O = 2; alors FN AI7*Q+ 1) est équivalent à FN AI-7*2+ 1) ou FN AI-131: la fonction sera évaluée pour 2*(-13)+(-13) ou -26-13 ou 39.

DEL 23,56

enlève l'étendue spécifiée de lignes dans le programme. Dans l'exemple les lignes 23 à 56 seront effacées dans le programme. Pour effacer une seule ligne, disons la ligne 350, utilisez la forme DEL 350,350 ou tapez simplement le numéro de ligne puis enfoncez la touche RETURN.

DIM AGE (20,3), NAMES (50)

Lorsqu'une instruction s'exécute, elle réserve la place du tableau spécifié, avec des indices DIM de 0 à l'indice indiqué. Dans l'exemple à NAMES 150) seront réservées 50 + 1 ou 51 chaînes de longueur quelconque; au tableau AGE (20,3) seront réservés (20+ 1)*(3+ 1) ou 21 *4 ou 84 éléments nombres réels. Si un élément de tableau est utilisé dans le programme avant d'avoir été dimensionné, un indice de valeur maxima 10 est réservé pour chaque dimension dans les indices de l'élément. Les éléments du tableau sont positionnés à zéro en exécutant RUN ou CLEAR.

DRAW 4 AT 50, 100

dessine une forme dont le numéro de définition est 4 à partir d'un tableau de forme précédemment chargé, en graphique haute-résolution, en partant de x = 50, y = 100. La couleur, la rotation et l'échelle de la forme à dessiner doivent avoir été spécifiées avant exécution de l'instruction DRAW.

END

amène un programme à cesser son exécution et rend le contrôle à l'utilisateur. Aucun message n'est imprimé.

esc A ou esc B ou esc C ou esc D

la touche peut s'utiliser en liaison avec les touches des lettres A. B ou C ou D pour déplacer le curseur sans affecter les caractères sur lesquels passe le curseur. Pour déplacer le curseur d'un espace, enfoncez d'abord la touche Escape puis relâchez cette touche et enfoncez la touche de la lettre appropriée.

Commande	**Déplace le curseur d'un espace**
esc A	vers la droite
esc B	vers la gauche
esc C	vers le bas
esc D	vers le haut

EXP (2)

donne la valeur "e" de élevée à la puissance* indiquée par l'argument. Avec 6 décimales, e=2.718289, et dans cet exemple on aura 7.3890561

FLASH

règle le mode vidéo en clignotement, le signal émis par l'ordinateur apparaissant alternativement sur l'écran TV en caractères blancs sur fond noir puis inversement en caractères noirs sur fond blanc. Utilisez l'instruction NORMAL pour revenir à un affichage non clignotant en lettres blanches sur fond noir.

FOR W= 1 TO 20 . . . NEXT W
FOR 0=2 TO -3 STEP -2 . . . NEXT Q
FOR Z=5 TO 4 STEP 3 . . . NEXT Z

vous permet d'écrire une boucle pour effectuer un nombre spécifié de fois les instructions quelconques qui se trouvent entre l'instruction FOR (la tête de la boucle) et l'instruction NEXT (la base de la boucle) Dans le premier exemple la variable W compte combien de fois il faut exécuter les instructions; les instructions qui se trouvent à l'intérieur de la boucle seront exécutées pour W égale à 1, 2, 3 . . . 20, puis la boucle s'arrête (avec W=21) et c'est l'instruction qui se trouve après NEXT W, qui s'exécute. Le second exemple illustre comment indiquer le pas (STEP) si vous voulez qu'il soit différent de 1. La vérification se place à la fin de la boucle; par conséquent dans le troisième exemple les instructions qui se trouvent à l'intérieur de la boucle ne seront exécutées qu'une seule fois.

FRE (0)

fournit la valeur de la mémoire, en octets, dont dispose encore l'utilisateur. Ce que vous mettez à l'intérieur des parenthèses n'a pas d'importance dans la mesure où il peut être évalué par APPLESOFT.

GET ANS $

amène un caractère unique du clavier sans le montrer sur l'écran TV et sans qu'il soit besoin d'enfoncer la touche RETURN. Dans l'exemple le caractère tapé au clavier est enregistré dans la variable ANS $.

