

GUIDE DE L'APPLE

L'APPLE STANDARD

Benoît de Merly

« Voici le livre que nous attendions : complet, clair et pratique, il devrait rapidement devenir le compagnon familier et indispensable de tous les utilisateurs de l'Apple.

Nous sommes particulièrement heureux de saluer son arrivée au moment où nous sortons notre nouvel Apple II : le Apple II e. »

Jean-Louis Gassée
Directeur Général de Apple France

Ce livre s'adresse à toute personne qui utilise un ordinateur personnel Apple, ou qui désire approfondir ses connaissances de l'Apple.

Chacun y trouvera l'information pratique dont il a besoin, quelle que soit l'application envisagée : gestion, calcul scientifique, jeux, graphiques, acquisition de données, contrôle de processus...

GUIDE APPLE VOL. 1 80F

L'auteur

L'auteur, Ingénieur civil des Ponts et Chaussées, titulaire d'une licence en informatique, a eu l'idée de ce livre en utilisant lui-même un Apple : il s'est donc posé les questions que l'utilisateur sera amené à se poser s'il veut tirer de son ordinateur toutes ses ressources, et le maîtriser parfaitement.

GUIDE DE L'APPLE

TOME 1	TOME 2	TOME 3
L'APPLE STANDARD	LES EXTENSIONS	LES APPLICATIONS

GUIDE DE L'APPLE

TOME 1



L'APPLE STANDARD

Benoît de Merly



Edimicro

L'APPLE STANDARD

par

Benoît de MERLY

© F.D.S./Edimicro 1983
Première édition

Imprimé en France. Droits mondiaux réservés.

« La loi du 11 mars 1957 n'autorisant, aux termes des alinéas 2 et 3 de l'article 41, d'une part, que les « copies ou reproductions strictement réservées à l'usage privé du copiste et non destinées à une utilisation collective » et, d'autre part, que les analyses et les courtes citations dans un but d'exemple et d'illustration, « toute représentation ou reproduction intégrale, ou partielle, faite sans le consentement de l'auteur ou de ses ayants droit ou ayants cause, est illicite » (alinéa 1^{er} de l'article 40) ».
« Cette représentation ou reproduction, par quelque procédé que ce soit, constituerait une contrefaçon sanctionnée par les articles 425 et suivants du Code pénal ».

Tome 1 ISBN : 2-904457-00-3

EDIMICRO

DÉPARTEMENT ÉDITIONS DE F.D.S. SARL

10, rue Henri-Pape, 75013 Paris. Tél. : 588-76-53

 **Edimicro**

Préface

Lorsqu'ils lancèrent sur le marché cette petite boîte de couleur crème qu'ils avaient baptisée APPLE II, Steve JOBS et ses collègues étaient loin de se douter de l'avenir qui attendait le premier microordinateur personnel.

Bien sûr, quelques amis de leur entourage s'étaient montrés enthousiastes devant les possibilités de l'appareil. Mais à la fin de 1977, qui pouvait prévoir un tel succès ? Quelques années plus tard, un million d'exemplaires étaient entrés dans les bureaux, les usines, les ateliers, les écoles et les foyers du monde entier.

Quand le mot microordinateur est apparu, les plus clairvoyants imaginaient déjà les « gros » ordinateurs réduits à la taille micro, mais personne n'aurait parié que vous, moi, nos enfants, allions très rapidement pouvoir l'utiliser pour jouer, apprendre et travailler.

Steve JOBS avait une certaine idée de ce que serait son APPLE II. Les utilisateurs ont démultiplié cette idée : en fouinant dans les instructions, en essayant différentes commandes et divers modes d'utilisation, les « users » ont en fait inventé la microinformatique.

Pour moi, qui dois présenter souvent cette nouvelle discipline à un public de télévision très vaste et très peu habitué à ces machines, la différence entre microinformatique et Informatique (notez la majuscule !) est au moins aussi grande qu'entre une Formule 1 et la voiture de Monsieur-tout-le-monde. Vous pouvez bien sûr, avec de l'entraînement et beaucoup d'argent, piloter une voiture de course sur circuit rapide, ce qui équivaut dans notre domaine à maîtriser les grosses machines des centres de calcul.

Si, plus simplement, vous aimez votre métier, si vous voulez améliorer votre vie quotidienne au travail et à la maison, sachez que la microinformatique est à votre portée, surtout grâce à des outils comme celui que vous tenez en main.

Ce guide va se révéler très vite indispensable, dans la recherche des meilleures possibilités de votre APPLE. Vous pouvez, grâce aux différents niveaux de compréhension, n'aborder en première lecture que les chapitres simples, les chapitres sur le BASIC par exemple. Plus vous en saurez, plus vous aurez envie d'aller de l'avant.

Vous deviendrez celui ou celle qui en sait un peu plus, qui comprend mieux et donc maîtrise mieux sa machine. Vos possibilités en seront décuplées, et le bénéfice de votre investissement vous apparaîtra pleinement : c'est précisément le but de cet ouvrage.

Georges LECLERE
mars 1983

Table des matières

TOME 1 : L'APPLE II STANDARD

CHAPITRE 1. — Présentation générale.....	1
1.1 Introduction.....	1
1.2 Unité centrale.....	4
1.3 Cartes d'extension classiques.....	6
1.4 Sortie télévision et variantes.....	8
1.5 Autres extensions possibles.....	9
CHAPITRE 2. — Le basic APPLESOFT.....	11
2.1 Introduction.....	11
2.2 Description syntaxique de l'APPLESOFT.....	11
2.2.1 Variables simples.....	11
2.2.2 Variables tableaux.....	12
2.2.3 Opérateurs numériques.....	12
2.2.4 Opérateurs relationnels.....	13
2.2.5 Opérateurs logiques.....	13
2.2.6 Instructions de contrôle.....	14
2.2.7 Accès à la mémoire et appel de sous-programmes assembleur.....	15
2.2.8 Edition de texte sur l'écran.....	16
2.2.9 Instructions d'entrée-sortie sur l'écran.....	18
2.2.10 Création d'une fenêtre de texte sur l'écran.....	20
2.2.11 Instructions d'entrée-sortie avec les périphériques..	21
2.2.12 Traitement des erreurs.....	22
2.2.13 Commandes diverses.....	24
2.2.14 Fonctions traitant les chaînes de caractères.....	24
2.2.15 Fonctions numériques.....	25

4.2.3	Tableau des instructions du 6502	81
4.2.4	Techniques de base de programmation en assembleur 6502.....	87
●	Branchements	87
●	Opérations logiques	88
●	Décalages et rotations	88
●	Opérations arithmétiques.....	90
●	Sous-programmes	91
●	Utilisation de la pile	91
●	Entrées-sorties	91
4.2.5	Conseils au lecteur débutant	92
4.3	Le moniteur	93
4.3.1	Présentation	93
4.3.2	Accès au moniteur à partir du Basic.....	93
●	Accès au moniteur	93
●	Sortie du moniteur	94
4.3.3	Les commandes du moniteur.....	95
●	Examen et modification de la mémoire	95
	Pas à pas	95
	Par zone mémoire	96
●	Recopie et comparaison de zones mémoires	99
	Recopie	99
	Comparaison	100
	Initialisation.....	101
●	Sauvegarde et restauration d'une zone mémoire.....	102
	Cassette	102
	Disquette	104
	Vérification	105
●	Mise au point de programmes en langage machine	107
●	Examen et modification des registres	107
●	Création et exécution d'un programme	108
●	Désassemblage d'un programme.....	109
●	Le miniassembleur	110
	* Accès.....	110
	* Accès aux commandes du moniteur.....	111

