POM's – ÉDITIONS MEV 12, RUE D'ANJOU 78000 VERSAILLES
✆ (1) 39 51 24 43

Le compilateur BEAGLE
d'Alan Bird

Édité par Beagle Bros Micro Software, Inc.
6215 Ferris Square, Suite 100
San Diego, California 92110 U.S.A.

Accélérateur de programmes Applesoft

Table des matières

Introduction 2
Pour vendre des programmes compilés 4

Comment faire pour...
...Exécuter un programme Applesoft à la vitesse du langage machine 6
...Sauvegarder un programme en format compilé 7
...Lancer un programme compilé depuis un disque 8
...Modifier un programme compilé 8
...Faire un programme interruptible par CTRL-C 9
...Exécuter un programme à vitesse normale 9
...Changer l'adresse de chargement d'un programme 10
...Obtenir des programmes compilés encore plus rapides 10
...Compiler des programmes qui utilisent la commande CHAIN 11
...Compiler des programmes qui utilisent les commandes STORE et RESTORE 12
...Écrire des routines ampersand pour programmes compilés 12

Erreurs 13
Compiler et compiler.system 16
Les autres programmes du disque 17
Modifier l'Applesoft à l'aide du compilateur 19
Utiliser le compilateur avec une extension mémoire 24

Introduction

Bienvenue !

Le compilateur Beagle est un utilitaire qui "réécrit" les programmes en Basic Applesoft, de telle façon qu'ils s'exécutent plus rapidement, exactement comme s'ils avaient été écrits en langage machine. Les programmes en langage machine s'exécutent bien plus rapidement que les programmes Applesoft parce qu'il n'y a pas de temps perdu à interpréter les commandes-utilisateur telles HOME, GOTO, IF, THEN, etc.
Bien que le compilateur Beagle ne convertisse pas vraiment le Basic en langage machine (en réalité il les convertit dans son *propre* langage), le résultat est le même.

Le Basic Applesoft est peut-être plus lent que le langage machine, mais il est beaucoup plus facile à utiliser (pour la plupart d'entre nous) – et il est bien plus lisible. Le compilateur Beagle vous donne tout : la simplicité du Basic *et* la rapidité du langage machine.

Sauvegardez-le

Le compilateur Beagle n'est pas protégé contre les copies, ainsi vous pouvez (et vous devriez) faire une sauvegarde au cas où un problème surviendrait sur l'original. Utilisez le programme de copie qui vous a été livré avec votre Apple ou bien le programme DISK.COPY (35 secondes) de la disquette Beagle Extra K. Vous pouvez également copier des fichiers d'un volume à l'autre en utilisant le programme FILEMOVER de la disquette Beagle Big U ou l'un des nombreux utilitaires Apple.
De grâce, ne donnez pas de copies des disquettes et programmes Beagle à vos amis. Chaque copie pirate *encourage* la protection des logiciels et *décourage* les éditeurs de logiciels conviviaux. Si vous avez l'intention de donner des copies de vos programmes compilés, reportez-vous à la page 4.

Ce que fait le compilateur

Exactement ce qu'en dit la publicité : «Après avoir démarré sur la disquette du compilateur Beagle, vous pouvez exécuter pratiquement tous les programmes Applesoft à la vitesse du langage machine. *Vite* !»

À quelle vitesse ?

Un simple essai montre que les programmes compilés s'exécutent de 2 à 15 fois plus vite que les programmes Basic ; tout dépend de ce que fait le programme. Certaines fonctions telles les manipulations de chaîne et de variables numériques sont considérablement accélérées. D'autres, comme les calculs en virgule flottante ne changent pas du tout.

Quand vaut-il mieux ne pas compiler ?

La vitesse du langage machine n'est pas toujours un avantage ; certains programmes, programmes interactifs par exemple (questions et réponses), fonctionnent fort bien avec le bon vieil Applesoft. Une vitesse trop élevée en rendrait l'utilisation impossible.
Des programmes gagnent à être compilés, mais par endroit doivent être ralentis. Comme vous ne pouvez pas compiler qu'une partie d'un programme, vous devrez faire quelques modifications en Basic avant de compiler.

Ce qu'on ne peut pas compiler

La plupart des programmes Applesoft se compilent sans problème. Il arrive que certains programmes soient trop grands pour être compilés ou comportent des ordres incompatibles avec la compilation.

Les programmes non Applesoft ne peuvent être compilés (par exemple vous ne pourrez pas compiler AppleWorks car il n'est pas écrit en Applesoft).

Les programmes protégés ne pourront pas être compilés tant que vous n'aurez pas enlevé la protection. Ne nous demandez pas comment, on ne sait pas.

Les programmes en DOS 3.3 se compilent en principe après conversion en ProDOS (utilisez l'un des programmes Apple pour la conversion). Vérifiez bien que le programme une fois converti fonctionne encore avant de le compiler.

Les avantages du compilateur Beagle

Le compilateur Beagle est supérieur à tous les compilateurs que nous avons vus. Et nous en avons vu un certain nombre...

- les autres compilateurs ne fonctionnent pas sous ProDOS ;
- les autres compilateurs ne compilent pas sur le champ en faisant simplement RUN ;
- les autres compilateurs génèrent un code bien plus volumineux que le programme original ;
- avec le compilateur Beagle, c'est l'inverse ;
- pour les autres compilateurs, le temps de compilation se compte en minutes et non en secondes. Et cela ne vous empêche pas d'avoir des problèmes ;
- les autres compilateurs butent sur des instructions classiques de l'Applesoft comme HIMEM, LOMEM, DEF FN, etc. ;
- les autres compilateurs demandent plus de modifications au programme Basic que le compilateur Beagle. Par exemple :

```
10 MAX = 100: DIM A$(MAX),B$(MAX),C$(MAX*2)
```

D'autres compilateurs verront d'un mauvais œil la ligne de programme ci-dessus et vous demanderont de la remplacer par :

```
10 MAX = 100: DIM A$(100),B$(100),C$(200)
```

et il vous faudra alors recompiler.

Il y a des contraintes de capacité mémoire

Démarrer sur la disquette du compilateur Beagle occupera 11 Ko de la mémoire principale. Vous pourrez diviser ce chiffre par deux en ne chargeant que l'un des deux fichiers du compilateur. Reportez-vous page 17.

Un mot sur les éditeurs de programme

Les programmeurs en Basic utilisent toutes sortes d'éditeurs. Malheureusement, vous ne pouvez pas avoir en mémoire des programmes non relogeables tels que le GPLE de Beagle Bros avec le fichier COMPILER.SYSTEM — désolé, mais il n'y a plus de place. En revanche, Program Writer d'Alan Bird, lui, fonctionnera bien.
Si vous voulez utiliser GPLE, démarrez normalement avec GPLE pour écrire et essayer les programmes, puis démarrez sur la disquette compilateur pour les exécuter. Le fichier COMPILER peut cohabiter en mémoire avec GPLE, dans la mesure où vous chargez d'abord GPLE, puis le fichier COMPILER.
Pour utiliser Program Writer, vous devrez charger les fichiers dans le bon ordre :
1- le fichier COMPILER.SYSTEM
2- Program Writer version carte langage.
3- le fichier COMPILER

Pour vendre (ou donner) des programmes compilés

Sur le plan légal, vous avez le droit de donner ou de vendre des copies des programmes que vous possédez et que vous avez compilés avec le compilateur Beagle. Du fait que le compilateur Beagle est protégé par le copyright, l'acheteur ou le donataire devra utiliser le compilateur qu'il aura acheté lui-même pour les exécuter.

