Ce livre de référence est destiné à se trouver en permanence à côté de votre Apple II. Son but est de vous faire accéder rapidement à l'information dont vous avez besoin : syntaxe des commandes, codes caractères, messages d'erreurs, codes machines, adresses utiles. Il comporte également un recueil de 25 "trucs" utiles, les "Comment...?".

CLEFS POUR L'APPLE II

APPLE II plus et APPLE//e

Editions du P.S.I.
B.P. 86
77402 Lagny/Marne
France

ISBN : 2.86595.073.5

105 FF
Imprimé en France

Couverture Mariane de Nayer

CLEFS POUR L'APPLE II

Nicole Bréaud-Pouliquen

CLEFS POUR L'APPLE II

APPLE II plus et APPLE//e

Nicole Bréaud-Pouliquen

Editions du P.S.I.

Autres ouvrages relatifs au matériel APPLE

Certains de ces ouvrages disposent d'une disquette d'accompagnement*.

— Apple pour tous — Jacques Boisgontier et Sophie Brébion
— 36 programmes Apple II pour tous — Jacques Boisgontier
— La pratique de l'Apple II, tome 1 — Nicole Bréaud-Pouliquen
— La pratique de l'Apple II, tome 2 — Nicole Bréaud-Pouliquen
— La pratique de l'Apple II, tome 3 — Nicole Bréaud-Pouliquen et Daniel-Jean David
— Exercices pour Apple II — Frédéric Lévy, traduit par André Babéanu.
— La découverte de l'Apple II — Dominique Schraën et Frédéric Lévy.
— 102 programmes pour Apple II — Jacques Deconchat, adapté par Jacques Labidurie
— Recueil Pom's n° 1
— Recueil Pom's n° 2
— Gestion de fichiers et de périphériques pour Apple II — Hervé Haut
— * Visicalc sur Apple — Hervé Thiriez
— Pascal UCSD sur Apple II, tome 1 — Jacques Rouault et Patrice Girard
— Pascal UCSD sur Apple II, tome 2 — Jacques Rouault et Patrice Girard
— * Multiplan pour Apple II — Hervé Thiriez
— Les Bases de données sur Apple II — Michel Keller
— * L'Apple et ses fichiers — Jacques Boisgontier
— Microbook : base de données pour Apple II — Ted Lewis, traduit par André Babéanu
— Du Logo pour Apple II — Nicole Bréaud-Pouliquen
— ProDOS sur Apple — Francis Verscheure

A paraître :
— Clefs pour Apple IIe — Nicole Bréaud-Pouliquen
— Techniques de programmation sur Apple II — René Belle
— Destination aventure, Programmes de jeux de rôle et d'aventure sur Apple II — Delton Horn

ISBN : 2-86595-073-5

CLEFS POUR L'APPLE II

APPLE II plus et APPLE //e

par

Nicole Bréaud-Pouliquen

Editions du P.S.I.
1984

Nicole BREAUD-POULIQUEN *est ingénieur-conseil en informatique individuelle. Dans ce cadre, elle enseigne la programmation depuis plusieurs années.*

Elle pratique l'APPLE II depuis sa commercialisation en France et s'intéresse plus particulièrement à l'insertion des ordinateurs individuels dans l'enseignement.

Du même auteur :

Aux Editions du P.S.I.

- La Pratique de l'Apple II - volume I
  Basic Applesoft Système Apple Graphiques
- La Pratique de l'Apple II - volume II
  Périphériques et gestion des fichiers
- La Pratique de l'Apple II - volume III
  (avec Daniel-Jean DAVID)
  Langage machine et Assembleur du 6502
- LISP sur Apple II
- Du Logo pour Apple

P R E S E N T A T I O N

Ce livre est destiné à se trouver en permanence à côté de votre Apple II lorsque vous l'utilisez. Il renferme toutes les informations de référence que vous pouvez souhaiter retrouver rapidement : syntaxe des commandes, codes caractères, messages d'erreur, langage machine et adresses du système.

Les informations sont données sans détails excessifs, le but principal de ce livre étant l'accès rapide à l'information : pour un exposé plus introductif et plus complet, vous pouvez vous reporter à la série "La Pratique de l'Apple II" (volumes 1, 2 et 3).

Le chapitre "comment... ?" rassemble astucieusement tous les "trucs" de différents niveaux qu'il est utile de savoir. Un index placé à la fin de cette section vous permettra de les retrouver rapidement.

Nous serons reconnaissant à nos lecteurs de toute suggestion d'informations complémentaires à incorporer aux futures éditions de ce livre.

S O M M A I R E

FONCTIONS DU BASIC APPLESOFT

Une fonction demande un argument (ou plusieurs) et renvoie une valeur qui est le résultat de l'application de cette fonction à la valeur de l'argument.

Fonctions mathématiques

ABS Valeur absolue de l'argument entre parenthèses.

ATN Arc tangente - le résultat est en radians, compris entre $-\pi/2$ et $+\pi/2$.

COS Cosinus - l'argument doit être en radians. Exemple : cos(x en degrés) ⇒ COS(π*X/180).

EXP Exponentielle e^X. L'argument doit être ≤ 88 sinon il se produit un dépassement de capacité.

INT Partie entière, plus exactement le plus grand entier inférieur ou égal à l'argument : INT(0.5) vaut 0 ; INT(5) vaut 5 ; INT(-0.5) vaut -1 ; INT(-3) vaut -3.

LOG Logarithme naturel (népérien ou en base e). Pour obtenir le logarithme de X en base Y, utiliser LOG(X)/LOG(Y). Exemple : logarithme décimal de X⇒LOG(X)/LOG(10).

RND Fournit un nombre pseudo-aléatoire supérieur ou égal à 0 et inférieur à 1, avec un argument positif.
Exemple : PRINT RND(1) ⇒.103112573
Si les appels de la fonction se succèdent, les résultats suivront toujours la même suite de nombres aléatoires. Un appel de la fonction avec un argument négatif permet d'amorcer une suite particulière. RND(0) renvoie le dernier nombre généré.

SGN Fonction "signe" : 1 si X>0, -1 si X<0 et 0 si X=0.

SIN Sinus - l'argument est supposé en radians.

SQR Racine carrée - l'argument doit être supérieur ou égal à 0.

TAN Tangente - l'argument est supposé en radians.

Fonctions de tabulation

POS POS(Ø) fournit la prochaîne position d'affichage libre sur la ligne d'écran (position horizontale du curseur)

SPC Ne peut s'employer que dans une instruction PRINT. PRINT SPC(X); imprime X espaces. X doit être entier compris entre Ø et 255.

TAB Ne peut s'employer que dans une instruction PRINT TAB(X) fait aller à la position d'impression n°X (1 est la position la plus à gauche d'une ligne, 40 la plus à droite) X doit être compris entre 1 et 255 TAB(Ø) déplace le curseur à la position 256. Si X<position courante du curseur, alors il n'y a pas d'action. TAB ne déplace jamais le curseur vers la gauche.

HTAB Positionne le curseur horizontalement avant l'instruction PRINT. HTAB 1 correspond à la position la plus à gauche de la fenêtre d'écran. La position extrême est 255 (5 lignes plus loin). Attention HTAB ne déplace pas correctement le curseur dans l'affichage 8Ø colonnes.

VTAB Positionne le curseur verticalement avant l'instruction PRINT. VTAB 1 correspond à la 1ère ligne du haut de l'écran quelle que soit la valeur de la marge haute de la fenêtre.

VTAB 24 positionne à la dernière ligne d'écran. Si l'argument est supérieur à la marge basse de la fenêtre d'écran, alors l'affichage ne pourra se faire qu'à la ligne pointée par l'argument pour toutes les instructions PRINT suivantes.

Fonctions système

FRE Quelle que soit la valeur de l'argument, fournit le nombre d'octets restés libres en mémoire. Provoque un nettoyage des chaînes abandonnées.

PEEK Fournit le contenu (compris entre Ø et 255) de la case mémoire dont l'adresse est égale à son argument (qui doit être entier et compris entre Ø et 65 535).

USR Appel d'un programme utilisateur en langage machine. L'unique argument est transmis à travers l'accumulateur flottant. L'adresse du sous-programme doit être préenregistrée en $ØB et $ØC avec JMP ($4C) en $ØA. Le résultat est placé dans l'accumulateur flottant.

Fonctions chaînes de caractères

LEN(X$) Longueur (de Ø à 255).

LEFT$(X$,N) Extraction des N caractères les plus à gauche.

RIGHT$(XS,N) Extraction des N caractères les plus à droite.

MID$(X$,K) ou MID$(X$,K,N) Extraction au milieu de tous ou de N caractères à partir de la K ième position. K doit être égal ou supérieur à 1.

Fonction de conversion

ASC(X$) Renvoie le code ASCII du premier caractère de la chaîne X$. ASC("A") vaut 65.

CHR$(K) Renvoie le caractère dont le code ASCII vaut K. CHR$(4) est 'Ctrl D'.

STR$(A) Donne la représentation d'un nombre en chaîne de caractères à partir de sa valeur numérique A.

VAL(X$) Donne la valeur numérique représentée par la chaîne X$.

Fonctions graphiques (basse résolution)

COLOR= Donne une couleur (Ø à 15) pour le prochain tracé en basse résolution.

PLOT X,Y Place un petit domino à l'abscisse X et à l'ordonnée Y. X et Y vont de Ø à 39. Ø,Ø est la position en haut, à gauche.

HLIN X1,X2 AT Y Trace une ligne horizontale entre X1 et X2 à l'ordonnée Y.

VLIN Y1,Y2 AT X Trace une ligne verticale entre Y1 et Y2 à l'abscisse X.

SCRN(X,Y) Renvoie la couleur du domino tracé en X,Y.

Fonctions graphiques (haute résolution)

H COLOR= Fixe la couleur (Ø,1,2,3) du prochain point à tracer.

HPLOT X,Y Place un point à l'abscisse X et à l'ordonnée Y. X va de Ø à 279. Y de Ø à 159 (HGR) ou à 191 (HGR2). Ø,Ø est en haut et à gauche.

HPLOT X1,Y1 TO X2,Y2 Trace une ligne entre 2 points, la commande peut être étendue à d'autres points... TO XN,YN.

DRAW F AT X,Y Dessine la forme n° F de la table des formes en commençant au point X,Y.

XDRAW F AT X,Y Dessine la forme n° F de la table des formes. La couleur de chaque point est le complément de la couleur actuelle du point sur l'écran.

ROT= L'argument est proportionnel à un angle de rotation à faire subir à la forme à dessiner par DRAW. ROT=16 correspond à 90°.

SCALE= Donne une valeur d'agrandissement à la forme à dessiner de 1 à 255.

Fonctions : manettes de jeux

PDL(X) Renvoie un nombre de Ø à 255 proportionnel à la position angulaire de la manette (potentiomètre). X vaut Ø,1,2 ou 3.

PEEK(X-16287) Donne un résultat >127 si le bouton poussoir n° X a été pressé. X vaut Ø,1 ou 2.

Fonction haut-parleur

PEEK(-16336) Emet un 'clicks' par le haut-parleur.

Mode obligatoire	*Mot-clé*	Définition - *exemples*
	&	Fait démarrer l'exécution d'un sous-programme en langage machine dont l'adresse figure en $3F6, $3F7 avec JMP ($4C) dans $3F5 *&16→1Ø* si le sous-programme convertit du décimal en hexadécimal
	CALL	Fait démarrer l'exécution d'un sous-programme en langage machine à l'adresse indiquée. *CALL -151* Un argument négatif correspond au complément à 65536 de l'adresse cherchée
	CLEAR	Remise à zéro de toutes les variables. Les chaînes ont la longueur nulle
Direct	**CONT**	Continuer dans le programme après interruption
Direct	**'Ctrl C'**	Arrêt d'exécution d'un programme en cours. Le programme reste intact
	'Ctrl D'	Début d'une commande du SED précédée de PRINT"
	'Ctrl G'	Emission d'un 'click' sonore
	'Ctrl H'	Déplacement du curseur d'une position sur la gauche (identique à ←)
	'Ctrl I'	Début d'une commande à l'imprimante précédée de PRINT" permettant d'ajuster le nombre de caractères par lignes imprimées : *PRINT"'Ctrl I'8ØN"* pour 8Ø caractères
	'Ctrl J'	Déplacement du curseur d'une position vers le bas
//e,8Ø col.	**'Ctrl K'**	Effacement jusqu'au bas de la fenêtre
//e,8Ø col.	**'Ctrl L'**	Voir HOME c'est-à-dire écran effacé et curseur en haut
	'Ctrl M'	Curseur au début de la ligne suivante
//e,8Ø col.	**'Ctrl N'**	Affichage normal blanc sur noir
et programmé	**'Ctrl O'**	Affichage inverse noir sur blanc
//e,8Ø col.	**'Ctrl Q'**	Affichage sur 4Ø colonnes
et programmé	**'Ctrl R'**	Affichage sur 8Ø colonnes

(précédé de PRINT")

Mode obligatoire	*Mot-clé*	Définition - *exemples*
//e,8Ø col.	'Ctrl Reset'	Voir 'Reset'
Direct	'Ctrl S'	Suspension de l'affichage, l'image reste fixe tant qu'on n'appuie pas sur une touche quelconque
//e,8Ø col. et programmé	'Ctrl U'	Permet de désactiver la MEM de gestion des 80 colonnes. (Equivaut à **'Esc''Ctrl Q'** utilisable seulement en mode direct).
//e,8Ø col.	'Ctrl W'	Défilement de l'écran d'une ligne vers le bas
//e,8Ø col.	'Ctrl Y'	Effacement de l'écran sans déplacer le curseur
//e,8Ø col.	'Ctrl Z'	Efface toute la ligne où se trouve le curseur
//e,8Ø col.	'Ctrl ç'\	Déplace le curseur vers la droite
//e,8Ø col.	'Ctrl §']	Efface jusqu'au bout de la ligne courante
Direct	'Ctrl X'	Annulation de l'entrée d'une ligne ou d'une donnée en cours
	DATA	Définit une liste de constantes qui seront "lues" par une instruction READ *1Ø DATA ABC,5,Ø.15*
	DEL	Avec deux arguments séparés par une virgule, délimite une partie de programme à supprimer *DEL 1Ø,5Ø* supprime les instructions de 1Ø à 5Ø
	DEF FN	Définition d'une fonction utilisateur à un seul argument : *1Ø DEF FN F(X)=X-256*INT(X/256)*
	DIM	Dimensionnement d'un tableau (fixe le nombre et les valeurs maxima des indices) *1Ø DIM A(1ØØ),B%(5ØØ),C$(1Ø)* *2Ø DIM T(N)* *3Ø DIM M(1Ø,1Ø,1Ø)* 88 indices au plus
Programmé	**END**	Arrêt d'exécution de la suite d'instructions

Mode obligatoire	*Mot-clé*	Définition - *exemples*
	FOR	Introduit une boucle : toutes les instructions comprises entre *FOR I=A TO B STEP C* et le NEXT I seront répétées pour toutes les valeurs de I allant de A à B, C par C *1Ø FOR I=1 TO 1ØØØ* *2Ø FOR X=1.5 TO 2 STEP.1* *3Ø FOR J=N TO -N STEP -2* Si plusieurs boucles se succèdent avec le même indice, ne pas interrompre la progression de l'indice jusqu'à sa valeur maxima *1Ø FOR I=1 TO 1ØØ* *2Ø IF N$(I)=X$ THEN T=I:I=1ØØ* *3Ø NEXT I* *4Ø IF T=Ø THEN PRINT"NON TROUVE":END* *5Ø PRINT"TROUVE EN";T*
	FLASH	Affichage des caractères en mode clignotant. Ce mode n'est pas neutralisé par 'Reset'. La touche → de recopie va modifier les caractères sur l'écran. Faire NORMAL pour rétablir la situation
Programmé	**GET**	Saisie 1 caractère au clavier, il n'est pas affiché. 'Ctrl C' n'a aucun effet *1Ø GET A$* n'est pas recommandé avec des instructions du SED dans le programme. Sauf si 'Ctrl D' est précédé de 'Return' *D$=CHR$(13)+CHR$(4)*
	GOSUB	Appel d'un sous-programme *1Ø GOSUB 1ØØØ*
	GOTO	Saut à une autre instruction numérotée *1Ø GO TO 5Ø*
	GR	Met une partie de l'écran en affichage graphique de 40 x 40 dominos et l'efface. Laisse 4 lignes de texte dans le bas

Mode obligatoire	Mot-clé	Définition - *exemples*
	HGR	Met une partie de l'écran en affichage graphique de 280 x 160 et l'efface Laisse 4 lignes de texte
	HGR2	Met tout l'écran en graphique de 280 x 192 et l'efface. (Le curseur disparaît)
	HIMEM:	Spécifie la plus haute position de mémoire vive, disponible au programme
	HOME	Efface la fenêtre d'écran et positionne le curseur en haut et à gauche de cette fenêtre. Précédée de TEXT : tout l'écran est effacé
	IF	Saut conditionnel, de la forme IF condition THEN instruction. Si la condition n'est pas satisfaite (résultat faux ou Ø) on passe à la ligne suivante ; si la condition est satisfaite, on effectue l'instruction qui suit THEN IF C THEN GOTO x s'écrit aussi IF C THEN x ou encore IF C GOTO x *1Ø IF A>B THEN Y=K* *2Ø IF A$=""THEN 5* *3Ø IF A<Ø OR A>1ØØ THEN 1ØØ*
Programmé	INPUT	Acquisition de données au clavier *1Ø INPUT A* *2Ø INPUT A,B,C$,D* *3Ø INPUT"VOTRE NOM?";N$*
	IN #	Connecte le périphérique branché sur le connecteur indiqué comme argument, en entrée du micro-ordinateur
	INVERSE	Provoque l'affichage des caractères en noir sur blanc. Pour revenir en blanc sur noir faire NORMAL
	LET	Est l'instruction d'affectation de valeur à une variable. N'est pas obligatoire *LET X$="AOUT"*
	LIST	Liste du programme *LIST tout le programme* *LIST 1Ø,1ØØ de 1Ø à 1ØØ* *LIST 1ØØ, de 1ØØ à la fin* *LIST, 1Ø jusqu'à 1Ø* La virgule peut être remplacée par -

Mode obligatoire	Mot-clé	Définition - *exemples*
	LOAD	Chargement d'un programme de la cassette vers la MEV
	LOMEM:	Spécifie la plus basse position de mémoire vive disponible pour les variables du programme
Direct	NEW	Simule l'effacement du programme actuellement en mémoire vive. (2 pointeurs sont mis à zéro)
	NEXT	Fait passer à l'itération suivante dans un FOR *NEXT I NEXT J,I NEXT*
	NORMAL	Rétablit l'affichage sur écran en blanc sur noir
	NOTRACE	Déconnecte le mode TRACE
	ON	ON I GOTO 1Ø,2Ø,3Ø Si I vaut 1, on va en 1Ø, s'il vaut 2, on va en 2Ø, en 3Ø s'il vaut 3 Si I est nul ou faux, on passe à l'instruction suivante ON I GOSUB 1ØØØ,3ØØØ Si I vaut 1, le sous-programme 1ØØØ est appelé, 2 c'est 3ØØØ
	ONERR	ONERR GOTO 5ØØ Permet d'intercepter une erreur avant qu'elle ne provoque l'arrêt de l'exécution du programme. Quand une erreur se produit il y a saut à l'instruction indiquée
	POKE	POKE a,b écrit la donnée b à l'adresse absolue a. (a et b exprimés en décimal) *POKE 2ØØØ,65*
	POP	Désempilement de la dernière adresse de retour (d'un sous-programme) du stack Le prochain RETURN fera revenir à l'instruction suivant l'avant-dernier GOSUB exécuté
	PRINT	Affiche en résultat sur écran ou sur imprimante *PRINT A* *1Ø PRINT A;B;J (juxtaposés)* *3Ø PRINT "X=";X* *2Ø PRINT A,B,J (en zones fixes)*

Mode obligatoire	*Mot-clé*	Définition - *exemples*
	PR #	Transfère la sortie au périphérique dont la carte d'interface est dans le connecteur spécifié par l'argument *PR#1* permet l'impression si l'interface de l'imprimante est sur le connecteur n°1
(car.F)	**PR £**	PR £ 3 permet d'activer la MEM et la MEV pour l'affichage en 8Ø colonnes ; faire *'Esc' 'Ctrl/Q'* pour les désactiver.
Programmé	**READ**	Lecture de données dans une instruction DATA associée *1Ø READ A* *2Ø READ B$,C*
	RECALL	Récupération de données numériques de la cassette vers la MEV. L'argument est une variable préalablement et correctement dimensionnée *5 DIM B(1ØØ)* *1ØØ RECALL B*
Programmé	**REM**	Introduit un commentaire dans le listing du programme
][et][Plus	**'RESET'**	Cette touche équivaut à 'Ctrl C' pendant l'exécution d'un programme. Le programme en cours s'arrête mais est intact. Les périphériques en ligne ne sont plus sélectés.
	'RESET'	Retour à l'interpréteur ou à l'adresse prévue dans SOFTEV ($3F2,$3F3) si PWERDUP ($3F4) est conforme, sinon le système redémarre comme si on venait de le mettre sous tension (COLDSTART)
	RESTORE	Revient au début des DATA
	RETURN	Retour de sous-programme *1ØØ RETURN*
	RESUME	Revient à l'instruction d'où était issue l'erreur traitée dans le programme par ONERR GOTO
	RUN	Déclenche l'exécution d'un programme Met à zéro toutes les variables. *RUN RUN 3Ø*

Mode obligatoire	*Mot-clé*	Définition - *exemples*
	SAVE	Sauvegarde d'un programme sur cassette
	SPEED=	Modifie la cadence d'affichage sur écran de 1 (minimum) à 255 (standard)
	STEP	Introduit le pas d'incrémentation dans FOR
	STOP	Arrête l'exécution d'un programme *1Ø STOP* et affiche le message *?BREAK IN 1Ø* On peut continuer avec CONT (si les instructions ne sont pas modifiées)
	STORE	Sauvegarde un tableau de valeurs numériques sur cassette. Ne fonctionne pas directement avec des chaînes de caractères *STORE A*
	TEXT	Affichage en mode texte après un mode graphique. Restaure les valeurs standards de fenêtre d'écran 40 caractères par ligne 24 lignes par écran
	THEN	Introduit l'instruction à effectuer quand un IF est satisfait
	TO	Introduit la valeur limite d'un FOR
	TRACE	Met en mode détection et supression éventuelle d'erreurs (debugging). Affiche les numéros d'instructions exécutées sans 'Return' donc entre les lignes de résultats du programme
	WAIT	Pause conditionnelle dans un programme *WAIT A,B* Suspend l'exécution du programme jusqu'à ce que le contenu de l'adresse A ET (bit à bit) l'équivalent binaire de B soit différent de Ø *WAIT -16384,128* est l'attente qu'une touche soit tapée au clavier

+	Addition de nombres ou concaténation de chaînes de caractères
-	Soustraction ou prendre l'opposé
*	Multiplication
/	Division
^	Elévation à la puissance
=	Egal <>différent
<	Inférieur >supérieur
<=	Inférieur ou égal
=<	Inférieur ou égal
>=	Supérieur ou égal
=>	Supérieur ou égal
NOT	Non logique, agit sur 1 seul opérande Si A est vrai NOT A est faux Si A est faux NOT A est vrai
AND	ET logique sur 2 opérandes P AND Q n'est vrai que si P et Q sont vrais
OR	OU logique sur 2 opérandes P OR Q n'est faux que si P et Q sont faux

] est le caractère de sollicitation du Basic Applesoft de l'Apple II et de l'Apple //e en clavier QWERTY.

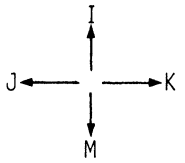
§ est le caractère de sollicitation du Basic Applesoft de l'Apple //e en clavier AZERTY.

Les différentes formes de curseur en Apple //e rappellent si la MEM gérant l'affichage sur 8Ø colonnes a été activée et si le curseur est en mode déplacement :

- Damier carré clignotant : MEM 8Ø col. non activée ; 4Ø col.
- Rectangle blanc : MEM 8Ø col. active ; 8Ø col.
- Plus inversé dans rectangle : 8Ø col. et curseur en déplacement.
- Carré blanc : MEM 8Ø col. active ; 4Ø col.
- Plus inversé dans carré : 4Ø col. et curseur en déplacement.

- *Déplacement du curseur*
- *Effacement de lignes sur l'écran*
- *Recopie de lignes en MEV*
- *Suppression de lignes de la MEV*

Avec un monitor AUTOSTART ROM en MEM : les commandes de déplacement du curseur sont obtenues par les 4 touches ci-contre, précédées de 'esc'.



En n'appuyant que sur ces 4 touches, on reste en mode déplacement du curseur dans les 4 directions.

Pour revenir au mode normal d'insertion et de correction, on appuie à nouveau sur 'esc'.

Pour effacer à partir de la position du curseur jusqu'en bas de la page, faire 'esc' E.

Pour effacer à partir de la position du curseur jusqu'au bas de la page, faire 'esc' F.

Pour effacer tout l'écran et positionner le curseur en haut à gauche faire 'esc' (SHIFT/P).

En se servant de la touche←(BS), on annule le dernier caractère tapé.

En se servant de la touche→on réenregistre en MEV le caractère sous le curseur. Pour réenregistrer une série de caractères utiliser→et la touche REPT, vous recopierez plus vite.

Pour supprimer une ligne d'instructions du programme en mémoire vive, taper le numéro de la ligne et 'Return'.

Avec le modèle Apple //e, le déplacement du curseur s'obtient par les commandes citées ci-dessus lorsque l'interpréteur Basic ou le Moniteur sont en ligne.

De plus, les 4 touches de direction flêchées jouent aussi le rôle de déplacement du curseur à condition de les faire précéder de la frappe de 'Esc'.

Les touches 'Del' ou effacement, 'flèche en haut', 'flèche en bas', 'Tab' ne sont opérantes que si le logiciel en cours en tient compte (traitement de texte par exemple).

Les nouvelles possibilités sur Apple //e concernent la sélection majuscule/minuscule avec la touche 'blocage de LETTRES majuscules' ou la touche 'Shift', cette dernière étant absolument nécessaire pour la sélection des chiffres ou des caractères de ponctuation gravés sur le haut des touches.

Dans l'écriture d'un programme en Basic, il est impératif de bloquer la touche 'lettres majuscules' ou bien de se mettre en mode majuscule restreint en donnant la commande 'Esc' R, mode qui n'enverra des minuscules qu'après PRINT". On quitte le mode majuscule restreint par 'Esc' T.

En mode d'affichage 8Ø colonnes obtenu par PR£3, la commutation en affichage 4Ø colonnes est obtenue par 'Esc' 4 et le retour en 8Ø colonnes par 'Esc' 8.

Les mots-clés par ordre alphabétique et les codes correspondants en hexadécimal.

Lettres A à S

Mot-clé	*Code hexadécimal*	*Mot-clé*	*Code hexadécimal*	*Mot-clé*	*Code hexadécimal*
&	$AF	GR	$88	ON	$B4
ABS	$D4	HCOLOR=	$92	ONERR	$A5
AND	$CD	HGR	$91	OR	$CE
ASC	$E6	HIMEM:	$A3	PDL	$D8
AT	$C5	HLIN	$8E	PEEK	$E2
ATN	$E1	HOME	$97	PLOT	$8D
CALL	$8C	HPLOT	$93	POKE	$B9
CHR$	$E7	HTAB	$96	POP	$A1
CLEAR	$BD	IF	$AD	POS	$D9
COLOR	$AØ	IN#	$8B	PRINT	$BA
CONT	$BB	INPUT	$84	PR#	$8A
COS	$DE	INT	$D3	READ	$87
DATA	$83	INVERSE	$9E	RECALL	$A7
DEF	$B8	LEFT$	$E8	REM	$B2
DEL	$85	LEN	$E3	RESTORE	$AE
DIM	$86	LET	$AA	RESUME	$A6
DRAW	$94	LIST	$BC	RETURN	$B1
END	$8Ø	LOAD	$B6	RIGHT$	$E9
EXP	$DD	LOG	$DC	RND	$DB
FLASH	$9F	LOMEM:	$A4	ROT	$98
FN	$C2	MID$	$EA	RUN	$AC
FOR	$81	NEW	$BF	SAVE	$B7
FRE	$D6	NEXT	$82	SCALE	$99
GET	$BE	NORMAL	$9D	SCRN(	$D7
GOSUB	$BØ	NOT	$C6	SGN	$D2
GOTO	$AB	NOTRACE	$9C	SHLOAD	$9A
				SIN	$DF

Les mots-clés (suite)

Lettres S à X

Mot-clé	*Code hexadécimal*	*Mot-clé*	*Code hexadécimal*	*Mot-clé*	*Code hexadécimal*
SPC(	$C3	TAB(	$CØ	VAL	$E5
SPEED=	$A9	TAN	$EØ	VLIN	$8F
SQR	$DA	TEXT	$89	VTAB	$A2
STEP	$C7	THEN	$C4	WAIT	$B5
STOP	$B3	TO	$C1	XPLOT	
STORE	$A8	TRACE	$9B	XDRAW	$95
STR$	$E4	USR	$D5		

XPLOT est codé en : $58 $8D
'X' 'PLOT'

Il ne faut pas l'utiliser comme nom de variable.