GOSUB 250

amène le programme à se brancher sur la ligne indiquée (250 par exemple). Lorsque l'on exécute une instruction RETURN le programme se branche sur l'instruction qui suit immédiatement la plus récente instruction GOSUB exécutée.

GOTO 250

amène le programme à se brancher sur la ligne indiquée (250 dans l'exemple).

GR

commute en mode graphique basse-résolution (40 X 40) pour l'écran TV, en laissant 4 lignes de texte en bas du graphique. L'écran est effacé, devient noir, le curseur est amené dans la fenêtre de texte et la couleur est réglée à 0 (noir).

HCOLOR = 4

positionne la couleur du graphique haute résolution à la couleur spécifiée par HCOLOR. Les noms de couleur et leurs valeurs associées sont:
0 noir
1vert (selon votre TV)
2 bleu (selon votre TV)
3 blanc 1
4 noir 2
5 (selon votre TV)
6 (selon votre TV)
7 blanc 2

HGR

n'existe que dans la version sur carte de APPLESOFT. Positionne le mode graphique haute-résolution (281 X 160) pour l'écran en laissant 4 lignes pour le texte en bas de l'écran. L'écran est effacé pour passer au noir et la page 1 de la mémoire est affichée. L'exécution de HGR n'affecte ni l'instruction HCOLOR ni la mémoire de la partie de l'écran réservée au texte. Le curseur n'est pas déplacé dans la fenêtre de texte.

HGR 2

positionne le mode graphique haute résolution sur la totalité de l'écran (281 X 1921. L'écran est effacé pour revenir au noir et la page 2 de la mémoire est affichée. La mémoire de l'écran correspondant au texte n'est pas affectée.

HIMEM: 16384

positionne l'adresse de l'emplacement en mémoire le plus élevé disponible pour un programme APPLESOFT, y compris les variables. S'utilise pour protéger les zones de mémoire pour les données, les graphiques haute résolution ou les routines de langage machine. HIMEM n'est pas remis à l'état initial par CLEAR, RUN, NEW, DEL par le fait de modifier ou d'ajouter une ligne de programme ou par reset.

HLIN 10, 20, AT 30

utilisé pour tirer des droites horizontales en mode graphique basse résolution en utilisant la couleur la plus récemment spécifiée par COLOR. L'origine (x = 0 et y = O) du système est le point supérieur le plus à gauche de l'écran. Dans l'exemple on tire la droite allant de x 10 à x = 20 pour y = 30. Autre façon de s'exprimer: on tire la droite allant du point (10, 30) au point (20, 301.

HOME

amène le curseur dans la position et à gauche de l'écran dans la fenêtre de texte et efface tout le texte qui se trouve dans la fenêtre.

HPLOT 10,20
HPLOT 3, 40 TO 50, 60
HPLOT TO 70, 80

trace des points et des droites en mode graphique haute résolution en utilisant la valeur de HCOLOR la plus récemment spécifié. L'origine est le point supérieur le plus à gauche de l'écran (x = 0, y = 01. Le premier exemple trace un point haute résolution à x = 10, y = 20. Le second exemple trace une droite haute résolution allant du point x = 30, y = 40 au point x = 50,y = 60. Le troisième exemple trace une droite partant de ce dernier point tracé et allant au point x = 70, y = 80, en utilisant la couleur du dernier point tracé et non pas nécessairement la plus récente instruction HCOLOR.

HTAB 23

déplace le curseur soit vers la gauche, soit vers la droite pour venir dans la colonne spécifiée (1 à 40) sur l'écran. Dans l'exemple le curseur sera positionné dans la colonne 23.

IF AGE< 18 THEN A=0: B= 1: C=2
IF ANS $ "YES" THEN GOTO 100
IF N<MAX THEN 25
IF N<MAX GOTO 25

Si l'expression qui suit IF s'évalue comme vraie (i.e. non zéro), alors la ou les instructions qui suivent THEN sur la même ligne seront exécutées. Sinon toutes les instructions qui suivent THEN sont ignorées et l'exécution passe à l'instruction de la ligne de numéros suivants du programme. Les expressions chaînes sont évaluées par ordre alphabétique. Les exemples 3 et 4 se comportent de la même façon malgré la façon différente de s'exprimer.