Il y a une autre possibilité : si vous voulez mettre le fichier COMPILER.SYSTEM du compilateur Beagle sur les disques que vous vendez ou que vous donnez, vous pourez le faire en versant à Alan Bird, l'auteur du compilateur, un droit de licence peu coûteux. Pour plus de renseignements :

The Software Touch
c/o Compiler Licensing
9625 Black Mountain Road, £204
San Diego, California 92126
✆ (619) 549-3091

Après avoir signé la licence et payé les droits, vous pourrez mettre sur les disques que vous vendez ou que vous donnez le fichier COMPILER.SYSTEM. C'est ce fichier qui permet l'exécution des programmes compilés. En aucun cas, vous n'êtes autorisés à mettre le fichier COMPILER sur les disques que vous vendez ou que vous donnez.

Ce manuel est-il à jour ? Lancez NOTES pour le savoir

Lancez le programme Basic NOTES de la disquette Beagle pour connaître les modifications et les corrections apportées à ce manuel.

Notions de base

Dans ce manuel, nous supposons que vous connaissez les bases, tel que charger et sauvegarder des fichiers, lancer des programmes Basic, etc. Même si vous ne maîtrisez pas ces notions, cela ne vous empêchera pas de tirer le maximum de profit du compilateur Beagle en lisant les pages 2 à 9. Nous ne saurions trop vous recommander la lecture des excellents manuels de programmation Apple, et tout particulièrement, le 'Manuel d'utilisation du Basic avec ProDOS' et le 'Guide de l'utilisateur ProDOS'.

Comment utiliser le compilateur Beagle ?

Règle numéro 1 : Uniquement les programmes Applesoft

Le compilateur Beagle ne fonctionne qu'avec des programmes Applesoft non protégés, sous ProDOS. Votre programme doit être sauvegardé sur un disque ProDOS. Quand vous listez le catalogue d'un volume (en tapant `CAT`), "BAS" indique que le fichier est en Basic Applesoft :

```
/DISQ.EXEMPLE
   NAME    TYPE
FICHIER.A  BAS      Vous pouvez compilez les programmes Basic.
FICHIER.B  BIN      Vous ne pouvez pas compiler des fichiers binaires.
FICHIER.C  TXT      Vous ne pouvez pas compilez des fichiers textes.
FICHIER.D  SYS      Laissez tomber.
FICHIER.E  VAR      Idem.
FICHIER.F  COM*     Ceci est un programme sauvegardé en format compilé.
```

*COM deviendra INT si le compilateur n'est pas en mémoire.

Règle numéro 2 : Vérifier que le compilateur est chargé

Les ordres du compilateur seront sans effet tant que le compilateur ne sera pas chargé en mémoire. Le moyen le plus simple pour y arriver est de démarrer sur la disquette du compilateur Beagle

(mettez le disque dans le lecteur numéro 1 et mettez en route votre Apple).
Il y a d'autres méthodes pour charger le compilateur en gagnant de la place en mémoire et/ou sur disque — voir page 16 et 17.

Règle numéro 3 : Faire quelques contrôles

- Les programmes utilisant les commandes ProDOS CHAIN, STORE et RESTORE nécessitent un traitement particulier, voir pages 11-12.
- Les instructions Ampersand (&) avec paramètres et les routines qu'elles appellent doivent être modifiées par un programmeur qui maîtrise l'assembleur, voir page 12.
- Certaines commandes Applesoft sont incompatibles avec le compilateur : CONT, DEL, LIST, LOAD, NOTRACE, RECALL, SAVE, SHLOAD, TRACE et STORE (en revanche les commandes ProDOS LOAD, SAVE et STORE sont compilables). Retirer ces commandes ne pose aucun problème à 99,6502 % des programmes que nous avons vus.
- Des POKEs inattendus en mémoire ont des effets imprévisibles. Si le programme que vous voulez compiler fait des POKEs dans la page 0 ou dans le BASIC.SYSTEM ou quelque autre endroit exotique, tant pis, essayez de compiler ; si vous avez de la chance, vous n'aurez aucun problème.
- Les programmes Applesoft très volumineux sont en principe compilables si vous les compilez sur disque sans le fichier COMPILER.SYSTEM en mémoire, voir page 17.

Essayez avec le programme TESTPROGRAM

Il y a un petit programme Applesoft nommé TESTPROGRAM sur la disquette du compilateur Beagle. Ce n'est pas grandiose, mais il utilise de nombreuses commandes Applesoft et est une bonne démonstration du fonctionnement du compilateur. Dans les exemples des pages qui suivent, pratiquement tout programme Applesoft pourra remplacer TESTPROGRAM.

Comment exécuter un programme Applesoft à la vitesse du langage machine

C'est facile. Tout d'abord, assurez-vous que le compilateur est bien en mémoire (c'est le cas si vous avez démarré sur la disquette Beagle). Maintenant lancez votre programme comme vous le faites habituellement en tapant :

```
RUN NOM
```

ou bien : - `NOM`

(`NOM` est le nom – ou le chemin d'accès – de votre programme Applesoft).
Le message «COMPILING...» (compilation en cours) s'affiche un court instant, puis votre programme devrait s'exécuter à la vitesse du langage machine. Si tel n'est pas le cas, le compilateur n'est certainement pas chargé. Si un message d'erreur s'affiche à l'écran, lire les pages 13 à 16.
Du fait que les programmes compilés n'ont pas de numéro de ligne, la commande ProDOS `RUN NOM,@123` ne fonctionnera pas (123 représente un numéro de ligne de programme).

Quitter un programme

On peut généralement quitter un programme en faisant CTRL-C. Toutefois il se peut que cela ne fonctionne pas, lors de l'exécution compilée (voir page 9 ce qu'il faut faire dans ce cas).
CTRL-RESET permet presque toujours de stopper le programme. Toutefois certains programmes ne permettent absolument pas de quitter et vous devrez *rebooter* pour vous en sortir.

Relancer un programme

Après l'exécution d'un programme compilé, vous pouvez taper RUN pour le relancer. Si le message «NOT A COMPILED PROGRAM» (ceci n'est pas un programme compilé) apparaît à l'écran, quelque chose a affecté le programme compilé en mémoire.

Exemple :

Pour exécuter TESTPROGRAM de la disquette Beagle à la vitesse du langage machine :
1- Démarrer sur la disquette Beagle.
2- Taper `RUN TESTPROGRAM` ou bien - `TESTPROGRAM`
3- Pour arrêter le programme, faites CTRL-RESET
4- Taper `RUN` pour relancer TESTPROGRAM.

Comment sauvegarder un programme en format compilé

Pour sauvegarder la version compilée d'un programme Applesoft sur le disque, tapez :

```
COMPILE NOM, NOUVEAU NOM
```

(`NOM` est le nom — ou le chemin d'accès — de votre programme Applesoft sur le disque. `NOUVEAU NOM` est le nom — ou le chemin d'accès — sous lequel sera sauvegardé le fichier compilé).

Votre programme Applesoft sera chargé, compilé puis sauvegardé sur le disque. Le message FILE TYPE MISMATCH signifie dans ce cas que vous avez donné le même nom aux deux fichiers. Vous pouvez utiliser le même nom pour le fichier compilé si vous sauvegardez sur un autre disque ou catalogue. Par exemple vous pourriez taper :

```
COMPILE /DISK1/NOM, /DISK2/NOM
```

Ceci provoque le chargement de NOM depuis DISK 1 et la sauvegarde sous le nom NOM sur DISK 2 en format compilé.

En faisant le catalogue du disque vous reconnaîtrez les fichiers compilés à leur type COM (COM pour COMpilé tout comme vous avez BAS pour BASic). Si vous faites le catalogue alors que le compilateur n'est pas installé, COM sera remplacé par INT.