	* Sortie	111
	* Instructions	111
●	Mise au point.....	113
	* Exécution d'un programme pas à pas.....	113
	* Obtention d'une trace	114
	* Exemple	114
●	Modification des entrées-sorties	118
●	Création d'une commande utilisateur.....	119
4.3.4	Adresses mémoire et sous-programmes du moniteur de l'APPLE II +.....	120
●	Description de la carte mémoire	120
●	Présentation générale	120
●	Utilisation de la page zéro	121
●	Pages un à trois	126
●	Pages de texte et graphique résolution	127
●	Entrées-sorties	129
	* Adresses spéciales	129
	* Cartes d'extension	134
●	Sous-programmes d'entrées-sorties	135
●	Clavier	135
●	Ecran	139
	* Affichage de texte dans la fenêtre	139
	* Gestion des modes de l'écran.....	140
	* Manipulations des données	140
	Fonctions ESC	141
	Mouvements du curseur.....	142
	Scrolling	142
	* Affichage de texte à l'extérieur de la fenêtre.....	143
	* Gestion du mode graphique basse résolution	143
●	Haut-parleur	145
●	Cassette	145
●	Manettes de jeux et E/S analogiques	147
●	Attente d'un délai	147
●	Gestion des interruptions.....	148
●	Généralités	148
●	NMI	149
●	RESET	150
●	IRQ/BRK.....	151

XIV TABLE DES MATIÈRES

- Utilisation des commandes du moniteur comme sous-programmes 151

ANNEXES

A1	Codes ASCII	159
A2	Codes ASCII saisissables au clavier	161
A3	Codes ASCII affichables à l'écran	162
A4	Tableau des sous-programmes assembleur classés par adresse croissante	163
A5	Tableau des adresses utiles dans la zone des E/S par ordre croissant	166

CHAPITRE 1

Présentation de l'APPLE II

1.1 INTRODUCTION

L'APPLE II est un microordinateur utilisable par les amateurs et les professionnels. Il se présente dans un boîtier compact, facile à transporter. Le clavier, incorporé sur la face avant du boîtier, possède les caractéristiques suivantes :

- 63 touches permettant de générer les 128 codes ASCII,
- caractères minuscules et majuscules,
- agencements AZERTY (machine à écrire française) ou QWERTY sélectionnables par un interrupteur.

La sortie vidéo est présente à l'arrière de l'appareil et délivre des signaux selon le code PAL.

La photo A montre un APPLE II e muni d'une unité de disquettes et d'un moniteur vidéo.

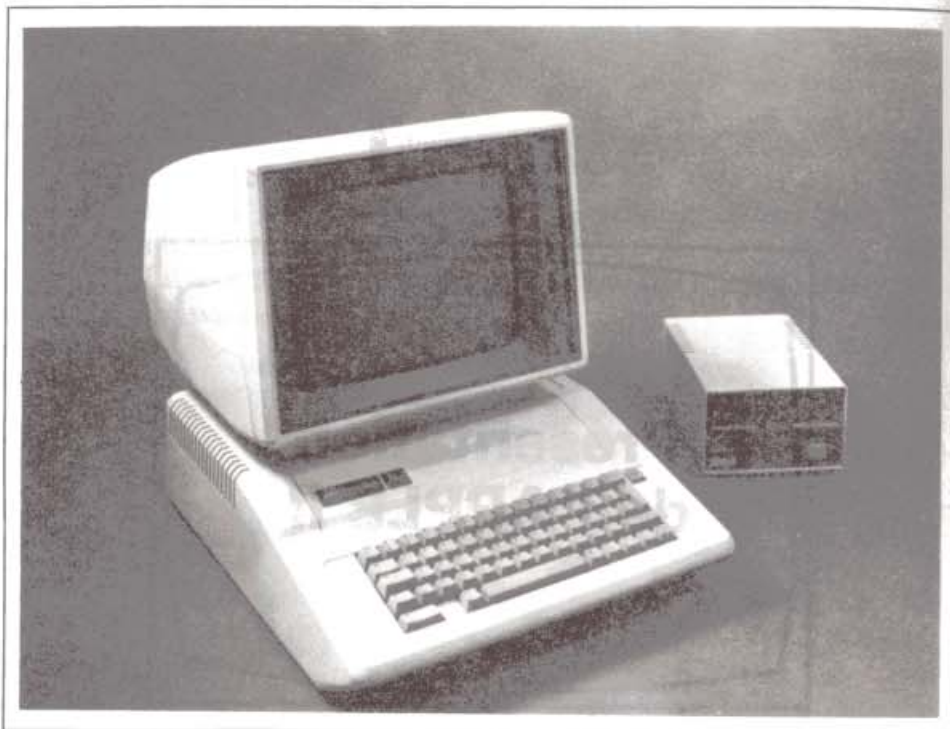


Photo A : Système APPLE II e

Mais de quoi se compose l'intérieur de la machine ? Comme le montre la photo B, appuyez vers le haut au centre de la face arrière pour ôter le couvercle.

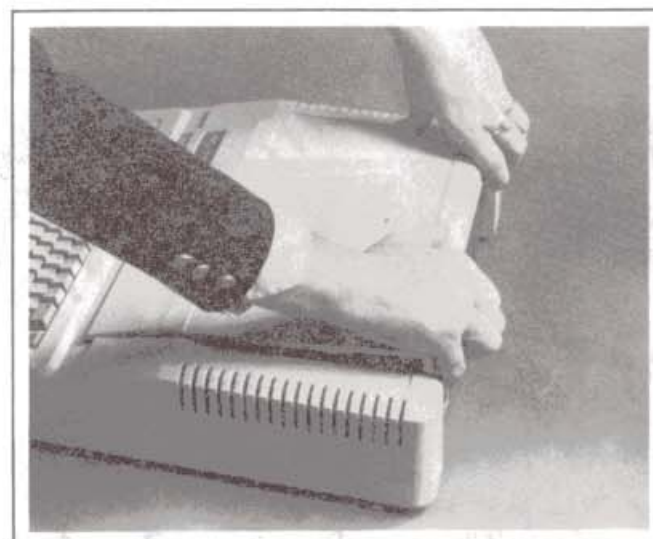


Photo B : Enlèvement du couvercle

- Vous pouvez alors distinguer les éléments suivants sur la photo C :
- le bloc alimentation autonome qui génère les tensions continues ± 5 V, ± 12 V,
 - la carte principale qui contient le microprocesseur 6502 et sur laquelle nous reviendrons au paragraphe suivant,
 - le haut-parleur permettant de générer des sons de durée et de fréquence variables, situé sous le clavier (cf. chap. 3),
 - l'interface vidéo et l'étage PAL,
 - l'interface cassette,
 - une prise de connexion de manettes de jeux ou d'entrées-sorties analogiques,
 - sept connecteurs d'extension permettant d'insérer des cartes ajoutant diverses fonctions à l'APPLE II,
 - le panneau arrière conçu pour des branchements et des débranchements rapides,
 - un connecteur auxiliaire destiné aux cartes vidéo.