Mots-clés par ordre alphabétique Lettre A à V

Mot-clé	*Code hexadécimal*	*Mot-clé*	*Code hexadécimal*	*Mot-clé*	*Code hexadécimal*
ABS	$31	INPUT	$52	PRINT	$61
AND	$1D		$53		$62
ASC	$3C		$54		$63
AT	$6B	LEN	$3B	PR#	$7E
	$6E	LET	$5E	REM	$5D
AUTO	$ØD	LIST	$74	RETURN	$5B
CALL	$4D		$75	RND	$2F
CLR	$ØC	LOAD	$76	RUN	$Ø7
COLOR=	$66	LOMEM	$Ø4		$Ø8
CON	$Ø6	MAN	$11	SAVE	$Ø5
DEL	$Ø9	MOD	$ØF	SCRN(	$3D
DIM	$4E	NEW	$1F	SGN	$3Ø
	$4F	NEXT	$ØB	STEP	$58
DSP	$7B	NOT	$59	TAB	$5Ø
	$7C	NOTRACE	$37	TEXT	$4B
END	$51	NODSP	$7A	THEN	$24
FOR	$55		$78		$25
GOSUB	$5C	OR	$79	TO	$57
GOTO	$5F	PDL	$1E	TRACE	$7D
GR	$4C	PEEK	$32	VLIN	$6C
HLIN	$69	PLOT	$2E	VTAB	$6F
IF	$6Ø	POKE	$67		
IN#	$7F	POP	$64		
			$77		

Fonctions mathématiques

ABS	Valeur absolue.
MOD	Reste de la division du 1er opérande par le second. PRINT 15 MOD 4 donne 3
RND	Génère un nombre entier pseudo-aléatoire inférieur à l'argument et positif. PRINT RND(1Ø) donne 3
SGN	Signe de l'argument : -1, Ø, 1

Fonctions de tabulation

TAB	Equivaut à HTAB en Applesoft. Positionnement du curseur de 1 à 255 dans la fenêtre d'écran.
VTAB	Positionnement vertical du curseur (en absolu) de 1 à 24.

Fonctions système

PEEK	Contenu d'une position de mémoire dont l'adresse est égale à l'argument. L'argument est compris entre -32768 et +32767.

Fonction de conversion

ASC("Z")	Renvoie le code ASCII du caractère.

Fonctions graphiques

COLOR=	Donne une couleur (Ø à 15) pour le prochain tracé en basse résolution.
PLOT X,Y	Place un petit domino à l'abscisse X et à l'ordonnée Y. X et Y vont de Ø à 39. Ø,Ø est la position en haut, à gauche.
HLIN X1,X2 AT Y	Trace une ligne horizontale entre X1 et X2 à l'ordonnée Y.
VLIN Y1,Y2 AT X	Trace une ligne verticale entre Y1 et Y2 à l'abscisse X.
SCRN(X,Y)	Renvoie la couleur du domino tracé en X,Y.

Fonction manettes de jeux

PDL(X) Renvoie un nombre de Ø à 255 proportionnel à la position angulaire de la manette (potentiomètre). X vaut Ø,1,2 ou 3.

PEEK(X-16287) Donne un résultat >127 si le bouton poussoir n° X a été pressé. X vaut Ø,1 ou 2.

Fonction haut-parleur

PEEK(-16336) Emet un 'clicks' par le haut-parleur.

Mot-clé	Définition - *exemples*
AUTO	Numérotation automatique à partir du numéro donné en argument. AUTO 1ØØ,5 : numérotation de 5 en 5 à partir de 1ØØ. Si le 2ème argument n'est pas spécifié la numérotation va de 1Ø en 1Ø
CALL	Fait exécuter un sous-programme qui commence à l'adresse indiquée <32767. CALL-936 (efface l'écran)
CLR	Remise à zéro de toutes les variables
CON	Poursuite du programme après interruption
'Ctrl C'	Arrêt du programme
DEL	Suppression de lignes d'instructions *DEL 1Ø,1ØØ*
DIM	- Dimensionnement d'un tableau à 1 seule dimension maximum - Dimensionnement de la longueur maximum d'une variable chaîne de caractère *1Ø DIM N$(2Ø)* (obligatoire pour toutes les variables chaînes de caractères)
DSP	Affiche les nouvelles valeurs prises par la variable spécifiée, pendant l'exécution du programme *1Ø X=RND(1Ø)* *15 DSP X* *2Ø GOTO 1Ø* *RUN → #1Ø X=3* *#1Ø X=7 etc.*
END	Dernière instruction du texte en Basic Si elle manque, produit un message d'erreur
FOR	Boucle FOR-NEXT de répétition d'instructions encadrées par FOR et NEXT avec à chaque boucle, comptage et test de fin de boucle
GOSUB	Appel d'un sous-programme à une adresse qui peut être une expression arithmétique ou une variable *GOSUB 3*X+1Ø*
GOTO	Saut à une autre instruction dont le numéro peut être calculé dans le programme *GOTO N*1Ø*
GR	Ecran graphique 4Øx4Ø

Mot-clé	Définition - *exemples*
IF	Saut conditionnel de la forme : IF condition THEN instruction V : instruction F Si la condition n'est pas satisfaite seule l'instruction V juste après le THEN est ignorée, donc on passe à l'instruction F. Si la condition est satisfaite l'instruction V est exécutée puis l'instruction F
INPUT	Acquisition de données au clavier. Il faut séparer le message de la variable par une virgule *1Ø INPUT "VOTRE NOM?",N$* A l'exécution, un point d'interrogation est affiché devant le curseur s'il y avait ou non un message et si la variable est numérique
IN #	Entrée de données par un périphérique. L'argument est le numéro du connecteur (1 à 7) correspondant
LEN(	Nombre de caractères d'une variable chaîne
LET	Affectation de valeur à une variable
LIST	Liste des instructions d'un programme
LOAD	Chargement d'un programme depuis la cassette vers la MEV
LOMEM:	Modifie le début d'implantation des variables
MAN	Pour cesser le mode automatique de numérotation. A l'apparition d'un nouveau numéro d'instruction, reculer le curseur jusqu'au > et taper *MAN*
NEW	Efface le programme en MEV
NEXT	Fait passer à l'itération suivante dans un FOR
NODSP	Pour faire cesser la visualisation des changements de valeurs demandée par DSP
NOTRACE	Pour faire cesser le mode TRACE
POKE	POKE a,b écrit la donnée b (inférieure à 256) dans la case mémoire d'adresse b *(-32768 à 32767)*
POP	Elimination d'un niveau de retour de sous-programme
PRINT	Affichage de résultats sur l'écran
PR	Changement de périphérique de sortie
REM	Introduit un commentaire
RETURN	Retour de sous-programme

Mot-clé	Définition - *exemples*
RUN	Exécution, les variables dimensionnées n'ont pas été mises à zéro
SAVE	Sauvegarde sur cassette
STEP	Pas d'incrémentation dans FOR
TEXT	Retour au mode texte après le mode graphique. Restauration de la fenêtre standard
THEN	Introduit l'instruction à effectuer quand un IF est satisfait. La **seconde** instruction après THEN s'effectuera si la condition est fausse ainsi que les suivantes
TO	Précède la variable valeur limite dans FOR
TRACE	Si plusieurs instructions sont sur la même ligne, la trace n'indiquera que le passage par la première instruction de la ligne.

Les noms de variables en INTEGER sont conservés entièrement quelle que soit leur longueur.

Seuls les opérations entières sont possibles en INTEGER.

Les erreurs de syntaxe sont détectées dès que la ligne est validée par 'Return'.

Opérateurs

+ - * / (division entière) ^(puissance)
= (égal) #(différent) >,<
NOT, AND, OR

Configuration standard

+ 48K de mémoire vive
+ carte langage de 16K de MEV sur le connecteur n° Ø
+ 2 lecteurs de disquettes
+ 1 contrôleur avec le PROM P5A et P6A (16 secteurs) dans le connecteur n° 6.

Disquettes du système Pascal

APPLE 1 :
APPLE 2 : } système { Editeur, Filer, Apple, Library
Compilateur, Linker, Assembler, etc.
APPLE 3 : Démographiques, Formater, Library
BASIC S : passage en BASIC (sous DOS 3.2) et chargement d'un interpréteur sur la carte-langage.

Formatage des disquettes vierges sous PASCAL :

X (exécution) du programme APPLE 3 : FORMATER
Disquette vierge en D2
Répondre 5 à FORMAT WHICH DISK (4,5,9...12) ? ou 'Return' pour arrêter.

Copier des disquettes :

F (gestion des disquettes) puis **T**(ransfert)
TRANSFER ? Nom de la disquette à recopier :
TO WHERE ? BLANK :
TRANSFER 280 BLOCKS ? **Y**
DESTROY BLANK : ? **Y**

Commandes

E(DIT	Appel de l'éditeur
R(UN	Appel d'un programme source (.TEXT), compilation et exécution
F(ILER	Gestion des fichiers et programmes sur disquettes
C(OMP	Compilation
L(INK	Liaison de programmes déjà compilés
X(ECUTE	Exécution d'un programme objet (.CODE)
A(SSEM	Appel de l'assembleur
D(EBUG	A ne pas utiliser
H	Redémarrage avec une disquette en DOS 3.3 ou BASIC S (pour le DOS 3.2)

'CTRL A'	Visualisation des caractères de la colonne 41 à 80 (partie droite du texte)
'CTRL K'	Pour entrer le caractère [
'SHIFT M'	Pour entrer le caractère]

>EDIT **A**(DJUST **C**(OPY **D**(ELETE **F**(IND **I**(NSERT **J**(UMP
R(EPLACE **Q**(UIT **X**(CHANGE **Z**(AP **S**(ET **V**(ERIFY

Sens de déplacement dans le texte

>normal avec → et ← ; 'CTRL A' visualise la partie droite
<arrière 'CTRL L' déplacement ver le bas
'CTRL O' déplacement ver le haut
'CTRL Q' début de ligne
'CTRL Z' visualisation suit le curseur

Changement de sens

- ou , pour <
+ ou . pour >

<BS> un caractère en arrière avec la flèche gauche ←
<ETX> fin d'opération par 'CTRL C'
<ESC> annulation d'opération par la touche 'ESC'
<DEL> annulation d'une ligne de texte par 'CTRL X'

Mise en page >ADJUST : L(JUST R(JUST C(ENTER <LEFT, RIGHT, UP, DOWN ARROWS> [<ETX> TO LEAVE]

L décalage du texte sur la marge gauche
R justification du texte sur la marge droite
C équilibrage du texte au centre
LEFT flèche ← } déplacement du texte à gauche
RIGHT flèche → } déplacement du texte à droite
UP 'CTRL O' alignement de la ligne précédente
DOWN 'CTRL L' alignement de la ligne suivante
<EXT> 'CTRL C' pour sortir

Copie >COPY : B(UFFER F(ROM FILE <ESC>

B copie le tampon (texte venant d'être effacé par exemple)
F copie d'un fichier de la disquette là ou se trouve le curseur
<ESC> 'ESC' pour sortir

Suppression >DELETE : < > MOVING COMMANDS [< EXT > TO DELETE, <ESC> TO ABORT]

flèche ← pour effacer un caractère
flèche → pour retrouver le texte effacé
EXT 'CTRL C' pour valider la suppression
ESC 'ESC' pour annuler la suppression

Vérification >**V**(ERIFY Vérification de l'écran après modifications

Recherche >FIND [1] : L(IT <TARGET> ⇒

Le mot à rechercher est à taper entre / et /. L(ITTERAL si le texte est inclus dans un mot.[n] à la nième occurence du mot. Ce nombre est à indiquer avant de taper F.

Insertion >INSERT TEXT [<BS> A CHAR, <DEL> A LINE] [<ETX> TO ACCEPTS, <ESC> ESCAPES]

Le texte à insérer est tapé normalement avec correction par la flèche ← ou <BS> pour le caractère précédent ou 'CTRL X' pour annuler une ligne. <ETX> ou 'CTRL C' pour valider l'insertion. <ESC> pour annuler l'opération.

Saut >JUMP : B(EGINNING E(ND M(ARKER <ESC>

B pour placer le curseur en début de texte
E pour placer le curseur en fin de texte
M pour placer le curseur à des marques préfixées (voir S(ET M(ARKER)
<ESC> pour annuler

Sortir de l'éditeur >QUIT : **U**(PDATE THE WORFILE AND LEAVE
E(XIT WITHOUT UPDATING
R(ETURN TO THE EDITOR WITHOUT UPDATING
W(RITE TO A FILE NOME AND RETURN

U en mettant à jour le fichier de travail appelé SYSTEM.WRK.TEXT
E sortie sans mise à jour
R retour à l'éditeur sans sauvegarde
W sauvegarde sur un fichier spécifié et retour à l'éditeur.

Marges >SET : E(NVIRONNEMENT M(ARKER <ESC>
>ENVIRONNEMENT : [OPTIONS] <ETX> OR <SP> TO LEAVE

A(UTO INDENT	TRUE	indentation automatique (alignement sur la ligne précédente)
F(ILLING	FALSE	remplissage jusqu'à la marge droite
L(EFT MARGIN	Ø	marge gauche
R(IGHT MARGIN	78	marge droite
P(ARA MARGIN	5	marge de paragraphe
C(OMMAND CH	^	caractère de commande en mode M(argin)
T(OKEN DEF	TRUE	affecte le mode F(IND et R(EPLACE

En édition d'un programme écrit en langage PASCAL, A doit rester TRUE et F doit être FALSE.
<SP> barre d'espacement pour sortir

Substitution >REPLACE [n] : L(IT V(FY <TARGET> <SUB> ⇨

Remplacement du texte compris entre / et / par un nouveau texte tapé entre / et / de longueur quelconque.
L pour remplacer une partie d'un mot
n nombre d'opérations de substitution (à donner avant d'appeler R)

Echange de caractères >EXCHANGE : TEXT [<BS> A CHAR]
[<ESC> ESCAPES, <ETX> ACCEPTS]

Le caractère remplace celui sous le curseur
<BS> flèche ← pour le caractère précédent
<ESC> 'ESC' pour annuler
<ETX> 'CTRL C' pour valider

Effacement >ZAP effacement depuis la position courante jusqu'à celle de début du dernier texte trouvé, remplacé ou inséré.

FILER **G**(ET **S**(AVE **N**(EW **L**(IST DIRECTORY **E**(XTENDED-DIRECTORY LIST **R**(EMOVE **C**(HANGE **T**(RANSFER **D**(ATE **Q**(UIT **V**(OLUME **W**(HAT **B**(AD-BLOCKS **X**(AMINE **Z**(ERO **P**(REFIX

G GET ? **Nom d'une disquette : Nom du programme**
Chargement en mémoire du programme désigné, il remplace le SYSTEM.WRK.TEXT (fichier de travail)
TEXT FILE LOADED, l'opération est réalisée.

S SAVE AS ? **Nom d'une disquette : Nom du programme**
Sauvegarde du fichier de travail sous le nom spécifié dans la disquette désignée.
TEXT FILE SAVED, l'opération est réalisée.

N Effacement du fichier de travail en MEV et sur disquette
Réponse : WORKFILE CLEARED.

L DIR LISTING OF ? **Nom de la disquette :**
Affiche le contenu de la disquette. (catalogue)
Faire suivre le nom par ,PRINTER : pour imprimer sur papier.

E Affichage du contenu avec des informations diverses comme les zones inutilisées.
DIR LISTING OF ? **Nom de la disquette :**

R Suppression d'un fichier.

C Changement de nom d'un fichier ou d'une disquette.

T Transfert d'une disquette ou d'un fichier sur une autre disquette.
TRANSFER ? **Nom de la disquette : [Nom du programme]**
TO WHERE ? **Nom de la disquette : [Nom du programme]**
Pour imprimer un programme source répondre PRINTER: à la question TO WHERE ?

D Mise à jour de la date.
Jour - Mois (3 lettres) - Année (2 chiffres)

Q Sortir du Filer.

V Liste des volumes connus du système par leur numéro et leur nom.

W WHAT donne le nom du fichier de travail et indique s'il a été sauvegardé ou non.

B Bad-blocks teste les 280 blocs d'une disquette et signale les blocs en mauvais état physique.

X Examine les mauvais blocs et essaie de les rendre cohérents (les répare). Si ce n'est pas possible il permet le marquage des mauvais blocs.
(Opération utile avant l'utilisation d'une disquette vierge).

Z Zéro efface le DIRECTORY (liste des fichiers).

P Préfix permet le changement de nom du volume courant pris par défaut (si on tape seulement :) par le volume spécifié.

Le signe indicatif est le caractère *

Les données sont fournies en numération hexadécimale.

Les adresses sont données sous forme de 4 chiffres hexadécimaux ou moins.

Commande	Définition - *exemples*
Adresse G	Exécution du programme commençant à cette adresse. *3DØ G : redémarrage du Basic à chaud*
Adresse L	Listing des 20 instructions en langage machine à partir de cette adresse ; désassemblage des codes hexadécimaux en code mnémonique du mini-assembleur. *3DØ L* *3DØ - 4C BF 9D JMP Ø9DBF* *3D3 - 4C 84 9D JMP Ø9D84* *...etc*
Adresse 1. adresse 2	Affichage des contenus des positions de mémoire à partir de l'adresse 1 jusqu'à l'adresse 2. *3DØ.3D7* *3DØ - 4C BF 9D 4C 84 9D 4C FD* Les adresses de début de ligne sont toujours de la forme XXXØ ou XXX8 sauf éventuellement l'adresse 1.
Adresse	Si une seule adresse est spécifiée, le Monitor renvoie le contenu de cette position de mémoire. *3D1* *3D1 - BF*
Adresse : valeur 'esp' valeur	Modification ou écriture de valeurs dans des positions de mémoire adjacentes. *73: ØØ 2Ø (modification de HIMEM)*

Commande	Définition - *exemples*
Adresse 1 < adresse 2. adresse 3 M	Déplacement d'une zone de valeurs contenues dans adresse 2 à adresse 3, dans la zone commençant en adresse 1. *6000 < 400.7FF M* sauvegarde de la page écran texte ou graphique entre $6000 et $63FF
Adresse 1 < adresse 2. adresse 3 V	Vérification de l'identité de 2 zones de mémoire. Dès qu'une différence apparaît elle est signalée : *F10B < B1.C8 V* *00B8 - 05 (60)* *00B9 - 02 (EA)* Le sous-programme CHRGET sous sa forme d'origine diffère de la forme de travail par le contenu de $B8 et $B9 (pointeur dans le texte BASIC)
N	Affichage en mode normal et séparateur de commandes successives au MONITOR.
I	Affiche en mode inverse (noir sur blanc).
Valeur + valeur	Opérations d'addition et de soustraction dans le système de numération hexadécimale.(à 2 chiffres) *3F+01 ou 40-01* *40 3F*
Numéro de connecteur 'Ctrl K'	Transfert sur une entrée issue du périphérique branché sur le connecteur spécifié. Transfert du contrôle des sorties au périphérique dont la carte d'interface est installée sur le connecteur spécifié. *6 'ctrl P'* stimule la PROM de la carte contrôleur du lecteur de disquette et provoque l'amorçage du chargement du SED en mémoire vive depuis une disquette.
'Ctrl B'	Amorçage de l'interpréteur BASIC en MEM (démarrage à froid).

Commande	Définition - *exemples*
'Ctrl C'	Re-amorçage de l'interpréteur BASIC en MEM (idem à 'Reset').
'Ctrl E'	Affichage du contenu des registres du microprocesseur. *A= X= Y= P= S=* Modification de ces registres
'Ctrl Y'	Branchement au programme qui commence à l'adresse. *$3F8* *3F8 : 4C 00 03 JMP $300* $3F8 doit être préparée pour provoquer un branchement vers le début du programme appelé.
Adresse 1. adresse 2 W	Enregistrement sur cassette de la zone de mémoire située entre l'adresse 1 et l'adresse 2 **4000.5FFF W*
Adresse 1. adresse 2 R	Lecture sur cassette. Chargement en MEV à partir de l'adresse 1 jusqu'à l'adresse 2 des données ou du programme déjà enregistré sur cassette.

Commande	Définition - *exemples*
* F666 G ou CALL-2458	Entrée dans le mini-assembleur La réponse est !
adrs : COP 'esp' OPE	Instruction à assembler à l'adresse adrs (4 chiffres hexa au plus) ; le code de l'instruction est exprimé en langage mnémonique (3 lettres) : COP parmi les 56 codes d'opération du 6502 ; la partie opérande OPE a un format variable suivant le mode d'adressage. *!300: LDA#ØØ 'Return'* donne *300-A9 ØØ LDA#ØØ* instruction permettant de mettre à zéro le registre accumulateur.
'esp' COP OPE	Assembler l'instruction à l'adresse suivante.
Format de la partie opérande OPE	Si l'adresse est sur 2 chiffres il s'agit de la page zéro.
#$ valeur	Adressage immédiat
$ adresse	Adressage absolu (page zéro ou non suivant COP)
$ adresse, X	Adressage indexé par X (page zéro ou non suivant COP)
$ adresse, Y	Adressage indexé par Y (page zéro ou non suivant COP)
$ adresse	Adressage relatif (pour les instructions de branchement à adresse)
($ adresse, X)	Adressage indirect pré-indexé par X
($ adresse), Y	Adressage indirect post- indexé par Y
($ adresse)	Adressage indirect (pour l'instruction de saut JMP) *!300:LDA $ØØ,X* *300-B5 ØØ* *!'esp' DEX* *303-CA* *!'esp' BNE $300* *304-DØ FB*
$	Appel au Monitor *!$300.305* pour vérifier le contenu d'une zone assemblée.

Sauvegarde - chargement de programmes

LOAD NOM,D1 Charge en MEV le programme NOM depuis la disquette placée dans le lecteur 1.

SAVE NOM,D2 Sauve le programme en BASIC de la MEV sur la disquette placée dans le lecteur 2.

BLOAD BINAIRE Charge en MEV le fichier binaire BINAIRE depuis la disquette, à l'adresse absolue indiquée dans l'en-tête du fichier. Cette adresse et la longueur sont en MEV à $AA72 et $AA60 après le chargement.

BSAVE BINAIRE, A$300,L$7F sauve le programme en langage machine implanté à partir de l'adresse absolue $300 sur une longueur de $7F octets, sur la disquette actuelle sous le nom BINAIRE.

BSAVE IMAGE,A$2000,L$1FF8 sauve l'image graphique haute résolution page 1, sur la disquette sous le nom IMAGE.

HGR:BLOAD IMAGE Restitue sur l'écran graphique haute résolution, les points enregistrés dans IMAGE.

RUN NOM Charge en MEV et exécute le programme intitulé NOM sur la disquette.

BRUN BINAIRE Charge en MEV et exécute le programme en langage binaire BINAIRE enregistré sur la disquette.

CHAIN ENTIER Charge en MEV le programme écrit en Basic INTEGER sans effacer la zone des variables du programme précédent. Le programme ENTIER ne doit pas redimensionner les variables communes.

D$=CHR$(13)+CHR$(4)

Fichiers séquentiels (type T)

PRINT D$"OPEN TS" Ouvre le fichier dénommé TS sur la disquette actuellement en ligne. Place le pointeur au début du fichier séquentiel.

PRINT D$"OPEN"F$",D1" Ouvre un fichier de nom variable F$ sur la disquette placée sur le lecteur n° 1.

PRINT D$"READ TS" Prépare une opération de lecture au début du fichier séquentiel TS, préalablement ouvert par OPEN.

**INPUT A$** Lit dans le fichier TS une chaîne de caractères qui sera mémorisée en MEV sous le nom A$. Le pointeur est déplacé au début du champ de données suivant.

GET C$ Lit un seul caractère et déplace le pointeur d'un caractère.

PRINT D$"POSITION TS,R"P positionne le pointeur après le Pième 'Return' depuis la position actuelle.

PRINT D$"WRITE TS" Prépare une opération d'écriture sur le fichier TS là où se trouve le pointeur.

**PRINT X$** Ecrit dans le fichier TS la chaîne X$.

**PRINT Y$** Ecrit la chaîne Y$ séparée de la chaîne X$, précédemment enregistrée, par le caractère 'Return'.

PRINT CHR$(4)"CLOSE TS" ferme le fichier TS en sauvant sur disquette le tampon de sortie alloué à ce fichier par OPEN, contenant le dernier secteur utilisé.

PRINT D$"APPEND TS" Réouvre le fichier TS en positionnant le pointeur à la fin du fichier. Permet d'écrire des données en les rajoutant à la fin du fichier.

PRINT D$"READ TS,B17" Positionne le pointeur à l'octet n° 17 pour une future lecture. (Ø est le 1er octet).

D$=CHR$(13)+CHR$(4)

Fichiers à accès direct (type T)

PRINT D$"OPEN TACT,L21" ouvre le fichier TACT sur une disquette en prévoyant des enregistrements de longueur constante ; ici 21 octets y compris le 'Return' pris comme fin d'enregistrement.

PRINT D$"OPEN"F$",L"N",D2" ouvre un fichier de nom variable F$, de longueur égale à la variable N, sur la disquette placée dans le lecteur n° 2.

PRINT D$"READ TACT,R"I positionne le pointeur sur le début du Iième enregistrement pour une future lecture.

**INPUT A$** Récupère dans A$, le contenu du Iième enregistrement du fichier TACT.

GET C$ Lit le caractère situé sous le pointeur dans l'enregistrement n° I.

PRINT D$"WRITE TACT,R"J Positionne le pointeur sur le début du Jième enregistrement pour une future écriture.

**PRINT X$** Ecrit la chaîne X$ dans l'enregistrement n° J.

**PRINT Y$** Ecrit la chaîne Y$, séparée de la chaîne précédemment enregistrée, par 'Return'.

PRINT CHR$(4)"CLOSE TACT" ferme le fichier TACT en sauvegardant le contenu du tampon de sortie alloué à ce fichier.

PRINT D$"READ TACT,RØ,B"K positionne le pointeur à l'octet n° K de l'enregistrement n° Ø, pour une lecture ultérieure.

PRINT CHR$(4) Annule l'effet d'une commande du SED, comme READ, par exemple pour donner à INPUT le sens d'une entrée par le clavier.

Commandes diverses

CATALOG D2 Affiche la liste des programmes et des fichiers enregistrés sur la disquette placée en D2.

* fichier verrouillé
I, A, T, B types de fichiers :
- I : Basic Integer
- A : Basic Applesoft
- T : Fichiers T séquentiels ou directs
- B : Binaires, données ou sous-programmes en langage machine

ØØ2 : nombre de secteurs occupés par le fichier (modulo 256)

Un secteur comprend 256 octets utiles.
Une piste comprend 13 ou 16 secteurs suivant le SED (3.2 ou 3.3)
Une disquette comprend 35 pistes dont 31 sont utilisables et 4 sont réservées au SED.

Le nombre maximum de références dans le répertoire d'une disquette est de 84 fichiers en DOS 3.2 et de 105 en DOS 3.3.

MON C,I,O
Affiche les commandes, les entrées et les sorties telles qu'elles sont reçues ou envoyées par le SED pendant l'exécution d'un programme.
Ce mode est annulé par 'Reset'.

NOMON C,I,O
Annule le mode précédent.

PRINT D$"PR#"S
Met le périphérique branché sur le connecteur n° S en ligne pour l'ordre PRINT à suivre.
Si l'imprimante à sa carte d'interface dans le connecteur n° 1 :
PRINT D$"PR#1"

PRINT D$"PR#Ø"
Désactive le périphérique de sortie et seul l'écran reste en ligne.

PRINT D$"IN#"S
Connecte le périphérique branché sur le connecteur S, pour qu'il puisse envoyer des données au système.

PRINT D$"IN#Ø"
L'entrée des données est limitée au clavier.

MAXFILES 4
Prévoit d'utiliser 4 tampons d'entrée/sortie en parallèle correspondant à 4 fichiers ouverts disponibles en MEV.
Chaque tampon occupe 595 octets.
Par défaut le système réserve 3 zones tampons.
Cette commande doit être exécutée avant de charger et d'exécuter un programme.