Exemple :

Pour sauvegarder TESTPROGRAM sur disque en format compilé :
1- Assurez-vous que le compilateur est chargé
2- Insérer le disque Beagle dans le drive 1
3- Précisez le préfixe si nécessaire en tapant `PREFIX,D1`
4- Tapez `COMPILE TESTPROGRAM, TESTRAPIDE`, ainsi TESTPROGRAM sera sauvegardé sous le nom TESTRAPIDE en format compilé. Quand vous faites le catalogue (en tapant CAT), vous verrez que TESTRAPIDE est du type COM.

Pourquoi sauvegarder en format compilé ?

- Vous gagnez du temps parce que vous n'avez pas besoin de compiler à chaque fois que vous lancer le programme.
- Vous gagnez de la place sur vos disques car les fichiers compilés sont moins volumineux que les fichiers Basic.
- Vous gagnez de la place en mémoire parce que le fichier COMPILER (voir page 17) n'a pas besoin d'être en mémoire quand vous lancez le programme.
- Les programmes ne sont pas listables et les petits curieux ne peuvent pas les lire et les modifier (ceci peut être aussi un inconvénient).

Comment lancer un programme compilé depuis un disque ?

Vous lancez un fichier compilé (COM) comme un programme Basic (BAS). Tapez :

```
RUN NOM
ou - NOM
```

(NOM est le nom – ou le chemin d'accès – de votre programme compilé)

Exemple :

Pour lancer TESTPROGRAM en format compilé :

1- Compiler TESTPROGRAM pour créér le fichier "COM" nommé TESTRAPIDE. (Suivre la procédure de l'exemple précédent)
2- Taper `RUN TESTRAPIDE` ou `- TESTRAPIDE`

Le fichier COMPILER.SYSTEM doit être chargé pour exécuter des programmes compilés (ce qui est le cas si vous avez démarré sur le disque Beagle). Vous pouvez gagner de la place mémoire en ne chargeant pas le fichier COMPILER.

Comment modifier un programme compilé

Vous ne pouvez pas modifier un programme compilé. Il faudra apporter les modifications au programme "source" en Basic, (celui que vous avez compilé). Faites les modifications comme vous en avez l'habitude et assurez vous d'avoir bien sauvegardé votre programme modifié avant de compiler.
Lire les remarques sur les éditeurs de programme GPLE et Program Writer page 4.

Exemple :

Pour modifier TESTPROGRAM :

1- Taper `LOAD TESTPROGRAM`
2- Taper `LIST 10` pour lire la ligne de programme numéro 10
3- Taper `10 x = 5` pour modifier la ligne 10. Cela modifiera les motifs de l'écran lors de l'exécution du programme.
4- Taper `SAVE TEST2` pour sauvegarder le Basic modifié (on pourra utiliser n'importe quel nom

de fichier valide).

5- Compilateur installé, tapez `RUN TEST2` ou bien – `TEST2` pour l'exécuter à la vitesse du langage machine.
   Ou bien tapez `COMPILE TEST2, NOUVEAUTEST2` pour le sauvegarder en format compilé.
6- Pour faire d'autres modifications, faites `LOAD TEST2` et retournez à l'étape numéro 2.

Comment faire un programme interruptible par CTRL-C

En principe, CTRL-C interrompt l'exécution d'un programme Basic. En revanche, il n'a pas d'effet sur un programme compilé (sauf en réponse à une instruction INPUT). Pour rendre le CTRL-C actif sur un programme compilé, ajouter quelque part dans le programme à compiler l'instruction RESUME. Bien réfléchir avant d'utiliser cette technique car elle a l'effet pervers de ralentir l'exécution du programme compilé.
La fonction RESUME a un effet indésirable sur les programmes Applesoft, aussi il faut la mettre à un endroit du programme où elle ne risque pas d'être exécutée ; par exemple après la fin du programme (END : RESUME).

Exemple :

Pour que TESTPROGRAM puisse être interrompu par CTRL-C après compilation :

1- Taper `LOAD TESTPROGRAM`
2- Taper `60000 END : RESUME` pour ajouter la ligne de programme 60000. (On peut remplacer 60000 par un numéro de ligne de 1 à 63999 ; assurez-vous simplement que l'instruction RESUME ne sera pas exécutée).
3- Tapez SAVE TESTC pour sauvegarder la version modifiée (tout autre nom de fichier valide conviendra).
4- Tapez – `TESTC` pour lancer le programme.
5- Faire CTRL-C pour interrompre le programme.

Un nouveau message : «?BREAK»

Quand votre programme compilé est interrompu par CTRL-C (ou par une instruction STOP ou par une erreur), un message d'erreur tel ?BREAK AT $ 0ABC s'affichera à l'écran. Cela vous indique l'adresse mémoire en hexadécimal où le programme s'est interrompu (et non du numéro de ligne puisqu'il n'y a pas de numéro de ligne dans les programmes compilés, voir page 14 pour plus de détails sur cette adresse.

Comment exécuter un programme à la vitesse normale

La meilleure solution pour lancer un programme à la vitesse normale est de supprimer le compilateur de la mémoire en démarrant sur un autre disque ProDOS.

! Attention : Il n'est pas garanti que la méthode décrite ci-dessous fonctionne. En effet, avec certains programmes, cela risque de 'planter' le système vous conduisant à "rebooter".

Sauvegardez toujours vos programmes avant de les lancer !

En gardant en mémoire les remarques ci-dessus, chargez un programme Applesoft de la disquette et tapez : `RUN`. Notez que cette commande commence par un ":". Si vous ne mettez pas le ":", vous aurez le message NOT A COMPILED PROGRAM (Ce n'est pas un programme compilé).

Exemple :

Pour lancer TEST PROGRAM à la vitesse normale avec le compilateur chargé :
0- Lire la remarque ci-dessus
1- Tapez `LOAD TESTPROGRAM`
2- Tapez : `RUN`
3- Pour sortir, faire CTRL-C ou CTRL-RESET
4- Pour recommencer, tapez à nouveau : `RUN`

Comment changer l'adresse de chargement d'un programme

(pour les programmeurs confirmés)

Les programmes compilés se chargent et s'exécutent en principe à l'adresse 2049 ($0801), comme les programmes Applesoft. Vous pouvez lancer un programme compilé à une autre adresse en modifiant le programme Basic avant compilation.
Par exemple, insérer la ligne suivante au début de TESTPROGRAM pour l'exécuter au-dessus de la page HGR1 en 16384 ($4000) :

```
1 IF PEEK (104)<>64 THEN POKE 16384,0:POKE 104,64: PRINT CHR$(4)"RUN
    TESTPROGRAM"
```

Remplacer les 64 par des 96, et le 16384 par 24576 pour charger le programme au-dessus de la page HGR2 en 24576 ($6000).
Vous pouvez aussi préciser l'adresse avec la commande RUN suivie d'une virgule et de l'adresse. La commande suivante compilera et exécutera un programme Basic — ou exécutera un programme déjà compilé — au-dessus de la page HGR1 et HGR2 respectivement.

```
RUN NOM, A$4000
RUN NOM, A$6000
```

(Note : Dans cette commande, RUN ne peut pas être remplacé par un tiret.)

Comment obtenir des programmes compilés encore plus rapides

(pour programmeurs confirmés)

Le point le plus important à garder à l'esprit pour que les programmes compilés soient le plus rapides possible est d'éviter quand c'est possible les calculs en virgule flottante. Le compilateur fait ses calculs mathématiques avec des entiers à chaque fois qu'il le peut : le calcul sur les entiers est bien plus rapide que le calcul en virgule flottante.