La carte de gestion du clavier de l'APPLE II + a été intégrée dans la carte mère de l'APPLE II e, qui comporte en outre un connecteur pour clavier numérique séparé.

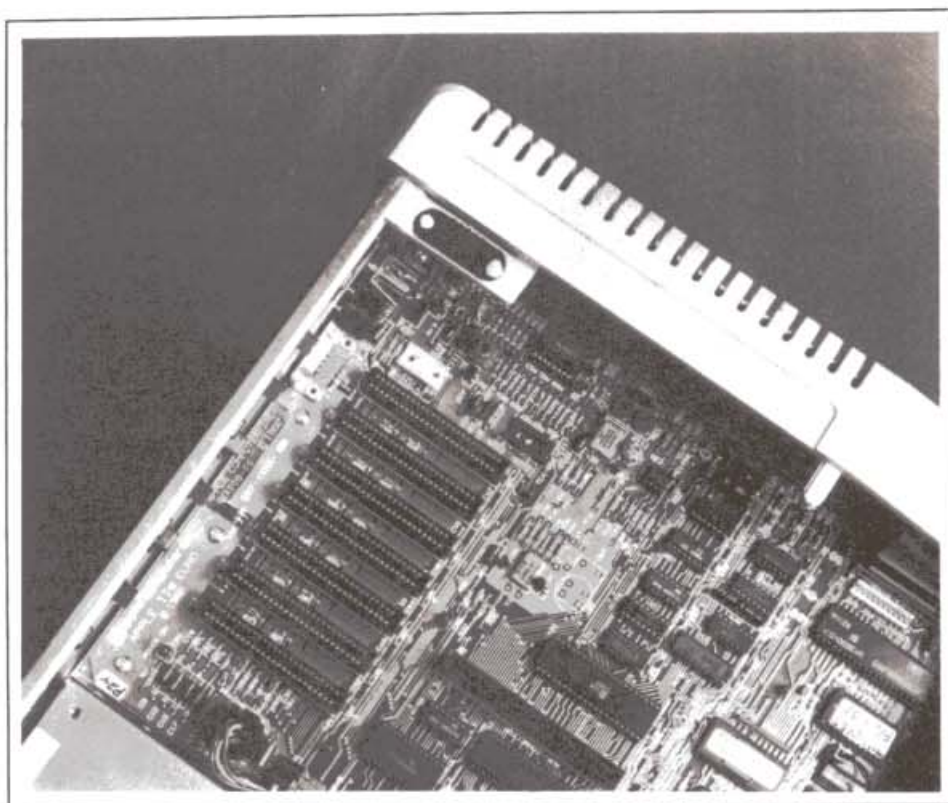


Photo C : Vue interne de l'APPLE II e

1.2 UNITÉ CENTRALE

Examinons de plus près la carte principale de l'APPLE II e.

Sur la photo D, figurent les éléments suivants :

- le microprocesseur 6502A, pièce maîtresse du système, qui effectue les opérations de calcul, exécute les programmes, communique avec la mémoire et les périphériques, etc. Il fonctionne à 1 MHz et effectue jusqu'à 500 000 opérations par seconde sur 8 bits,

- deux boîtiers de mémoire ROM (8 K \times 8 bits) contenant dans 16 K octets, le Basic APPLESOFT, le programme moniteur dont nous parlons au chapitre 4 et qui gère les entrées-sorties, et des programmes d'auto-test du système,

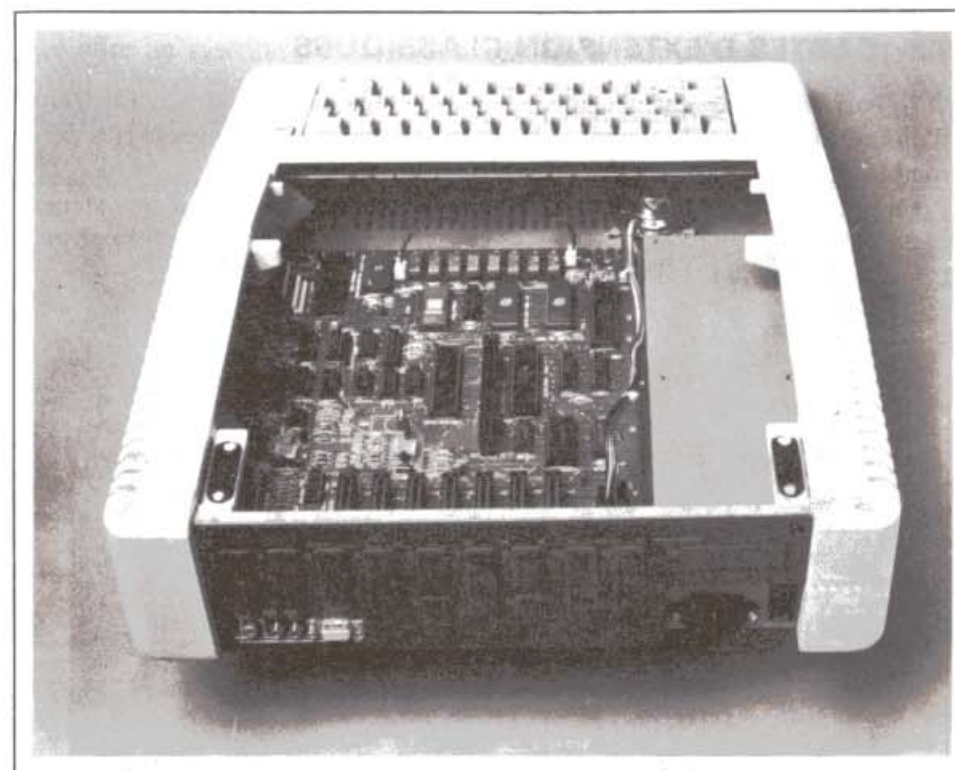


Photo D : Unité centrale

- huit boîtiers de mémoire RAM (64 K \times 1 bit) effaçables à la coupure du courant, offrant 64 K octets disponibles aux programmes utilisateurs,

- une unité de gestion de la mémoire permettant d'accéder aux mémoires RAM et ROM alternativement.

Nous supposerons dans la suite du livre que la ROM Basic est sélectionnée, ce qui permet d'avoir simultanément 48 K octets de RAM disponibles.

1.3 CARTES D'EXTENSION CLASSIQUES

A l'arrière de la carte centrale figurent des connecteurs permettant d'ajouter des cartes d'extension. Nous décrivons ci-dessous les cartes permettant de changer de système d'exploitation ou de langage.

En premier lieu, citons la carte langage qui permet d'obtenir le système d'exploitation P-SYSTEM U.C.S.D. et les langages Pascal, Fortran, Logo, etc. (cf. chap. 2 du tome 2.)