VERIFY NOM
Teste le bon enregistrement physique du programme ou du fichier NOM. Si un secteur de la disquette utilisée n'est pas bon à l'écriture, le message I/O ERROR sera affiché (est réalisée automatiquement après SAVE)

Manipulation de fichiers

INIT HELLO [,V254]
Initialisation d'une disquette vierge.
Le programme Basic en MEV est chargé sur la disquette sous le nom HELLO, ainsi que le SED.
La disquette porte un numéro de volume qui peut servir de contrôle.
Le formatage de la disquette dépend du SED en MEV.

DOS 3.2 : 13 secteurs par piste
DOS 3.3 : 16 secteurs par piste
La place disponible est
DOS 3.2 : 103168 octets par disquette
DOS 3.3 : 126976 octets par disquette
Le système d'exploitation enregistré sur la disquette est de type esclave il ne contient plus de programme de relocation pour s'adapter au mieux à la configuration-mémoire.
Cette disquette maintenant initialisée est suffisante pour démarrer le système.

DELETE NOM
Supprime le fichier NOM de la disquette sauf s'il est LOCKé (verrouillé).

RENAME ANCIEN, NOUVEAU
Change le nom d'un fichier ou d'un programme.

LOCK NOM
Verrouille le fichier NOM contre toute tentative de suppression par DELETE ou de réenregistrement par SAVE ou WRITE.

UNLOCK NOM
Déverrouille le fichier NOM.

Fichier de commandes

```
LIST
  1Ø PRINT D$"OPEN CMD"          Création du fichier
  2Ø PRINT D$"WRITE CMD"
  3Ø PRINT "PR#1"                Enregistrement des ordres
  4Ø PRINT "CHR$(9)"6Ø N"
  5Ø PRINT "LIST"
  6Ø PRINT "PR#Ø"
  7Ø PRINT D$"CLOSE CMD"         Fermeture du fichier
RUN                              Réalisation du fichier
```

EXEC CMD
Exécution automatique des commandes enregistrées dans CMD c'est-à-dire dans l'exemple :
impression d'un listing sur imprimante à raison de 60 caractères par ligne.

Toutes les instructions et commandes exécutables en mode direct sont susceptibles d'appartenir à un fichier de ce type.

Changement d'interprêteur Basic

INT	Mise en opération de l'interprêteur Integer Basic (en MEM carte-mère, ou en MEV carte-langage ou en MEV à banc-commuté de l'Apple //e) Le curseur succède à >
FP	Mise en opération de l'interprêteur Basic Apple-soft (en MEM carte-mère ou en MEV à banc commuté) Le curseur apparaît à côté de] ou §

CONVERSION HEXADECIMALE/DECIMALE/HEXADECIMALE
des 256 premiers nombres $00 à $FF

$00	:0	1	2	3	4	5	6	7
$08	:8	9	10	11	12	13	14	15
$10	:16	17	18	19	20	21	22	23
$18	:24	25	26	27	28	29	30	31
$20	:32	33	34	35	36	37	38	39
$28	:40	41	42	43	44	45	46	47
$30	:48	49	50	51	52	53	54	55
$38	:56	57	58	59	60	61	62	63
$40	:64	65	66	67	68	69	70	71
$48	:72	73	74	75	76	77	78	79
$50	:80	81	82	83	84	85	86	87
$58	:88	89	90	91	92	93	94	95
$60	:96	97	98	99	100	101	102	103
$68	:104	105	106	107	108	109	110	111
$70	:112	113	114	115	116	117	118	119
$78	:120	121	122	123	124	125	126	127
$80	:128	129	130	131	132	133	134	135
$88	:136	137	138	139	140	141	142	143
$90	:144	145	146	147	148	149	150	151
$98	:152	153	154	155	156	157	158	159
$A0	:160	161	162	163	164	165	166	167
$A8	:168	169	170	171	172	173	174	175
$B0	:176	177	178	179	180	181	182	183
$B8	:184	185	186	187	188	189	190	191
$C0	:192	193	194	195	196	197	198	199
$C8	:200	201	202	203	204	205	206	207
$D0	:208	209	210	211	212	213	214	215
$D8	:216	217	218	219	220	221	222	223
$E0	:224	225	226	227	228	229	230	231
$E8	:232	233	234	235	236	237	238	239
$F0	:240	241	242	243	244	245	246	247
$F8	:248	249	250	251	252	253	254	255

LES CARACTERES ALPHANUMERIQUES ET LEUR CODES ASCII ET ECRAN

Dec	Hex	Caractère Clavier	Nom	Ecran	Dec	Hex	Caractère Clavier	Ecran
ØØ	ØØ	c à	NUL	à i	32	2Ø	Espace	Espace i
Ø1	Ø1	c A	SOH	A i	33	21	!	! i
Ø2	Ø2	c B	STX	B i	34	22	"	" i
Ø3	Ø3	c C	ETX	C i	35	23	£	£ i
Ø4	Ø4	c D	EOT	D i	36	24	$	$ i
Ø5	Ø5	c E	ENQ	E i	37	25	%	% i
Ø6	Ø6	c F	ACK	F i	38	26	&	& i
Ø7	Ø7	c G	BEL	G i	39	27	'	' i
Ø8	Ø8	c H	BS	H i	4Ø	28	(	(i
Ø9	Ø9	c I	HT	I i	41	29	)	) i
1Ø	ØA	c J	LF	J i	42	2A	*	* i
11	ØB	c K	VT	K i	43	2B	+	+ i
12	ØC	c L	FF	L i	44	2C	, virgule	, i
13	ØD	c M	CR	M i	45	2D	- tiret	- i
14	ØE	c N	SO	N i	46	2E	. point	. i
15	ØF	c O	SI	O i	47	2F	/ division	/ i
16	1Ø	c P	DLE	P i	48	3Ø	Ø	Ø i
17	11	c Q	DC1	Q i	49	31	1	1 i
18	12	c R	DC2	R i	5Ø	32	2	2 i
19	13	c R	DC3	S i	51	33	3	3 i
2Ø	14	c T	DC4	T i	52	34	4	4 i
21	15	c U	NAK	U i	53	35	5	5 i
22	16	c V	SYN	V i	54	36	6	6 i
23	17	c W	ETB	W i	55	37	7	7 i
24	18	c X	CAN	X i	56	38	8	8 i
25	19	c Y	EM	Y i	57	39	9	9 i
26	1A	c Z	SUB	Z i	58	3A	: deux pts	: i
27	1B	Esc	ESC	" i	59	3B	; pt.virg.	; i
28	1C	c ç	FS	ç i	6Ø	3C	<	< i
29	1D	c §	GS	§ i	61	3D	= égal	= i
3Ø	1E	c ^	RS	^ i	62	3E	>	> i
31	1F	c _	US	_ i	63	3F	?	? i

c X pour Ctrl X

i pour affichage INVERSE

LES CARACTERES ALPHANUMERIQUES ET LEURS CODES ASCII ET ECRAN

Dec	Hex	Caractère-écran- Clavier	Prim	Alt	Dec	Hex	Caractère-écran- Clavier (//e)	Prim	Alt
64	4Ø	à	à cl	à i	96	6Ø	`	esp. cl	i
65	41	A	A cl	A i	97	61	a	! cl	a i
66	42	B	B cl	B i	98	62	b	" cl	b i
67	43	C	C cl	C i	99	63	c	£ cl	c i
68	44	D	D cl	D i	1ØØ	64	d	$ cl	d i
69	45	E	E cl	E i	1Ø1	65	e	% cl	e i
7Ø	46	F	F cl	F i	1Ø2	66	f	& cl	f i
71	47	G	G cl	G i	1Ø3	67	g	' cl	g i
72	48	H	H cl	H i	1Ø4	68	h	(cl	h i
73	49	I	I cl	I i	1Ø5	69	i	) cl	i i
74	4A	J	J cl	J i	1Ø6	6A	j	* cl	j i
75	4B	K	K cl	K i	1Ø7	6B	k	+ cl	k i
76	4C	L	L cl	L i	1Ø8	6C	l	, cl	l i
77	4D	M	M cl	M i	1Ø9	6D	m	- cl	m i
78	4E	N	N cl	N i	11Ø	6E	n	. cl	n i
79	4F	O	O cl	O i	111	6F	o	/ cl	o i
8Ø	5Ø	P	P cl	P i	112	7Ø	p	Ø cl	p i
81	51	Q	Q cl	Q i	113	71	q	1 cl	q i
82	52	R	R cl	R i	114	72	r	2 cl	r i
83	53	S	S cl	S i	115	73	s	3 cl	s i
84	54	T	T cl	T i	116	74	t	4 cl	t i
85	55	U	U cl	U i	117	75	u	5 cl	u i
86	56	V	V cl	V i	118	76	v	6 cl	v i
87	57	W	W cl	W i	119	77	w	7 cl	w i
88	58	X	X cl	X i	12Ø	78	x	8 cl	x i
89	59	Y	Y cl	Y i	121	79	y	9 cl	y i
9Ø	5A	Z	Z cl	Z i	122	7A	z	: cl	z i
91	5B	"	" cl	" i	123	7B	é	; cl	é i
92	5C	ç	ç cl	ç i	124	7C	ù	< cl	ù i
93	5D	§	§ cl	§ i	125	7D	è	= cl	è i
94	5E	^	^ cl	^ i	126	7E	¨	> cl	¨ i
95	5F	_	_ cl	_ i	127	7F	Del	? cl	⊡ i

cl pour affichage clignotant

i pour affichage inversé

/ Prim pour caractères primaires

/ Alt pour caractères alternatifs

Le changement du jeu Primaire au jeu Alternatif est obtenu par POKE -16369,Ø.

Le changement du jeu Alternatif au jeu Primaire par POKE -1637Ø,Ø.

Dec	Hex	Caractère Clavier	Caractère Ecran	Dec	Hex	Caractère Clavier	Caractère Ecran
128	8Ø	c à NUL	à N	16Ø	AØ	Espace	Espace N
129	81	c A SOH	A N	161	A1	!	! N
13Ø	82	c B STX	B N	162	A2	"	" N
131	83	c C ETX	C N	163	A3	£	£ N
132	84	c D EOT	D N	164	A4	$	$ N
133	85	c E ENQ	E N	165	A5	%	% N
134	86	c F ACK	F N	166	A6	&	& N
135	87	c G BEL	G N	167	A7	'	' N
136	88	c H BS	H N	168	A8	(	(N
137	89	c I HT	I N	169	A9	)	) N
138	8A	c J LF	J N	17Ø	AA	*	* N
139	8B	c K VT	K N	171	AB	+	+ N
14Ø	8C	c L FF	L N	172	AC	,	, N
141	8D	c M CR	M N	173	AD	-	- N
142	8E	c N SO	N N	174	AE	.	. N
143	8F	c O SI	O N	175	AF	/	/ N
144	9Ø	c P DLE	P N	176	BØ	Ø	Ø N
145	91	c Q DC1	Q N	177	B1	1	1 N
146	92	c R DC2	R N	178	B2	2	2 N
147	93	c S DC3	S N	179	B3	3	3 N
148	94	c T DC4	T N	18Ø	B4	4	4 N
149	95	c U NAK	U N	181	B5	5	5 N
15Ø	96	c V SYN	V N	182	B6	6	6 N
151	97	c W ETB	W N	183	B7	7	7 N
152	98	c X CAN	X N	184	B8	8	8 N
153	99	c Y EM	Y N	185	B9	9	9 N
154	9A	c Z SUB	Z N	186	BA	:	: N
155	9B	Esc ESC	" N	187	BB	;	; N
156	9C	c ç FS	ç N	188	BC	<	< N
157	9D	c § GS	§ N	189	BD	=	= N
158	9E	c ^ RS	^ N	19Ø	BE	>	> N
159	9F	c _ US	_ N	191	BF	?	? ?

c X pour Ctrl X.

N pour affichage NORMAL.

Dec	Hex	Caractère Clavier	Caractère Ecran	Dec	Hex	Caractère-écran- Clavier	//e	II
192	CØ	à	à N	224	EØ	`	` N	Esp n
193	C1	A	A N	225	E1	a	a N	! n
194	C2	B	B N	226	E2	b	b N	" n
195	C3	C	C N	227	E3	c	c N	£ n
196	C4	D	D N	228	E4	d	d N	$ n
197	C5	E	E N	229	E5	e	e N	% n
198	C6	F	F N	23Ø	E6	f	f N	& n
199	C7	G	G N	231	E7	g	g N	' n
2ØØ	C8	H	H N	232	E8	h	h N	(n
2Ø1	C9	I	I N	233	E9	i	i N	) n
2Ø2	CA	J	J N	234	EA	j	j N	* n
2Ø3	CB	K	K N	235	EB	k	k N	+ n
2Ø4	CC	L	L N	236	EC	l	l N	, n
2Ø5	CD	M	M N	237	ED	m	m N	- n
2Ø6	CE	N	N N	238	EE	n	n N	. n
2Ø7	CF	O	O N	239	EF	o	o N	/ n
2Ø8	DØ	P	P N	24Ø	FØ	p	p N	Ø n
2Ø9	D1	Q	Q N	241	F1	q	q N	1 n
21Ø	D2	R	R N	242	F2	r	r N	2 n
211	D3	S	S N	243	F3	s	s N	3 n
212	D4	T	T N	244	F4	t	t N	4 n
213	D5	U	U N	245	F5	u	u N	5 n
214	D6	V	V N	246	F6	v	v N	6 n
215	D7	W	W N	247	F7	w	w N	7 n
216	D8	X	X N	248	F8	x	x N	8 n
217	D9	Y	Y N	249	F9	y	y N	9 n
218	DA	Z	Z N	25Ø	FA	z	z N	: n
219	DB	"	" N	251	FB	é	é N	; n
22Ø	DC	ç	ç N	252	FC	ù	ù N	< n
221	DD	§	§ N	253	FD	è	è N	= n
222	DE	^	^ N	254	FE	¨	¨ N	> n
223	DF	_	_ N	255	FF	DEL	⊡ N	? n

N pour NORMAL en Apple //e (minuscules).

n pour NORMAL en Apple II et PLUS (pas de minuscules sur l'écran).

Table d'équivalence des caractères français et américains

Code décimal	35	64	91	92	93	96	123	124	125	126
Code hexadécimal	23	4Ø	5B	5C	5D	6Ø	7B	7C	7D	7E
Caractères français	£	à	"	ç	§	`	é	ù	è	¨
Caractères américains	#	@	[	\	]	`	{	|	}	~

de nombres de 4 chiffres hexadécimaux $H3 H2 H1 HØ

COEFF	H3	H2	H1	H0
0	0	0	0	0
1	4096	256	16	1
2	8192	512	32	2
3	12288	768	48	3
4	16384	1024	64	4
5	20480	1280	80	5
6	24576	1536	96	6
7	28672	1792	112	7
8	32768	2048	128	8
9	36864	2304	144	9
A	40960	2560	160	10
B	45056	2816	176	11
C	49152	3072	192	12
D	53248	3328	208	13
E	57344	3584	224	14
F	61440	3840	240	15

```
CONVERSION HEXADECIMALE DECIMALE
LE NOMBRE DECIMAL EST OBTENU
EN FAISANT LA SOMME DES VALEURS
PRISES A L'INTERSECTION
DE LA LIGNE DU CHIFFRE HEXA
ET DE LA COLONNE DE LA POSITION
DE CE CHIFFRE DANS LE NOMBRE HEXA

    EXEMPLE
  $AFF6 DEVIENT 40960 (A EN H3)
                + 3840 (F EN H2)
                +  240 (F EN H1)
                +    6 (6 EN H0)
                -------
           SOIT: 45046
```


Les messages d'erreur dans un programme en APPLESOFT sont de la forme :

? message ERROR IN numéro de ligne

'message'est le nom de l'erreur.
Le numéro de ligne est celui de l'instruction dans laquelle une erreur a été rencontrée. (Les erreurs ne sont détectées dans un programme qu'au moment de l'exécution des instructions).

Dès l'erreur détectée, l'interpréteur Basic Applesoft provoque l'arrêt du programme et l'affichage du message. Les variables et les instructions ne sont pas modifiées mais le programme ne peut pas se poursuivre. Les compteurs de boucles FOR-NEXT sont mis à zéro et la trace des GOSUB est annulée.

Grâce à l'instruction ONERR GOTO et à un sous-programme de traitement des erreurs (attendues!) un programme peut malgré tout se poursuivre normalement.

Les instructions données en mode immédiat (sans numéro de ligne) peuvent aussi déclencher un message d'erreur, il ne comportera pas d'indication de numéro de ligne.

Chaque type d'erreur est associé à un code qui figure à l'adresse décimale 222 (ou $DE) au moment de l'erreur.

Le numéro de ligne où l'erreur s'est produite figure aux adresses 218 et 219 (ou $DA, $DB). La valeur du pointeur TXPTR dans l'instruction erronée figure aux adresses 220 et 221 (ou $DC, $DD). La valeur du pointeur du stack au moment de l'erreur est conservée à l'adresse 223 (ou $DF). Cette information doit être restaurée avant tout traitement d'erreur grâce au sous-programme suivant :

```
300   68      PLA
301   A8      TAY
302   68      PLA
303   A6 DF   LDX $DF
305   9A      TXS
306   48      PHA
307   98      TYA
308   48      PHA
309   60      RTS
```

Le sous-programme est inscriptible en mémoire par les instructions suivantes dans le programme :

```
Ø POKE 216,0 : 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
```

A l'adresse décimale 216 figure une indication d'activation ($8Ø) ou de désactivation ($ØØ) de l'instruction ONERR GOTO.

Le schéma de programmation de la prise en compte des erreurs avant l'arrêt du programme est le suivant :

```
   Ø  Mise en mémoire du sous-programme ci-dessus
   1  ONERR GOTO 1ØØØ : activation du système
  1Ø  Déroulement normal du programme
 999  END
1000  CALL 768 : exécution du sous-programme
1010  IF PEEK(222)=5 THEN Prise en compte de l'erreur n° 5
1020  Détection d'autres codes d'erreurs
1030  RESUME pour revenir à l'instruction erronée si nécessaire
```

Le sous-programme qui contrôle le déroulement en cas d'erreur est à l'adresse $D412. Il provoque l'exécution du sous-programme HANDLERR à l'adresse $F2E9 si ONERR a été activé. HANDLERR met en place les mémoires $DA à $DF.

Si ONERR n'a pas été utilisée, alors le sous-programme $D412 arrête l'exécution et affiche le message d'erreur.

Dans l'interpréteur APPLESOFT, la table des messages d'erreur est implantée à partir de l'adresse $D260 et leurs codes correspondent à leur position dans cette table.

```
NEXT WITHOUT FORSYNTAXRETURN WITHOUT GO
SUBOUT OF DATAILLEGAL QUANTITYOVERFLOWOU
T OF MEMORYUNDEF'D STATEMENTBAD SUBSCRIP
TREDIM'D ARRAYDIVISION BY ZEROILLEGAL DI
RECTTYPE MISMATCHSTRING TOO LONGFORMULA
TOO COMPLEXCAN'T CONTINUEUNDEF'D FUNCTIO
N ERROF IN
BREAK
```


Liste des messages d'erreurs par ordre alphabétique

Code	*Message*	*Origine*	*Commentaire*
1Ø7	?BAD SUBSCRIPT (mauvais indice)	DIM	Tentative d'appeler un élément de tableau d'indice supérieur à la limite fournie dans DIM. Exemple : DIM A(15) avec A(20) ou encore avec un nombre d'indices différent de celui spécifié dans DIM. L'applesoft dimensionne automatiquement à 11 les variables indicées non déclarées.
21Ø	CAN'T CONTINUE (on ne peut pas continuer)	CONT	Impossibilité de reprendre l'exécution d'un programme par CONT. En cas d'erreur ou d'insertion ou de modification d'une instruction. Dans certains on peut repartir avec GOTO numéro de ligne.
133	?DIVISION BY ZERO (division par zéro)	/Ø	Peut venir d'une variable non initialisée à une valeur différente de zéro.
	?EXTRA IGNORED (donnée de trop ignorée)	INPUT	Si les données (séparées par des virgules) sont en nombre supérieur à celui des variables prévues pour les recevoir. Le programme se poursuit pourtant.
191	?FORMULA TOO COMPLEX (formule trop complexe)	IF "chaîne de caractère" THEN	Le test ne peut être demandé plus de 2 fois dans un programme.
149	?ILLEGAL DIRECT (illégal en mode direct)	Mode direct	Les instructions INPUT GET DEF FN et DATA ne sont pas autorisées en mode direct.
53	?ILLEGAL QUANTITY (valeur erronée)	Fonctions maths	Le paramètre donné à une fonction dépasse les limites permises - l'indice d'une variable est négatif - l'argument de LOG est négatif ou nul - l'argument de SQR est négatif. A PUISSANCE B si A est négatif et B n'est pas entier.

Code	Message	Origine	Commentaire
53	?ILLEGAL QUANTITY	MID$ LEFT$ RIGHT$	La longueur ou l'indice de positionnement ne sont pas compris entre 1 et 255.
		CHR$	Le code n'est pas compris entre Ø et 255 (bornes incluses).
		ASC	Le caractère est de longueur Ø (vide).
		CALL	L'adresse n'est pas comprise entre -65535 et +65535.
		POKE	L'adresse n'est pas comprise entre -65535 et 65535 ; la valeur n'est pas comprise entre Ø et 255.
		HIMEM:	L'adresse n'est pas comprise entre -65535 et 65535.
		HPLOT	X,Y<Ø ou X>278 et Y>191
		DRAW	X,Y<Ø ou X>278 et Y>191
		PLOT,VLIN, HLIN	X,Y<Ø ou X,Y>39
		PDL	X<Ø ou X>255
		HTAB	X<Ø ou X>255
		VTAB	X<Ø ou X>24
		SPC	X<0 ou X>255
		TAB(	X<Ø ou X>255
		ON..GOTO ON..GOSUB	L'index ne doit pas dépasser 255 ou être inférieur à Ø. Si la valeur de l'index est nulle ou plus élevée que le nombre de numéros de ligne spécifiés, l'exécution continue à l'instruction suivante.
Ø	NEXT WITHOUT FOR (NEXT sans FOR correspondant)	FOR,NEXT	.Des boucles FOR-NEXT ont été mal imbriquées exemple : FOR X = 1 TO... FOR Y = 1 TO... PRINT X,Y NEXT : NEXT Y (Réécrire NEXT Y : NEXT X) .Il manque un FOR pour un NEXT isolé.

Code	Message	Origine	Commentaire
42	?OUT OF DATA (Data épuisées)	READ	Essai d'exécution d'un READ alors que toutes les données du DATA ont déjà été lues. Prévoir de tester un caractère de fin de données ou une variable de comptage ou faire RESTORE pour relire les données au début de DATA.
		RECALL STORE	Ne pas utiliser des variables dont le nom commence par RECALL ou STORE.
77	?OUT OF MEMORY	DIM	Ne pas dépasser le nombre maximum d'indices : 88.
	(Mémoire épuisée)	GOSUB	Ne peut gérer plus de 24 niveaux d'appels imbriqués.
		HIMEM: LOMEM:	Ne pas la fixer trop basse Ne pas la fixer trop haute ou inférieure à la valeur actuelle. Le programme est trop grand ou les variables sont trop nombreuses.
66	?OVERFLOW ERROR (dépassement de capacité)	Nombre réel	Résultat supérieur à 1.7E38 (Un nombre réel est mémorisé avec 1 octet d'exposant et 4 octets de mantisse. Si le résultat est inférieur à 2.9E-39, il équivaut à 0 sans message d'erreur.
		STR$	Si le nombre à convertir en chaîne de caractères est trop grand.
		VAL	Si la valeur absolue du nombre cherché est supérieure à 1E38 ou si le nombre contient plus de 38 chiffres (dont les zéros les plus à droite).

Code	Message	Origine	Commentaire
120	?REDIM'D ARRAY (tableau redimensionné)	DIM	Un même tableau ne peut être dimensionné deux fois. (On a repassé deux fois sur l'instruction DIM).
	?REENTER (refaire l'entrée de données)	INPUT	On a fourni une quantité alphanumérique ; il faut reprendre en redonnant toutes les valeurs *numériques* attendues par l'instruction INPUT.
22	?RETURN WITHOUT GOSUB (retour sans GOSUB)	RETURN	Un sous-programme a été placé après la fin logique du programme où END a été oublié.
			Dans le traitement d'une erreur, reprendre dans un sous-programme sans exécuter GOSUB.
176	?STRING TOO LONG (chaîne de caractères trop longue)	X$	Ne pas créer une chaîne par concaténation dont la longueur dépasse 255, si l'argument est une chaîne de longueur totale supérieure à 255.
		LEN VAL	
		PRINT	A$+B$ a plus de 255 caractères (écrire PRINT AB)
16	?SYNTAX (erreur de syntaxe)		Instruction incompréhensible pour l'interpréteur APPLESOFT. - parenthèses non appariées - caractères illégaux - mauvaise ponctuation - faute d'orthographe dans un mot clé.
		ASC	Sur "CTRL @" ou CHR$(0)
		CONT INPUT	Si une entrée de donnée en INPUT est interrompue par 'CTRL C' et que CONT est essayé pour repartir.
16	?SYNTAX (erreur de syntaxe)	DATA	Une chaîne de caractères contenant ? n'est pas acceptée.

Code	Message	Origine	Commentaire
16	?SYNTAX	DEL	Doit être suivi de 2 numéros de ligne dans l'ordre croissant
		FOR..NEXT	Ne pas utiliser une variable de type entier (%) comme indice de boucle.
		HGR HGRZ TEXT	Ne pas utiliser ces mots-clés comme premières lettres d'un nom de variable (Ils seront exécutés avant affichage de l'erreur).
		IF..THEN	Il manque THEN avec le IF correspondant.
		LIST, Q	Affiche le programme complet puis le message SYNTAX.
		RECALL STORE SHLOAD	Ne pas utiliser ces mots-clés comme premières lettres d'un nom de variable
		RESUME	Elle est rencontrée avant qu'une erreur se soit produite. Peut être une erreur fatale.
		N° d'instruction incorrect	Si la lettre O est tapée au lieu du chiffre 0 ou la lettre I au lieu du chiffre 1.
163	?TYPE MISMATCH (désaccord entre numérique et alphanumérique	LET MID$ LEFT$ RIGHT$	- Une variable chaîne ne peut recevoir une valeur numérique et vice-versa - erreur de type d'arguments.
224	?UNDEF'D FUNCTION (fonction non définie)		Référence à une fonction pour laquelle il n'existe pas d'instruction DEF FN préalable
90	?UNDEF'D STATEMENT	GOTO GOSUB ON. GOTO RUN THEN	Renvoient à un numéro de ligne inexistant.

Dans le système d'exploitation des disquettes chargé en MEV (dans une configuration de 48K), la table des messages d'erreurs est implantée à partir de l'adresse $A971. Le premier message est 'Return','Bell','Return'. Les suivants sont :

```
LANGUAGE NOT AVAILABLERANGE ERRORWRI
TE PROTECTEDEND OF DATAFILE NOT FOUNDDUOL
UME MISMATCHI/O ERRORDISK FULLFILE LOCKE
DSYNTAX ERRORNO BUFFERS AVAILABLEFILE TY
PE MISMATCHPROGRAM TOO LARGENOT DIRECT C
OMMAND
```

Dans la zone suivante $AA3F à $AA4F, sont enregistrées les positions du début de chaque message dans la table des messages précédents :

Dec	Ø	3	25	25	36	51	62	76	91	1ØØ	1Ø9	12Ø	132	152	17Ø	187
Code	0	1	2	3	4	5	6	7	8	9	1Ø	11	12	13	14	15
Hexa				$24	$33											

Exemple : le message 'WRITE PROTECTED' est le 4ième de la liste, son code d'erreur est 4 et le texte est mémorisé de l'adresse $A971+$24 à l'adresse $A971+$32.

Cette analyse permet de traduire en français les messages d'erreur envoyés par le système d'exploitation des disquettes. Ne vous en privez pas !