L'ordinateur utilise les routines en virgule flottante dans les cas suivants :

- quand le nombre a une partie fractionnaire (par exemple 3.5) ;
- quand le nombre est supérieur à 32767 ou inférieur à - 32768 ;
- quand une expression mathématique utilise la division ;
- quand l'une des fonctions suivantes est utilisée : ATN, COS, EXP, LOG, RND, SIN, SQR ou TAN.

Les techniques pour accélérer le Basic ne sont pas efficaces

Les méthodes de programmation ci-dessous accélèrent l'exécution des programmes Basic mais n'accélèrent pas les programmes compilés (ceci dit, elles ne perturbent rien).

- Utiliser des variables entières et non des variables réelles (Utiliser A% = 3 au lieu de A = 3 n'améliorera pas la vitesse après compilation).
- Utiliser des variables et non des constantes numériques (Dans un programme compilé A = PI ne va pas plus vite que A = 3.14159).
- Mettre les lignes fréquemment appelées et les sous-programmes au début du programme.
- Déclarer les variables fréquemment utilisées au début du programme.

Comment compiler des programmes qui utilisent la commande CHAIN

La commande ProDOS CHAIN fonctionne comme la commande ProDOS RUN, mais les variables existantes ne sont pas effacées. Les programmes qui utilisent CHAIN partagent des variables communes et doivent être soumis à un traitement particulier pour que la compilation aboutisse. À défaut vous aurez le message d'erreur FILE TYPE MISMATCH (erreur de type de fichier).
Tous les programmes concernés par la commande CHAIN doivent être compilés sur disque, en utilisant la commande COMPILE pour l'un et la commande COMMON pour le ou les autres. La syntaxe de COMMON est semblable à celle de COMPILE :

```
COMMON NOM, NOUVEAUNOM
```

COMMON doit être utilisé immédiatement après COMPILE. Si vous modifiez un programme ou si vous ajoutez un nouveau fichier qui sera chaîné à des fichiers déjà compilés vous devrez reprendre tout à zéro et recompiler tous les fichiers.

Exemple :

Supposons que vous ayez trois programmes — PRINCIPAL, PROG.A et PROG.B — qui ont des variables communes. PRINCIPAL est le programme sur lequel on démarre auquel sera chaîné PROG.A, lui-même chaîné à PROG.B, ce dernier chaîné à PRINCIPAL. Voici comment compiler ces programmes :

1 - Compiler PRINCIPAL avec la commande habituelle COMPILE :

```
COMPILE PRINCIPAL, PRINCIPAL.COMP
```

(Tout nom de fichier valide convient).

2 - Compiler aussitôt chaque programme qui partage les données avec la commande COMMON :

```
COMMON PROG.A, PROG.A.COMP
COMMON PROG.B, PROG.B.COMP
```

3 - Pour lancer le ou les programmes, compilateur installé, tapez :

```
RUN PRINCIPAL.COMP
```

ou - `PRINCIPAL.COMP`

(Remarque : même les fichiers compilés par COMMON peuvent être lancés directement).

Comment compiler des programmes qui utilisent les commandes STORE et RESTORE

Remarque : ceci concerne les commandes STORE et RESTORE de ProDOS. Le RESTORE de l'Applesoft se compile sans problème. La commande STORE de l'Applesoft n'a plus lieu d'être.

La commande ProDOS STORE sauvegarde les variables en mémoire sur disque dans un fichier de type VAR. La commande RESTORE recharge ces variables en mémoire.
Les programmes qui utilisent les commandes STORE et RESTORE partagent des variables communes et doivent faire l'objet d'un traitement particulier pour que la compilation aboutisse. Les programmes doivent être compilés sur disque, avec la commande COMPILE sur un programme et la commande COMMON sur les autres. Voir CHAIN, ci-dessus.
COMMON doit être utilisé aussitôt après COMPILE. Si vous faites des modifications de programmes, ou si vous ajoutez un nouveau fichier qui partage des variables avec des fichiers déjà compilés, vous devez tout reprendre à zéro et recompiler tous les fichiers.
STORE et RESTORE dans les programmes compilés génèrent et utilisent des fichiers de type CVR et non VAR.

Exemple :

Supposons que vous ayez un programme qui s'appelle DONNEES.MAJ qui utilise la commande STORE pour écrire des variables qui seront chargées avec la commande RESTORE par un programme nommé GRAND.JEU. Voici comment compiler ces deux programmes :

1- Tapez `COMPILE GRAND.JEU, GRAND.JEU.COMP`
(tout autre nom valide convient)
2- Tapez aussitôt :
`COMMON DONNEES.MAJ, DONNEES.MAJ.C`
3- Pour lancer l'un des programmes, compilateur chargé, tapez :
`RUN GRAND.JEU.COMP` ou bien `-DONNEES.MAJ.C`

Comment écrire des routines ampersand pour des programmes compilés

(pour les programmeurs confirmés en assembleur)

Une routine ampersand sans paramètres (& tout seul) se compilera sans problème. Pour les routines ampersand avec paramètres (comme &XXX ou &XXX,YYY,ZZZ) c'est une autre histoire. La commande ampersand et la routine en langage machine elle-même doivent être modifiées.

Modification numéro 1 : Utiliser && au lieu de &

Quand vous voulez appeler une routine ampersand depuis un programme compilé, vous devez utiliser deux ampersands de suite (par exemple vous utiliserez &&TRI au lieu de &TRI). C'est ainsi que le compilateur voit si les programmes ont été modifiés ou non.

Modification numéro 2 :
Changer la façon de saisir des paramètres

Compilateur chargé, un JSR en $98FD évaluera le paramètre suivant après un ampersand :

- **Si le paramètre est une chaîne**, on trouvera le pointeur sur la chaîne en $F6,$F7. Toutes les chaînes dans un programmes compilés sont stockées avec la longueur dans le premier octet.
- **Si le paramètre est une valeur numérique** et que la retenue est à zéro (c = 0), il s'agit d'un entier avec l'octet de poids faible dans le registre X et l'octet de poids fort dans l'accumulateur.
- **Si le paramètre est une valeur numérique** et que la retenue est à 1 (c = 1), il s'agit d'une valeur en virgule flottante stockée dans le FAC (accumulateur flottant en $9D).

Le compilateur ne peut pas savoir si les paramètres passés à votre routine ampersand sont corrects. Si les paramètres ne conviennent pas, le programme se *plantera* lamentablement.

Ex&mple :

Écrivons une routine ampersand qui affiche N fois le premier caractère de la chaîne S$. Notre programme Applesoft ressemblera à ceci :

```
20 S$"VACHE":N = 80
50 &&N,S$
```

Le but de ce programme est d'afficher 80 fois «V». L'assembleur ressemblera à ceci :

```
          jsr  $98FD      ;évalue N
          bcs  erreur     ;refuse un nombre décimal
          cmp  £0         ;controle le poids fort
          bne  erreur     ;pas plus de 256
          txa
          pha             ;sauvegarde N sur la pile
          jsr  $98FD      ;évalue S$
          pla             ;dépile N
          tax             ;l'utilise comme compteur
          beq  fini       ;compteur vaut 0
          ldy  £0
          lda  ($F6),y    ;charge la longueur de la chaîne
          beq  fini       ;chaîne vide
          iny
          lda  ($F6),y    ;prend le premier caractère de la chaîne
          ora  £$80       ;positionne le bit de poids fort
boucle    jsr  $FDED      ;affiche le caractère
          dex
          bne  boucle     ;boucle
fini      rts             ;retourne au compilateur
erreur    ldx  £53        ;Illegal quantity error
          jmp  $98F1      ;s'occupe de l'erreur
```

Erreurs lors de l'exécution des programmes compilés

Une erreur ProDOS ou Applesoft dans un programme interrompra un programme compilé tout comme un programme non compilé. La seule différence est que les programmes compilés génèrent

de curieux messages d'erreurs tels que :

```
?ILLEGAL QUANTITY ERROR AT $0ABC
```

Les programmes non compilés générent, comme vous le savez, ce type de message :

```
?ILLEGAL QUANTITY ERROR IN 123
```

Dans cet exemple, $0ABC est l'adresse en hexadécimal où l'erreur est survenue, et 123 est le numéro de la ligne qui a provoqué l'erreur (les programmes compilés n'ont pas de numéro de ligne). Comme il est plus facile de travailler avec des numéros de ligne qu'avec des adresses mémoire, le moyen le plus efficace pour déceler les erreurs est d'essayer vos programmes et de les mettre au point avant compilation.