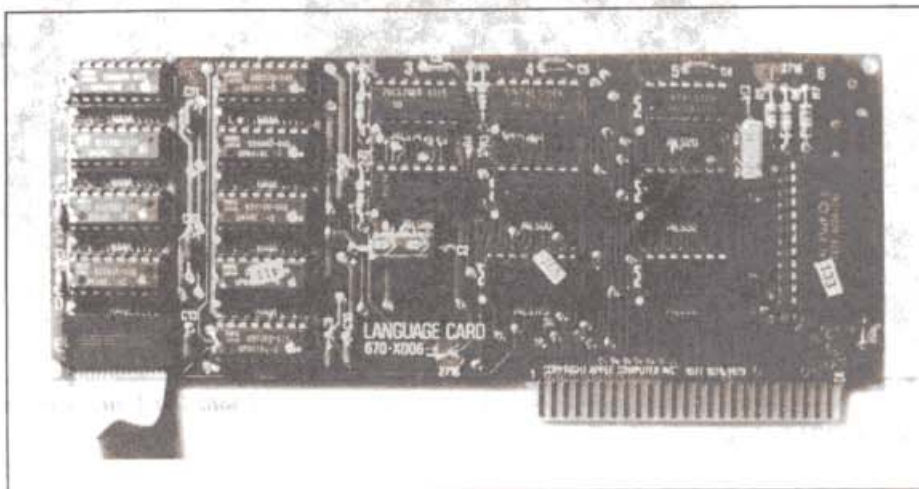


Photo E : Carte langage APPLE

Cette carte a été intégrée dans le modèle APPLE IIe et il suffit d'avoir les disquettes correspondantes pour disposer des langages Pascal, Fortran, Logo, etc.

Dans le même style de carte, la carte M/DOS, créée par la société MIS, qui offre un système d'exploitation beaucoup plus puissant que le DOS APPLE, avec gestion de masques d'entrées-sorties, de fichiers séquentiels indexés multiclés, etc.

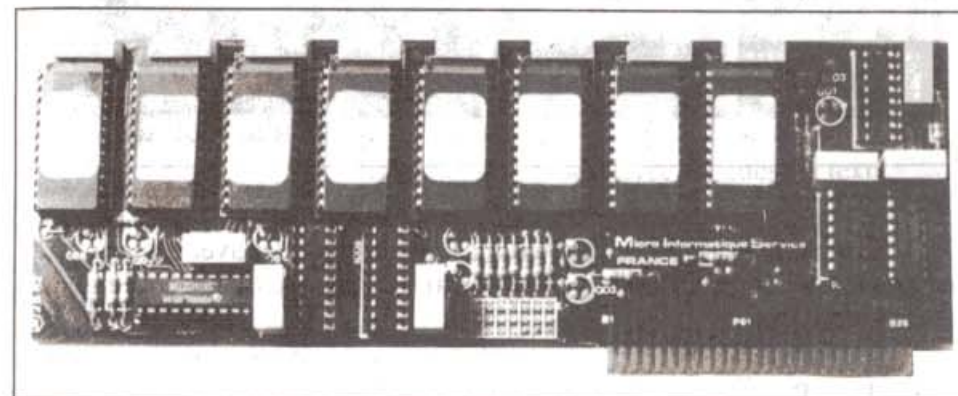


Photo F : Carte M/DOS

Parmi les autres cartes classiques, deux types de cartes sont à évoquer :

— Carte d'interface parallèle permettant de communiquer avec une imprimante.

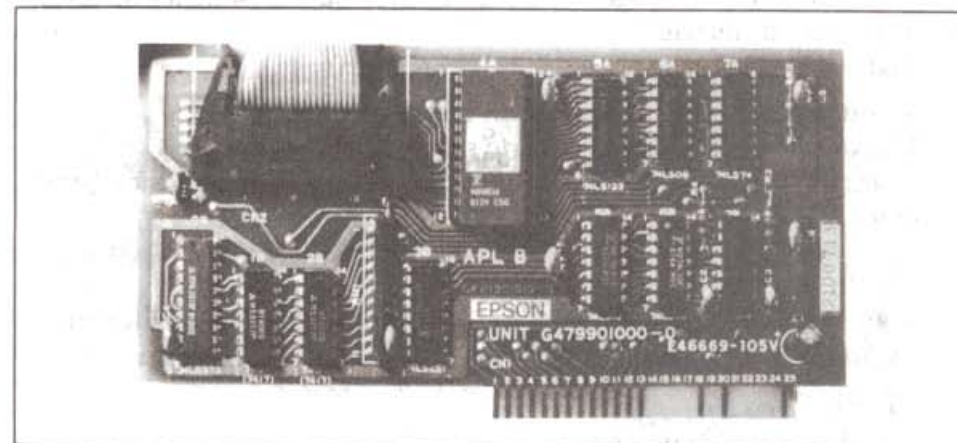


Photo G : Carte d'interface parallèle Epson

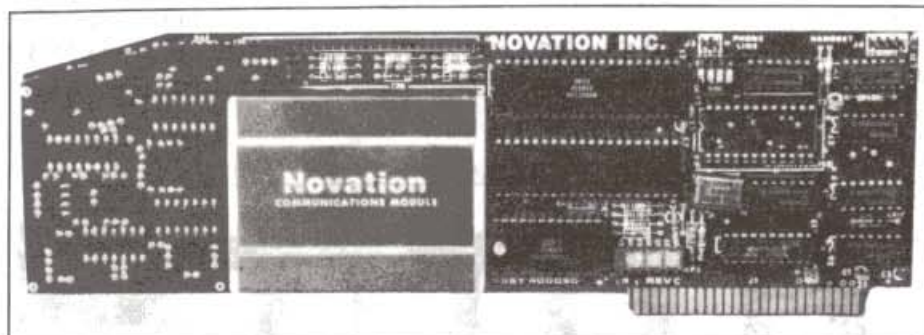


Photo H : Carte modem

— Carte d'interface série permettant de communiquer avec un terminal écran-clavier, une imprimante, avec une autre machine ou avec un réseau téléinformatique.

— Carte modem permettant de communiquer à distance avec un autre ordinateur, par l'intermédiaire ou non du téléphone.

1.4 SORTIE TÉLÉVISION ET VARIANTES

Par sélection logicielle, un APPLE II affiche du texte, des graphiques basse ou haute résolution, et lorsqu'il est relié à un moniteur couleur ou à un poste de télévision couleur (avec option RVB Péritel), il affiche des graphiques en couleur. Les commandes graphiques permettent d'afficher l'une ou l'autre des 2 « pages » de l'écran, avec ou sans 4 lignes de texte sous la zone d'affichage.

Modes d'affichage (matrice 5 × 7) :

- texte sur 40 colonnes (télévision ou moniteur),
- texte sur 80 colonnes avec carte sur option et moniteur,
- graphiques couleur basse résolution (moniteur ou télévision avec carte RVB Péritel),
- graphiques couleur haute résolution (moniteur ou télévision avec carte RVB Péritel),
- graphiques très haute résolution (carte 80 colonnes plus 64 K octets de mémoire).

Capacité en texte :

- 24 lignes de 40 colonnes,
- 24 lignes de 80 colonnes avec carte sur option.