INPUT A%
INPUT "TYPE AGE THEN COMMA THEN NAME": B, C$

dans le premier exemple, INPUT imprime un point d'interrogation, et attend que l'utilisateur tape un numéro, qui sera assigné à la variable entière A%. Dans le deuxième exemple, INPUT imprime la chaîne optionnelle exactement comme elle est vue puis attend que l'utilisateur tape un numéro (qui sera assigné à la variable réelle B) puis une virgule, puis l'entrée chaîne (qui sera assignée à la variable chaîne C$. Des entrées multiples pour INPUT peuvent être séparées par des virgules ou des returns.

INT (NUM)

fournit l'entier le plus grand inférieur ou égal à l'argument donné. Dans l'exemple si NUM est 2.389, on obtiendra 2; si NUM est -45.123345, on obtiendra -46.

INVERSE

Commute le mode vidéo de façon que les signaux provenant de l'ordinateur s'impriment en lettres noires sur fond blanc. Utilisez NORMAL pour revenir aux lettres blanches sur fond noir.

IN# 4

Spécifie le slot (de 1 à 7) du périphérique qui fournira l'entrée suivante pour l'ordinateur. In #0 rétablit l'entrée en provenance du clavier au lieu du périphérique.

LEFT $ ("APPLESOFT" 5)

donne le nombre spécifié de caractères les plus à gauche dans la chaîne. Dans l'exemple on aura APPLE (les cinq caractères les plus à gauche).

Flèche à gauche

voir touches à flèche

LEN ("AN APPLE DAY")

fournit le nombre de caractères d une chaîne entre 0 et 255. Dans l'exemple on obtiendra 14.

LET A = 23.567
A$= "DELICIOUS"

le nom de la variable qui se trouve à gauche de = est assignée à la valeur de la chaîne ou .à l'expression qui se trouve à la droite de =. Le LET est optionnel.

LIST
LIST 200 – 3000
LIST 200, 3000

Le premier exemple amène la totalité du programme à être affichée sur l'écran TV; le second exemple amène les lignes du programme 200 à 3000 à être affichées. Pour obtenir l'affichage du début du programme jusqu'à la ligne 200, utilisez LIST -200; pour obtenir l'affichage de la

ligne 200 à la fin du programme, utilisez LIST 200-. Le troisième exemple donne le même résultat que le second. L'instruction LIST est annulée par ctrl C.

LOAD

lit un programme APPLESOFT sur bande cassette pour l'enregistrer dans la mémoire de l'ordinateur. II n'est pas fourni de caractères indicateurs: l'utilisateur doit rembobiner la bande et enfoncer "play" sur le lecteur de cassette avant l'exécution LOAD. On entend un bip lorsque l'information est trouvée sur la bande à charger. Lorsque le chargement a été correctement exécuté, on entend un second bip et on obtient le caractère h) de APPLESOFT. Un reset peut seul interrompre une instruction LOAD.

LOG (2)

donne le logarithme naturel de l'expression arithmétique spécifiée. Dans l'exemple on obtient .69314 7181.

LOMEM: 2060

positionne l'adresse de l'emplacement de mémoire le plus faible disponible pour un programme BASIC. Ceci permet de protéger les variables provenant des graphiques haute résolution dans les ordinateurs à grande capacité de mémoire.

MID $ ("AN APPLE A DAY", 4)
MID $ ("AN APPLE A DAY", 4,9)

donne la sous-chaîne spécifiée. Dans le premier exemple on obtiendra du 4e au dernier caractère de la chaîne: APPLE A DAY. Dans le second exemple on obtiendra les 9 caractères qui commencent au 4e caractère de la chaîne: APPLE A D.

NEW

efface le programme en cours et toutes les variables.

NEXT

voir la discussion de FOR . . . TO . . . STEP.

NORMAL

Commute le mode vidéo pour donner les lettres blanches habituelles sur fond noir à la fois pour les signaux d'entrée et de sortie.

NOTRACE

met hors-circuit le mode TRACE. Voir TRACE.

ON ID GOSUB 100, 200,23,4005,500

exécute un GOSUB pour rejoindre le n' de ligne indiqué par la valeur de l'expression arithmétique qui suit ON. Dans l'exemple si ID est un, c'est GOSUB 100 qui est exécuté; si ID est 2, c'est GOSUB 200 qui est exécuté et ainsi de suite. Si la valeur de l'expression est 0 ou est supérieur au nombre du numéro de ligne variantes de la liste, l'exécution du programme passe à l'instruction suivante.