Le programme PRINT.LINES convertit les adresses hexadécimales en numéros de ligne

(pour programmeurs confirmés)

Au cas où vous passeriez outre notre conseil ci-dessus : pour connaître le numéro de ligne qui correspond à l'adresse hexadécimale de l'erreur, compiler le programme en utilisant COMPILE ou RUN puis :

1- Facultativement, activer votre imprimante en faisant PR#1
2- Avec le compilateur Beagle dans le lecteur courant, taper :
   `BRUN PRINT.LINES` ou `-PRINT.LINES`
3- Faites PR#0 pour désactiver votre imprimante si nécessaire.

Le programme PRINT.LINES affiche les adresses hexadécimales et les numéros de ligne en décimal.

Erreurs survenant pendant la compilation

Si votre programme Applesoft comporte des erreurs, le compilateur fera de son mieux pour les déceler pendant la phase de compilation. Chaque ligne erronée sera affichée avec indication de l'endroit où se situe l'erreur. Alors le compilateur demandera «CONTINUE WITH ERRORS ? » (Continuez avec les erreurs ?), pour savoir si vous voulez continuer et exécuter tout de même le programme (Certaines "erreurs" sont volontaires ou bien ne posent pas de problème). Si vous répondez NO, la compilation s'interrompra et vous pourrez faire vos corrections. Sauvegardez votre programme corrigé sur disque, puis compilez-le à nouveau.
Certaines erreurs ne seront pas décelées par le compilateur – c'est le cas des erreurs ProDOS et des erreurs situées à l'intérieur de guillemets. Voici quelques erreurs qui seront détectées pendant la compilation :

<?> (Erreur de syntaxe)

- Le symbole <?> peut être provoqué par l'utilisation d'un mot-clé du Basic incompatible avec le compilateur ; particulièrement CONT, DEL, LIST, LOAD, NOTRACE, RECALL, SAVE, SHLOAD, STORE et TRACE. Les programmes contenant ces commandes ne se compileront pas. (Remarque : les commandes ProDOS LOAD, SAVE et STORE sont compilables).
- Les autres responsables de ce message sont des erreurs que vous avez déjà rencontrées, comme des paramètres omis (par exemple HPLOT sans coordonnées), les erreurs de type (par exemple A$ = 3), des mots-clé mal orthographiés (par exemple PIRNT), des virgules et des deux-points omis, etc. Votre mode d'emploi Applesoft vous aidera pour apporter les modifications nécessaires.

<£> (ligne inexistante)

Le symbole <£> signifie généralement que votre programme comporte un GOTO ou un GOSUB

vers une ligne inexistante.

<A> (Erreur de dimension de tableau)

Le symbole <A> signifie que votre programme comporte des tableaux mal déclarés ou mal utilisés. Par exemple A(25) = 3 : A(6,3) = 3.

<*> (Valeur invalide)

Les seules valeurs invalides que votre compilateur détectera concernent les adresses (par exemple : POKE 90000,0). Les autre valeurs invalides (comme HPLOT 90000,0) ne seront pas détectées avant "plantage".

INCLUDE APPENDED MACHINE CODE ? (inclure le code machine attaché ?)

Ce message signifie que votre compilateur a détecté une zone excédentaire au-delà de la fin de votre programme. Il peut s'agir de caractères inutiles, ou au contraire de données nécessaires ou encore d'une routine appelée par le programme. En cas d'hésitation, jouez la sécurité en répondant Y (oui, l'inclure).

Les erreurs en mode direct

Ces erreurs peuvent survenir aussitôt après avoir tapé une commande au clavier :

FILE TYPE MISMATCH (Erreur de type de fichier)

- Peut-être avez-vous utilisé deux noms identiques pour compiler un fichier sur disque (`COMPILE NOM, NOM`).
- Ou peut-être avez vous utilisé une commande telle que `COMPILE NOM, NOUVEAUNOM` alors que NOUVEAUNOM (d'un autre type que COM) était déjà utilisé sur le disque.
- Ou encore vous avez lancé un programme qui contient les ordres CHAIN, STORE et RESTORE et vous ne l'avez pas compilé avec COMMON (voir pages 11-12).
- Ou bien votre programme STARTUP est en basic non compilé (BAS). STARTUP doit être un fichier compilé (COM).

NO BUFFER AVAILABLE (Pas de buffer disponible)

- Peut-être avez vous essayé de charger une deuxième fois le fichier COMPILER.SYSTEM ou le fichier COMPILER. Une seule fois suffit...
- Ou peut-être avez vous essayé de charger le programme au-dessous de l'adresse $0801.
- Ou bien vous avez fait CTRL-RESET lors d'un catalogue.

Solution : recommencez ou bien "rebootez".

NOT A COMPILED PROGRAM (Programme non compilé)

Le compilateur en mémoire, vous avez tapé RUN après avoir chargé un programme Applesoft. (voir pages 7-9).

PATH NOT FOUND (Chemin d'accès inconnu)

Autrement dit : fichier non trouvé. Si vous êtes certain que le "fichier non trouvé" est bien sur le disque et que vous avez correctement orthographié son nom, essayez de taper `PREFIX/` ou `PREFIX,S6,D1` (utilisez les numéros de port et de lecteur qui conviennent), ou bien `PREFIX/CAT/SOUS` (utilisez les bons noms de catalogue et de sous-catalogue).

PROGRAM TOO LARGE (Programme trop volumineux)

- Votre programme Applesoft est trop volumineux pour tenir en mémoire.
- Ou bien vous essayez de lancer un programme à une adresse trop élevée.

Essayez de compiler sur disque puis d'enlever le fichier COMPILER (voir page 17).