Jeux de caractères :

- 96 caractères ASCII imprimables (majuscules et minuscules),
- caractères français et anglais.

Formats d'affichage :

- normal, inversé, clignotant.
- Graphiques basse résolution :
 - 16 couleurs,
 - résolution : * 40 horizontal × 48 vertical
 - * 40 horizontal × 40 vertical avec 4 lignes de texte.
- Graphiques haute résolution :
 - 6 couleurs : noir, blanc, vert, bleu, violet, orange,
 - résolution : * 280 horizontal × 192 vertical,
 - * 280 horizontal × 160 vertical avec 4 lignes de texte.
- Graphiques très haute résolution : 560 horizontal × 192 vertical.

Les cartes d'extension vidéo doivent être placées dans le connecteur auxiliaire, situé au centre de la carte mère.

1.5 AUTRES EXTENSIONS POSSIBLES

La force commerciale de l'APPLE II réside dans le fait qu'avec une carte d'extension, le comportement de la machine peut être totalement bouleversé. Le meilleur exemple que nous pouvons donner réside dans les cartes

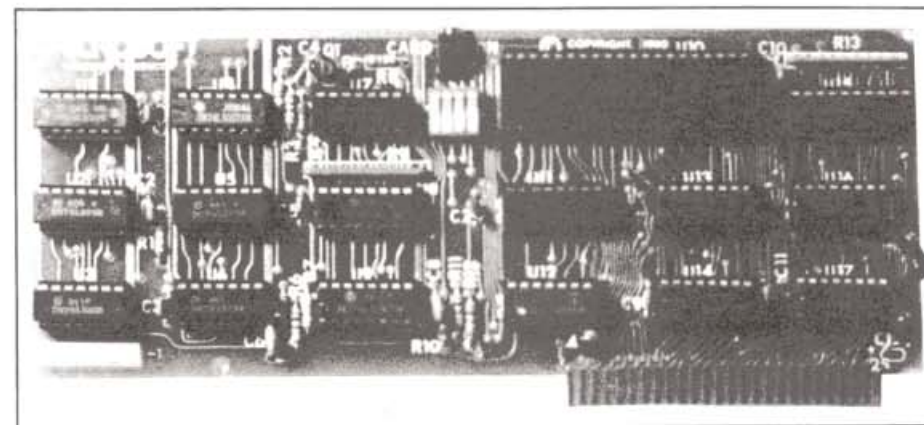


Photo I : Softcard Z80

qui comportent d'autres processeurs (Z80, 6809, 8088, ...), ou dans les cartes qui permettent de faire de la synthèse vocale, du traitement d'images vidéo, ou qui transforment le système en oscilloscope.

Il existe en outre cinq types de cartes d'extension non classiques :

— Les cartes d'extension de la mémoire de 16 K, 32 K, 64 K et même 128 K octets. Exemple : la Ramcard de Microsoft ou la carte Legend 128 K qui peut être utilisée comme simulation d'une unité de disquettes.

— Les cartes « processeur » (Z80, 6809, 8088) auxquelles se rattache la Softcard Z80 décrite dans le tome 2.

— Les cartes d'interface avec des appareils externes (bus IEEE/488, instruments de mesure, convertisseur analogique-digital et digital-analogique, oscilloscope, ...).

— cartes de traitement de la parole,

— cartes de digitalisation du monde extérieur.

Nous donnons dans le tome 2, un tour d'horizon des cartes d'extension disponibles sur l'APPLE II.

CHAPITRE 2

Le Basic APPLESOFT

2.1 INTRODUCTION

Le Basic APPLESOFT est présent en mémoire morte dans l'APPLE II^e et dans l'APPLE II⁺. Ce chapitre a donc pour objet la syntaxe du langage et l'emploi d'une imprimante. Les possibilités graphiques seront abordées au prochain chapitre.

2.2 DESCRIPTION SYNTAXIQUE DE L'APPLESOFT

2.2.1 Variables simples

Les trois types suivants de variables simples existent :

Type	Nom de variable	Valeurs possibles
Entier	AB %	± 32767
Réel	AB	$\pm 9.99999999E + 37$
Chaîne de caractères	AB\$	0 à 255 caractères alphanumériques

où A représente une lettre et B une lettre ou un chiffre.

Seuls sont pris en compte, en plus du type, les deux premiers caractères du nom d'une variable.

Par exemple : PR1 % et PR2 % désignent la même variable, mais PR1 % et PR2\$ sont des variables différentes (types différents).

L'instruction CLEAR initialise à zéro toutes les variables (simples ou non).

2.2.2 Variables tableaux

Nous retrouvons les trois types de variables :

Type	Nom de variable	Déclaration
Entier	AB %(4, 6, 3)	DIM AB %(5, 6, 7)
Réel	AB(2, 3)	DIM AB(2, 4)
Chaîne de caractères	AB\$(2, 8, 6)	DIM AB\$(2, 8, 6)

La taille des tableaux est limitée uniquement par la **taille** mémoire disponible. Avant d'utiliser une variable tableau, il faut la **dimensionner** par une instruction DIM (cf. partie déclaration du tableau ci-dessus).

L'index utilisable dans un tableau est compris entre zéro et la valeur indiquée lors de la déclaration. Par exemple, le tableau T1(2, 3, 4) contiendra $(2 + 1) * (3 + 1) * (4 + 1) = 60$ éléments différents.

Le nombre de dimensions d'un tableau est limité à quatre-vingt-huit dimensions.

La taille des éléments « chaîne de caractères » d'un tableau peut varier au cours du temps (0 à 255 caractères).

Par défaut, le Basic considérera un tableau non dimensionné comme ayant 11 éléments par dimension.

2.2.3 Opérateurs numériques

Ces opérateurs sont traités selon l'ordre décroissant des **priorités** :

- () parenthèses de délimitation d'expression
- + - opérateurs unaires (* + 1, * - 1)

- ^ élévation à la puissance
- * multiplication
- / division
- + addition
- soustraction

2.2.4 Opérateurs relationnels

Leur priorité, inférieure à celle des opérateurs numériques, s'établit selon l'ordre décroissant :

- > supérieur strictement à
- < inférieur strictement à
- > = supérieur ou égal à
- < = inférieur ou égal à
- = > supérieur ou égal à
- = < inférieur ou égal à
- < > différent de
- > < différent de
- = égal à

Les chaînes de caractères peuvent être comparées entre elles selon les codes ASCII des caractères contenus.

2.2.5 Opérateurs logiques

Il existe trois types d'opérateurs logiques :

- NOT négation logique
(priorité immédiatement supérieure à l'élévation à la puissance)
- AND et logique
- OR ou logique
(opérateurs de plus faible priorité).

Le résultat d'une comparaison ou d'une opération logique peut être égal à un (vrai) ou à zéro (faux).

Par la suite, nous désignerons sous le terme d'expression toute série d'opérations entre variables et constantes réelles, ou entières, ou entre chaînes de caractères.

Exemple : $(AB↑E) < (D - E) \text{ AND } (A + 1 > B)$

Nous noterons X, Y, Z les expressions numériques, VR, VI les variables réelles ou entières.