Code	*Message*	*Origine*	*Cause*
9	DISK FULL (disquette pleine)	SAVE WRITE	Le répertoire arrive à sa limite ou tous les secteurs sont utilisés.
5	END OF DATA (fin des données)	INPUT	Les données sont insuffisantes pour satisfaire l'instruction INPUT.
		APPEND READ	Après cette instruction seule l'instruction WRITE est autorisée.
		POSITION READ	La position atteinte ne correspond à aucune donnée enregistrée.
		EXEC F,Rr	Si r correspond au 2ième champ *après* la fin du fichier !!
		READ F,Rr	Si r correspond à un enregistrement non encore effectué (codé Ø).
10	FILE LOCKED (fichier ou programme verrouillé)	SAVE DELETE BSAVE WRITE	Un fichier comportant une astérisque sur le CATALOG ne peut être modifié en ré-écriture (sauf avec APPEND).
6	FILE NOT FOUND (fichier inconnu)	LOAD RUN BLOAD BRUN DELETE	Le fichier demandé n'existe pas sur la disquette. Vérifiez l'orthographe de son nom.
13	FILE TYPE MISMATCH (désaccord sur le type du fichier)	LOAD BLOAD RUN BRUN	Un fichier de type T ou B ne peut être appelé par LOAD ou RUN et un fichier de type I ou A ne peut être appelé par BLOAD ou BRUN.
13	FILE TYPE MISMATCH (désaccord sur le type de fichier)	OPEN READ POSITION WRITE APPEND EXEC CLOSE	Ces commandes ne sont opérationnelles que sur un fichier de *données* (de type T)

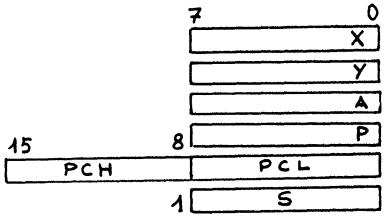
Code	*Message*	*Origine*	*Cause*
		CHAIN	Un programme en Basic Applesoft ne peut être 'chaîné' par cette seule commande qui ne concerne que les programmes en Basic Integer.
8	I/O ERROR (erreur d'entrée/ sortie)	Toute commande	- lecteur sans disquette - n° de connecteur (sans contrôleur) - disquette abimée - disquette non initialisée - porte du lecteur ouverte
		VERIFY	S'il y a une erreur après vérification qu'un fichier est correctement ou mal enregistré.
1	LANGUAGE NOT AVAILABLE (interpréteur Basic absent du système)	LOAD FP INT	Un programme en Basic ne peut être exécuté sans que l'interpréteur soit présent.
		APPLE][PLUS	N'a pas d'interpréteur BASIC INTEGER en standard.
		CARTE LANGAGE	Si la carte-mère contient en MEM un interpréteur, la carte langage pourra être chargée avec l'autre.
12	NO BUFFERS AVAILABLE (trop de fichiers ouverts en MEV)	MAXFILES n	Le nombre maximum est 16 (le système en utilise 1 pour chaque commande). Le nombre par défaut est 3.
15	NOT DIRECT COMMAND (commande directe illégale)	OPEN READ WRITE APPEND POSITION	Ne pas les utiliser en mode direct. - *Ecrire un programme* contenant ces commandes dans un PRINT.
14	PROGRAM TOO LARGE (programme trop grand)	LOAD RUN	Le HIMEM est trop bas (le SED compare le nombre de secteurs du programme avec l'octet de poids fort de HIMEM)

Code	*Message*	*Origine*	*Cause*	
2,3	RANGE ERROR (valeur erronée)			Mn-Max
		V	Volume	Ø-254
		D	Lecteur	Ø-2
		S	Connecteur	1-7
		L	Longueur	1-32767
		R	Numéro	Ø-32767
		B	Numéro	Ø-32767
		A	Adresse	Ø-65535
		MAXFILES	Nombre de fichiers ouverts	1-16
11	SYNTAX ERROR (erreur de syntaxe dans une commande du SED	INT	Commande sans paramètres instruction BASIC non valide	
		EXEC		
		Aa	a ni négatif	
		Ll	l ni > 65535	
		IN#S	s ne peut être supérieur à 7.	
		PR#s		
7	VOLUME MISMATCH (désaccord de numéros de volume)	V	Le volume de la disquette courante est différent de celui de la disquette demandée. Si la demande est faite avec VØ, la détection de volume ne sera pas faite.	
4	WRITE PROTECTED (protection en écriture)	L'encoche recouverte, les fichiers ne seront accessibles qu'en lecture. La disquette est peut-être placée à l'envers. La disquette SYSTEM MASTER est toujours protégée.		

MESSAGES D'ERREUR FICHIERS SED

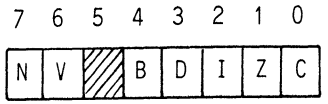
Code	*Message*	*Origine*	*Cause*
		CHAIN	Un programme en Basic Applesoft ne peut être 'chaîné' par cette seule commande qui ne concerne que les programmes en Basic Integer.
8	I/O ERROR (erreur d'entrée/ sortie)	Toute commande	- lecteur sans disquette - n° de connecteur (sans contrôleur) - disquette abimée - disquette non initialisée - porte du lecteur ouverte
		VERIFY	S'il y a une erreur après vérification qu'un fichier est correctement ou mal enregistré.
1	LANGUAGE NOT AVAILABLE (interpréteur Basic absent du système)	LOAD FP INT	Un programme en Basic ne peut être exécuté sans que l'interpréteur soit présent.
		APPLE][PLUS	N'a pas d'interpréteur BASIC INTEGER en standard.
		CARTE LANGAGE	Si la carte-mère contient en MEM un interpréteur, la carte langage pourra être chargée avec l'autre.
12	NO BUFFERS AVAILABLE (trop de fichiers ouverts en MEV)	MAXFILES n	Le nombre maximum est 16 (le système en utilise 1 pour chaque commande). Le nombre par défaut est 3.
15	NOT DIRECT COMMAND (commande directe illégale)	OPEN READ WRITE APPEND POSITION	Ne pas les utiliser en mode direct. - *Ecrire un programme* contenant ces commandes dans un PRINT.
14	PROGRAM TOO LARGE (programme trop grand)	LOAD RUN	Le HIMEM est trop bas (le SED compare le nombre de secteurs du programme avec l'octet de poids fort de HIMEM)

REGISTRES INTERNES DU 6502



registres
index
accumulateur
registre d'état
compteur ordinal
pointeur de pile ou stack

Détail du registre d'état P

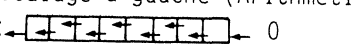


N	signe	D	mode décimal
V	débordement	I	inhibition interruptions
5	inutilisé	Z	résultat nul
B	indicateur de BReaK	C	retenue

ADC : Addition avec retenue (ADd with Carry) : A ← A + M + C
On ajoute à l'accumulateur la mémoire spécifiée plus le bit de retenue.
Opère en mode binaire ou décimal. Agit sur N, V, Z, C.

AND : Et logique (AND) : A ← A ∧ M
Fait le et bit à bit entre accumulateur et mémoire.
Agit sur N,Z.

ASL : Décalage à gauche (Arithmetic Shift Left)



Décale à gauche l'accumulateur ou une mémoire. Agit sur N,Z,C.

BCC : Branchement si pas de retenue (Branch on Carry Clear)
Si le bit C = 0, on saute à l'instruction indiquée ; sinon, on continue en séquence.

BCS : Branchement si retenue (Branch on Carry Set)
Si le bit C = 1, on saute à l'instruction indiquée ; sinon, on continue en séquence.

BEQ : Branchement si résultat = 0 (Branch on EQual)
Si le bit Z = 1 (c'est-à-dire si le dernier résultat est 0 ou si la dernière comparaison a donné l'égalité), on saute à l'instruction indiquée ; sinon, on continue en séquence.

BIT : Test de bits (BIt Test) $Z \leftarrow \sum_i \overline{A_i \wedge M_i}$, N ← M7, V ← M6.

Effectue le et virtuel de l'accumulateur et de la mémoire spécifiée et positionne Z en conséquence. En outre, les bits 7 et 6 de la mémoire sont copiés respectivement dans N et V.

BMI : Branchement si négatif (Branch on MInus).
Si le bit N = 1, on saute à l'instruction indiquée ; sinon, on continue en séquence.

BNE : Branchement si non égal à 0 (Branch on Not Equal)
Si le bit Z = 0 (c'est-à-dire si le dernier résultat est ≠ 0 ou si la dernière comparaison n'a pas donné l'égalité), on saute à l'instruction indiquée ; sinon, on continue en séquence.

BPL : Branchement si positif ou nul (Branch if PLus)
Si le bit N = 0, on saute à l'instruction indiquée ; sinon, on continue en séquence.

BRK : Interruption software (BReaK)
Met le bit B à 1 et simule une interruption (saut à l'adresse contenue en FFFE, FFFF).

BVC : Branchement si pas de débordement (Branch on oVerflow Clear).
Si le bit V = 0, on saute à l'instruction indiquée ; sinon, on continue en séquence.

BVS : Branchement si débordement (Branch on oVerflow Set)
Si le bit V = 1, on saute à l'instruction indiquée ; sinon, on continue en séquence.

CLC : Annuler la retenue (CLear Carry)
Force à 0 le bit de retenue.

CLD : Annuler le mode décimal (CLear Decimal mode)
Force à 0 le bit D pour mettre l'UAL en mode binaire.

CLI : Autorise les interruptions (CLear Interrupt inhibit flag)
Force à 0 le bit I d'inhibition des interruptions.

CLV : Annuler l'indicateur de débordement (CLear oVerflow flag)
Force à 0 le bit V.

CMP : Comparer avec l'accumulateur (CoMPare accumulator) A-M
Effectue la soustraction virtuelle registre A - mémoire et positionne les indicateurs N, Z et C en conséquence.

CPX : Comparer avec X (ComPare with X) X-M
Effectue la soustraction virtuelle registre X - mémoire et positionne les indicateurs N, Z et C en conséquence.

CPY : Comparer avec Y (ComPare with Y) Y-M
Effectue la soustraction virtuelle registre Y - mémoire et positionne les indicateurs N, Z et C en conséquence.

DEC : Décrémenter en mémoire (DECrement memory) M ← M - 1.
Diminue de 1 le contenu de la mémoire indiquée. Agit sur N et Z.

DEX : Décrémenter X (DEcrement X) X ← X - 1
Diminue de 1 le contenu du registre X. Agit sur N et Z.

DEY : Décrémenter Y (DEcrement Y) Y ← Y - 1
Diminue de 1 le contenu du registre Y. Agit sur N et Z.

EOR : Ou exclusif (Exclusive OR)
Effectue le ou exclusif entre l'accumulateur et la mémoire indiquée. Agit sur N et Z.

INC : INCrémenter en mémoire (Increment memory)
Augmente de 1 le contenu de la mémoire indiquée. Agit sur N et Z.

INX : INCrémenter X (Increment X)
Augmente de 1 le contenu du registre X. Agit sur N et Z.

INY : INCrémenter Y (increment Y)
Augmente de 1 le contenu du registre Y. Agit sur N et Z.

JMP : Saut inconditionnel (JuMP) PC ← adresse
Saute à l'adresse indiquée.

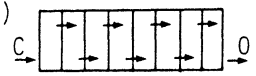
JSR : Appel d'un sous-programme (Jump to Sub-Routine) PC ↓ ; PC ← Ad
Sauve PC dans la pile (adresse de retour) puis saute à l'adresse indiquée.

LDA : Charger l'accumulateur (LoaD Accumulator) A ← M
Met dans l'accumulateur le contenu de la mémoire spécifiée. Agit sur N et Z.

LDX : Charger X (LoaD X register) X ← M
Met dans le registre X le contenu de la mémoire spécifiée. Agit sur N et Z.

LDY : Charger Y (LoaD Y register) Y ← M
Met dans le registre Y le contenu de la mémoire spécifiée. Agit sur N et Z.

LSR : Décalage à droite (Logical Shift Right)
Décale à droite l'accumulateur ou une mémoire. Agit sur N, Z et C.



NOP : Pas d'opération (No OPeration) PC ← PC + 1
Instruction muette. S'exécute en 2 cycles.

ORA : Ou inclusif (OR Accumulator) A ← AvM
Effectue le ou inclusif entre l'accumulateur et la mémoire indiquée. Agit sur N et Z.

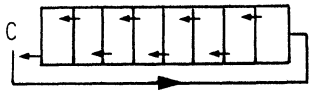
PHA : Empiler A (PusH Accumulator) A ↓ : (S) ← A ; S ← S - 1
Met l'accumulateur au sommet de la pile et met à jour le pointeur de pile.

PHP : Empiler P (PusH Processor status register)
P ↓ : (S) ← P ; S ← S - 1
Met le registre d'état P au sommet de la pile et met à jour le pointeur de pile.

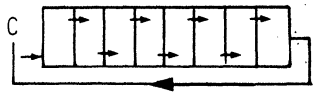
PLA : Dépiler vers A(PuLl Accumulator) A ↑ : S ← S + 1 ; A←(S)
Transfère vers A le contenu du sommet de la pile et met à jour le pointeur de pile. Affecte N et Z.

PLP : Dépiler vers P (PuLl P register) P ↑ : S ← S + 1 ; P←(S)
Transfère vers le registre d'état P le contenu du sommet de la pile et met à jour le pointeur de pile. Affecte tous les indicateurs.

ROL : Rotation à gauche (ROtate Left)
Décale à gauche l'accumulateur ou une mémoire. L'ancienne valeur du bit de retenue rentre par la droite tandis que le bit qui sort par la gauche devient la nouvelle valeur de C. Affecte N, Z et C.



ROR : Rotation à droite (ROtate Right)
Décale à droite l'accumulateur ou une mémoire. L'ancienne valeur du bit de retenue rentre par la gauche tandis que le bit qui sort par la droite vient remplacer C. Affecte N, Z et C.



RTI : Retour d'interruption (ReTurn from Interrupt)
P ↑ ; PC ↑
Retour de routine d'interruption : récupère sur la pile PC et P qui y avaient été sauvés par le mécanisme d'interruption.

RTS : Retour du sous-programme (ReTurn from Subroutine) PC ↑
Récupère sur la pile PC qui y avait été sauvé par le dernier JSR.

SBC : Soustraction avec retenue (SuBtract with Carry)
A ← A - M - C
On soustrait à l'accumulateur la mémoire spécifiée et aussi l'opposé du bit de retenue (emprunt). Opère en mode binaire ou décimal. Agit sur N, V, Z, C.

SEC : Mettre à 1 la retenue (SEt Carry flag)
Force à 1 le bit C.

SED : Mettre en mode décimal (SEt Decimal mode)
Force à 1 le bit D (influe sur ADC et SBC).

SEI : Inhiber les interruptions (SEt Interrupt inhibit flag)
Force à 1 le bit I.

STA : Ranger l'accumulateur (STore Accumulator) M ← A
Transfère le contenu de l'accumulateur dans la mémoire indiquée.

STX : Ranger X (STore X register) M ← X
Transfère le contenu du registre X dans la mémoire indiquée.

STY : Ranger Y (STore Y register) M ← Y
Transfère le contenu du registre Y dans la mémoire indiquée.

TAX : Transfert de A dans X. X ← A. Agit sur N et Z.

TAY : Transfert de A dans Y. Y ← A. Agit sur N et Z.

TSX : Transfert de S dans X. X ← S. Agit sur N et Z.

TXA : Transfert de X dans A. A ← X. Agit sur N et Z.

TXS : Transfert de X dans S. S ← X. N'agit pas sur les indicateurs.

TYA : Transfert de Y dans A. A ← Y. Agit sur N et Z.

Tableau de désassemblage

Ce tableau est l'inverse du suivant. En fonction du code hexadécimal AB, il donne le mnémonique et le mode d'adressage correspondant. Exemple : A9 → LDA IMM (ligne A, col 9). (Pas de mode d'adressage = inhérent ou relatif).

A \ B	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	BRK	ORA IND,X				ORA PZ,X	ASL PZ,X		PHP	ORA IMM	ASL A			ORA ABS	ASL ABS	
1	BPL	ORA IND,Y				ORA PZ,X	ASL PZ,X		CLC	ORA ABS,Y				ORA ABS,X	ASL ABS,X	
2	JSR	AND IND,X			BIT PGE Z	AND PGE Z	ROL PGE Z		PLP	AND IMM	ROL A		BIT ABS	AND ABS	ROL ABS	
3	BMI	AND IND,Y				AND PZ,X	ROL PZ,X		SEC	AND ABS,Y				AND ABS,X	ROL ABS,X	
4	RTI	EOR IND,X				EOR PGE Z	LSR PGE,Z		PHA	EOR IMM	LSR A		JMP ABS	EOR ABS	LSR ABS	
5	BVC	EOR IND,Y				EOR PZ,X	LSR PZ,X		CLI	EOR ABS,Y				EOR ABS,X	LSR ABS,X	
6	RTS	ADC IND,X				ADC PGE Z	ROR PGE Z		PLA	ADC IMM	ROR A		JMP IND	ADC ABS	ROR ABS	
7	BVS	ADC IND,Y				ADC PZ,X	ROR PZ,X		SEI	ADC ABS,Y				ADC ABS,X	ROR ABS,X	
8		STA IND,X			STY PGE Z	STA PGE Z	STX PGE Z		DEY		TXA		STY ABS	STA ABS	STX ABS	
9	BCC	STA IND,Y			STY PZ,X	STA PZ,X	STX PZ,Y		TYA	STA ABS,Y	TXS			STA ABS,X		
A	LDY IMM	LDA IND,X	LDX IMM		LDY PGE Z	LDA PGE Z	LDX PGE Z		TAY	LDA IMM	TAX		LDY ABS	LDA ABS	LDX ABS	
B	BCS	LDA IND,Y			LDY PZ,X	LDA PZ,X	LDX PZ,Y		CLV	LDA ABS,Y	TSX		LDY ABS,X	LDA ABS,X	LDX ABS,Y	
C	CPY IMM	CMP IND,X			CPY PGE Z	CMP PGE Z	DEC PGE Z		INY	CMP IMM	DEX		CPY ABS	CMP ABS	DEC ABS	
D	BNE	CMP IND,Y				CMP PZ,X	DEC PZ,X		CLD	CMP ABS,Y				CMP ABS,X	DEC ABS,X	
E	CPX IMM	SBC IND,X			CPX PGE Z	SBC PGE Z	INC PGE Z		INX	SBC IMM	NOP		CPX ABS	SBC ABS	INC ABS	
F	BEQ	SBC IND,Y				SBC PZ,X	INC PZ,X		SED	SBC ABS,Y				SBC ABS,X	INC ABS,X	

Codes opération / nombre de cycles (n) / nombre d'octets occupés (#)

MNEMO	EFFET	IMMEDIAT OP	n	#	ABSOLU OP	n	#	PAGE ZERO OP	n	#	ACCUM. OP	n	#	INHERENT OP	n	#	(IND,X) OP	n	#	(IND),Y OP	n	#	PAGE Z,X OP	n	#	ABS,X OP	n	#	ABS,Y OP	n	#	RELATIF OP	n	#	INDIRECT OP	n	#	PAGE Z,Y OP	n	#	REGISTRE D'ETAT 7 6 5 4 3 2 1 0 N V . B D I Z C	MNEMO
ADC	A+M+C→A (1) (4)	69	2	2	6D	4	3	65	3	2							61	6	2	71	5	2	75	4	2	7D	4	3	79	4	3									N V . . . . Z C	ADC
AND	A∧M→A (1)	29	2	2	2D	4	3	25	3	2							21	6	2	31	5	2	35	4	2	3D	4	3	39	4	3									N . . . . . Z .	AND
ASL	C ← [7 0] ← 0				0E	6	3	06	5	2	0A	2	1										16	6	2	1E	7	3												N . . . . . Z C	ASL
BCC	BRANCH.SI C=0 (2)																															90	2	2							. . . . .	BCC
BCS	BRANCH.SI C=1 (2)																															B0	2	2							. . . . .	BCS
BEQ	BRANCH.SI Z=1 (2)				2C	4	3	24	3	2																					F0	2	2							. . . . .	BEQ
BIT	A∧M																																								M_7 M_6 . . . . Z .	BIT
BMI	BRANCH.SI N=1 (2)																															30	2	2							. . . . .	BMI
BNE	BRANCH.SI Z=0 (2)																															D0	2	2							. . . . .	BNE
BPL	BRANCH.SI N=0 (2)																															10	2	2							. . . . .	BPL
BRK	BREAK													00	7	1																								. . 1 . 1 . .	BRK
BVC	BRANCH.SI V=0 (2)																															50	2	2							. . . . .	BVC
BVS	BRANCH.SI V=1 (2)																															70	2	2							. . . . .	BVS
CLC	0 → C													18	2	1																								 0	CLC
CLD	0 → D													D8	2	1																								. . . . 0 . .	CLD
CLI	0 → I													58	2	1																								. . . . 0 . .	CLI
CLV	0 → V													B8	2	1																								. 0	CLV
CMP	A - M	C9	2	2	CD	4	3	C5	3	2							C1	6	2	D1	5	2	D5	4	2	DD	4	3	D9	4	3									N . . . . . Z C	CMP
CPX	X - M	E0	2	2	EC	4	3	E4	3	2																														N . . . . . Z C	CPX
CPY	Y - M	C0	2	2	CC	4	3	C4	3	2																														N . . . . . Z C	CPY
DEC	M - 1 → M				CE	6	3	C6	5	2													D6	6	2	DE	7	3												N . . . . . Z .	DEC
DEX	X - 1 → X													CA	2	1																								N . . . . . Z .	DEX
DEY	Y - 1 → Y													88	2	1																								N . . . . . Z .	DEY
EOR	A ● M → A (1)	49	2	2	4D	4	3	45	3	2							41	6	2	51	5	2	55	4	2	5D	4	3	59	4	3									N . . . . . Z .	EOR
INC	M+1 → M				EE	6	3	E6	5	2													F6	6	2	FE	7	3												N . . . . . Z .	INC
INX	X+1 → X													E8	2	1																								N . . . . . Z .	INX
INY	Y+1 → Y													C8	2	1																								N . . . . . Z .	INY
JMP	SAUT INCOND.				4C	3	3																												6C	5	3					. . . . .	JMP
JSR	APPEL S/P				20	6	3																																			. . . . .	JSR
LDA	M → A (1)	A9	2	2	AD	4	3	A5	3	2							A1	6	2	B1	5	2	B5	4	2	BD	4	3	B9	4	3									N . . . . . Z .	LDA

Codes opération / nombre de cycles (n) / nombre d'octets occupés (#)

MNEMO.	EFFET	IMMEDIAT OP	n	#	ABSOLU OP	n	#	PAGE ZERO OP	n	#	ACCUM. OP	n	#	INHERENT OP	n	#	(IND,X) OP	n	#	(IND),Y OP	n	#	PAGE Z,X OP	n	#	ABS,X OP	n	#	ABS,Y OP	n	#	RELATIF OP	n	#	INDIRECT OP	n	#	PAGE Z,Y OP	n	#	REGISTRE D'ETAT 7 6 5 4 3 2 1 0 N V . B D I Z C	MNEMO.
LDX	M → X (1)	A2	2	2	AE	4	3	A6	3	2																			BE	4	3							B6	4	2	N . . . . . Z .	LDX
LDY	M → Y (1)	A0	2	2	AC	4	3	A4	3	2													B4	4	2	BC	4	3												N . . . . . Z .	LDY
LSR	0 → [7 0] → C				4E	6	3	46	5	2	4A	2	1										56	6	2	5E	7	3												0 . . . . . Z C	LSR
NOP	PAS D'OPERATION													EA	2	1																								. . . . .	NOP
ORA	A∨M→A	09	2	2	0D	4	3	05	3	2							01	6	2	11	5	2	15	4	2	1D	4	3	19	4	3									N . . . . . Z .	ORA
PHA	A → Ms S-1 → S													48	3	1																								. . . . .	PHA
PHP	P → Ms S-1 → S													08	3	1																								. . . . .	PHP
PLA	S+1 → S Ms → A													68	4	1																								N . . . . . Z .	PLA
PLP	S+1 → S Ms → P													28	4	1																								(RESTAURES)	PLP
ROL	[7 0] ← [C]				2E	6	3	26	5	2	2A	2	1										36	6	2	3E	7	3												N . . . . . Z C	ROL
ROR	[C] → [7 0]				6E	6	3	66	5	2	6A	2	1										76	6	2	7E	7	3												N . . . . . Z C	ROR
RTI	RETOUR INTER													40	6	1																								(RESTAURES)	RTI
RTS	RETOUR S/P													60	6	1																								. . . . .	RTS
SBC	A - M - C → A (1) (4)	E9	2	2	ED	4	3	E5	3	2							E1	6	2	F1	5	2	F5	4	2	FD	4	3	F9	4	3									N V . . . . Z (3)	SBC
SEC	1 → C													38	2	1																								 1	SEC
SED	1 → D													F8	2	1																								. . . . 1 . .	SED
SEI	1 → D													78	2	1																								. . . . . 1 . .	SEI
STA	A → M				8D	4	3	85	3	2							81	6	2	91	6	2	95	4	2	9D	5	3	99	5	3							96	4	2	. . . . .	STA
STX	X → M				8E	4	3	86	3	2													94	4	2																. . . . .	STX
STY	Y → M				8C	4	3	84	3	2																																. . . . .	STY
TAX	1 → X													AA	2	1																								N . . . . . Z .	TAX
TAY	A → Y													A8	2	1																								N . . . . . Z .	TAY
TSX	S → X													BA	2	1																								N . . . . . Z .	TSX
TXA	X → A													8A	2	1																								. . . . .	TXA
TXS	X → S													9A	2	1																								. . . . .	TXS
TYA	Y → A													98	2	1																								N . . . . . Z .	TYA

(1) Ajouter 1 à "n" si changement de page

(2) Ajouter 1 à "n" si branchement dans une page différente

(3) Emprunt = $\overline{\text{retenue}}$

(4) En mode décimal, l'indicateur Z est inopérant. L'accumulateur doit être testé pour savoir si le résultat est nul.

X	Index X
Y	Index Y
A	Accumulateur
M	Mémoire
Ms	Mémoire (pile)
M_7	Mémoire (bit 7)
M_6	Mémoire (bit 6)
+	Addition
-	Soustraction
∧	Et
∨	Ou
●	Ou exclusif
n	Nombre de cycles
#	Nombre d'octets

1 - Positionnement du curseur

Maintenir le curseur en face de la question jusqu'à ce que la réponse soit correcte.

La position du curseur est **mémorisée** avant la saisie et restaurée pour une nouvelle saisie.

Si la réponse est incorrecte, elle est **effacée.**

Exemple :

```
5  HOME
6 Q$ = "REPONDEZ PAR O(UI OU N(ON "
10  PRINT Q$;: GOSUB 100
20  INPUT "";R$
30  IF R$ = "O" THEN  VTAB PV: PRINT Q$;"OUI"
40  IF R$ = "N" THEN  VTAB PV: PRINT Q$;"NON"
45  IF  LEFT$ (R$,1) <  > "O" AND  LEFT$ (R$,1) <
      > "N" THEN  VTAB PV: PRINT Q$;: CALL  - 868:
      GOTO 20
50  PRINT "FIN": PRINT : GOTO 10
100  REM
200 PV =  PEEK (37) + 1
210  RETURN
```

PV est la position verticale (ligne)

CALL-868 efface le reste de la ligne depuis l'endroit où se trouve le curseur.

2 - Simuler INPUT X$

La chaîne lue s'inscrit de $200 (512) à $2FF (768)

APPLESOFT	Assembleur
CALL-10964	JSR $D52C

Tous les caractères sont acceptés jusqu'à concurrence de 255 mais 'CTRL X' annule la ligne et 'Return' valide l'entrée.

3 - Empêcher le listing d'un programme

POKE 2049,Ø : POKE 2Ø5Ø,Ø

met à zéro le pointeur du début de la deuxième ligne d'instruction.

Pour retrouver sa valeur exacte, rechercher le premier octet ØØ de fin de la première ligne d'instruction et ajouter 1 à l'adresse de cet octet.

Exemple :

```
]LIST

10  REM  COMMENT NØ 3
20  PRINT : END

]POKE2049,0:POKE2050,0

]LIST


]CALL -151

*800.81F          14Ø8 )

0800- 00 00 00 0A 00 B2 20 43
0808- 4F 4D 4D 45 4E 54 20 4E
0810- 30 20 33 00 1C 08 14 00
0818- BA 3A 80 00 00 00 0A 00
```

Les commandes NEW et FP annulent aussi ce pointeur sans effacer le programme.

4 - Mettre deux programmes bout-à-bout

a) Charger le programme de tête en mémoire vive.

b) Modifier le pointeur de début de programme pour qu'il pointe après l'octet ØØ de la dernière instruction du programme de tête.

c) Charger le programme de queue en mémoire vive.

d) Modifier le pointeur de début de programme pour qu'il pointe au début du programme de tête.