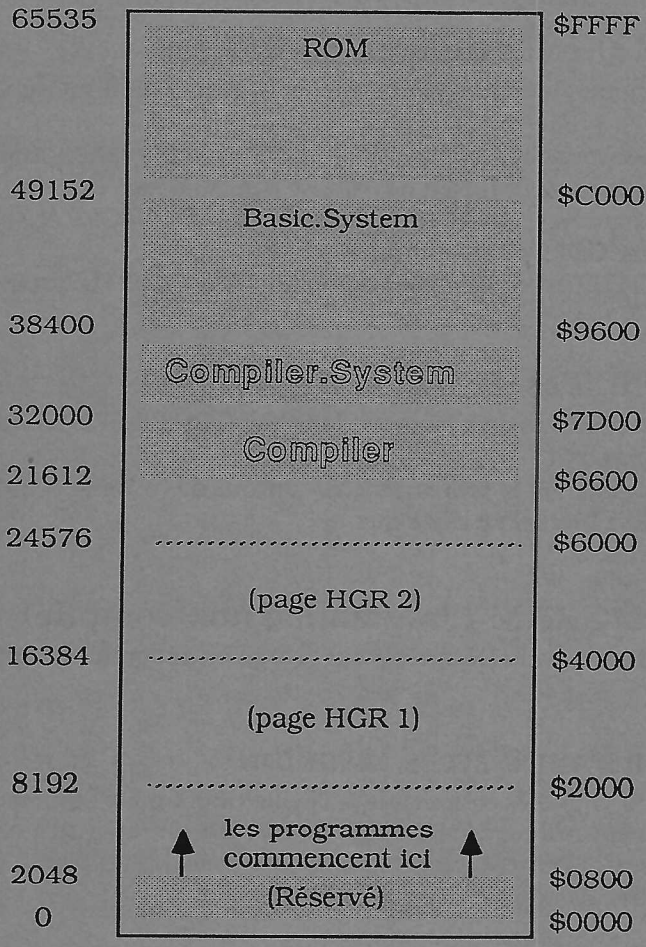
?SYNTAX ERROR

- Peut-être avez-vous mal orthographié une commande.
- Ou peut-être avez-vous utilisé la commande COMPILE ou COMMON sans avoir installé le compilateur.
- Si le message ?SYNTAX ERROR s'affiche alors que vous êtes certain que ce que vous avez tapé est valide (LIST par exemple), la mémoire est certainement perturbée, et il faudra "rebooter". D'abord essayer de faire CTRL-RESET. (Ça ne sert peut-être à rien, mais ça soulage).

Les fichiers COMPILER et COMPILER.SYSTEM

Quand vous faites le catalogue du disque Beagle, vous voyez les deux fichiers COMPILER (BIN) et COMPILER.SYSTEM (SYS) :

- COMPILER est le programme qui transforme les programmes Applesoft en programmes compilés.
- COMPILER.SYSTEM est le programme qui permet d'exécuter les programmes déjà compilés.



Quand vous démarrez sur le compilateur Beagle, voici ce qui se passe :

1- Le fichier COMPILER.SYSTEM se charge en mémoire.
2- Le fichier STARTUP est chargé et exécuté.
3- Le fichier COMPILER s'installe en mémoire.

S'il n'y a pas de fichier de type COM nommé STARTUP sur le disque, la séquence ci-dessus s'arrête après la première étape. Le fichier STARTUP de la disquette Beagle charge le fichier COMPILER, cependant vous pouvez remplacer ce STARTUP par le vôtre. STARTUP doit nécessairement être un fichier de type COM.

Pour gagner de la place : supprimez le fichier COMPILER

Si vous ne faites qu'exécuter des programmes compilés (COM) sans faire de compilatiion de programme Applesoft (BAS), vous pouvez gagner environ 6Ko de mémoire en ne chargeant pas le fichier COMPILER. Utiliser l'une des méthodes suivantes :

- Renommer le fichier STARTUP de la disquette Beagle avant de booter. Puis démarrer sur le disque. Ou remplacer le STARTUP de Beagle par votre propre version qui n'installera pas le fichier COMPILER. STARTUP doit obligatoirement être un fichier COM.
- Ou copier le fichier COMPILER.SYSTEM sur un autre disque qui contient le fichier ProDOS (mais pas les fichiers COMPILER ou BASIC.SYSTEM). Démarrez sur ce disque et vous pourrez exécuter les programmes compilés, mais pas faire de compilation.

Pour compiler des programmes volumineux : supprimer COMPILER.SYSTEM

Si vous voulez compiler un programme Applesoft très volumineux, il se peut qu'il n'y ait pas assez de place en mémoire pour votre programme et pour les deux fichiers Beagle. Ce problème peut trouver sa solution ainsi : installer le fichier COMPILER sans COMPILER.SYSTEM, puis faire la compilation sur disque.
Pour empêcher COMPILER.SYSTEM de se charger, démarrez sur un disque ProDOS ordinaire qui charge BASIC.SYSTEM, puis mettez le disque Beagle et tapez la commande `BRUN COMPILER` (ou bien `-COMPILER`). Ne faites ceci qu'une seule fois, parce que le fichier COMPILER consomme 6Ko de mémoire à chaque fois qu'il est installé.
Souvenez-vous que COMPILER.SYSTEM doit être installé (tout seul ou avec COMPILER) pour pouvoir exécuter des programmes compilés.

Le programme MENU

MENU est un fichier de type COM qui vous permet de choisir vos lecteurs de disques et de lancer vos programmes à l'aide de menus du style AppleWorks. Vous pouvez faire que MENU s'exécute dès le démarrage sur un disque en le renommant STARTUP. Ou bien vous pouvez faire que votre programme STARTUP lance MENU.
Pour appeler MENU, tapez - `MENU`. La liste de tous les lecteurs de disques reconnus par ProDOS s'affichera en haut de l'écran. S6,D1 signifie Port 6, Lecteur 1. S6,D2 signifie Port 6, Lecteur 2 etc. S3,D2 correspond au disque virtuel créé par ProDOS. En dessous s'affichera la liste des fichiers exécutables (BAS, BIN et COM) du lecteur dont le numéro apparaît en vidéo inversée.

Pour lancer un programme de l'un de vos lecteurs :

1- Utilisez les touches ← et → ou tapez un chiffre pour choisir le lecteur de disquette qui convient. Les noms de fichiers de ce lecteur s'afficheront à l'écran.
2- Utilisez les flèches ou la touche tabulation pour choisir votre programme.
3- Appuyez sur la touche RETURN pour lancer le programme désigné. Si vous faites RETURN alors qu'un sous-catalogue est sélectionné, les noms des fichiers qu'il contient s'afficheront.

Reprenez alors à l'étape numéro 2.

Vous pouvez quitter MENU à tout moment en appuyant sur ESC.

Messages d'erreur de MENU

- I/O ERROR (Erreur d'entrée/sortie) peut signifier que la porte d'un lecteur est restée ouverte.
- NO DEVICE ERROR (Pas de périphérique) signifie généralement que vous essayez de lire sur le lecteur 2 alors qu'il n'y a pas de lecteur 2.
- PATH NOT FOUND (Chemin d'accès non trouvé) signifie que vous avez probablement enlevé un disque.

Autres problèmes

- **Si votre programme "se plante"**, il n'était peut-être pas prévu qu'il soit exécuté. (Par exemple, il peut s'agir d'une image HGR et non d'un programme).
- **Si votre programme n'apparaît pas sur l'écran** alors qu'il est bien sur le disque, il est peut être d'un type non exécutable comme TXT ou VAR. Ou bien il n'y avait pas assez de place à l'écran pour lui. En 80 colonnes le maximum est 60 fichiers plus 10 lecteurs. En 40 colonnes, le maximum est 30 fichiers plus 5 lecteurs.

Extensions au COMPILER.SYSTEM

Les programmes ci-dessous sont des modifications du compilateur. Il suffit de BRUNer simplement le fichier après avoir démarré sur la disquette du compilateur Beagle.

INPUT.ANYTHING (Input étendu)

Ce patch remplace l'instruction INPUT du compilateur par une autre qui autorise la saisie des virgules et des deux-points. C'est très pratique pour récupérer des données d'un fichier texte.

SLOW.PDL

SLOW.PDL introduit un petit délai dans la fonction PDL (paddle, poignée de jeu), ainsi la valeur retournée sera la bonne.
Le manuel de référence du programmeur BASIC vous conseille, si vous faites des lectures répétitives de la poignée de jeu avec la fonction PDL, de mettre une petite boucle entre les lectures, FOR X = 1 TO 10 : NEXT par exemple. Du fait que le compilateur augmente beaucoup la vitesse des programmes, vous devrez augmenter ce délai. Ou bien utilisez SLOW.PDL.