2.2.6 Instructions de contrôle

Les instructions suivantes correspondent aux boucles, aux branchements calculés, conditionnels ou inconditionnels, à l'utilisation de sous-programmes, etc.

```
FOR VR = 2 TO 10 STEP 4
  .
  .
  .
NEXT VR
```

} boucle
de
programme

1. VR représente, pour les instructions FOR et NEXT, une variable réelle. Les autres termes de l'instruction FOR sont des expressions numériques, réelles ou non.

2. Les instructions comprises dans la boucle sont exécutées au moins une fois, car le test a lieu au niveau de l'instruction NEXT.

IF <expression> THEN instructions	exécution des instructions indiquées ou branchement à la ligne indiquée si la condition est vraie
ON X GOTO L1, L2, ...	branchement calculé à la ligne Li si l'expression X est égale à i
GOTO n° ligne	branchement inconditionnel à la ligne précisée
ON X GOSUB S1, S2, ...	appel calculé de sous-programmes
GOSUB S	appel du sous-programme situé à la ligne S
RETURN	retour d'un sous-programme
POP	suppression d'un niveau de sous-programme (équivalent à RETURN mais avec enchaînement à l'instruction qui suit POP)

STOP	arrêt du programme et impression du numéro de la ligne comprenant STOP
END	arrêt normal du programme
RUN	lancement de l'exécution d'un programme
CTRL-C } RESET }	arrêt du programme provoqué au clavier par l'utilisateur
CONT	reprise d'un programme après STOP, END, CTRL-C

2.2.7 Accès à la mémoire et appel de sous-programme assembleur

POKE X, Y	positionnement à l'adresse X de l'octet Y
PEEK(X)	lecture de la valeur de l'octet d'adresse X

Les instructions POKE et PEEK vous seront très utiles pour l'élaboration de programmes complexes. Vous pourrez vous en passer pour les programmes simples.

FRE(0)	lecture du nombre d'octets non utilisés en mémoire et réorganisation de la zone mémoire occupée par les variables
HIMEM	positionnement de l'adresse maximale utilisable en mémoire par le Basic
LOMEM	positionnement de l'adresse minimale utilisable en mémoire par le Basic

Si vous désirez vous servir des possibilités graphiques de votre système, ou réaliser des sous-programmes assembleurs, nous vous conseillons de limiter la place occupée par le Basic et ses programmes pour ne pas risquer d'écraser quoi que ce soit.

CALL X appel du sous-programme assembleur d'adresse indiquée par l'expression X

USR(X) appel de la fonction assembleur avec passage du paramètre égal à la valeur de l'expression X

Lors de l'appel de la fonction USR, le Basic convertit le paramètre en nombre réel, le range dans « l'accumulateur flottant » situé aux adresses 157 à 163 et effectue un branchement à l'adresse 10 qui doit contenir un autre branchement vers le sous-programme assembleur.

WAIT X, Y, Z attente jusqu'à ce que soit obtenu que (X) XOR Z AND Y soit non nul

(X) désigne l'octet situé à l'adresse X.

& branchement à l'adresse \$3F5 (équivalent Basic de la fonction CTRL-Y du moniteur, cf. chap. 4)

2.2.8 Edition de texte sur l'écran

CTRL-X suppression de la ligne en cours
→ recopie du caractère sous le curseur en mémoire et déplacement du curseur d'une position vers la droite

← suppression d'un caractère et déplacement du curseur d'une position vers la gauche

ESC A déplacement du curseur d'une position vers la droite

ESC B déplacement du curseur d'une position vers la gauche

ESC C déplacement du curseur d'une position vers le haut

ESC D déplacement du curseur d'une position vers le bas

Pour ces quatre dernières fonctions, le caractère situé sous le curseur n'est pas recopié en mémoire.

ESC E suppression des caractères situés entre le curseur et la fin de la ligne

ESC F suppression des caractères situés entre le curseur et le bas de l'écran

ESC @ effacement de l'écran

Exemple : transformation de la ligne

10 PRINT "APPLE"
en 10 PRINT "POMME"

scénario : 1. Recopie de 10 PRINT "

- a. positionner le curseur sur le 1 du nombre 10
- b. taper 11 fois →

2. Suppression de APPLE
taper 5 fois ESC A

3. Ajout de POMME
 - a. se placer à la ligne supérieure
taper ESC D
 - b. taper POMME
 - c. retour sur le second "
taper 5 fois ESC B
 - d. validation du "
taper →
 - e. fin de la modification
taper RETURN

2.2.9 Instructions d'entrées-sorties sur l'écran

VTAB X	positionnement du curseur à la ligne X (X = 1 à 24)
HTAB X	positionnement du curseur à la colonne X (X = 1 à 255)

La ligne logique gérée par l'instruction HTAB consiste en plusieurs lignes physiques sur l'écran. Par exemple X=1 à 40 se situe sur la ligne courante, X=41 à 80 sur la suivante et ainsi de suite.

POS(0)	obtention de la position du curseur sur l'écran
HOME	effacement de l'écran et positionnement du curseur dans le coin supérieur gauche
NORMAL INVERSE FLASH	affichage des caractères dans le mode écran choisi <div style="display: inline-block; vertical-align: middle; margin-left: 10px;"> { (blanc sur noir) (noir sur blanc) (clignotant) </div>
SPEED=X	modification de la vitesse d'affichage des caractères sur l'écran (X = 0 correspond à la vitesse minimale)
GET A\$	lecture d'un caractère alphanumérique au clavier et enchaînement sans attendre la touche RETURN
INPUT A%	affichage d'un point d'interrogation et saisie d'une variable (numérique ou non) terminée par la touche RETURN
INPUT "XYZ"; A, B%, C\$	affichage de la chaîne de caractères indiquée sans point d'interrogation et saisie de variables au clavier
PRINT "A="; A, "B>="; C; "D<="; B	impression sur l'écran des chaînes "A=", "B>=", "D<=" et des valeurs des variables A, C, B selon le format suivant : A=valeur B>=valeur D<=valeur

Les points-virgules concatènent les champs alors que les virgules les positionnent au début d'une zone de tabulation de seize caractères (c'est-à-dire en colonne 1, 17, 32). Un point-virgule situé à la fin d'une instruction PRINT supprime le passage à la ligne suivante.

Exemples :

	colonne 1	colonne 17	colonne 32
PRINT "XYZ", "2"; 3	XYZ	23	
PRINT "X"; "Y"; "Z2", 3	XYZ2	3	
PRINT "X", "Y"; "Z", 32	X	YZ	32

SPC(X)	cette fonction est utilisée dans une instruction PRINT pour insérer X blancs dans la ligne affichée
TAB(X)	cette fonction est utilisée dans une instruction PRINT pour positionner le curseur à la colonne X (0-255)

Les instructions

10 HTAB 23 : PRINT "ABC"
et
10 PRINT TAB(22); "ABC"

sont équivalentes, car HTAB numérote les colonnes en commençant au chiffre 1 alors que TAB débute à zéro.