TEXTTAB pointeur de début de programme $67,$68
PRGEND pointeur de fin de programme $AF,$BØ


```
]NEW

]100 REM PG DE QUEUE

]SAVE PQ
]NEW

]10 REM PG DE TETE          a

]CALL -151

*AF.B0

00AF- 15
00B0- 08
*800.815

0800- 00 12 08 0A 00 B2 20 50
0808- 47 20 44 45 20 54 45 54
0810- 45 00 00 00 64 0A
*67:12 08                    b

*3D0G

]LOAD PQ                     c
]CALL -151

*67:01 08                    d

*3D0G

]LIST

10  REM  PG DE TETE
100  REM  PG DE QUEUE
```

5 - Empêcher l'accès au clavier

En dehors du blocage de la touche 'RESET' (cf 6) il faut prévoir de neutraliser aussi la touche 'CTRL C' qui provoque l'interruption du programme en cours avec le message:

BREAK IN n° de l'instruction où l'arrêt a été déclenché.

La solution proposée utilise **le traitement d'erreur** : 'CTRL C' correspond au code d'erreur n° 255 et une fois détectée, l'erreur est annulée par RESUME :

```
1 ON ERR GO TO 1000
1000 IF PEEK(222) = 255 THEN RESUME
```

6 - Toutes les commandes sont interprêtées comme un RUN

POKE 214,128

Une valeur supérieur ou égale à 128 dans l'adresse 214 en $D6 a un effet irréversible sur toutes les commandes ou instructions BASIC, elles sont transformées en RUN. Sont épargnées les commandes d'accès aux programmes sur disquettes.

Faire PR#6 pour réinitialiser le système.

7 - Inhiber la touche 'RESET'

L'effet de 'Reset' sur le système dépend du contenu des mémoires $3F2 et $3F3.

L'adresse contenue dans ces mémoires est celle vers laquelle se branchera le système si 'Reset' est tapé.

Adresses		*valeurs* par défaut		
Dec	*Hex*	*Dec*	*Hex*	
1010	3F2	191	BF	retour à BASIC sous DOS
1011	3F3	157	9D	(arrêt du programme en cours)
1012	3F4	56	38	OR exclusif de (1011) et #A5

L'octet d'adresse $3F4 doit être recalculé pour modifier l'effet de 'Reset'.

Sa valeur est obtenue par CALL-1169 ($FB6F) puis PRINT POKE (1012).

a) Inhibition : (le programme en cours ne s'arrêtera pas avec 'Reset'

```
* 3F2 : ØØ Ø3 A6

* 3ØØ : 2Ø EA Ø3      JSR $Ø3EA (DOS)
* 3Ø3 : 2Ø 98 D8      JSR $D898 (CONT)
* 3Ø5 : 4C D2 D7      JSR $D7D2 (NEWSTT)
```

b) Inhibition de tout le système (après avoir tapé 'Reset')

```
* 3F2 : 00 03 A6
* 3ØØ : 4C ØØ Ø3
```

Le redémarrage de tout le système n'est possible qu'après coupure du courant.

c) La touche 'Reset' fait redémarrer le système comme si on mettait sous tension : (démarrage à froid)

Il suffit d'un POKE 1012,Ø

d) Désinhibition * 3F2 : BF 9D 38

8 - Attendre un caractère au clavier

a) 1Ø X = PEEK(-16384) : IF X < 128 THEN 1Ø
   2Ø POKE - 16368,Ø : X$ = CHR$(X-128)

b) 1Ø WAIT -16384,128 : X = PEEK(-16384)-128 : POKE - 16368,Ø

c) 10 GET X$

d) 1Ø CALL-756 (RDKEY)

9 - Modifier l'affichage d'un listing de programme en Basic

(en mode 4Ø colonnes exclusivement)

- POKE 33,33

La fenêtre d'écran est réduite à 33 colonnes de largeur. La commande LIST affiche les instructions **sans marge.**

- TEXT annule la commande précédente

- Le caractère : permet d'introduire une indentation de ligne d'instruction.

- POKE 33,28
  facilite le cadrage des REM : la disposition à l'enregistrement n'est pas modifiée par LIST.

- TEXT ou POKE 33,4Ø pour revenir au mode standard.

10 - GOTO calculé par l'intermédiaire de l'ampersand &

On écrit & expression.

Le sous-programme est mémorisé à partir de $3ØØ. Donc les adresses $3F5 $3F6 et $3F7 doivent être pré-enregistrées avec l'instruction JMP $3ØØ pour que l'interpréteur se branche sur l'adresse $3ØØ dès qu'il rencontrera &.

```
* 3F5 : 4C ØØ Ø3
```

ou bien :

```
10 POKE 1013,76 : POKE 1014,Ø : POKE 1015,3
```

Sous-programme d'évaluation de l'expression et de branchement à la ligne calculée

```
*300L
0300-   20 7B DD    JSR   $DD7B
0303-   20 52 E7    JSR   $E752
0306-   20 1A D6    JSR   $D61A
0309-   90 03       BCC   $030E
030B-   4C 41 D9    JMP   $D941
030E-   A2 5A       LDX   #$5A
0310-   4C 12 D4    JMP   $D412
```

$DD7B	FRMEVL	évaluation de l'expression le résultat va dans FAC
$E752	GETADR	conversion FAC en valeur entière le résultat va dans $50,$51
$D61A	FNDLIN	recherche si la ligne calculée fait partie du programme
$D941	GOTO+	saut à la ligne trouvée
$D412	ERROR	erreur éventuelle avec code #5A = 90 UNDEF'D STATEMENT

11 - Imprimer avec D décimales

a) DEF FNF(X) = INT(X*10^D)/10^D

au lieu d'imprimer X, on imprimera FNF(X)

Note : L'instruction PRINT d'un nombre réel n'affiche pas les zéros les plus à droite de la partie fractionnaire ni les zéros les plus à gauche de la partie entière.

Si 0.01 ⩽ |X| < 999 999 999.2 le nombre est en notation virgule fixe, sinon il est sous forme mantisse, exposant
sx.xxx xxx xxEstt

s est le signe
. est la séparation entre partie entière et partie fractionnaire
E veut dire 10 à la puissance
x et T sont des chiffres de 0 à 9

b) Arrondir à D décimales

La fonction INT(X) a pour résultat le plus petit entier inférieur à X, ce qui pose quelque problème si X est négatif.

Ainsi ?INT(-5.3)
-6

Il faut donc tenir compte du signe de X en arrondissant

Ainsi ?INT(ABS(-5.3))/SGN(-5.3)
-5

```
DEF FN AR(X) = INT(ABS(X)*10^D+.5)/10^D*SGN(X)
```

Faire PRINT FNAR(X)

c) Tronquer à D décimales avec la notation flottante

```
10 X$ = STR$(X)
20 FOR I = 1 TO LEN(X$):IF MID$(X$,I,1)<>"E"THEN NEXT I
30 FOR J = 1 TO I-1:IF MID$(X$,J,1)<>"."THEN NEXT J
40 IF J+D< = I-1 THEN N=J+D:GO TO 60
50 N = I-1
60 PRINT LEFT$(X$,N)+MID$(X$,I)
```


12 - Justifier à droite dans une zone de C caractères

```
A$ = STR$(FN AR(X))
C$ = REM C caractères 'espace'
PRINT RIGHT$(C$+A$,C)
```

13 - Connaître l'adresse d'une variable

Il faut distinguer les variables numériques simples des variables alphanumériques.

Après les deux octets représentant les deux premières lettres du **nom**, le système réserve 5 octets pour conserver la **valeur** réelle ou entière d'une variable numérique.

Exemple : A = 1

41	00	81	00	00	00	00

NOM — VALEUR

B% = 32767

C2	80	7F	FF	00	00	00

NOM — VALEUR

Le cas des variables alphanumériques dont la valeur est une chaîne de caractères est différent puisque dans les 5 octets suivants le nom de la variable, on trouve **la longueur** de la chaîne et **l'adresse de début** de la chaîne.

Exemple : A$ = ""

41	80	00			00	00

NOM — POINTEUR — LONGUEUR

L'adresse cherchée est celle de la valeur d'une variable **numérique** donc du pointeur de valeur.

L'adresse d'une chaîne de caractères est celle contenue dans les octets +1 et +2 par rapport au pointeur de valeur.

Ce pointeur de valeur (VARPNT) est mémorisé dans $83, $84 (131,132) et contient celui de la dernière variable traitée par l'APPLESOFT.

En APPLESOFT on écrit :

```
10 X=A : REM ON recherche l'adresse de A
20 A=PEEK(131) + 256*PEEK(132)
30 PRINT A
40 A=X : REM on rétablit la valeur de A
```

En langage machine, on peut se servir de la routine PTRGET d'adresse $DFE3 pour récupérer dans l'accumulateur et le registre Y, les poids faibles et forts du pointeur de la variable dont on cherche l'adresse.

Grâce à l'opérateur & suivi du nom de la variable, on entre dans ce sous-programme en langage machine qui renvoie aux adresses 778($30A) et 779($30B) la valeur cherchée.

```
*300L

0300-   20 E3 DF     JSR   $DFE3
0303-   8D 0A 03     STA   $030A
0306-   8C 0B 03     STY   $030B
0309-   60           RTS
```

Dans un tableau de valeurs numériques entières, la valeur n'occupe que deux octets pour chaque variable indicée. Le pointeur de valeur sera utilisé directement : l'adresse de la variable indicée contient l'octet de poids fort, suivi de l'octet de poids faible.

```
          1  POKE 1013,76: POKE 1014,0: POKE 1015,3
          2 X = 0:L = 0:P = 0:A$ = "APPLESOFT"
          5  DIM A%(100):A%(1) = 32767
          10  & A%(1)
          30 X =  PEEK (778) +  PEEK (779) * 256
          40  PRINT X
          45  PRINT 256 *  PEEK (X) +  PEEK (X + 1)
          50 A$ =  LEFT$ (A$,5)
          60  & A$
          70 X =  PEEK (778) +  PEEK (779) * 256
Longueur  80 L =  PEEK (X)
Pointeur  90 P =  PEEK (X + 1) +  PEEK (X + 2) * 256
          100  PRINT L,P;" ";
          110  FOR X = P TO P + L - 1
Chaîne    120  PRINT  CHR$ ( PEEK (X));: NEXT X


          ]RUN
Adresse   2365
Valeur    32767
          5                    38395 APPLE
```

14 - Listing sur imprimante

Si la carte d'interface de l'imprimante est dans le connecteur n° 1 :

PR#1
LIST

Si les lignes d'instructions dépassent 30 caractères, il faut modifier le nombre de caractères par ligne éditée sur l'imprimante pour éviter le format classique (image de l'écran).

PR#1
PRINT "CTRL I 80 N" pour 80 caractères par ligne.

Cet ordre rend l'affichage sur écran impossible.

Pour sortir de cet état, taper 'Reset' ou 'Ctrl/Reset' ou

PR#0

15 - Changer de page d'écran

POKE -16299,0	affiche la page n° 2	*C055
POKE -16300,0	affiche la page n° 1	*C054
POKE -16304,0	affiche en graphique	*C050
POKE -16303,0	affiche en texte	*C051
POKE -16297,0	affiche en HGR sans effacement	*C057
POKE -16298,0	affiche en GR sans effacement	*C056
POKE -16302,0	graphique sur tout l'écran	*C053
POKE -16301,0	graphique et 4 lignes de texte	*C052

16 - Modifier la fenêtre d'écran-texte

TEXT règle la fenêtre aux valeurs maxima

Largeur	: $21(33)	: WNDWDTH = $28(40)
Marge gauche	: $20(32)	: WNDLFT = $0(0)
Marge haute	: $22(34)	: WNDTOP = $$0(0)
Marge basse	: $23(35)	: WNDBTM = $18(24)

POKE 33, largeur	comprise entre 1 et 40
POKE 32, marge gauche	marge gauche + largeur inférieure à 39
POKE 34, marge haute	comprise entre 0 et 23
POKE 35, marge basse	supérieure à la marge haute et inférieure à 24

La marge gauche ne se positionne qu'après un 'Return' (PRINT)

17 - Pour que PRINT imprime des minuscules (Apple II ou Plus)

PR#1 (imprimante en ligne)

```
]POKE243,32:PRINT"10 BONJOUR":NORMAL
10 bonjour
```

L'adresse 243 ou $F3 ou ORMASK sert dans l'APPLESOFT pour forcer le mode FLASH au groupe des chiffres ($60-$7F) par l'instruction OR ORMASK.

avec ORMASK = $40 en mode FLASH
= $0 en mode NORMAL et INVERSE

Les minuscules diffèrent des majuscules par le bit 5 de leur code qui vaut 1 pour les minuscules ($E0-$FF) et 0 pour les majuscules ($C0-$DF).

Par une opération OR ORMASK
avec OR MASK = $20(32), ce bit 5 est mis à 1 d'où POKE 243,32 pour imprimer des minuscules.

18 - Effacer dans l'écran (durant l'écriture d'un programme)

HOME ou ou 'esc' @ } CALL -936 ou FC58 G efface tout l'écran de texte et positionne le curseur en haut et à gauche de la fenêtre.

'esc' E ou CALL-868 ou FC9CG efface depuis la position du curseur jusqu'au bout de la ligne courante (ou jusqu'au bord droit de la fenêtre).

'esc' F ou CALL-958 ou FC42G efface depuis la position du curseur jusqu'au bas de l'écran (ou de la fenêtre).

Sur l'Apple //e 'esc' 4 et 'esc' 8 effacent l'écran et changent le format.

19 - Ecrire sur l'écran de bas en haut

CALL-998 : CALL-998 entre chaque PRINT

20 - Ecrire sur l'écran de droite à gauche

CALL-1008 : CALL-1008 entre chaque PRINT X$

21 - Déplacer tout le texte vers le haut (SCROLL)

CALL-912 ou FC70G

Cette opération est **systématique** dès que le curseur atteint la '25ème ligne' qui apparaitra à la 24ème avec un décalage de tout le texte d'une ligne vers le haut.

22 - Protéger un INPUT avec une valeur par défaut

La valeur par défaut est d'une longueur 1 caractère


```
10  REM  SAISIE PAR DEFAUT
20 DE$ = "0": REM   VALEUR PAR DEFAUT
30  PRINT "QUESTION? ";DE$;
40 PH =  PEEK (36):PV =  PEEK (37) + 1
41  IF PV > 23 THEN PV = 23: REM  ATTENTION AU SCROLL

42  CALL  - 1008: REM  RECUL D'UNE POSITION
50  INPUT "";RE$
60  IF RE$ = "" THEN RE$ = DE$
70  HTAB PH: VTAB PV: PRINT RE$
```

Lorsque l'INPUT s'exécute, on voit le curseur clignoter sur la valeur par défaut. Si en réponse on tape 'Return', la valeur prise sera celle par défaut. Si l'on veut entrer une autre valeur, on tape cette valeur sur celle présentée, puis 'Return'.

23 - Prévoir la taille d'un programme

En gros un programme occupe autant d'octets qu'il renferme de caractères, puisqu'il est stocké tel quel, comme chaîne de caractères sauf les MOTS-CLES qui sont remplacés par un code en 1 octet.

En ce qui concerne les variables, chaque variable numérique simple réelle ou entière occupe 7 octets, chaque chaîne occupe (7 + longueur) octets.

Un tableau occupe x(n+1) + 2d + 3
n est la taille du tableau (y compris l'élément n° 0)
x = 5 (nombres réels)
x = 2 (entiers)
x = 3 (chaînes de caractères)
d nombre de dimensions

On gagne de la place en mémoire en supprimant tous les blancs inutiles, en mettant plusieurs instructions par ligne, en évitant les REM, en utilisant des variables plutôt que des constantes.

Utilisez les GOSUB dès qu'il faut faire appel plusieurs fois à une séquence d'instructions identiques.

24 - Faire jouer de la musique à l'APPLE

Un air est défini par une liste de paires I,J
I est la hauteur de la note ou sa fréquence
J est la durée de cette note (à quel temps, ronde, blanche, noire, croche ? etc.).

Un programme, écrit en langage machine, permet de stimuler le haut parleur par LDA $C030, à intervalles réguliers :
- le registre X est initialisé avec la valeur I, avant un 'bip', et diminue jusqu'à 0, jusqu'au prochain 'bip' ;
  Plus I est faible, plus la fréquence, donc la hauteur, est élevée.
- le registre Y diminue aussi et son passage par zéro fait diminuer J, la durée, laquelle, en atteignant zéro, va provoquer la fin de l'exécution d'une note.

GAMME : (avec le programme ci-dessous). Valeurs de I :
255, 242, 230, 216, 204, 192, 182, 172, 162, 152, 144, 136, 128,
128, 121, 115, 108, 102, 96, 91, 86, 81, 76, 72, 68, 64,
64, 60, 57, 54, 51, 48, 45, 43, 40, 38, 36, 34, 32
SOL, SOL# , LA, LA# , SI, DO, DO# , RE, RE# , MI, FA, FA # , SOL

```
0302-   AD 30 C0     LDA   $C030
0305-   88           DEY
0306-   D0 05        BNE   $030D
0308-   CE 01 03     DEC   $0301
030B-   F0 09        BEQ   $0316
030D-   CA           DEX
030E-   D0 F5        BNE   $0305
0310-   AE 00 03     LDX   $0300
0313-   4C 02 03     JMP   $0302
0316-   60           RTS
```

Pour jouer un air, il faut appeler ce sous-programme pour chacune des notes successivement.

En Basic, les paires I,J sont lues sur un fichier DATA jusqu'aux valeurs 0,0.

Le programme ci-dessus est mémorisé en début de programme, par des instructions POKE A,V.

Exemple : deux petites musiques "synthétisées"

```
10  REM  MUSIQUE
20  POKE 770,173: POKE 771,48: POKE
     772,192: POKE 773,136: POKE
     774,208: POKE 775,5: POKE 77
     6,206: POKE 777,1: POKE 778,
     3: POKE 779,240: POKE 780,9:
      POKE 781,202
30  POKE 782,208: POKE 783,245: POKE
     784,174: POKE 785,0: POKE 78
     6,3: POKE 787,76: POKE 788,2
     : POKE 789,3: POKE 790,96: POKE
     791,0: POKE 792,0
```



```
40  READ I,J: IF J = 0 THEN 70
50  POKE 768,I: POKE 769,J: CALL
     770
60  GOTO 40
70  IF F = 1 THEN  END
80 F = 1: INPUT " ENCORE UNE?";R$

90  GOTO 40
100  DATA  114,120,144,60,114,255
     ,1,120,128,120,144,60,128,12
     0,114,60,144,120,171,255,228
     ,255,0,0
200  DATA  0,160,128,255,152,40,1
     71,80,192,40,228,255,1,40,0,
     160,192,255,192,40,171,80,15
     2,40,128,255,0,0
```

25 - Faire dessiner, tourner, agrandir une forme

Le codage d'une forme peut être simplifié en utilisant un octet pour chaque vecteur élémentaire tracé :
- dirigé vers le haut → 4
- dirigé vers la droite → 5
- dirigé vers le bas → 6
- dirigé vers la gauche → 7

Les vecteurs successifs sont rangés dans une zone choisie par l'utilisateur : on l'appelle une table de forme.

Elle doit contenir dans
- le 1er octet : le nombre de forme (1 par exemple)
- le 3ème et le 4ème : l'offset pour repérer le début de la 1ère forme (04 00 pour une forme)

puis les vecteurs de la forme.

Il faut préciser en début de programme, l'adresse de début de cette table de forme. Cette adresse est rangée en $E8 ou 232 et $E9 ou 233 avec la partie basse de l'adresse en $E8 (poids faibles).

En Basic, la succession des vecteurs du dessin rangée en DATA avec 0 comme fin de liste, est lue et mémorisée à partir du 5ème octet de la table (si une seule forme est prévue).

La forme choisie dans l'exemple suivant est un pétale "stylisée".

Une fleur avec un pétale comme forme définie

```
]LIST

 10  HGR
 20  HCOLOR= 3
 25  REM  LA TABLE DES FORMES EST
      A L'ADRESSE $300 OU 768
 30  POKE 232,0: POKE 233,3
 35  REM   UNE SEULE FORME ... UN
      PETALE
 40  POKE 768,1: POKE 769,0: POKE
      770,4: POKE 771,0
 42  RESTORE :T = 0
 43  READ D: POKE 772 + T,D: IF D =
      0 THEN 48
 45 T = T + 1: GOTO 43
 48 X = 140:Y = 80
 50  SCALE= 3
 52  REM  VOICI UNE FLEUR
 55  FOR R = 0 TO 64 STEP 4
 58  ROT= R
 60  DRAW 1 AT X,Y
 70  NEXT R
 80  END
100  DATA  4,4,4,5,4,4,4,5,4,4,4,
      5,4,4,4,5,4,5,4,5,4,5,4,5,5,
      5,4,5,5,5,4,5,5,5,4,5,5,5
200  DATA  6,6,6,7,6,6,6,7,6,6,6,
      7,6,6,6,7,6,7,6,7,6,7,6,7,7,
      7,6,7,7,7,6,7,7,7,6,7,7,7 ,0

300  REM  LA DIRECTION D'UN VECTE
      UR ELEMENTAIRE VAUT 4,5,6,7
      POUR HAUT,DROITE,BAS,GAUCHE
      RESPECTIVEMENT
```


(Les numéros sont les numéros des "trucs")

Page zéro

Dec	*Hex*	*Nom*	*Rôle*
Ø,1	ØØ,Ø1	LOC1,LOC2	utilisations diverses
2-31	Ø2-1F	non utilisées	disponibles
32,33, 34,35	2Ø,21, 22,23	WNDL,W,T,B	les 4 paramètres de définition de la fenêtre
36,37	24,25	CH,CV	positions horizontales et verticales du curseur
38,39	26,27	GBASL,GBASH	contiennent l'adresse de base de la ligne graphique calculée par GBASCALC d'après Acc.
40,41	28,29	BASL,BASH	contiennent l'adresse de base de ligne texte calculée par BASCALC d'après l'acc.
42,43	2A,2B	BAS2L,BAS2H	adresse de base de ligne modifiée par le dérouiement du texte en SCRL1
44,45	2C,2D	H2,V2	paramètres de tracé de verticale et d'horizontale graphiques
44,45	2C,2D	LMNEM,RMNEM	codes de mnémonique (3 car. en 2 octets) pour le désassembleur
46	2E	MASK	ØF (lignes paires), FØ (impaires) en couleur GR
46	2E	CHKSUM	indice d'erreur de lecture cassette pour READ
46	2E	FORMAT	pour le désassemblage des opérandes
47	2F	LASTIN	indicateur de fin en lecture cassette
47	2F	LENGTH	longueur d'une instruction (1, 2 ou 3 octets)
48	3Ø	COLOR	indicateur de couleur pour 2 lignes adj.

Dec	Hex	Nom	Rôle
49	31	MODE	conserve les caractères de commande : . + - ou zéro
5Ø	32	INVFLG	masque d'inversion des caractères à afficher
51	33	PROMPT	caractère '*' ou autre annonçant la saisie par GETLN de car. au clavier
52	34	YSAV	mise en réserve de l'index Y pour NXTITM
53	35	YSAV1	mise en réserve de Y avant COUTZ
54,55	36,37	CSWL,CSWH	contiennent l'adresse de la routine de sortie à utiliser par le système
56,57	38,39	KSWL,KSWH	contiennent l'adresse de la routine d'entrée à utiliser par le système ex clavier : FD1B (sans SED)
58,59	3A,3B	PCL,PCH	sauvegarde du compteur ordinal avant BREAK par exemple
60,61	3C,3D	A1L,A1H	mémoires de travail pour MOVE, VFY, READ, WRITE
62,63	3E,3F	A2L,A2H	
64,65	4Ø,41	A3L,A3H	
66,67	42,43	A4L,A4H	
68,69	44,45	A5L,A5H	
69	45	ACC	mémoires de sauvegarde des registres utilisés par RESTORE et SAVE
70	46	XREG	
71	47	YREG	
72	48	STATUS	
73	49	SPNT	
78,79	4E,4F	RNDL,RNDH	compteur incrémenté pendant KEYIN. Sert d'amorce pour la fonction RND
74-77	4A-4D	non utilisées	disponibles à d'autres systèmes
8Ø-255	5Ø-FF	non utilisées	disponibles à d'autres systèmes

Page un

256-511	1ØØ-1FF	Stack	pile dont le haut est pointé par le registre S

Page deux

512-758	2ØØ-2F8	IN	tampon d'entrée remplit par GETLN avec ADDIMP

Page trois

Dec	Hex	Nom	Rôle
1016-1018	3F8-3FA	USRADR	JMP adresse si 'CRTL Y'
1019-1021	3FB-3FD	NMI	JMP adresse si interruption non masquée comme 'Reset'
1022,1023	3FE,3FF	IRQLOC	adresse si interruption hardware

AUTOSTART seulement

Dec	Hex	Nom	Rôle
1008,1009	3FØ,3F1	BRKV	adresse de reprise si BRK par défaut 59,FA (OLDBREAK)
1010,1011	3F2,3F3	SOFTEV	adresse de reprise si 'Reset' par défaut 03,EØ (BASCONT)
1012	3F4	PWREDUP MASK	octet de décision pour la reprise à chaud ou à froid si 'Reset'

Page quatre

Dec	Hex	Nom	Rôle
1024	4ØØ	LINE 1	début page écran texte ou GR
1024-1063	4ØØ-427	ligne Ø en texte	lignes Ø et 1 en GR
1064-1103	428-44F	ligne 8 en texte	lignes 16 et 17 en GR
1104-1143	45Ø-477	ligne16 en texte	lignes 32 et 33 en GR
1144+s	478+s		mémoires RAM accessibles au programme d'interfaces des périphériques (1 octet par connecteur)

Page quatre (suite) à sept

Dec	Hex	Nom	Utilisation
1152-1191	480-4A7	ligne 1 TEXT	lignes 2 et 3 GR
1192-1231	4A8-4CF	ligne 9 TEXT	lignes 18 et 19 GR
1232-1271	4DØ-47F	ligne 17 TEXT	lignes 34 et 35 GR
1272+s	4F8+s		mémoire MEV utilisable pour le périphérique connecté en s de Ø à 7
1280-1319	5ØØ-527	ligne 2 TEXT	lignes 4 et 5 GR
1320-1359	528-54F	ligne 10 TEXT	lignes 20 et 21 GR
1360-1399	550-577	ligne 18 TEXT	lignes 36 et 37 GR
14ØØ+s	578+s		mémoire MEV utilisable pour le périphérique connecté en s
1408-1447	580-5A7	ligne 3 TEXT	lignes 6 et 7 GR
1448-1487	5A8-5CF	ligne 11 TEXT	lignes 12 et 13 GR
1488-1527	5DØ-5F7	ligne 19 TEXT	lignes 38 et 39 GR
1528+s	5F8+s		périphérique connecté en s
1536-1575	600-627	ligne 4 TEXT	lignes 8 et 9 GR
1576-1615	628-64F	ligne 12 TEXT	lignes 24 et 25 GR
1616-1655	650-677	ligne 20 TEXT	lignes 40 et 41 GR
1656+s	678+s		périphérique connecté en s
1664-1703	680-6A7	ligne 5 TEXT	lignes 10 et 11 GR
1704-1743	6A8-6CF	ligne 13 TEXT	lignes 26 et 27 GR
1744-1783	6DØ-6F7	ligne 21 TEXT	lignes 42 et 43 GR
1784+s	6F8+s		périphérique connecté en s
1792-1831	7ØØ-727	ligne 6 TEXT	lignes 12 et 13 GR
1832-1871	728-74F	ligne 14 TEXT	lignes 28 et 29 GR
1872-1911	75Ø-777	ligne 22 TEXT	lignes 44 et 45 GR
1912+s	778+s		périphérique connecté en s
1920-1959	78Ø-7A7	ligne 7 TEXT	lignes 14 et 15 GR
1960-1999	7A8-7CF	ligne 15 TEXT	lignes 30 et 31 GR
2000-2039	7DØ-7F7	ligne 23 TEXT	lignes 46 et 47 GR
2040	7F8	SLOT#	contient $Cs si s est le
2040+s	7F8+s		périphérique actuellement utilisé. Mémoire pour périphérique en s