ON ID GOTO 100, 200, 23, 4005, 500

Identique à ON ID GOSOB (voir ci-dessus) mais exécuté un GOTO branchement au numéro de ligne indiqué par la valeur de l'expression arithmétique qui suit ON.

ONERR GOTO 500

utilisé pour éviter un message d'erreur qui arrête l'exécution lorsqu'une erreur se produit. Lors de son exécution, ONERR GOTO positionne un flag qui cause un saut inconditionnel sur le numéro de ligne indiqué (500 dans l'exemple) si une erreur se rencontre par la suite.

PDL (3)

donne la valeur en cours, un nombre entre 0 et 255, de la commande de jeu indiquée. Les numéros de commande de jeu 0 à 3 sont valides.

PEEK (37)

fournit les contenus, en décimales, de l'octet qui se trouve à l'adresse décimale spécifiée (37 par exemple).

PLOT 10, 20

en mode graphique basse résolution, place un point à l'emplacement spécifié. Dans l'exemple le point sera en X = 10, Y = 20. La couleur du point est déterminée par la valeur la plus récente de COLOR, qui est 0 (noir) si aucune valeur n'a été spécifiée précédemment.

POKE -16302, 0

enregistre l'équivalent binaire du second argument (0 dans l'exemple) dans l'emplacement de mémoire dont l'adresse décimale est donnée par le premier argument (-16302 dans l'exemple)

POP

entraîne l'élimination de la dernière adresse de retour de l'instruction RETURN. Le prochain RETURN que l'on rencontrera après un POP effectuera un branchement sur une instruction qui se trouve au-delà du second GOSUB exécuté en dernier lieu.

POS (O)

donne la position horizontale en cours du curseur. C'est un nombre allant de 0 (sur la marge gauche) à 39 (sur la marge droite). Ce que vous mettez dans les parenthèses n'a pas d'importance pour autant qu'il puisse être évalué par APPLESOFT. -

PRINT
PRINT A$; "X = " ; X

le premier exemple amène l'impression d'une ligne et un retour du chariot sur l'écran. Les items d'une liste à afficher dorent être séparés par des virgules si chacun doit être affiché dans un champ distinct du tabulateur. Les items doivent être séparés par des points et virgules s'ils doivent être imprimés immédiatement à droite l'un de l'autre sans aucun espace blanc intermédiaire. Si A$ contient "COKE" et si X vaut 3, le second exemple donne l'affichage de COREX = 3

PR# 2

Transfère les signaux de sortie au slot spécifié, de 1 à 7. PR#0 renvoie les signaux de sortie vers l'écran TV.

READ A, B%, C$

l'exécution de cette instruction assigne aux variables de l'instruction READ les valeurs successives des éléments qui se trouvent dans les instructions DATA du programme. Dans l'exemple, les deux premiers éléments des instructions DATA doivent être des nombres et le troisième doit être une chaire (qui peut être un nom). Ils seront assignés respectivement aux variables A,B% et C$.

RECALL MX

rappelle un tableau de réels ou d'entier s qui a été enregistré sur une bande cassette. Un tableau peut être rappelé par un nom différent de celui utilisé lorsqu'on l'a enregistré sur la bande. Si on le rappelle, MX doit avoir été dimensionné par le programme. Les indices ne sont as utilisés par STORE ou RECALL. Dans l'exemple c'est le tableau dont les éléments sont MX (0), MX(1), . . . qui sera rappelé; la variable non indicée MX ne sera pas affectée. Aucun signal souffleur ou autre n'est fourni: vous devez enfoncer la touche 'play' sur le lecteur de cassette pour exécuter l'instruction RECALL; un bip signale le début et la fin du tableau enregistré. Un reset peut seul interrompre un RECALL.

REM THIS A REMARK

permet d'insérer un texte dans un programme sous forme de remarque . ... ,
repeat si vous maintenez enfoncée la touche repeat, indiquée REPT, tout en enfonçant une touche de caractère quelconque, ce caractère sera répété.

RESTORE
repositionne le pointeur de la liste de données sur le premier élément de DATA. Amène la prochaine instruction READ rencontrée à relire les instructions de DATA, à partir de la première.