FAST.HPLOT

FAST.HPLOT remplace l'instruction HPLOT par une autre bien plus rapide. En temps normal il n'est pas installé dans le compilateur parce qu'il occupe beaucoup de mémoire. N'utilisez ce patch que si votre programme utilise essentiellement la haute résolution.

Modifier l'Applesoft grâce au compilateur Beagle

En réalité, dans les pages suivantes, nous ne vous en dirons pas beaucoup plus mais nous voulons donner aux passionnés, aux curieux et autres gens bien une idée sur le fonctionnement du compilateur. S'il vous plaît, prenez ces renseignements uniquement comme un petit bonus et n'appelez pas Beagle Bros pour demander de l'aide dans l'analyse du fonctionnement du compilateur.

Amusez-vous à améliorer (ou dégrader) l'Applesoft !

Les programmeurs ont toujours envie de modifier l'Applesoft pour donner plus de puissance et d'efficacité à un langage quelque peu dépassé. Du fait que l'interpréteur est en ROM, les modifications sont un peu plus délicates. En revanche, l'interpréteur du compilateur Beagle est en RAM aussi vous pouvez intervenir sur les instructions et les fonctions comme vous le souhaitez.
La table des sauts listée ci-dessous occupe la zone de $9900 à $99FF et contient les vecteurs (adresses) de chaque routine du compilateur qui traite une instruction ou une fonction. Quelques POKEs bien placés depuis le Basic ou l'assembleur détourneront l'interprétation des commandes vers la zone mémoire que vous indiquerez.

Par exemple, le programme suivant après compilation rendra l'instruction HOME identique à HGR. À partir de là, à vous de jouer.

```
10 POKE 39208,PEEK(39282): REM $9928, $9972
20 POKE 39209,PEEK(39283): REM $9929, $9973
21 : REM $9928-29 est l'adresse de HOME
22 : REM $9972-73 est l'adresse de HGR
30 HOME: REM Home maintenant fonctionne comme HGR
40 HCOLOR= 3: HPLOT 0,0 TO 279,191
```

Attention : un programme comme celui-ci peut rapidement devenir un bazar complet...

Adresses des routines du compilateur Beagle

Adresses	Noms	Fonctions
$9900	init	Initialisation et exécution d'un programme
$9902	run	RUN
$9904	clear	CLEAR
$9906	restore	RESTORE
$9908	on	ON GOTO/GOSUB
$990A	goto	GOTO
$990C	gosub	GOSUB
$990E	return	RETURN
$9910	pop	POP
$9912	end	END stoppe le programme
$9914	let	affecte une valeur à une variable numérique simple
$9916	for	FOR positionne la borne TO et initialise la boucle
$9918	step	STEP
$991A	numprt	évalue et affiche une valeur numérique
$991C	strprt	évalue et affiche une chaîne

$991E	spc	SPC
$9920	tab	TAB
$9922	comma	gère la virgule dans les PRINTs
$9924	crout	émet un retour-chariot
$9926	text	TEXT
$9928	home	HOME
$992A	normal	NORMAL
$992C	inverse	INVERSE
$992E	flash	FLASH
$9930	next	NEXT
$9932	nextvar	instruction NEXT avec variable
$9934	letstr	affecte une valeur à une variable chaîne simple
$9936	onerr	ONERR GOTO
$9938	prnum	PR#
$993A	innum	IN#
$993C	readn	READ d'une valeur numérique
$993E	reads	READ d'une valeur chaîne
$9940	getn	GET d'une valeur numérique (non utilisé)
$9942	gets	GET d'une valeur chaîne
$9944	plot	PLOT
$9946	vlin	VLIN
$9948	hlin	HLIN
$994A	if	IF
$994C	inputn	INPUT d'une valeur numérique
$994E	inputs	INPUT d'une valeur chaîne
$9950	prtqm	affiche un "?"
$9952	gr	GR
$9954	draw	DRAW
$9956	drawat	DRAW suivi de AT
$9958	xdraw	XDRAW
$995A	xdrawat	XDRAW suivi de AT
$995C	hplot	HPLOT
$995E	hplotto	HPLOT TO
$9960	stop	STOP
$9962	hcolor	HCOLOR
$9964	htab	HTAB
$9966	vtab	VTAB
$9968	color	COLOR
$996A	speed	SPEED=
$996C	poke	POKE
$996E	call	CALL
$9970	hgr2	HGR2
$9972	hgr	HGR
$9974	scale	SCALE=
$9976	rot	ROT=
$9978	usr	USR
$997A	pdl	PDL
$997C	peek	PEEK
$997E	letint	affecte un entier à une variable simple
$9980	letary	affecte une valeur à un tableau

$9982	letstrary	affecte une valeur chaîne à un tableau
$9984	dim	DIM
$9986	himem	HIMEM:
$9988	lomem	LOMEM:
$998A	resume	RESUME
$998C	ampersand	& (en prend 2)
$998E	wait2	WAIT avec 2 paramètres
$9990	wait3	WAIT avec 3 paramètres
$9992	def	DEF FN
$9994	byte	constante numérique sur 1 octet
$9996	integer	constante numérique entière
$9998	fp	constante en virgule flottante (format compacté)
$999A	literal	constante chaîne
$999C	getnvar	attend une valeur numérique
$999E	getsvar	attend une variable chaîne
$99A0	or	OR
$99A2	and	AND
$99A4	relop	opération de comparaison, fonction de l'octet suivant : 0 : < — 3 : <= — 6 : = — 9 : <> — 12 : >= — 15 : >
$99A6	strcomp	idem, pour les chaînes
$99A8	add	+
$99AA	minus	–
$99AC	times	*
$99AE	div	/
$99B0	power	^
$99B2	not	NOT
$99B4	asc	ASC
$99B6	chr	CHR$
$99B8	pos	POS
$99BA	len	LEN
$99BC	left	LEFT$
$99BE	right	RIGHT$
$99C0	mid2	MID$ avec 2 paramètres
$99C2	mid3	MID$ avec 3 paramètres
$99C4	str	STR$
$99C6	abs	ABS
$99C8	rnd	RND
$99CA	sgn	SGN
$99CC	int	INT
$99CE	val	VAL
$99D0	neg	négation
$99D2	concat	+, concaténation de chaînes
$99D4	scrn	SCRN
$99D6	fre	FRE
$99D8	sqr	SQR
$99DA	log	LOG
$99DC	exp	EXP
$99DE	cos	COS
$99E0	sin	SIN
$99E2	tan	TAN
$99E4	atn	ATN

$99E6	getnay	attend une valeur pour un tableau numérique
$99E8	getsary	attend une valeur pour un tableau chaîne
$99EA	fn	FN fonction utilisateur
$99EC	savres	sauve le pointeur de programme si l'instruction RESUME est active. appelée avant chaque instruction susceptible de générer une erreur.
$99EE	error	gère les erreurs
$99F0	input	initialise l'attente d'une valeur numérique ou chaîne.
$99F3 à $99FF		si vous touchez à cette zone, on appelle la police !