Page douze : CØØØ-CFFF - Apple II et Apple //e

Adresses Dec	Hex	Nom	Rôle	e=Apple//e R=Lire,PEEK W=Ecrire,POKE
-16384	CØØØ	KBD	Adresse contenant le code du caractère tapé au clavier (bit 7 à 1)	R
-16384	CØØØ	CLR8ØSTORE	Com. log. de mise à Ø de 8ØSTORE pour revenir sur la mémoire principale	w e
-16383	CØØ1	SET8ØSTORE	Com. log. de mise à 1 de 8ØSTORE pour utiliser la mémoire auxiliaire	W e
-16382	CØØ2	CLRRAMRD	Com. log. de mise à Ø de RAMRD pour lire en mémoire principale	W e
-16381	CØØ3	SETRAMRD	Com. log. de mise à 1 de RAMRD pour lire en mémoire auxiliaire	W e
-1638Ø	CØØ4	CLRRAMWRT	Com. log. de mise à Ø de RAMWRT pour écrire en mémoire principale	W e
-16379	CØØ5	SETRAMWRT	Com. log. de mise à 1 de RAMWRT pour écrire en mémoire auxiliaire	W e
-16378	CØØ6	SETSLOTCXROM	Com. log. de mise à 1 de SLOTCXROM pour utiliser les MEM des interfaces	W e
-16377	CØØ7	SETINTCXROM	Com. log. de mise à Ø de SLOTCXROM pour pouvoir utiliser la MEM interne située entre C1ØØ et CFFF sur l'Apple //e	W e
-16376	CØØ8	CLRALTZP	Com. log. de mise à Ø de ALTZP pour utiliser la page Zéro et la pile de la mémoire principale	W e
-16375	CØØ9	SETALTZP	Com. log. de mise à 1 de ALTZP pour utiliser la page Zéro de la pile de la mémoire auxiliaire	W e
-16374	CØØA	SETINTC3ROM	Com. log. de mise à Ø de SLOTC3ROM pour utiliser la MEM interne C3ØØ à C3FF	W e
-16373	CØØB	SETSLOTC3ROM	Com. log. de mise à 1 de SLOTC3ROM pour utiliser la MEM interne C3ØØ à C3FF	W e

Adresses Dec	Hex	Nom	Rôle	e=Apple//e R=Lire,PEEK W=Ecrire,POKE	
-16372	CØØC	CLR8ØCOL	Com. log. de mise à Ø de 8ØCOL pour désactiver l'affichage sur 8Ø colonnes	W	e
-16371	CØØD	SET8ØCOL	Com. log. de mise à 1 de 8ØCOL pour mettre en ligne l'affichage de 8Ø colonnes	W	e
-1637Ø	CØØE	CLRALTCHAR	Com. log. de mise à Ø de ALTCHARSET pour afficher les caractères du jeu primaire	W	e
-16369	CØØF	SETALTCHAR	Com. log. de mise à 1 de ALTCHARSET pour afficher les caractères du jeu alternatif	W	e
-16368	CØ1Ø	KBDSTRB	Echantillonnage du clavier pour lire une nouvelle touche	W	
-16367	CØ11	RDLCBNK2	Etat de lecture sur la MEV à banc commuté ou sur le banc 2 ou sur le banc 1	R	e
-16366	CØ12	RDLCRAM	Etat de lecture sur l'espace DØØØ à FFFF ou sur la MEM interne ou sur la MEV	R	e
-16365	CØ13	RDRAMRD	Etat du commutateur logiciel RAMRD. Lecture en MEV principale ou auxiliaire	R	e
-16364	CØ14	RDRAMWRT	Etat du commutateur logiciel RAMWRT. Ecriture en MEV principale au auxiliaire	R	e
-16363	CØ15	RDCXROM	Etat du commutateur SLOTCXROM. Sélection de la MEM interne ou externe sur l'espace C1ØØ à CFFF	R	e
-16361	CØ17	RDC3ROM	Etat du commutateur SLOTC3ROM. Sélection de la MEM interne ou externe sur l'espace d'adresses C3ØØ à C3FF	R	e
-1636Ø	CØ18	RD8ØSTORE	Etat du commutateur logiciel 8ØSTORE. MEV principale ou auxiliaire en ligne	R	e
-16359	CØ19	RDVBL	Lecture du signal d'effacement vertical	R	e
-16358	CØ1A	RDTEXT	Etat du commutateur logiciel TEXT	R	e

Adresses Dec	Hex	Nom	Rôle	e=Apple//e R=Lire,PEEK W=Ecrire,POKE	
-16357	CØ1B	RDMIXED	Etat du commutateur logiciel MIXED	R	e
-16356	CØ1C	RDPAGE2	Etat du commutateur logiciel PAGE2	R	e
-16355	CØ1D	RDHIRES	Etat du commutateur logiciel HIRES	R	e
-16354	CØ1E	RDALTCHAR	Etat du commutateur logiciel ALTCHARSET	R	e
-16353	CØ1F	RD8ØCOL	Etat du commutateur logiciel 8ØCOL	R	e
-16352	CØ2Ø	TAPEOUT	Sortie sur cassette	R	
-16336	CØ3Ø	SPKR	Envoi d'une impulsion sur le haut-parleur	R	
-1632Ø	CØ4Ø	STROBE'	Envoi d'une impulsion de la durée de Ø8 sur la patte n°5 du connecteur de jeux	R	
-163Ø4	CØ5Ø	TXTCLR	Com. log. de mise à Ø de TEXT pour utiliser le mode graphique	W	
-163Ø3	CØ51	TXTSET	Com. log. de mise à 1 de TEXT pour utiliser le mode texte de visualisation	W	
-163Ø2	CØ52	MIXCLR	Com. log. de mise à Ø de MIXTE pour utiliser tout l'écran en graphique	W	
-163Ø1	CØ53	MIXCLR	Com. log. de mise à 1 de MIXTE pour visualiser 4 lignes de texte sous le graphique	W	
-163ØØ	CØ54	TXTPAGE1	Com. log. de mise à Ø de PAGE2 pour demander l'affichage de la page 1	W	
-16299	CØ55	TXTPAGE2	Com. log. de mise à 1 de PAGE2 pour obtenir l'affichage de la page 2	W	
-16298	CØ56	LORES	Com. log. de mise à Ø de HIRES pour visualiser en Basse-Résolution graphique	W	
-16297	CØ57	HIRES	Com. log. de mise à 1 de HIRES. Visualiser en haute-Résolution graphique	W	
-16296	CØ58	SETABØ	La sortie tout-ou-rien ANØ est mise au niveau de tension Ø.3v.	W	

Adresses Dec	Hex	Nom	Rôle	e=Apple//e R=Lire,PEEK W=Ecrire,POKE
-16295	CØ59	CLRANØ	La sortie tout-ou-rien ANØ est mise au niveau de tension 3.5v	W
-16294	CØ5A	SETAN1	idem	W
-16293	CØ5B	CLRAN1		W
-16292	CØ5C	SETAN2	pour les trois sorties	W
-16291	CØ5D	CLRAN2		W
-1629Ø	CØ5E	SETAN3	AN1, AN2, AN3	W
-16289	CØ5F	CLRAN3		W
-16288	CØ6Ø	TAPEIN	Lecture ou entrée du signal sur cassette capté sur le bit 7 de cette adresse	R
-16287	CØ67	PBØ	Etat du bouton-poussoir Ø (dans le bit 7) et de la touche POMME-OUVERTE ♁	R
-16286	CØ62	PB1	Etat du bouton-poussoir 1 (dans le bit 7) et de la touche POMME-PLEINE	R
-16285	CØ63	PB2	Etat du bouton-poussoir 2 (dans le bit 7)	R
-16284	CØ64	PADDLØ	Etat de sortie du monostable relié à la manette de jeux n°Ø (bit 7)	R
-16283	CØ65	PADDL1	Idem pour la manette n°1	R
-16282	CØ66	PADDL2	Idem pour l'entrée analogique n°2	R
-16281	CØ67	PADDL3	Idem pour l'entrée analogique n°3	R
-1628Ø	CØ68		Même effet que CØ6Ø	
-16279	CØ69		Même effet que CØ61	
-16278	CØ6A		Même effet que CØ62	
-16277	CØ6B		Même effet que CØ63	
-16276	CØ6C		Même effet que CØ64	
-16275	CØ6D		Même effet que CØ65	
-16274	CØ6E		Même effet que CØ66	
-16273	CØ6F		Même effet que CØ67	
-16272	CØ7Ø	PTRIG	Remise à Ø des monostables pour débuter la lecture des manettes de jeux	R
-16256	CØ8Ø	CØ8Ø	Com. log. de lecture seulement en MEV de l'espace DØØØ-FFFF et banc n°2	W
-16255	CØ81	CØ81	Com. log. de lecture en MEM et écriture en MEV dans l'espace DØØØ-FFFF et banc n°2	W

Adresses Dec	Hex	Nom	Rôle	e=Apple//e R=Lire,PEEK W=Ecrire,POKE
-16254	CØ82	CØ82	Com. log. de sélection de la MEM entre DØØØ et FFFF	W
-16253	CØ83	CØ83	Com. log. de sélection de la MEV entre DØØØ et FFFF et sur le banc n°2	W (2 fois)
-16252	CØ84		Même effet que CØ8Ø	
-16251	CØ85		Même effet que CØ81	
-1625Ø	CØ86		Même effet que CØ82	
-16249	CØ87		Même effet que CØ83	
-16248	CØ88		Com. log. de lecture seulement en MEV et éventuellement sur le banc n°1	W
-16247	CØ89	CØ89	Com. log. de lecture en MEM et écriture en MEV et éventuellement sur le banc n°1	W
-16246	CØ8A		Com. log. de sélection de la MEM	W
-16245	CØ8B		Com. log. de sélection de la MEV dans la zone d'adresses DØØØ-FFFF et le banc n°1	W
-16244	CØ8C		Même effet que CØ88	
-16243	CØ8D		Même effet que CØ89	
-16242	CØ8E		Même effet que CØ8A	
-16241	CØ8F		Même effet que CØ8B	

Page douze ($C0-CFFF)

Dec	Hex	Nom	Rôle
-16240,-16225	C090-C09F	DEV SELECT 1	périphérique 1 sélectionné avec 16 adresses réservées au dialogue
-16234,	C0A0-C0AF	DEV SELECT 2	périphérique 2
-16208,	C0B0-C0BF	DEV SELECT 3	périphérique 3
-16192,	C0C0-C0CF	DEV SELECT 4	périphérique 4
-16176,	C0D0-C0DF	DEV SELECT 5	périphérique 5
-16160,	C0E0-C0EF	DEV SELECT 6	périphérique 6
-16144,	C0F0-C0FF	DEV SELECT 7	périphérique 7
	Cs00-CsFF	- Adresses des sous-programmes en MEM dans chaque interface de périphérique connectée en s - Chaque sous-programme est limité à 256 octets	
-14336,-12289	C800-CFFF	- Extension d'espace de mémoire pour une MEM installée sur une carte d'interface activée par DEV SELECT s désactivée par CLRROM	
-12289	CFFF	CLRROM	désactivation de l'extension MEM $C800-$CFFF

MEM Mini-assembleur - sweet 16
Integer + Monitor
Prog's Aid

Adresses Dec	Adresses Hex	Rôle
-2816	F500	Mini-assembleur
-2458	F666	Entrée
-2423	F689	Sweet 16 Interpréteur pour programmer une
-2054	F7FA	pseudo-machine à 16 bits
-12288	D000	Prog's Aid (graphiques haute résolution)
-11265	D3FF	
-11076	D4BC	Prog's Aid (ajout de programmes Integer)
-10955	D535	Prog's Aid (vérification d'enregnt cassette)
-10531	D6DD	Prog's Aid (renumérotation Integer)
-10521	D6E7	Prog's Aid (renumérotation partielle)
-10473	D717	Prog's Aid (musique)
-8192	E000	INTEGER
-8189	E003	Entrée à froid ('ctrl B')
-2049	F7FF	Entrée à chaud ('ctrl C')

Commandes		Adresses des sous-programmes	
Codage	Nom	Nom	Adresse
BC	'Ctrl C'	BASCONT	FEB3
B2	'Ctrl Y'	USR	FECA
BE	'Ctrl E'	REGZ	FEBF
B2 ou ED	'Ctrl Y' T	USR ou TRACE	FECA (AUTOSTART) FEC2 (MONITOR)
EF	V	VFY	FE36
C4	'Ctrl K'	INPRT	FE8D
B2 ou EC	'Ctrl Y' S	USR ou STEPZ	FEC5 (AUTOSTART) FEC4 (MONITOR)
A9	'Ctrl P'	OUTPRT	FE97
BB	'Ctrl B'	XBASIC	FEBØ
A6	-	SETMODE	FE18
A4	+	SETMODE	FE18
Ø6	M	MOVE	FE2C
95	<	LT	FE2Ø
Ø7	N	SETNORM	FE84
Ø2	I	SETINV	FE8Ø
Ø5	L	LIST	FE5E
FØ	W	WRITE	FECD
ØØ	G	GO	FEB6
EB	R	READ	FEFD
93	:	SETMODE	FE18
A7	.	SETMODE	FE18
C6	'Return'	CRMON	FEF6
99	espace	BLANK	FEØ4

Le codage des caractères de commande est tel qu'il apparaît dans la table des commandes : CHRTBL $FFCC-$FFE2. La formule de dérivation du code ASCII à ce code consiste en 2 opérations successives EOR #BØ
ADC #88 (C = 1) (voir NXTCHR en $FFAD)

La table des vecteurs de commandes (adresses des sous-programmes) SUBTBL: $FFE3-$FFF9 ne contient que la partie basse des adresses -1 ; la partie haute et constante est égale à $FE.

Les grandes fonctions du MONITOR

1 - Début

-155	FF65	MON	entrée avec 'bip' dans le monitor
-151	FF69	MONZ	entrée sans 'bip' dans le monitor

2 - Entrée de données (placées en $2ØØ sur X caractères)

-665	FD67	GETLINZ	lecture d'une ligne de commandes
-662	FD6A	GETLN	comme GETLINZ avec affichage de '*'
-651	FD75	NXTCHAR	lecture du prochain caractère détecté '→'
-715	FD35	RDCHAR	lecture d'une touche avec détection de 'esc'
-756	FDOC	RDKEY	lecture d'une touche avec curseur clignotant
-741	FD1B	KEYIN	caractère dans l'A. Inc. de RNDL,H
-721	FD2F	ESC	gestion du déplacement du curseur
-707	FD3D	NOTCR	détection de 'Ctrl X' et BS et 248 caractères maximum
-636	FD84	ADDINP	rajout dans le tampon d'entrée $2ØØ jusqu'à 'Return'

3 - Analyse des données et interprétation

-144	FF7Ø	Analyse du tampon $2ØØ avec MODE = Ø	
-141	FF73	NXTITM	analyse de l'item suivant
-89	FFA7	GETNUM	récupération d'un nombre hexa
-83	FFAD	NXTCHR	dispatching du type de caractère et
-118	FF8A	DIG	décodage ASCII vers A2L,H du nombre
-134	FF7A	CHRSRCH	recherche de la commande
-66	FFBE	TOSUB	appel du sous-programme correspondant
-488	FE18	SETMODE	MODE = ':' ou '.' ou '+' ou '-'
-132	FF7C	LZMODE	met à zéro MODE

4 - Affichage des registres

-550	FDDA	PRBYTE	contenu de A en 2 chiffres hexa
-1724	F944	PRNTX	contenu de X
-1727	F941	PRNTAX	contenu de A,X en 4 chiffres hexa
-1728	F940	PRNTYX	contenu de Y,X en 4 chiffres hexa
-541	FDE3	PRHEX	poids faible de A en 1 chiffre hexa
-321	FEBF	REGZ	tous les registres
-1321	FAD7	RGDSP	sont affichés
-1318	FADA	RGDSP1	A,X,Y,P,S

5 - Sortie des caractères

-626	FD8E	CROUT	saut à la ligne suivante
-531	FDED	COUT	JMP (CSWL)
-528	FDFØ	COUT1	affichage d'1 carac. sur écran (ACC)
-522	FDF6	COUTZ	avec sauvegarde de A et Y
-1160	FB78	VIDWAIT	suspension d'affichage par 'Ctrl S'

-1027	FBFD	VIDOUT	dispatching des types de caractères
-1040	FBFØ	STORADV	affichage et déplacement du curseur
-198	FF3A	BELL	déclenche un bip
-1063	FBD9	BELL1	retarde le bip
-1052	FBE4	BELL2	stimule le
-1720	F948	PRBLNK	3 espaces
-384	FE8Ø	SETINV	mode inversé d'affichage
-380	FE84	SETNORM	mode normal d'affichage

6 - Gestion du curseur

-1036	FBF4	ADVANCE	déplacement d'une position à droite
-926	FC62	CR	au début de la ligne suivante
-922	FC66	LF	à la ligne suivante
-1008	FC1Ø	BS	à une position à gauche
-998	FC1A	UP	à la ligne précédente
-990	FC22	VTAB	à la ligne spécifiée dans Acc CV
-1189	FB5B	TABV	à la ligne spécifiée dans Acc CV
-1087	FBC1	BASCALC	calcul de l'adresse de base BASL, BASH d'après A

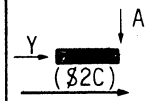
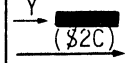
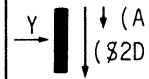
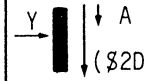
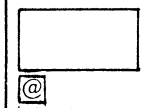
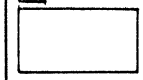
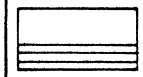
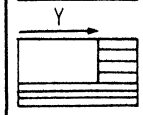
7 - Gestion de l'écran texte

-1223	FB39	SETTXT	mode texte
-1205	FB4B	SETWND	fenêtre texte
-936	FC58	HOME	effacement dans les limites de la fenêtre
-958	FC42	CLREOP	effacement jusqu'au bas de l'écran
-868	FC9C	CLREOL	effacement jusqu'au bout de la ligne
-912	FC7Ø	SCROLL	déroulement vers le haut

8 - Graphique basse résolution

-1216	FB4Ø	SETGR	mode GR mixte. Effacement
-1948	F864	SETCOL	couleur spécifiée dans A COLOR
-2048	F8ØØ	PLOT	voir détails page ci-contre
-2034	F8ØE	PLOT1	
-2023	F819	HLINE	
-2020	F81C	HLINE1	
-2010	F826	VLINEZ	
-2008	F828	VLINE	
-1998	F832	CLRSCR	
-1994	F836	CLRTOP	
-1977	F847	GBASCALC	calcule l'adresse-base de la ligne graphique spécifiée dans A
-1935	F871	SCRN	met dans A, la couleur du domino situé en Y,A

Commandes monitor

Appel BASIC	Appel Monitor	Nom	Résultat
CALL-2048	F8ØØG	PLOT	 calcul de GBASL,H
CALL-2034	F8ØEG	PLOT1	 à l'ordonnée courante
CALL-2023	F819G	HLINE	 A et Y sont modifiés
CALL-2020	F81CG	HLINE1	 à la ligne courante
CALL-2010	F826G	VLINEZ	 A est modifié
CALL-2008	F828G	VLINE	
CALL-1998	F832G	CLRSCR	 effacement de l'écran graphique et mise à @ du texte
CALL-1994	F836G	CLRTOP	 effacement de l'écran graphique. Les 4 lignes du bas ne sont pas touchées
CALL-1992	F838G	CLRSC2	 Y effacement jusqu'à la ligne Y
CALL-1988	F83CG	CLRSC3	 effacement de l'écran dans la partie haute et à gauche

9 - Entrée - sortie

-1425	FA6F	INITAN	Initialisation des sorties logiques (C058-C05F)
-1250	FB1E	PREAD	lecture des manettes, n° dans X, résultat dans Y
-371	FE8D	INPRT	si "IN s" pour nouvelle entrée
-375	FE89	SETKBD	mettre le clavier en ligne
-361	FE97	OUTPRT	si 'PR s' pour nouvelle sortie
-365	FE93	SETVID	mettre l'écran en ligne (PR#Ø)
-307	FECD	WRITE	écriture en cassette depuis (A1) jusqu'à (A2)
-259	FEFD	READ	lecture et mémorisation entre (A1) et (A2)
-823	FCC9	HEADR	écriture sur cassette
	FAB4	détection	du connecteur disquette

10 - Désassemblage

-418	FE5E	LIST	si 'L', 20 instructions désassemblées
-413	FE63	LIST2	désassemblage de (A) instructions
-184Ø	F8DØ	INSTDSP	une instruction
-1918	F882	INSDS1	son adresse
-1906	F88C	INSDS2	son code d'opération ?
-1879	F8A9	GETFMT	recherche du type d'instruction
-1709	F953	PCADJ	ajustement du compteur ordinal

11 - Affichage des mémoires

-589	FDB3	XAM	affichage de (A1) à (A2)
-622	FD92	PRA1	affichage adresse suivie de -
-586	FDB6	DATAOUT	affichage contenus
-838	FCBA	NXTA1	incrémentation de A1 jusqu'à A2 (C=1)

12 - Déplacement de mémoires et vérification

-468	FE2C	MOVE	déplacement de (A1)-(A2) vers (A4)
-844	FCB4	NXTA4	incrémentation de A4L,H
-458	FE36	VFY	vérification de (A1)-(A2) avec (A4)
-48Ø	FE2Ø	LT	transfert de A2 en A4 et A5

13 - Arithmétique hexadécimale

		XAMPM	
-57Ø	FDC6	Addition ou soustraction de A1L et A2L avec '+' ou '-'	dans l'acc. Résultat affiché

14 - Système superviseur

-1472	FA4Ø	IRQ	si interruption hardware
-1447	FA59	OLDBREAK	si "BRK" en langage machine
-1438	FA62	RESET	si 'Reset'
-1407	FA81	NEWMON	dispatching du démarrage
-1370	FAA6	PWRUP	démarrage à froid

-1367	FAA9	SETPG3	mise en place des vecteurs $3FØ à 3F3
-1169	FB6F	SETPWRC	calcul de la valeur à mettre en $3F4
-856	FCA8	WAIT	temporisation
-336	FEBØ	XBASIC	redémarrage en BASIC
-333	FEB3	BASCONT	continuer en BASIC
-330	FEB6	GO	si G
-266	FEF6	CRMON	si 'Return' seul
-310	FECA	USR	JMP $3F8 si 'Ctrl Y'
-193	FF3F	RESTORE	restauration de A,X,Y,P,S
-182	FF4A	SAVE	sauvegarde de A,X,Y,P,S

Différences

AUTOSTART		*MONITOR*	
STEP	n'existe pas	FA4Ø-FA85 FAA5-FAD6 FAFD-FB18	-désassemblage de l'instruction en cours, mode pas à pas
IRQ	FA40	FA86	-prise en compte d'une in-
BREAK	FA4C	FA92	terruption ou d'un arrêt
OLDBREAK	FA59	XBRK FA9C	-affichage PC et registres
RESET	FA62-FAA3	n'existe pas	-auto-démarrage
APPLE II	FB6Ø	n'existe pas	-Affichage APPLE II au départ
SETPWRC	FB6F	n'existe pas	-mémorisation du PWREDUP
VIDWAIT	FB78	n'existe pas	-interruption d'affichage
KBDWAIT	FB88	n'existe pas	par 'Ctrl S' et reprise
NOWAIT	FB94	n'existe pas	
ESCOLD	FB97	n'existe pas	-déplacement du curseur par
ESCNOW	FB9B	n'existe pas	'esc' I,J,K,M
ESCNEW	FBA5	n'existe pas	
MULPM	n'existe pas	FB60-FB80	-multiplication entière 16 bit
DIVPM	n'existe pas	FB81-FBCØ	-division entière
TRACE	n'existe pas	FEC2	-mode trace cf. STEP
STEPZ	FEC4	FEC4	-en AUTOSTART, conduit à USR

Nouvelles adresses du Monitor de l'Apple //e

Les sous-programmes standards ont les mêmes adresses d'entrée que dans les autres Monitor ; mais le déroulement des instructions diffère puisque les fonctions sont reportées sur le nouvel espace de MEM entre C1ØØ et CFFF.

Les instructions préliminaires consistent à prédisposer le registre d'index Y à une valeur correspondante à l'action désirée avant d'aller en FBB4 ou GOTOCX, sous programme qui sauvegarde

le registre d'état P sur la pile, inhibe les interruptions, sauvegarde l'état du commutateur logiciel SLOTCXROM puis met ce commutateur à l'état Ø puisqu'il s'agit d'allouer l'espace d'adresses C1ØØ-CFFF à la MEM interne qui en standard dans l'Apple //e contient les nouveaux sous-programmes du Monitor.

Effets de l'appel à FBB4 ou GOTOCX suivant la valeur de Y :

Y	*Action correspondante*	*Identification*
Ø	Effacement de l'écran jusqu'en bas de page	CLEROP
1	Effacement de tout l'écran et curseur en haut	HOME
2	Déroulement de tout l'écran	SCROLL
3	Effacement jusqu'au bout de la ligne courante	CLEOL
4	Effacement jusqu'au bout de la ligne repérée par BASL	CLEOL2
5	Démarrage ou redémarrage	INIT
6	Entrée d'un caractère au clavier	KEYIN
7	Sélection du mode Déplacement du curseur	EscFIX
8	Mise en place de la fenêtre écran	SETWND

Après exécution de ces sous-programmes, le retour dans la MEM F8ØØ-FFFF s'effectue par FD29 ou FUNCEXIT qui rétablit l'état avant le saut en C1ØØ préparé par FBB4 ou GOTOCX.

Adresses utiles dans la MEM C1ØØ-CFFF de l'Apple //e

(mettre SLOTCXROM à Ø au préalable par POKE-16377,Ø)

Adresse	*Nom*	*Explications*
-15799 C249	COLDSTART	Destruction volontaire de 2 octets par page et redémarrage à froid
-15775 C261	DIAGS	Appel du sous-programme d'auto-diagnostic
-15616 C3ØØ	BASICINIT	Préparation à la gestion des E/S en Basic
-15599 C311	AUXMOVE	JMP MOVE ; pour transférer des contenus entre les mémoires principales et auxiliaire
-15596 C314	AUXFER	JMP XFER ; pour transférer l'exécution entre les mémoires principale et auxiliaire
-15593 C317	BASICENT	Point d'entrée des E/S de base du Basic

Adresse	*Nom*	*Explications*
-15514 C363	MOVE	Sous-programmes de copie automatique de contenus de mémoires : C=1 mémoire principale à mémoire auxiliaire C=Ø mémoire auxiliaire à mémoire principale A1L,H adresse de début ; A2L,H adresse de fin ; A4L,H adresse de début de la copie
-15440 C3BØ	XFER	Sous-programme de transfert d'exécution entre mémoires : C=1 mémoire principale à mémoire auxiliaire C=Ø mémoire auxiliaire à mémoire principale V=1 page zéro et pile de la mémoire auxiliaire V=Ø page zéro et pile de la mémoire principale $3ED, $3EE adresse de début de programme.
-13532 CB24	TESTCARD	Teste la présence de la carte 8Ø colonnes dans le connecteur auxiliaire et agit sur N
-13487 CB51	BASCALC	Calcule de l'adresse de base d'une ligne
-13415 CB99	CTRLCHAR	Prise en compte des caractères de contrôle
-13380 BBBC	X.BELL	si Ctrl - G , sonnerie
-13449 XBDB	X.BS	si Ctrl - H , recul d'une position
-13332 CBEC	X.CR	si Ctrl - M , retour-chariot
-13299 CCØD	X.EM	si Ctrl - Y , exécute HOME
-13286 CC1A	X.SUB	si Ctrl - Z , effacement de la ligne
-13274 CC26	X.FS	si Ctrl - ç , avance d'une position
-13260 CC34	X.US	si Ctrl - , remonte d'une ligne
-13239 CC49	X.SØ	si Ctrl - N , affichage video normal
-13230 CC52	X.S1	si Ctrl - O , affichage video inverse
-13167 CC91	X.LF	si Ctrl - J , saut de ligne
-13148 CCA4	SCROLLUP	si Ctrl - W , déroulement vers le haut
-13142 CCAA	SCROLDN	si Ctrl - V , déroulement vers le bas
-13021 CD23	X.VT	si Ctrl - K , effacement jusqu'en bas
-12990 CD42	X.FF	si Ctrl - L , efface tout l'écran, le curseur n'avançant que d'une ligne
-12984 CD48	X.GS	si Ctrl - § , effacement jusqu'en fin de ligne
-12983 CD49	X.DC1	si Ctrl - Q , mode 4Ø col. dû à Esc 4
-12937 CD77	X.DC2	si Ctrl - R , mode 8Ø col. dû à Esc 8
-12912 CD9Ø	X.NAK	si Ctrl - U , quitte le mode 8Ø colonnes
-12901 CD9B	FULL8Ø	mise en place de la fenêtre 8Ø colonnes

Adresse	*Nom*	*Explications*
-12886 CDAA	QUIT	Retour à l'affichage 4Ø colonnes standard
-12837 CDDB	SCRN84	Conversion 8Ø colonnes en 4Ø colonnes
-12750 CE32	SCRN48	Conversion 4Ø colonnes en 8Ø colonnes
-12625 CEAF	SETCH	Affecte à CH et OURCH ($57B) la valeur Acc
-12579 CEDD	INVERT	Inversion du car. affiché sous le curseur
-12558 CEF2	STORCHAR	Affiche le car. de l'Acc. à la position CH donnée par Y
-12543 CFØ1	PICK	Récupère dans l'Acc. le car. situé en position horizontale Y.
-12538 CFØ6	SCREENIT	Pour afficher ou récupérer un car. écran
-12462 CF52	ESCON	Mise en mode ESC du curseur (+ inversé)
-12443 CF65	ESCOFF	Curseur en mode normal
-12424 CF78	COPYROM	Recopie de la MEM interne F8 sur MEV

SOFTEV ET PWREDUP $3F2, $3F3 ET $3F4

L'AUTOSTART ROM, le monitor en MEM disponible sur l'Apple II Plus, a de quoi déterminer son type de redémarrage en cas de 'Reset'.