RESUME
à la fin d'une routine de traitement d'erreur (voir ONERR GOTO), amène la reprise du programme à l'instruction où l'erreur s'est produite.

RETURN
effectue un branchement sur l'instruction qui suit immédiatement l'instruction GOSUB exécutée le plus récemment.

RIGHT $ ("SCRAPPLE", 5)
fournit le nombre spécifié de caractères les plus à droite de la chaîne. Dans l'exemple on obtiendra APPLE (les cinq caractères les plus à droite).
Flèche à droite (voir touches à flèche)

RND (S)
donne un nombre réel aléatoire supérieur ou égal à 0 et inférieur à 1. RND 10) donne le nombre aléatoire engendre en dernier lieu. Chaque argument négatif engendre un nombre aléatoire particulier qui est le même chaque fois que l'on utilise RND avec cet argument et les RND suivants à arguments positifs vont toujours suivre une séquence particulière répétitive. Chaque fois que l'on utilise RND avec un argument positif quelconque, on engendre un nouveau nombre aléatoire entre 0 et 1, à moins qu'il ne fasse partie d'une séquence de nombres aléatoires débutée par un argument négatif.

ROT = 16
définit un angle de rotation pour la forme à dessiner par DRAW ou XDRAW. ROT = 0 fait que la forme doit être dessinée orientée comme elle a été définie. ROT = 0 fait que la forme doit être dessinée après rotation de 90' sens d'horloge, etc . . . Le processus se répète à partir de ROT = 64.

RUN 500
efface toutes les variables les pointeurs et les piles et commence l'exécution au numéro de ligne indiqué (500 dans l'exemple). Si aucun numéro de ligne n'est spécifié, l'exécution commence au numéro de ligne le plus faible de programme.

SAVE
enregistre un programme sur une bande cassette. Aucun signal indicateur ou autre n'est fourni: l'utilisateur doit enfoncer "record" et "play" sur le lecteur de cassette avant d'exécuter l'instruction SAVE. SAVE ne vérifie pas que les boutons corrects du lecteur de cassette soient enfoncés; des bips signalent le début et la fin de l'enregistrement.

SCALE = 50
donne l'échelle pour la forme à dessiner par DRAW ou XDRAW. SCALE = 1 définit une reproduction point pour point de la définition de la forme. SCALE = 255 se traduit par le fait que chaque vecteur à tracer est agrandi 255 fois. Note: SCALE = 0 est l'échelle maxima et non un simple point.

SCRN (10,20)
en mode graphique basse résolution, donne le mot couleur du point spécifié. Dans l'exemple on obtient la couleur du point situé à x = 10, y = 20.

SGN (NUM)
donne moins 1 si l'argument est négatif, 0 si l'argument est 0 en 1 si l'argument est positif.

SHLOAD
charge un tableau de forme à partir de la bande en cassette. Le tableau de forme est chargé juste en dessous de HIMEM : puis HIMEM : est placé juste sous le tableau de forme pour le protéger.

SIN (2)
donne le sinus de l'argument, il doit être en radians. Dans l'exemple, on obtient 909297427.

SPC (8) -
doit s'utiliser dans une instruction PRINT. Introduit le nombre spécifié d'espaces blancs (8 dans l'exemple), entre le dernier item affiché et le suivant. Si des points et virgules précèdent et suivent l'instruction SPC.

SPEED = 50
définit la vitesse à laquelle les caractères doivent être envoyées à l'écran ou autres dispositifs d'entrée/sortie. La vitesse la plus faible est 0; la plus rapide est 255.

SQR (2)
donne la racine carrée positive de l'argument; dans l'exemple on obtient 1.414 21356. SQR exécute plus rapidement que ^.5.

STOP
amène un programme à cesser son exécution et affiche un message indiquant quel numéro de ligne contenait l'instruction STOP. Le contrôle de l'ordinateur est renvoyé à l'utilisateur.

STORE MX
enregistre un tableau d'entiers ou de réels sur bande. Aucun message indicateur ou autre n'est fourni: l'utilisateur doit enfoncer "record" et "play" sur le lecteur enregistreur de cassettes pour exécuter l'instruction STORE. Des bips signalent le début et la fin de l'enregistrement de la fin. L'indice du tableau
n'est pas indiqué lorsqu'on utilise l'instruction STORE. Dans l'exemple les éléments MX (0), MX(1), MX(2), . . . sont sauvegardés par enregistrement sur bande; la variable MX n'est pas affectée. Voir RECALL.