READ A1, A2%, A1\$, ...	lecture des données indiquées dans les instructions DATA et assignation aux variables précisées
DATA 1.3, 2, "erty", ...	
RESTORE	reprise de la lecture des DATA à la 1 ^{re} instruction DATA du programme

2.2.10 Création d'une fenêtre de texte sur l'écran

Il est possible de réduire la dimension de l'écran sur lequel s'affiche le texte, en vue par exemple de conserver un titre en haut d'une page, ou de créer des pseudo-masques de saisie de données, etc.

A cet effet, vous utiliserez l'instruction POKE de modification de la mémoire pour les adresses suivantes :

Adresse	Rôle
32	Marge gauche (1 à 40)
33	Largeur (1 à 40)
34	1 ^{re} ligne (1 à 24)
35	Dernière ligne (1 à 24)

Les dimensions de la fenêtre ne doivent pas dépasser celles de l'écran, car aucun contrôle n'étant fait, la fenêtre écraserait une zone mémoire qui ne lui est pas attribuée. Il est indispensable d'avoir toujours les relations suivantes :

$$\text{PEEK}(34) + \text{PEEK}(35) < = 24$$

$$\text{PEEK}(34) < = \text{PEEK}(35)$$

$$\text{PEEK}(32) + \text{PEEK}(33) < = 40$$

Voici le comportement des instructions d'entrée-sortie sur l'écran :

- HTAB tient compte de la nouvelle largeur de l'écran
- VTAB : effet nul
- PRINT et ses fonctions associées (TAB, SPC...) :
 1. la position actuelle du curseur n'est pas testée
 2. tous les caractères affichés respectent ensuite la fenêtre.

Par exemple, si vous avez positionné la première ligne à 10, la marge gauche à 5, la largeur à 20, vous obtiendrez la situation de la figure 2.1 en utilisant la ligne

VTAB 9 : HTAB 35 : PRINT "ABCDEFGG"

- INPUT : le principe est le même que pour l'instruction PRINT.

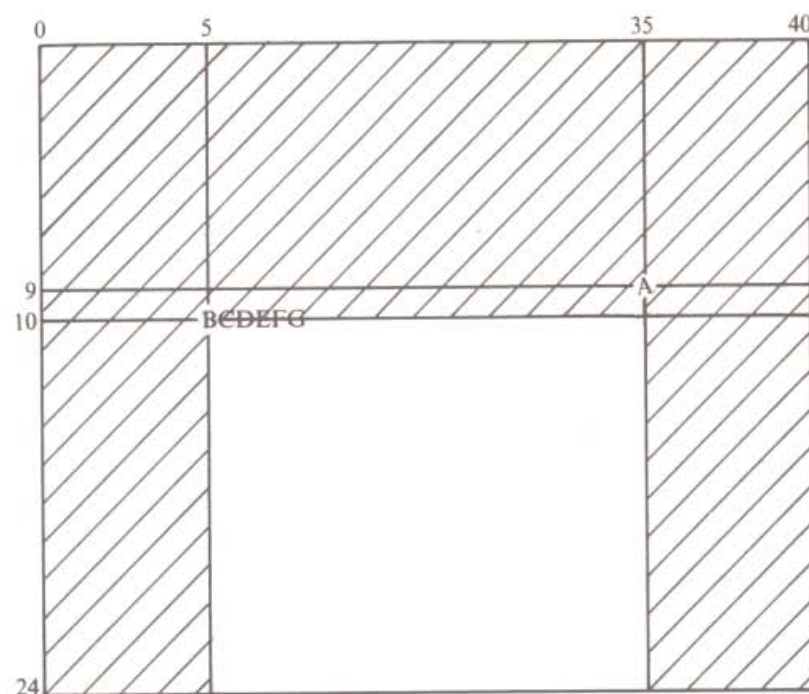


Figure 2.1

2.2.11 Instructions d'entrées-sorties avec les périphériques

Pour travailler avec les périphériques (imprimante, DOS 3.3, ...), les instructions nécessaires sont :

- | | |
|---------------------------|---|
| IN # numéro de connecteur | lecture des données provenant du périphérique relié au connecteur précisé |
| PR # numéro de connecteur | sortie des données vers le périphérique relié au connecteur précisé |
| LOAD | chargement d'un programme à partir de la cassette |
| SAVE | sauvegarde d'un programme sur cassette |
| LOAD nom de fichier | chargement |
| SAVE nom de fichier | sauvegarde |
| RUN | exécution |
- } — d'un programme
situé sur
disquette
(cf. tome 2 Le DOS 3.3)

STORE A	sauvegarde d'un tableau numérique sur cassette
RECALL A	restauration d'un tableau numérique précédemment dimensionné à partir de la cassette
PDL(I)	lecture de la valeur indiquée par la manette de jeux numéro I (cf. § 3.7).

2.2.12 Traitement des erreurs

Au cas où aucune instruction de traitement des erreurs n'est présente dans votre programme, le Basic APPLESOFT arrêtera votre programme dès qu'apparaît une erreur et affichera le message correspondant.

Pour l'éviter, vous pouvez reprendre le contrôle des opérations en plaçant au début de votre logiciel l'instruction suivante :

ON ERROR GOTO no Ligne

A la rencontre d'une quelconque erreur, le Basic effectuera un branchement à la ligne indiquée. Vous pouvez alors disposer des deux informations suivantes :

● Code de l'erreur

Il est situé à l'adresse 222 et peut prendre les valeurs indiquées dans le tableau ci-dessous :

Code	Message d'erreur
0	NEXT without FOR l'instruction NEXT a été rencontrée sans avoir été précédée par FOR
16	Syntax erreur de syntaxe
22	RETURN without GOSUB l'instruction RETURN a été rencontrée sans avoir été précédée par GOSUB
42	Out of DATA les données des instructions DATA ont été entièrement lues précédemment
53	Illegal Quantity
69	Overflow dépassement de capacité

77	Out of memory	il n'y a plus de mémoire libre
90	Undefined Statement	instruction (commande) inconnue
107	Bad Subscript	l'emploi d'un tableau ne correspond pas à sa déclaration (nombre de dimensions, index impossible...)
120	Redimensionned Array	il est interdit de dimensionner deux fois un même tableau
133	Division by Zero	division par zéro
163	Type mismatch	emploi d'un type de variable interdit
176	String Too Long	la longueur de la chaîne de caractères est trop importante ($> = 256$)
191	Formula Too Complex	le Basic n'est pas capable de traiter une expression aussi complexe
224	Undefined Function	la fonction utilisée n'est pas définie
254	Bad Response to INPUT Statement	réponse incorrecte à INPUT
255	Ctrl-C Interrupt Attempted	occurrence de CTRL-C

Le DOS utilisant la même adresse, nous compléterons ce tableau au tome 2.

● Numéro de la ligne où s'est produite l'erreur.

Pour cela, il suffit de lire le contenu des adresses 218 et 219 :

$$L = \text{PEEK}(218) + 256 * \text{PEEK}(219)$$

Le traitement de l'erreur terminé, deux possibilités vous sont offertes :

- l'instruction RESUME qui provoque la reprise du programme à l'instruction en "faute"
- l'instruction GOTO.