Le 'Reset' à chaud est programmable dans le vecteur SOFTEV et le monitor se sert d'un octet particulier PWREDUP comme trace du passage par un démarrage à froid. PWREDUP doit être le ou exclusif # $A5 et du contenu de $3F3 si on veut que 'Reset' produise un démarrage à chaud.

Valeur par défaut :

	SOFTEV	PWREDUP	
sans SED	$EØØ3	$45	(Basic à chaud)
avec SED	$9DBF	$38	

Si au démarrage à froid, dès la mise sous tension, une carte d'interface de lecteur de disquette est reconnue, alors le monitor laisse s'exécuter le programme de chargement du SED (Bootstrap) présent en MEM sur la carte d'interface.

Dans l'ensemble des fonctions de mise en route du SED, l'affectation d'une adresse de redémarrage à chaud (en cas de 'Reset') est réalisée dans SOFTEV ($9DBF) (poids faible en tête).

SOFTEV peut être modifié par l'utilisateur qui aura soin d'affecter à PWREDUP le ou exclusif de SOFTEV+1 et de $A5 pour que 'Reset' conduise à un programme spécifique et ne provoque pas l'équivalent d'un démarrage à froid.

AMPERV $3F5, $3F6, $3F7

L'adresse AMPERV sera utilisée (par indirection) pour déclencher l'exécution d'un programme en langage machine depuis un programme en APPLESOFT contenant le MOT-CLE &.

Exemple : * 3F5 : 4C ØØ Ø3 JMP $3ØØ

Le mot-clé & conduira à l'exécution du sous-programme commençant à l'adresse $3ØØ.

DOSWARMSTART $3DØ : 4C BF 9D

Cette adresse est à utiliser pour reprendre l'APPLESOFT (controlé par le SED) depuis le Monitor, à savoir :

```
* 3DØG
]
```

Le programme courant n'a pas été touché.

MONZ $FF69 (-151)

Adresse d'entrée dans le Monitor.
Depuis l'APPLESOFT, on tape

CALL - 151

alors *

Routines fondamentales

Impression d'un caractère

COUT $FDED — Affiche le caractère présent dans l'accumulateur et avance le curseur. Prend en compte 'Return' 'LF' et les 2 modes Normal/Inverse.

OUTDO $DB5C — Dans APPLESOFT, affiche le caractère présent dans l'accumulateur et prend en compte les 3 modes video Normal/Inverse/Clignotant.

Acquisition d'un caractère dans le texte BASIC

CHRGET $00B1 — Ce sous-programme (qui s'auto-modifie) pointe en $B8, $B9 le caractère à prendre qui sera chargé dans l'accumulateur Z = 1 si fin d'instruction ($3A ou $00) C = 0 si le caractère est un chiffre (les espaces du texte en BASIC ont été sautés).

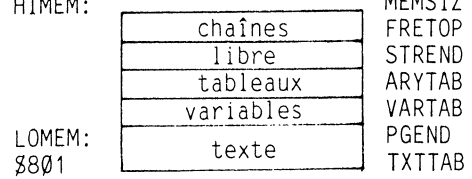
CHRGOT $00B7 — Le caractère est le caractère actuel non le suivant comme dans CHRGET.

Lecture d'un caractère tapé au clavier

RDKEY $FD0C — Attend qu'une touche soit pressée avec le curseur clignotant. Le code du caractère est chargé dans l'accumulateur

Nom	Adresse HEX	Adresse DEC	Rôle
TXTTAB	$67,$68	103,104	Début du texte BASIC = $801 (2049) par défaut.
VARTAB	$69,$6A	105,106	Début des variables simples, des pointeurs de chaînes, des pointeurs de fonction.
ARYTAB	$6B,$6C	107,108	Début des variables dimensionnées, pointeurs des tableaux de chaînes.
STREND	$6D,$6E	109,110	Début de l'espace libre.
FRETOP	$6F,$70	111,112	Fin de l'espace libre. Fin des chaînes.
MEMSIZ	$73,$74	115,116	Début des chaînes. Fin de l'espace mémoire +1. Les chaînes sont enregistrées du haut vers le bas.
CURLIN	$75,$76	117,118	Numéro de la ligne en cours d'exécution.
OLDLIN	$77,$78	119,120	N° de la ligne interrompue par 'Ctrl C', STOP ou END.
OLDTXT	$79,7A	121,122	Adresse du dernier octet ($00) de la ligne en cours d'exécution.
DATLIN	$7B,$7C	123,124	N° de la ligne dans laquelle sont lues les DATA.
DATPTR	$7D,$7E	125,126	Adresse du premier octet des DATA à lire.
INPPTR	$7F,$80	127,128	Pointe le tampon d'entrée par clavier pendant INPUT.
VARNAM	$81,$82	129,130	Contient le nom (2caractères) de la dernière variable référencée.
VARPNT	$83,84	131,132	Adresse de la valeur de la dernière variable référencée, ou de l'octet de longueur d'une chaîne.
PGEND	$AF,$B0	175,176	Fin du texte BASIC.

Résumé



Implantation d'un programme et des variables en MEV

```
*
JLIST

10 AA = 2
20 AA% = 4
30 AA$ = ""
40  DIM AA(1,3)
50  DIM AA%(2,1)
60  DIM AA$(3,2)
70  DEF FN AA(X)= X - 256 * INT
     (X / 256)
80  PRINT  FN AA(257)

JRUN
1

JCALL -151

*69.6A

0069- 74 08
*800.873

0800- 00 <0A 08>[0A 00]41 41(D0
0808- 32 00<14 08>[14 00] 41 41
0810- 25 (D0 34 00<1F 08>[1E 00]
0818- 41 41 24 (D0 22 22 00<2C
0820- 08>[28 00](86 41 41 (28 31
0828-(2C 33 29 00<3A 08>[32 00]
0830-(86 41 41 (25 28 32 2C 31
0838-(29 00<48 08>[3C 00](86 41
0840- 41 24(28 33 2C 32 29 00
0848-<63 08>[46 00](B8(C2 41 41
0850-(28 58 29(D0 58(C9 32 35
0858- 36 (CA (D3 (28 58 (CB 32 35
0860- 36 29 00<71 08>[60 00](BA
0868-(C2 41 41 (28 32 35 37(29
0870- 00 00 00 0A
```

AA variable réelle simple
AA% variable entière simple
AA$ variable chaîne de caractère simple

AA(1,3) variable réelle dimensionnée
AA%(2,1) variable entière dimensionnée
AA$(3,2) variable chaîne dimensionnée

FN AA(X) fonction définie par le programme

Texte Basic codé

< >adresse du début de la ligne d'instruction suivante

[ ] numéro de ligne d'instruction

(mot-clé

ØØ fin de ligne

ØØ ØØ fin du texte-programme

—— définition de fonction

Variables simples

```
*6B.6C

006B- 07 08
*874.896

0874-.41 41;82 00
0878- 00 00 00.C1 C1;00 04 00
0880- 00 00.41 C1;00,1D 08 00
0888- 00•C1 41;54 08,92 08,58.
0890-.58 00;00 00 00 00 00.
```

2 caractères du nom ; valeur.

Codage des 2 caractères du nom :

		1er car.	2e car.
AA	réelle	P	P
AA%	entière	N	N
AA$	chaîne	P	N
AA(X)	fonction	N	P

P est le code ASCII avec le bit 7=Ø
N est le code ASCII avec le bit 7=1

Valeur d'une variable sur 5 octets :

	1er octet	2e octet	3e octet	4e octet	5e octet
Réelle	exposant	mantisse			
Entière	poids forts	poids faibles	non utilisés		
Chaîne	longueur	adresse du début de chaîne		non utilisés	
Fonction	adresse de la définition		adresse de l'argument		code P du 1er car. après =

Les adresses sont exprimées avec les poids faibles sur le 1er octet et les poids forts sur le 2ème octet.

Implantation des variables dimensionnées

```
*6D.6E

006D- 0A 09
*897.909

0897-.41
0898- 41;<31 00>02 00 04 00 02
08A0-/00 00 00 00 00/00 00 00
08A8- 00 00/00 00 00 00 00/00
08B0- 00 00 00 00/00 00 00 00
08B8- 00/00 00 00 00 00/00 00
08C0- 00 00 00/00 00 00 00 00
08C8-.C1 C1;<15 00>02 00 02 00
08D0- 03/00 00/00 00/00 00/00
08D8- 00/00 00/00 00/.41 C1;<2D
08E0- 0C>02 00 03 00 04 00 00
08E8- 00/00 00 00/00 00 00/00
08F0- 00 00/00 00 00/00 00 00/
08F8- 00 00 00/00 00 00/00 00
0900- 00/00 00 00/00 00 00/00
0908- 00 00/
```

DIM AA(1,3)

DIM AA%(2,1)

DIM AA$(3,2)

. 2 caractères du nom ; offset / variable suivante

D nombre de dimensions ou d'indices

└──┘ nombre maximum d'éléments du tableau pour chaque dimension de la dernière à la première (valeur maximum de l'indice +1).

/ / valeur de chaque élément du tableau
- réelle 5 octets
- entière 2 octets seulement
- chaîne 3 octets : longueur, adresse

dans l'ordre AA(Ø,Ø), AA(1,Ø), AA(Ø,1), AA(1,1), AA(Ø,2), AA(1,2), AA(Ø,3), AA(1,3) l'indice le plus à droite augmentant le plus lentement.

Exemple d'implantation d'un programme et des variables en mémoire vive

```
JLIST

10  REM  APPLESOFT
20  INPUT "NOM ?";N$
30  FOR K = 1 TO  LEN (N$) - 2
40  PRINT  RIGHT$ (N$, LEN (N$) -
     K + 1);" "; MID$ (N$,K,2);"
     ";
50  NEXT K

JCALL -151

*69.6A
                                    VARTAB
0069- 63 08

*67.68
                                    TXTTAB
0067- 01
0068- 08

*800.862

0800- 00<11 08>0A 00(B2 20 41
0808- 50 50 4C 45 53 4F 46 54
0810- 00<21 08>14 00(84 22 4E
0818- 4F 4D 20 3F 22 3B 4E 24
0820- 00<32 08>1E 00(81 4B D0
0828- 31 C1 E3 28 4E 24 29 C9
0830- 32 00<5A 0C>28 00)8A (E9
0838- 28 4E 24 2C E3 28 4E 24
0840- 29 C9 4B C8 31 29 3B 22
0848- 20 22 3B (EA 28 4E 24 2C
0850- 4B 2C 32 29 3B 22 20 22
0858- 3B 00 <61 0C>32 00(82 4B
0860- 00 00 00
```

Texte BASIC codé

(mots-clés

ØØ fin de ligne

ØØ ØØ fin de programme

[ ] n° de ligne

< > adresse de la ligne suivante

Exemple d'implantation d'un programme et de ses variables en mémoire vive.
(suite)

```
]RUN
NOM ?DUAERB
DUAERB DU UAERB UA AERB AE ERB ER

]CALL -151

*6D.6E

006D- 71 08

*863.870

0863-.4E 80;06,FA 95/
0868- 00 00.4B 00;83 20 00 00
0870- 00/

*6F.70

006F- FA
0070- 95

*95FA.95FF

95FA- 44 55 41 45 52 42

*79.7A

0079- 60 08

*AF.B0

00AF- 63
00B0- 08
```

Exécution des instructions

STREND

Variables

-N$: longueur, pointeur
- K : valeur réelle

FRETOP

Valeur de N$

adresse du dernier octet de la dernière instruction exécutée

fin du programme (+2)

Les mots-clés par ordre croissant des codes et les adresses des sous-programmes correspondants dans l'interpréteur.

Codes $80 à $A9

Mot-clé	*Code Décimal*	*Adresse Décimale*
END	$80 128	$D870 55408
FOR	$81 129	$D766 55142
NEXT	$82 130	$DCF9 56569
DATA	$83 131	$D995 55701
INPUT	$84 132	$DBB2 56242
DEL	$85 133	$F331 62257
DIM	$86 134	$DFD9 57305
READ	$87 135	$DBE2 56290
GR	$88 136	$F390 62352
TEXT	$89 137	$F399 62361
PR#	$8A 138	$F1E5 61925
IN#	$8B 139	$F1DE 61918
CALL	$8C 140	$F1D5 61909
PLOT	$8D 141	$F225 61989
HLIN	$8E 142	$F232 62002
VLIN	$8F 143	$F241 62017
HGR2	$90 144	$F3D8 62424
HGR	$91 145	$F3E2 62434
HCOLOR=	$92 146	$F6E9 63209
HPLOT	$93 147	$F6FE 63230
DRAW	$94 148	$F769 63337
XDRAW	$95 149	$F76F 63343
HTAB	$96 150	$F7E7 63463
HOME	$97 151	$FC58 64600
ROT=	$98 152	$F721 63265
SCALE=	$99 153	$F727 63271
SHLOAD	$9A 154	$F775 63349
TRACE	$9B 155	$F26D 62061
NOTRACE	$9C 156	$F26F 62063
NORMAL	$9D 157	$F273 62067
INVERSE	$9E 158	$F277 62071
FLASH	$9F 159	$F280 62080
COLOR=	$A0 160	$F24F 62031
POP	$A1 161	$D96B 55659
VTAB	$A2 162	$F256 62038
HIMEM:	$A3 163	$F286 62086
LOMEM:	$A4 164	$F2A6 62118
ONERR	$A5 165	$F2CB 62155
RESUME	$A6 166	$F318 62232
RECALL	$A7 167	$F3BC 62396
STORE	$A8 168	$F39F 62367
SPEED=	$A9 169	$F262 62050

Les mots-clés par ordre croissant des codes.
Codes $AA à $D4

Mot-clé	Code Décimal	Adresse Décimale
LET	$AA 170	$DA46 55878
GOTO	$AB 171	$D93E 55614
RUN	$AC 172	$D912 55570
IF	$AD 173	$D9C9 55753
RESTORE	$AE 174	$D849 55369
&	$AF 175	$03F5 1013
GOSUB	$B0 176	$D921 55585
RETURN	$B1 177	$D96B 55659
REM	$B2 178	$D9DC 55772
STOP	$B3 179	$D86E 55406
ON	$B4 180	$D9EC 55788
WAIT	$B5 181	$E784 59268
LOAD	$B6 182	$D8C9 55497
SAVE	$B7 183	$D8B0 55472
DEF	$B8 184	$E313 58131
POKE	$B9 185	$E77B 59259
PRINT	$BA 186	$DAD5 56021
CONT	$BB 187	$D896 55446
LIST	$BC 188	$D6A5 54949
CLEAR	$BD 189	$D66A 54890
GET	$BE 190	$DBA0 56224
NEW	$BF 191	$D649 54857
TAB(	$C0 192	
TO	$C1 193	
FN	$C2 194	
SPC(	$C3 195	
THEN	$C4 196	
AT	$C5 197	
NOT	$C6 198	$DE98 -8552 NOT
STEP	$C7 199	
+	$C8 200	$E7BE -6210 +
-	$C9 201	$E7A7 -6233 -
*	$CA 202	$E97F -5761 *
/	$CB 203	$EA66 -5530 /
^	$CC 204	$EF09 -4349 ^
AND	$CD 205	$DF55 -8363 AND
OR	$CE 206	$DF4F -8369 OR
>	$CF 207	
=	$D0 208	$DF6A -8342 =
<	$D1 209	
SGN	$D2 210	$EB90 60304
INT	$D3 211	$EC23 60451
ABS	$D4 212	$EBAF 60335

Les mots-clés par ordre croissant des codes.
Codes $D5 à $EA

Mot-clé	Code Décimal	Adresse Décimale
USR	$D5 213	$000A 10
FRE	$D6 214	$E2DE 58078
SCRN(	$D7 215	$D412 54290
PDL	$D8 216	$DFCD 57293
POS	$D9 217	$E2FF 58111
SQR	$DA 218	$EE8D 61069
RND	$DB 219	$EFAE 61358
LOG	$DC 220	$E941 59713
EXP	$DD 221	$EF09 61193
COS	$DE 222	$EFEA 61418
SIN	$DF 223	$EFF1 61425
TAN	$E0 224	$F03A 61498
ATN	$E1 225	$F09E 61598
PEEK	$E2 226	$E764 59236
LEN	$E3 227	$E6D6 59094
STR$	$E4 228	$E3C5 58309
VAL	$E5 229	$E707 59143
ASC	$E6 230	$E6E5 59109
CHR$	$E7 231	$E646 58950
LEFT$	$E8 232	$E65A 58970
RIGHT$	$E9 233	$E686 59014
MID$	$EA 234	$E691 59025

Classement par grandes fonctions

1 - Début (points d'entrée)

-8192	E000	BASIC	démarrage à froid si 'Crtl B'
-3800	F128	COLDSTART	démarrage à froid si 'Ctrl B'
-8189	E003	BASIC2	2ème entrée dite à chaud si 'Ctrl C' ou 'Reset'
-11204	D43C	CMDLOOP	début de la boucle principale de l'interpréteur (à chaud)

2 - Entrée des données (dans le tampon $200)

-11201	D43F		entrée avec affichage de]
-10962	D52E	INLIN+2	entrée par le clavier d'une instruction avec (X) comme signal (PROMPT)
-10964	D52C	INLIN	entrée sans signal
-10951	D539	GDBUFS	mise à zéro du bit 7 de tous les caractères enregistrés par INLIN
-10925	D553	INCHR	entrée d'un caractère dans l'acc. et mise à zéro du bit 7

3 - Analyse des données (dans la zone du texte du programme)

177	B1	CHRGET	chargement dans l'accumulateur du caractère pointé par ($B8,$B9) ou TXTPTR (suivant ou actuel) et discrimination du type C = 0 pour un chiffre Z = 1 pour la fin d'une ligne ou d'une instruction
183	B7	CHRGOT	
-9716	DA0C	LINGET	chargement dans $50,$51 ou LINNUM du numéro de la ligne pointée par $B8, $B9 ou TXTPTR et poursuite de l'analyse
-6411	E6F5	GTBYTC	saisie d'un caractère par CHRGET et évaluation à partir de TXTPTR pour X
-6408	E6F8	GETBYT	évaluation de l'expression pointée par TXTPTR et mise du résultat dans FAC
-6405	E6FB	CONINT	puis FAC→entier ⩽255 dans X et FACLO
-10726	D61A	FNDLIN	recherche dans le programme l'adresse de l'instruction dont le n° est dans LINNUM ($50,$51). Si C = 1, résultat dans LOWTR ($9B,$9C) sinon LOWTR pointe à l'instruction de n° le plus élevé
-6234	E74C	COMBYTE	vérifie que TXTPTR pointe sur une virgule et continue l'analyse par GETBYT

(suite)

-6330	E746	GETNUM	saisie d'un nombre pour évaluation et test virgule sur caractère suivant
-8857	DD67	FRMNUM	évalue une expression pointée par TXPTR et met le résultat en FAC en s'assurant que c'est un nombre
-6318	E752	GETADR	FAC→entier (2 octets) en $50,51
-2087	F7D9	GETARYPT	recherche d'une variable par son nom pointé par TXTPTR et
-8221	DFE3	PTRGET	résultat dans $83,$84 ou VARPNT et Y,A - si elle n'existe pas déjà, **création** - la place du nom dans la table des variables est en $9B,9C ou LOWTR
-8067	E07D	ISLETC	l'acc a contient-il un code ASCII d'une lettre, oui C = 1
-8526	DEB2	PARCHK	vérification des parenthèses
-8520	DEB8	CHKCLS	TXTPTR pointe-t-il sur) ?
-8517	DEBB	CHKOPN	TXTPTR pointe-t-il sur (?
-8514	DEBE	CHKCOM	TXTPTR pointe-t-il sur , ?
-8512	DEC0	SYNCHR	sinon, erreur de syntaxe si oui l'analyse continue
	EC4A	FIN	enregistre le nombre flottant pointé par TXTPTR dans FAC

4 - Affichage de données

-4818	ED2E	PRNTFAC	affiche le FAC ($9D-A2) et le détruit
-9414	DB3A	STROUT	affiche le chaîne pointée par (Y,A)
-9411	DB3D	STRPRT	affiche la chaîne pointée par (FACMO,FACLO)
-9385	DB57	OUTSP	affiche un espace
-9477	DAFB	CRDO	retour chariot
-9382	DB5A	OUTQST	?
-9380	DB5C	OUTDO	affiche l'acc A avec les modes I,F,N
-4839	ED19	INPRT	affiche "IN" n° de ligne courante
-4828	ED24	LINPRT	affiche un entier dans X,A
-9515	DAD5	PRINT	instruction d'affichage

5 - Arithmétique et fonctions algébriques

CONSTANTES NUMERIQUES

E0FE	90	80	00	00	20	$-2^{16} = -32767.0005$
ED0A	9B	3E	BC	1F	FD	99 999 999,9
ED0F	9E	6E	6B	27	FD	999 999 999
ED14	9E	6E	6B	28	00	1 000 000 000 = 10^9

(suite)

FØ66	81	49	ØF	DA	A2	PI/2 = 1.57079633
FØ6B	83	49	ØF	DA	Ø2	2xPI = 6.28318531
FØ7Ø	7F	00	00	00	00	1/4
EE64	80	00	00	00	00	1/2
E913	81	00	00	00	00	1
E92D	80	35	04	F3	34	SQR(0 5) = 0.707106781
E932	81	35	04	F3	34	SQR(2) = 1.41421356
E937	80	80	00	00	00	-1/2
E93C	80	31	72	17	F8	LOG(2) = .693147181
EA50	84	20	00	00	00	10
EEDB	81	38	AA	3B	2A	LOG(e)/LOG(2) = 1.44269504

FONCTIONS

E7A0	FADDH	(FAC) ← (FAC) + 1/2
E7A7	FSUB	ARG ← (Y,A) et appel FSUBT
E7AA	FSUBT	FAC ← ARG-FAC
E7BE	FADD	ARG ← (Y,A) et appel FADDT
E7C1	FADDT	FAC ← FAC + ARG
E941	LOG	FAC ← LN(FAC)
E97F	FMULT	ARG ← (Y,A) et appel FMULTT
E982	FMULTT	ARG ← FAC * ARG
E9E3	CONUPK	ARG ← (Y,A)
EA39	MUL10	FAC ← FAC * 10
EA55	DIV10	FAC ← FAC/10
EA66	FDIV	ARG ← (Y(A) et appel FDIVT
EA69	FDIVT	FAC ← ARG/FAC
EB80	SGN	FAC ← signe de FAC
EB82	SIGN	A ← signe de FAC (1 si > 0, Ø si Ø, FF si < 0)
EB93	FLOAT	FAC ← A davient flottant
EBAF	ABS	valeur absolue FAC ← FAC
EC23	INT	plus grande valeur entière inférieure FAC ← FAC
EBF2	QINT	plus grande valeur entière inférieure si FAC < 32767
E10C	AYINT	plus grande valeur entière inférieure dans mantisse FAC

6 - Fonctions sur les chaînes de caractère

DEC	*HEX*		
- 8837	DD7B	FRMEVL	évaluation d'une expression à partir de TXTPTR
- 8575	DE81	STRTXT	TXPTR → Y,X puis appelle STRLIT
- 7193	E3E7	STRLIT	met un caractère de fin de chaîne en ENDCHR

- 7187	E3ED	STRLT2	construit un descripteur de chaîne en DSCTMP, FACMO, LO et conduit à PUTNEW
- 7126	E42A	PUTNEW	range DSCTMP dans un descripteur temporaire pointé par FACMO, LO
- 7203	E3DD	STRSPA	conduit à GETSPA et range le pointeur et la longueur en DSCTMP
- 7086	E452	GETSPA	libère de l'espace pour une chaîne en déplaçant FRESPC et FRETOP vers le bas - peut émettre "OUT OF MEMORY" - met à jour DSCTMP
- 6761	E597	CAT	concaténation de la chaîne décrite par (FACMO, LO) et celle pointée par TXTPTR + 1
- 6686	E5E2	MOVSTR	déplace la chaîne pointée par Y,X et A de longueur dans la position pointée par FRESPC ($71,$72)
- 6700	E5D4	MOVINS	déplace la chaîne dont le descripteur est pointé par STRNF1 vers FRESPC
- 6659	E5FD	FRESTR	vérifie que FAC adresse une chaîne et conduit à FREFAC
- 6656	E600	FREFAC	libère l'espace occupé par une chaîne temporaire
- 6603	E635	FRETMS	libère le descripteur temporaire sans libérer la chaîne
- 7036	E484	GARBAG	récupération de l'espace occupé par les chaînes abandonnées et déplacement vers le haut des autres

VARIABLES UTILISEES PAGE ZERO

DEC	*HEX*		
13	D	CHRAC = "	
14	E	ENDCHR = ØØ	
17	11	VALTYP = 1	si chaîne dans FAC
82	52	TEMPPT	pointeurs temporaires
83	53	LASTPT	pointeurs temporaires
	5E,5F	INDEX	pointeurs temporaires
111,112	6F,70	FRETOP	bas de la zone chaîne
113,114	71,72	FRESPC	fin de la zone libre
133,134	85,86	FORPNT	utilisé par COPY pour libérer de l'espace

DEC	HEX		
148,149	94,95	HIGHDS	utilisés par BLTU
150,151	96,97	HIGHTR	pour l'adresse de destination
155,156	9B,9C	LOWTR	utilisé par BLTU
157,158,159	9D,9E,9F	DSCTMP	descripteur de chaîne
160,161	AØ,A1	FACMO,LO	pointeur de descripteur
171,172	AB,AC	STRNG1	pointeur utilisé par MOVINS
173,174	AD,AE	STRNG2	pointeur utilisé par STRLT2

7 - Fonctions graphiques haute résolution

ADRESSES

page 1 : $2ØØØ - $3FF7
page 2 : $4ØØØ - $5FF7

ligne Ø : $2ØØØ - $2Ø27 (40 octets / ligne)
ligne 64 : $2Ø28 - $2Ø4F
ligne 128 : $2Ø5Ø - $2Ø77
ligne 8 : $2Ø8Ø - $2ØA7
ligne 72 : $2ØA8 - $2ØCF
ligne 136 : $2ØDØ - $2ØF7

lignes 16,80,144,24,88,152 : $21ØØ - $21F7
lignes 32,96,160,40,104,168 : $22ØØ - $22F7
lignes 48,112,176,56,120,184 : $23ØØ - $23F7

Soit n le n° de la ligne et $a son adresse des lignes précédemment listées alors :

la n+1e ligne a pour adresse $a + $4ØØ
la n+2e ligne a pour adresse $a + $8ØØ
la n+3e ligne a pour adresse $a + $CØØ
la n+4e ligne a pour adresse $a + $1ØØØ
la n+5e ligne a pour adresse $a + $14ØØ
la n+6e ligne a pour adresse $a + $18ØØ
la n+7e ligne a pour adresse $a + $1CØØ

VARIABLES DE LA PAGE ZERO

DEC	HEX	NOM	ROLE
26,27	$1A,1B	SHAPE L,H	pointeur dans la table des formes
28	$1C	HCOLOR1	dépend de la parité de l'abscisse x du HMASK et HCOLORØ
29	$1D	COUNT H	compteur dans le tracé de ligne
38,39	$26,$27	HBASL,H	adresse du début d'une ligne y

DEC	HEX	NOM	ROLE
48	$3Ø	HMASK	$81, $82, $84, $88 b0, b1, b2, b3 $9Ø, $AØ, $CØ b4, b5, b6
82	$52	DY	incrément de y pour HLINE
83	$53	QDRNT	angle de rotation pour DRAW

VARIABLES DE LA PAGE ZERO UTILISEES PAR LES FONCTIONS GRAPHIQUES

DEC	HEX	NOM	ROLE
224,225	$EØ,$E1	xH,xL	coordonnées écran du point
226	$E2	y	tracé par HPLOT
228	$E4	HCOLORØ	ØØ, 2A, 55, 7F noir, ocre, bleu, blanc 8Ø, AA', D5, FF noir, vert, rouge, blanc
229	$E5	XD7	n° d'octet dans une ligne pour le point d'abscisse x
230	$E6	HPAG	$2Ø : page 1 $4Ø : page 2
231	$E7	SCALE	facteur d'échelle d'une forme
232,233	$E8,$E9	SHPTAB	pointeur du début de la table de formes
234	$EA	CC	compteur de collision