Routines à disposition du programmeur

Les routines suivantes vous aideront à écrire des routines en assembleur interfacées avec l'Applesoft :

Adresse	Nom	Fonction
$98E8	MOVSTR	Déplace un chaîne (non précédée d'un octet pour sa longueur, elle doit se terminée par un octet à 0) dans la zone qui a été alloué par GETSPA. Paramètres d'entrée : A = poids faible de l'adresse de la chaîne, X = poids fort de l'adresse de la chaîne.
$98EB	BYTE	Évalue un paramètre numérique et vérifie qu'il tient sur un octet (0-255). Toute autre valeur provoque le message ILLEGAL QUANTITY. La valeur est retournée dans A et dans X.
$98EE	PTRNUM	Affiche une valeur numérique. Positionne la retenue s'il s'agit d'une valeur en virgule flottante (dans FAC). S'il s'agit d'un entier, la retenue est à 0 et la valeur est dans X (poids faible) et dans A (poids fort).
$98F1	ERROR	Appelle la routine ERROR pour gérer une erreur. X doit contenir le numéro du message d'erreur (voir le Manuel de Référence du Basic Applesoft). Si ONERR n'est pas actif, le programme s'arrête et le message d'erreur s'affiche.
$98F4	GETSPA	Alloue un espace-mémoire dans la zone réservée aux chaînes. A doit contenir la longueur de la chaîne à stocker. Un octet supplémentaire est alloué car le premier octet contient la longueur. Retourne la position de la chaîne en $F6-$F7.
$98F7	PRTSTR	Affiche la chaîne pointée par $F6-$F7.
$98FA	PROC	Exécute l'instruction suivante. Un saut à cette adresse est effectuée à la fin de chaque instruction Basic. Si vous remplacez une instruction Basic, votre code devrait se terminer par ce saut.
$98FD	EVAL	Évalue un paramètre numérique ou chaîne.

Variables

Les variables sont repérées par un octet ($00 à $FF) servant de décalage. On accède à la valeur d'une variable donnée en utilisant cet octet comme pointeur sur les tables dont les adresses sont données par les pointeurs suivants :

vtype	($78)	Type de variable bit 7 = 1 s'il s'agit d'un tableau, 0 sinon bit 6 = 1 s'il s'agit d'une chaîne, 0 sinon bit 5 = 1 s'il s'agit de FN (fonction définie par l'utilisateur) bit 4 = 1 s'il s'agit d'un entier bits 0 à 3 = nombre de dimensions dans le cas d'un tableau

Les valeurs suivantes dépendent du type de variable concerné :
FP = valeur en virgule flottante, LSB = octet de poids faible, MSB = octet de poids fort.

vval1	($7A)	Tableau : LSB de l'adresse de l'en-tête du tableau Chaîne : LSB de l'adresse de la chaîne FP : $7A sera différent de 0 et sera le premier octet de la valeur en format compacté. Entier : $7A est égal à 0
vval2	($7C)	Tableau : MSB de l'adresse de l'en-tête du tableau Chaîne : MSB de l'adresse de la chaîne FP : 2ème octet de la valeur Entier : LSB de la valeur
vval3	($7E)	Tableau : LSB de l'adresse du tableau FP : 3ème octet de la valeur Entier : MSB de la valeur
vval4	($80)	Tableau : MSB de l'adresse du tableau FP : 4ème octet de la valeur
vval5	($82)	Tableau : Nombre de dimension ; 0 s'il n'est pas déjà dimensionné. FP : 5ème octet de la valeur

Si l'index de A$ vaut 2, alors la routine suivante affectera à A$ la chaîne pointée par PTR :

```
LDY  #2          ;variable A$
LDA  PTR
STA  ($7A),Y     ;stocke le LSB de l'adresse
LDA  PTR+1
STA  ($7C),Y     ;stocke le MSB de l'adresse
```

Utiliser le Compilateur Beagle avec une extension mémoire

S'en graver la mémoire !

La dernière version du compilateur Beagle tire avantage de deux types d'extensions-mémoire, vous permettant de stocker chaînes et tableaux au-delà des 48Ko de mémoire principale. Cela vous laisse plus de place pour les programmes volumineux.

Utiliser AUX.SLOT.SYSTEM au lieu de COMPILER.SYSTEM avec :
- les Apple //c ou //e 128Ko
- les Apple IIGS (seuls 128Ko seront accessibles)
- les cartes RamWorks™ d'Applied Engineering (tous modèles)
- les cartes MultiRam™ Checkmate (tous modèles)

Utiliser APPLEMEM.SYSTEM au lieu de COMPILER.SYSTEM avec :
- les cartes d'extension mémoire Apple (en slot 1 à 7)
- les cartes RamFactor d'Applied Engineering
- Flipster

Pour utiliser l'un de ces fichiers System, copiez-le sur un autre disque avec le fichier ProDOS. Démarrez alors sur ce disque. Le fichier que vous voulez installer lors du 'boot' (Compiler.System, Aux.Slot.System ou Applemem.System) doit être le premier fichier de type SYS dont le nom se termine par '.SYSTEM' dans le catalogue.
Pour éviter de devoir transférer les fichiers de la disquette Beagle, vous pouvez simplement renommer les deux fichiers System que vous ne voulez pas utiliser. Utiliser des noms qui ne se terminent pas par '.SYSTEM'.

Note : Le programme de STARTUP de la disquette Beagle, si vous l'utilisez, affiche «COMPILER.SYSTEM INSTALLED» (Fichier Compiler.System installé) pour chacun des trois fichiers qui peuvent être installés.

Autre note : APPLEMEM.SYSTEM utilise mieux la mémoire qu'AUX.SLOT.SYSTEM qui ne gère que 28% de la mémoire disponible. (Vous n'avez pas de choix : c'est la configuration de votre matériel qui décide du fichier System que vous utiliserez.)

Ni STORE, ni RESTORE

Désolés, les commandes STORE et RESTORE de ProDOS ne fonctionnent ni avec APPLEMEM.SYSTEM, ni avec AUX.SLOT.SYSTEM.

Disques virtuels : remarques importantes

Quand vous installez APPLEMEM.SYSTEM ou AUX.SLOT.SYSTEM, tout ce que vous avez pu sauvegarder en disque RAM est écrasé. Assurez-vous d'avoir bien recopié vos fichiers sur des disques réels avant.

APPLEMEM.SYSTEM ne peut être utilisé avec un disque virtuel.
AUX.SLOT.SYSTEM peut être configuré pour fonctionner avec les disque RAM RamWorks ou MultiRam. Voici comment faire :

1- Avec BASIC.SYSTEM installé (et non avec l'un des fichiers System du compilateur), lancer le programme CONFIGURE.BANKS et indiquez les bancs de mémoires minimum et maximum que vous allouez au compilateur. Faites RETURN pour sauvegarder votre version reconfigurée de AUX.SLOT.SYSTEM.
   Assurez-vous de ne pas entamer les bancs de mémoire utilisés par votre disque RAM*.
2- Utilisez le programme de partitionnement livré avec votre disque RAM pour lui fixer ses limites. Assurez-vous que cette zone n'entame pas celle que vous avez allouée au compilateur*.

*La plupart des disques RAM n'utilisent pas le banc 0. Vous pourrez probablement éviter la phase 2 en indiquant au compilateur de n'utiliser que le banc 0 (dans la phase 1, mettez le minimum et le maximum à 0).

Dimensions maximales des tableaux

Avec APPLEMEM.SYSTEM ou AUX.SLOT.SYSTEM installé, les tableaux peuvent être dimensionnés avec les tailles maximales suivantes :

Type de tableau	AUX.SLOT.SYSTEM	APPLEMEM.SYSTEM
Réel	9419	13106
Entier	23550	32766
Chaîne	15699	21844

Vitesse des programmes

Vos programmes compilés s'exécuteront légèrement plus lentement avec les fichiers AUX.SLOT.SYSTEM et APPLEMEM.SYSTEM. La différence de vitesse dépendra de la quantité de manipulation de chaînes.

Quelle est la place disponible ?

Après avoir installé AUX.SLOT.SYSTEM ou APPLEMEM.SYSTEM, lancez le programme Applesoft HOW.MUCH pour connaître la place disponible pour les variables. Attendez-vous à voir des chiffres impressionnants !

Pour la traduction ©1988 Joëlle Piard & Pom's