STR $ (12.45)
fournit une chaîne qui représente la valeur de l'argument. Dans l'exemple on obtient la chaîne ' 12.45'.

TAB (23)
doit s'utiliser dans une instruction PRINT. L'argument doit être entre 0 et 255 et mis entre parenthèses. Pour les arguments de 1 à 255, si l'argument est supérieur à la valeur de la position en cours du curseur, alors l'instruction TAB déplace le curseur pour l'amener à la position d'affichage spécifiée, en comptant à partir du bord gauche de la ligne en cours du curseur. Si l'argument est inférieur à la valeur de la position en cours du curseur, alors le curseur ne se déplace pas. TAB (0) amène le curseur en position 256.

TAN (2)
donne la tangente de l'argument qui doit être en radians. Dans l'exemple on obtient - 2.18503987.

TEXT
commute l'écran sur le mode usuel texte non graphique, avec 40 caractères par ligne et 24 lignes. Repositionne également la fenêtre de texte sur la totalité de l'écran.

TRACE

amène le n° de ligne de chaque instruction à être affiché sur l'écran au moment de son exécution. TRACE n'est pas annulé par RUN, CLEAR, NEW DEAL ou reset, mais par NOTRACE.

USR (3)

cette fonction envoie son argument dans un sous programme de langage machine. L'argument est évalué et placé dans l'accumulateur à point décimal flottant (emplacements $9D à $A3) et il s'établit un SSR renvoyant à l'emplacement SDA. Les emplacements S0A à SOC doivent contenir un INP renvoyant à l'emplacement de début du sous-programme de langage machine. La valeur obtenue pour la fonction est placée dans l'accumulateur à point maximal flottant. Pour revenir à APPLESOFT, exécutez un RTS.

VAL("-3.7E4A5PLE")

s'efforce d interpréter une chaîne, jusqu'au premier caractère non-numérique, comme un réel ou un entier et fournit la valeur de ce nombre. Si aucun nombre n'apparaît avant le premier caractère non-numérique, on obtient un 0. Dans l'exemple on obtient -37000.

VLIN 10, 20 AT 30

dans le mode graphique basse résolution, tire une droite verticale dans la couleur indiquée par la plus récente instruction COLOR. La droite est tirée dans la colonne indiquée par le troisième argument. Dans l'exemple la droite est tirée de y = 10 à y = 20 pour x = 30.

VTAB (15)

déplace le curseur à la ligne de l'écran spécifiée par l'argument. La ligne de tête est la ligne 1; la ligne inférieure est la ligne 24; VTAB va déplacer le curseur vers le haut ou vers le bas mais non vers la gauche ou vers la droite.

WAIT 16000, 255
WAIT 16000, 255,0

permet d'insérer dans le programme une pause conditionnelle. Le premier argument est l'adresse décimale de l'emplacement en mémoire à tester pour voir si certains bits ont la valeur 1 et certains bits ont la valeur 0. Chaque bit de l'équivalent binaire du second argument décimal indique si vous êtes ou non intéressé au bit correspondant à l'emplacement en mémoire: la valeur 1 signifie que vous êtes intéressé, la valeur 0 signifie que vous devez ignorer ce bit. Chaque bit de l'équivalent binaire du troisième argument décimal indique quel état vous attendez (WAIT) pour le bit correspondant à l'emplacement en mémoire: la valeur 1 signifie que le bit doit avoir la valeur 0, la valeur 0 signifie que le bit doit avoir la valeur 1. S'il n'y a pas de troisième argument, on suppose la valeur 0. Si l'un quelconque des bits indiqués par un bit de valeur 1, dans le second argument correspondent à l'état pour ce bit indiqué par le bit correspondant du troisième argument, l'instruction WAIT est terminée.

XDRAW 3 AT 180, 120

dessine la forme définie de N' 3 à partir d'un tableau de formes précédemment chargé, en graphiques haute résolution, en commençant au point x = 180, y = 120. Pour chaque point tracé, la couleur est le complément de la couleur qui existe défia en ce point. Fournit un moyen facile pour effacer: si vous appliquez l'instruction XDRAW à une forme, puis encore une fois, vous effacerez la forme sans effacer le fond.

INDEX