FONCTIONS GRAPHIQUES

DEC	HEX	NOM	ROLE, RESULTAT
-3112	F3D8	HGR2	effacement des pages
-3102	F3E2	HGR	$2Ø→HPAG. $4Ø→HPAG
-3084	F3F4	BKGND	écran d'une couleur uniforme couleur→acc→HCOLOR1
	F6FØ	HCOLOR	couleur→X→HCOLORØ
-3055	F411	HPOSN	x→y, x→xH xL y→acc→y calcul de HBASL,H ; HMASK et XD7 et HCOLOR1

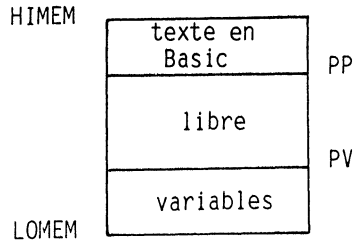
-3111 Affichage page HR 2 sans effacement

FONCTIONS GRAPHIQUES (suite)

DEC	HEX	NOM	ROLE, RESULTAT
-2971	F465	INTX	suivant, incrémentation ou décrémentation de xH, xL et y
-2861	F4D3	INTY	suivant. incrémentation ou décrémentation de y→HBASL,H
-2613	F5CB	IPOS	HBASL,H ; XD7→xH, xL et y
-2985	F457	HPLOT	tracé du point x→y,x et y→A avec le secours de HPOSN et PLOT
-2982	F45A	PLOT	tracé du point défini par HCOLOR1 ; HMASK ; XD7→Y ; HBASL,H suivant les instructions : LDA HCOLOR1 EOR (HBASL),Y AND HMASK EOR (HBASL),Y STA (HBASL),Y
-2758	F53A	HLINE	tracé d'une ligne quelle que soit sa direction du point actuel au point x→X,A et y→Y

Nom	Adresse HEX	Adresse DEC	Rôle
LOMEM	$4A,$4B	74,75	Début des variables
HIMEM	$4C,$4D	76,77	Fin du texte Basic
PP	$CA,$CB	202,203	Début du texte Basic
PV	$CC,$CD	204,205	Fin des variables
PR	$DC,$DD	220,221	N° de la ligne courante
PN	$DE,$DF	222,223	Nom de la dernière variable référencée
PX	$EØ,$E1	224,225	Début de l'instruction courante

Résumé



Exemple d'implantation d'un programme et de ses variables en mémoire vive. (Voir exemple n° 2 en applesoft).

```
>LIST
   10 REM  INTEGER
   15 DIM N$(20)
   20 INPUT "NOM ?",N$
   30 FOR K=1 TO LEN(N$)-2
   40 PRINT N$(K);" ";N$(K,K+1);"  "
      ;
   50 NEXT K
   60 END

>CALL -151

*4A.4B                                    LOMEM

004A- 00 08

*CA.CB

                                          PP
00CA- 9A 95

*4C.4D

                                          HIMEM
004C- 00 96

*959A.95FF

959A-<0D>[0A 00](5D A0 C9
95A0- CE D4 C5 C7 C5 D2 01 <0C>
95A8- [0F 00]4E CE (40 (22 B2 14
95B0- 00 (72 01 <0B>[14 00](53 (28
95B8- CE CF CD A0 BF (29 (26 CE
95C0- (40 01 <13>[1E 00](55 CB (56
95C8- B1 01 00 57 3B CE 40 72
95D0- 13 B2 02 00 01 <20>[28 00]
95D8- (61 CE (40 (2A CB (72 (45 (28
95E0- A0 (29 (45 CE (40 (2A CB (23
95E8- CB (12 B1 01 00 (72 (45 (28
95F0- A0 A0 (29 (47 01 <06>[32 00]
95F8- 59 CB 01 <05>[3C 00](51 01


*CC.CD

                                          PV
00CC- 00 08
```

Texte Basic

(Mots-clés

Ø1 fin de ligne

<>nombre d'octets de la ligne -1

[ ] n° de ligne

Exemple d'implantation d'un programme et de ses variables en mémoire vive.
(suite)

```
>RUN                                      Exécution
NOM ?DUAERB                               du programme

DUAERB DU  UAERB UA  AERB AE  ERB ER


>CALL -151

*CC.CD                                    PV

00CC- 20 08


*800.81F
                                          Variables
0800-.CE 40 00,1A<08>/C4 D5 C1
0808- C5 D2 C2/1E 53 4F 46 54             - N$, <pointeur pro-
0810- 00 21 08 14 00 84 22 4E             chaine variable>
0818- 4F 4D.CB 00 20<08>/05 00/           - valeur (20 carac-
                                          tères)
                                          - K,<>.valeur
```

Les mots-clés par ordre croissant des codes et les adresses des sous-programmes correspondants dans l'interpréteur Integer Basic. Codes $00 à $15.

Mots-clés	Code HEX	Code DEC	Adresse HEX	Adresse DEC-65536
[HIMEM:]	$00	0		
Fin de ligne	$01	1		
-	$02	2		
:	$03	3		
LOAD	$04	4	$F0DF	-3873
SAVE	$05	5	$F140	-3776
CON	$06	6	$F30A	-3318
RUN (n°de ligne)	$07	7	$EFF2	-4110
RUN	$08	8	$EFEC	-4116
DEL	$09	9	$E36F	-7313
, (pour DEL)	$0A	10		
NEW	$0B	11	$E5AD	-6739
CLR	$0C	12	$E5B7	-6729
AUTO	$0D	13	$E7E2	-6174
, (pour AUTO)	$0E	14		
MAN	$0F	15	$EE54	-4524
HIMEM:	$10	16	$F04D	-4019
LOMEM:	$11	17	$F0C9	-3895
+	$12	18	$E787	-6267
- (Op. num.)	$13	19	$E782	-6270
* (Op. num.)	$14	20	$E222	-7646
/ (Op. num.)	$15	21	$EF10	-4336

Les mots-clés (suite). Codes $16 à $33.

Mots-clés	Code HEX	Code DEC	Adresse HEX	Adresse DEC-65536
=	$16	22		
#	$17	23		
>=	$18	24		
>	$19	25		
<= (Op. logiques pour num.)	$1A	26		
<>	$1B	27		
<	$1C	28		
AND	$1D	29		
OR	$1E	30		
MOD	$1F	31	$E27A	-7558
^	$20	32	$F371	-3215
	$21	33		
((pour DIM)	$22	34		
, (pour DIM)	$23	35		
THEN (n°de ligne)	$24	36		
THEN (inst)	$25	37		
, (chaîne)	$26	38		
, (nombre)	$27	39		
" (début)	$28	40		
" (fin)	$29	41		
(var.$	$2A	42		
indicée	$2B	43		
	$2C	44		
(var. indicée	$2D	45		
PEEK	$2E	46	$EEF6	-4362
RND	$2F	47	$EF4E	-4274
SGN	$3D	48	$E75C	-6308
ABS	$31	49	$E74A	-6326
PDL	$32	50	$F33B	-3269
	$33	51		

Les mots-clés (suite). Codes $34 à $51.

Mots-clés	Code HEX	Code DEC	Adresse HEX	Adresse DEC-65536
(pour DIM	$34	52		
+ (signe)	$35	53		
- (signe)	$36	54		
NOT	$37	55	$E736	-6346
(	$38	56		
= comparaison	$39	57		
# de chaînes	$3A	58		
LEN(	$3B	59	$EE22	-4574
ASC(	$3C	60	$F31D	-3299
SCRN(	$3D	61	$E28A	-7542
, dans SCRN	$3E	62		
(	$3F	63		
$ (chaîne)	$40	64		
	$41	65		
(	$42	66		
,	$43	67		
,	$44	68		
;	$45	69		
;	$46	70		
;	$47	71		
,	$48	72		
,	$49	73		
,	$4A	74		
TEXT	$4B	75		
GR	$4C	76		
CALL	$4D	77	$EEAØ	-4448
DIM (chaînes)	$4E	78	$E13Ø	-7888
DIM (nombres)	$4F	79	$EF1E	-4322
TAB	$5Ø	80	$E7A4	-6236
END	$51	81		

Les mots-clés (suite). Codes $52 à $6F.

Mots-clés	Code HEX	Code DEC	Adresse HEX	Adresse DEC-65536
INPUT (chaînes)	$52	82	$E171	-7823
INPUT (message)	$53	83		
INPUT (nombre)	$54	84	$EBAA	-5206
FOR	$55	85	$E93A	-5830
= (FOR/NEXT)	$56	86		
TO (FOR)	$57	87	$E950	-5808
STEP	$58	88	$F279	-3463
NEXT	$59	89		
, (NEXT)	$5A	90		
RETURN	$5B	91	$E8A5	-5979
GOSUB	$5C	92	$E83C	-6084
REM	$5D	93		
LET	$5E	94		
GOTO	$5F	95	$E85B	-6053
IF	$6Ø	96	$E828	-6104
PRINT (chaîne)	$61	97	$EEØ3	-4605
PRINT (nombre)	$62	98		
PRINT	$63	99		
POKE	$64	100		
, (POKE)	$65	101		
COLOR=	$66	102	$EE4E	-4530
PLOT	$67	103	$EE3F	-4545
, (PLOT)	$68	104		
HLIN	$69	105	$EEBØ	-4432
, (HLIN)	$6A	106		
AT (HLIN)	$6B	107		
VLIN	$6C	108	$EEC6	-4410
, (VLIN)	$6D	109		
AT (VLIN)	$6E	110		
VTAB	$6F	111	$EE57	-4521

Les mots-clés (suite). Codes de $7Ø à $7F.

Mots-clés	Code HEX	Code DEC	Adresse HEX	Adresse DEC-65536
= (chaînes)	$7Ø	112		
= (nombre)	$71	113		
)	$72	114		
	$73	115		
LIST	$74	116		
, (LIST)	$75	117		
LIST	$76	118		
POP	$77	119	$F167	-3737
NODSP (chaîne)	$78	120		
NODSP (nombre)	$79	121		
NOTRACE	$7A	122	$F176	-3722
DSP (chaîne)	$7B	123	$F2EØ	-3360
DSP (nombre)	$7C	124		
TRACE	$7D	125	$F171	-3727
PR #	$7E	126	$F3C9	-3127
IN #	$7F	127	$F41A	-3046

BOOT : Mise en place du SED (et démarrage à froid)

Programmes	Localisation	Occupation	Rôle
1 - BOOT Ø	PROM carte-contrôleur $C6ØØ	256 octets	charge BOOT 1 en MEV
2 - BOOT 1	DISQUETTE : piste Ø, secteur Ø MEV : $8ØØ-$9ØØ	1 secteur 256 octets	charge BOOT 2 et lui-même
3 - BOOT 2	DISQUETTE : piste Ø, s. 1 à 9 MASTER / SLAVE (48 K) $37ØØ-$4ØØØ $B7ØØ-$CØØØ	9 secteurs 2304 octets	contient RWTS charge le SED et éventuellement le translateur
BOOT 1	DISQUETTE : piste Ø, secteur Ø MASTER / SLAVE (48 K) $36ØØ-$36FF $B6ØØ-$B6FF	1 secteur 256 octets	version de BOOT 1 disponible pour initialiser une disquette vierge
4 - SED	DISQUETTE : piste 2, sect. 4 à Ø piste 1, sect. F à Ø piste Ø, sect. F à C MASTER / SLAVE (48 K) $1DØØ-$36ØØ $9DØØ-$B6ØØ	25 secteurs 6400 octets	système d'exploitation des commandes et de gestion de l'espace sur disquette
Translateur (relocator)	DISQUETTE MASTER piste Ø, sect. A et B MEV : $1BØØ-$1DØØ (n'existe pas dans une disquette) SLAVE	2 secteurs 512 octets	réinstalle le SED à sa place définitive $9DØØ-$CØØØ (48 K)

Organisation d'une disquette

Quelle que soit la version du SED, une disquette est constituée de 35 pistes et les données sont transmises par secteur de 256 octets.

Version 3.2	*Version 3.3*
13 secteurs/piste 455 secteurs/disquette dont 403 utiles	16 secteurs/piste 560 secteurs/disquette dont 496 utiles
soit 103168 octets utiles	soit 126976 octets utiles

Occupation des pistes et secteurs

Pistes Ø, 1, 2 : SED (système d'exploitation)
Piste $11, secteur Ø : VTOC (occupation)
Piste $11, secteurs $F à $1 : DIRECTORY (catalog)

Pistes $12 à $22 et $16 à $3 : programmes et fichier utilisateurs

(Le fichier le plus long enregistrable sur une disquette a environ 126000 octets).

Le "directory" peut gérer un maximum de 105 références.

Une référence est un ensemble de 35 octets comprenant :
- l'adresse de la liste des secteurs occupés (n° de piste, n° de secteur) par le fichier référencé
- le type de fichier A, I, T, B, verrouillé ou non
- le nom du fichier (30 caractères)
- la longueur en nombre de secteurs occupés (2 octets)

Commandes par ordre d'apparition dans la table des commandes

COMMANDES SED

Index		ADRESSE D'ENTREE HEX	DEC
$00 0	INIT	$A54F	42319
$01 1	LOAD	$A413	42003
$02 2	SAVE	$A397	41879
$03 3	RUN	$A4D1	42193
$04 4	CHAIN	$A4F0	42224
$05 5	DELETE	$A263	41571
$06 6	LOCK	$A271	41585
$07 7	UNLOCK	$A275	41589
$08 8	CLOSE	$A2EA	41706
$09 9	READ	$A51B	42267
$0A 10	EXEC	$A5C6	42438
$0B 11	WRITE	$A510	42256
$0C 12	POSITION	$A5DD	42461
$0D 13	OPEN	$A2A3	41635
$0E 14	APPEND	$A298	41624
$0F 15	RENAME	$A281	41601
$10 16	CATALOG	$A56E	42350
$11 17	MON	$A233	41523
$12 18	NOMON	$A23D	41533
$13 19	PR#	$A229	41513
$14 20	IN#	$A22E	41518
$15 21	MAXFILES	$A251	41553
$16 22	FP	$A57A	42362
$17 23	INT	$A59E	42398
$18 24	BSAVE	$A331	41777
$19 25	BLOAD	$A35D	41821
$1A 26	BRUN	$A38E	41870
$1B 27	VERIFY	$A27D	41597

Localisation	*S.E.D.*	*point d'entrée*
$B6ØØ-B6FF	RWTS	$B7B5
$AAC9-B5FF	gestion des commandes	$AAFD
$9DØØ-AAC8	programme principal	$9DØØ
$96ØØ-9CFF	3 buffers de 595 octets	

Configuration 48K MEV

Sous-programme d'accès à 1 secteur : RWTS

Table des paramètres : IOB

L'adresse de IOB est à charger dans les registres A (poids forts) et Y (poids faibles) avant d'appeler RWTS :

```
Exemple :  LDA  #$1Ø
           LDY  #$ØØ
           JSR  $3D9
           RTS

* 1ØØØ : Ø1 6Ø Ø1 ØØ 11 ØC 11 1Ø
* 1Ø08 : ØØ Ø9 ØØ ØØ Ø1
* 1Ø11 : ØØ Ø1 EF D8
```

IOB : octet n° 4 : n° de la piste ($11)
octet n° 5 : n° du secteur ($ØC)
octets 6, 7 : adresse de la DCT ($1Ø11)
octets 8, 9 : adresse de la zone de transfert en MEV ($9ØØ)
octet C : code de la commande
ØØ positionnement
Ø1 lecture
Ø2 écriture
Ø3 formatage

DCT constantes : ØØ Ø1 EF D8 du périphérique

Adresse Hex	*Contenu*	*Rôle*
3DØ	JMP $9DBF	redémarrage à chaud
3D3	JMP $9D84	démarrage à froid
3D6	JMP $AAFD	gestion des commandes
3D9	JMP $B7B5	lecture-écriture d'1 secteur (RWTS)
3DC	LDA $9DØF LDY $9DØE RTS	recherche de l'adresse de la liste des paramètres pour la gestion des commandes
3E3	LDA $AAC2 LDY $AAC1 RTS	recherche de l'adresse de la table IOB des paramètres de RWTS
3EA	JMP $A851	pour remplacer les vecteurs d'E/S $38, $39 et $36, $37 avec les pointeurs du SED
3EF	JMP $FA59	en cas de BRK en MONITOR
3F3, 3F2	$9DBF	adresse de renvoi en cas de 'Reset' (SOFTEV)
3F4	$38	PWRUP = ($3F3) ⊕ $A5
3F5	JMP $FF58	en cas de &
3F8	JMP $FF65	en cas de 'CTRL Y'
3FB	JMP $FF65	si interruption non masquée
3FE	$FF65	si interruption

Adresses diverses du SED en MEV

- Fichier binaire chargé en MEV par BLOAD
  adresse en $AA72, AA73
  longueur en $AA6Ø, AA61
- Programme exécuté au démarrage à froid
  nom : piste 1, secteur 9, octet $75...
  type * 9E42 : 34 BRUN
  * 9E42 : 14 EXEC
- Retrait de la pause durant CATALOG
  AE34 : 6Ø

SYSTEM MASTER 3.3

B	FID	- recopie de programmes ou fichiers - taux d'occupation de la disquette
A I	COPY	- recopie d'une disquette entière (avec 1 ou 2 lecteurs)
B	BOOT13	- démarrage avec un SED version 3.2 (13 secteurs par disquette)
B	MUFFIN	- conversion de fichiers ou programmes écrits sous 3.2 en fichiers ou programmes sous 3.3
B	MASTER CREATE	- création d'une disquette MASTER à partir d'une disquette déjà initialisée et utilisable mais de type SLAVE. - possibilité de changer le nom du programme qui s'exécute au démarrage.

DAK IN5 programming aids 3.3

the Patcher	visualisation et modification de n'importe quel secteur
the Peeker	lecture d'un fichier

Implantation des fichiers et programmes sur une disquette

Exemple :

1 - extrait du CATALOG

```
DISK VOLUME 254
(T)(n)  (NOM)
*A 006 HELLO
*I 018 ANIMALS
*T 003 APPLE PROMS
*I 006 APPLESOFT
*I 026 APPLEVISION
*I 017 BIORHYTHM
*B 010 BOOT13
```

* verrouillé en écriture ; (T)A,I,T, B type
n : nombre de secteurs occupés
NOM du fichier

2 - extrait du DIRECTORY ou répertoire de la disquette

Piste $11 Secteur $0F

```
00- 00 <11 0E> 00 00 00 00 00  ........
08- 00 00 00[13 0F](82)/C8 C5  ......HE
10- CC CC CF A0 A0 A0 A0 A0  LLO
18- A0 A0 A0 A0 A0 A0 A0 A0
20- A0 A0 A0 A0 A0 A0 A0 A0
28- A0 A0 A0 A0/[06 00][14 0F]      ....
30- (81)/C1 CE C9 CD C1 CC D3  .ANIMALS
38- A0 A0 A0 A0 A0 A0 A0 A0
40- A0 A0 A0 A0 A0 A0 A0 A0
48- A0 A0 A0 A0 A0 A0 A0/[12          .
50- 00[15 0F](80)/C1 D0 D0 CC  ....APPL
58- C5 A0 D0 D2 CF CD D3 A0  E PROMS
60- A0 A0 A0 A0 A0 A0 A0 A0
68- A0 A0 A0 A0 A0 A0 A0 A0
70- A0 A0/[03 00][16 0F](81)/C1    .....A
78- D0 D0 CC C5 D3 CF C6 D4  PPLESOFT
80- A0 A0 A0 A0 A0 A0 A0 A0
88- A0 A0 A0 A0 A0 A0 A0 A0
90- A0 A0 A0 A0 A0/[06 00][17        ...
98- 0F](81)/C1 D0 D0 CC C5 D6  ..APPLEV
A0- C9 D3 C9 CF CE A0 A0 A0  ISION
A8- A0 A0 A0 A0 A0 A0 A0 A0
B0- A0 A0 A0 A0 A0 A0 A0 A0/
B8- [1A 00][18 0F](81)/C2 C9 CF  .....BIO
C0- D2 C8 D9 D4 C8 CD A0 A0  RHYTHM
C8- A0 A0 A0 A0 A0 A0 A0 A0
D0- A0 A0 A0 A0 A0 A0 A0 A0
D8- A0 A0 A0/[11 00][19 0F](84)     .....
E0- /C2 CF CF D4 B1 B3 A0 A0  BOOT13
E8- A0 A0 A0 A0 A0 A0 A0 A0
F0- A0 A0 A0 A0 A0 A0 A0 A0
F8- A0 A0 A0 A0 A0 A0[0A 00          ..
```

<> piste, secteur de la suite du répertoire

[ ] piste, secteur de la liste des secteurs occupés par ce fichier

◯ type de fichier
82 Applesoft
81 Integer
80 fichier T
84 binaire

/ / nom du fichier complété à 30 caractères par des espaces codés AØ

└── longueur du fichier en nombre de secteurs

3 - extrait du VTOC ou table d'occupation des secteurs

Piste $11 Secteur Ø

```
00- 04 [11 0F]03 00 00 FE 00
08- 00 00 00 00 00 00 00 00
10- 00 00 00 00 00 00 00 00
18- 00 00 00 00 00 00 00 00
20- 00 00 00 00 00 00 00 [7A]
28- 00 00 00 00 00 00 00 00
30- 0D FF 00 00 [23] 10 00 01
38- '00 00 00 00"00 00 00 00'
40- '00 00 00 00"00 00 00 00'
48- '00 00 00 00"00 00 00 00'
50- '00 00 00 00"00 00 00 00'
58- '00 00 00 00"00 0F 00 00'
60- 'FF FF 00 00"00 00 00 00'
68- '00 7F 00 00"01 FF 00 00'
70- '00 00 00 00"00 00 00 00'
78- '00 00 00 00"00 00 00 00'
80- 'FF E0 00 00"00 00 00 00'
88- '00 00 00 00"00 00 00 00'
90- '00 00 00 00"00 00 00 00'
98- '00 00 00 00"00 00 00 00'
A0- '00 00 00 00"00 03 00 00'
A8- '00 00 00 00"00 00 00 00'
B0- '00 00 00 00"00 00 00 00'
B8- '00 00 00 00"00 00 00 00'
C0- '00 00 00 00'00 00 00 00
```

[ ] piste secteur du 1er secteur du répertoire

version du SED (3.3)
n° de volume (254

7A = 122 secteurs max. dans une liste des adresses (piste secteur) des secteurs occupés par un fichier

23 = 35 pistes maximum par disquette

10 = 16 secteurs par piste

ØØ Ø1 = 256 octets par secteur

' ' secteurs occupés dans chaque piste successive

FEDCBA98	76543210	

si un secteur est libre le bit correspondant est à 1

FF FF ØØØØ la piste n° 1Ø a tous ses secteurs libres

ØØ 7F ØØØØ la piste n° 12 a ses secteurs F,E,D,C,A,9,8,7 occu occupés

4 - extrait d'une liste d'adresses (piste, secteur) des secteurs occupés par un fichier

Exemple : HELLO, piste $13, secteur $F

```
00- 00 00 00 00 00 00 00 00
08- 00 00 00 00 13 0E 13 0D
10- 13 0C 13 0B 13 0A 00 00
18- 00 00 00 00 00 00 00 00
```

└── adresses des secteurs successivement occupés par HELLO

5 - extrait du code d'un programme enregistré sur disquette

Exemple : HELLO, écrit en langage APPLESOFT

piste $13 secteur $E

```
00-[71 04]/19 08 0A 00 B2 20 ......2
08- 20 2D 2D 20 44 4F 53 20  -- DOS
10- 33 2E 33 20 48 45 4C 4C 3.3 HELL
18- 4F 00/20 08 14 00 B2 20 O. ...2
20- 00/28 08 1E 00 89 3A BA .(....::
28- 00/2E 08 28 00 97 00/59 ...(...Y
30- 08 32 00 BA 22 44 4F 53 .2.:"DOS
38- 20 56 45 52 53 49 4F 4E  VERSION
40- 20 33 2E 33 20 20 20 20  3.3
48- 20 20 20 20 20 20 20 20
50- 30 38 2F 32 35 2F 38 30 08/25/80
58- 22 00/8B 08 3C 00 BA 3A "...<.::
60- BA 22 41 50 50 4C 45 20 :"APPLE
68- 49 49 20 50 4C 55 53 20 II PLUS
70- 4F 52 20 52 4F 4D 43 41 OR ROMCA
78- 52 44 20 20 20 53 59 53 RD   SYS
```

[pf pF] longueur du programme en nombre d'octets de MEV occupés

pf : poids faibles
pF : poids forts

/ / instructions du programme

6 - extrait du code d'un programme en binaire

Exemple : BOOT13, piste $19, secteur $E

```
00-[00 17]E0 08/20 E3 03 84
08- 00 85 01 A0 01 B1 00 8D
10- 90 17 C8 B1 00 8D 91 17
18- 20 58 FC A0 FF C8 B9 96
20- 17 08 09 80 20 ED FD 28
28- 10 F3 A9 BF 85 33 20 6A
30- FD AD 00 02 C9 8D F0 0F
38- C9 B1 90 DC C9 B8 B0 D8
40- 0A 0A 0A 0A 8D 82 17 A9
48- 17 A0 81 20 00 1D B0 F7
50- AD FE 16 8D 8A 17 85 13
58- E6 13 AD FF 16 4A 4A 4A
60- 85 10 A9 17 A0 81 20 00
68- 1D B0 F7 EE 8A 17 EE 86
70- 17 AD 86 17 C5 10 F0 EA
78- 90 E8 AD 82 17 AA A9 00
```

[pf pF] longueur du programme en nombre d'octets en MEV

pf pF adresse d'implantation en MEV du début du programme écrit en langage machine

' instructions du programme en langage machine

BOOT13 a été sauvegardé sur disquette par la commande

BSAVE BOOT13, A$8F0, L$1700

Implantation des fichiers sur disquette

7 - extrait du contenu d'un fichier de type T

Exemple : APPLE PROMS, piste $15, secteur $E

```
00-(B7 B5(8D C4 C5 CC A0 B1 75.DEL 1
08-)B0 B0 B0 AC B1 B2 B5 B0 000,1250
10-(8D D3 C1 D6 C5 A0 D2 C1 .SAVE RA
18-)CE C4 CF CD(8D C8 CF CD NDOM.HOM
20-)C5(8D D2 D5 CE(8D 00 00 E.RUN...
28-(D0 C1 D2 C1 CC CC C5 CC PARALLEL
30-)A0 D0 D2 C9 CE D4 AC) B2  PRINT,2
38-)B5 B6 AC) B8 AC) B5 B0 B0 56,8,500
40-)8D 00 00 00 00 00 00 00 ........
48-)00 00 00 00 00 00 00 00 ........
50-(C3 CF CD CD D5 CE C9 C3 COMMUNIC
58-)C1 D4 C9 CF CE D3 AC) B2 ATIONS,2
60-)B5 B6 AC) B8 AC) B1 B2 B5 56,8,125
68-)B0 8D 00 00 00 00 00 00 0.......
70-)00 00 00 00 00 00 00 00 ........
78- A8 CE CF D4 A0 C1 D6 C1 (NOT AVA
```

(8D séparateur 'Return'

AC) séparateur ","

enregistrement de longueur fixe

00 aucune donnée écrite (END OF DATA)

Ce fichier a été ouvert par la commande

OPEN APPLE PROMS, L40

définissant la longueur de chaque enregistrement à 40 caractères

A partir de l'enregistrement n° 1 l'instruction de lecture des champs est

INPUT N$, BL, BW, ST

et d'écriture

PRINT N$;",";BL;",";BW;",";ST

8 - extrait du contenu d'un fichier séquentiel de type T

```
00- B3 (8D D0 D2 C5 CD C9 C5 3.PREMIE
08- D2 (8D D3 C5 C3 CF CE C4 R.SECOND
10- (8D D4 D2 CF C9 D3 C9 C5 .TROISIE
18- CD C5 8D 00 00 00 00 00 ME......
20- 00 00 00 00 00 00 00 00 ........
```

(8D le séparateur 'Return' termine chaque enregistrement dont la longueur est libre.

00 aucune donnée inscrite (END OF DATA)

Achevé d'imprimer en décembre 1984
sur les presses de l'imprimerie Laballery et Cie
58500 Clamecy
Dépôt légal : décembre 1984
N° d'impression : 412026
N° d'édition : 86595-73-5
ISBN : 2-86595-073-5