Le rôle de l'instruction ON ERROR GOTO n'est en fait que de positionner l'octet d'adresse 216 à une valeur supérieure à 128. Son effet peut être annulé par l'instruction suivante :

POKE 216,0

Un exemple d'utilisation de l'instruction "ON ERROR GOTO" est fourni au paragraphe 3.3.5.

2.2.13 Commandes diverses

NEW	suppression du programme en mémoire
LIST [L1,] [L2]	Liste des lignes comprises entre L1 et L2
DEL L1, L2	suppression des lignes de numéro comprises dans l'intervalle L1, L2
TRACE	impression des numéros des lignes exécutées dans un programme
NOTRACE	suppression de l'effet de TRACE
REM	instruction commentaire

2.2.14 Fonctions traitant les chaînes de caractères

LEN(A\$)	fonction indiquant la longueur (nombre de caractères) d'une chaîne de caractères
VAL(A\$)	fonction indiquant la valeur numérique d'une chaîne de caractères $VAL("23456") = 23456$
STR\$(X)	fonction convertissant un nombre en chaîne de caractères STR(2) = "2"$
ASC (A\$)	fonction retournant le code ASCII du premier caractère de la chaîne A\$
CHR\$(X)	conversion du code ASCII en une chaîne de caractères $ASC(CHR$(10)) = 10$ CHR(65) = "A"$
LEFT\$(A\$, X)	obtention de la chaîne des X premiers caractères de A\$
RIGHT\$(A\$, X)	obtention de la chaîne des X derniers caractères de A\$
MID\$(A\$, X, Y)	obtention de la chaîne des Y caractères extraits à partir du X-ième caractère de A\$ MID("ABCDE", 3, 2) = "CDE"$
+	concaténation de chaînes de caractères

2.2.15 Fonctions mathématiques

DEF FN A(X)=3*X+4 définition de la fonction FNA

FONCTIONS TRIGONOMÉTRIQUES

SIN(X)	sinus de X (en radians)
COS(X)	cosinus de X (en radians)
TAN(X)	tangente de X (en radians)
ATN(X)	arc tangente de X

GÉNÉRATION D'UN NOMBRE ALÉATOIRE

RND(X)	retourne un nombre aléatoire compris entre zéro et un
--------	---

- * Si X est positif, un nouveau nombre est généré à chaque appel
- * Si X est nul, le dernier nombre généré à nouveau.
- * Si X est négatif, le nombre correspondant à X est généré (table de nombres en mémoire).

$$RND(-3) = 4.48217179E-08$$

AUTRES FONCTIONS MATHÉMATIQUES

ABS(X)	valeur absolue de X
EXP(X)	exponentielle de X
INT(X)	partie entière de X
LOG(X)	logarithme népérien de X
SGN(X)	signe de X
SQR(X)	racine carrée de X

2.2.16 Organisation de la mémoire du Basic APPLESOFT**● Introduction**

Ce paragraphe décrit la carte mémoire de l'APPLESOFT, l'organisation des variables, des instructions, la place occupée. Même si vous ne voulez pas travailler en assembleur, la lecture des descriptions suivantes peut vous

être utile pour réduire la taille d'un programme, sa vitesse d'exécution ou pour comprendre le programme PHONE LIST situé sur la disquette système MASTER du DOS 3.3.

● Carte-mémoire

La configuration mémoire ci-dessous correspond au Basic APPLESOFT situé en mémoire ROM.

Adresses	Signification
\$F7FF à \$D000	APPLESOFT
HIMEM	Chaînes de caractères
	Espace libre
	Variables tableaux (numériques et chaînes)
LOMEM	Variables simples
	Programme Basic
\$B01	
\$FF à \$00	Variables systèmes (APPLESOFT, MONITEUR, DOS 3.3)

● Utilisation de la page mémoire zéro

Cette zone-mémoire est comprise entre les adresses \$0000 et \$00FF et contient les variables systèmes du Basic, du moniteur, et du DOS 3.3. Nous décrivons ci-après les adresses-mémoire utilisées par l'APPLESOFT dans la page zéro, qu'il est important de connaître.

Adresse décimale	Adresse hexadécimale	Utilisation
00-05	\$00-\$05	Ces adresses sont utilisées au démarrage du système pour déterminer la présence ou non d'un contrôleur et pour charger le DOS si nécessaire
10-12	\$0A-\$0C	Instruction de branchement à la fonction USR (cf. § 2.2.7)
13-25	\$0D-\$17	Variables internes de l'APPLESOFT
32-79	\$20-\$4F	Adresses utilisées par le moniteur (cf. § 4.3.4)
98-102	\$62-\$66	Résultat de la dernière multiplication/division
103-104	\$67-\$68	Adresse du début du programme Basic en mémoire
105-106	\$69-\$6A	Adresse du début des variables simples en mémoire. LOMEM
107-108	\$6B-\$6C	Adresse du début des variables tableaux en mémoire
109-110	\$6D-\$6E	Adresse de fin des variables tableaux en mémoire (début de la zone libre moins un)
111-112	\$6F-\$70	Adresse de début des chaînes de caractères en mémoire. Celles-ci sont stockées entre cette adresse et HIMEM
113-114	\$71-\$72	Pointeur d'usage général
115-116	\$73-\$74	HIMEM
117-118	\$75-\$76	Numéro de ligne de l'instruction en cours

Adresse décimale	Adresse hexadécimale	Utilisation
119-120	\$77-\$78	Numéro de la ligne dans laquelle a été rencontrée l'une des instructions CTRL-C, STOP, END
121-122	\$79-\$7A	Adresse de la prochaine instruction à exécuter
123-124	\$7B-\$7C	Numéro de la ligne de l'instruction DATA traitée
125-126	\$7D-\$7E	Adresse de la prochaine donnée — à lire — des instructions DATA
127-128	\$7E-\$80	Adresse des données lues : — pour une instruction INPUT, le code positionné est égal à \$201 — pour une instruction READ, ce pointeur contient l'adresse de la prochaine donnée de l'instruction DATA en cours de traitement
129-130	\$81-\$82	Nom de la dernière variable traitée
131-132	\$83-\$84	Adresse de la valeur de la dernière variable traitée
157-163	\$9D-\$A3	Accumulateur flottant principal (zone mémoire contenant les nombres réels à traiter et les résultats des calculs entre des nombres réels)
165-171	\$A5-\$AB	Second accumulateur flottant
175-176	\$AF-\$B0	Adresse de la fin du programme
177-200	\$B1-\$C8	Sous-programme d'acquisition d'un caractère CHRGET
184-185	\$B8-\$B9	Adresse du dernier caractère saisi par CHRGET

Adresse décimale	Adresse hexadécimale	Utilisation
201-205	\$C9-\$CD	Nombre aléatoire généré
216-223	\$D8-\$DF	Traitement des erreurs
216	\$D8	Indicateur d'activation du traitement des erreurs
218-219	\$DA-\$DB	Numéro de la ligne où s'est produite l'erreur
222		Code de l'erreur (cf. § 2.2.12)
224-226	\$E0-\$E2	Coordonnées X, Y d'un point graphique haute résolution
228	\$E4	Couleur haute résolution
232-233	\$E8-\$E9	Adresse de début de la table des figures graphiques (cf. § 3.3 et 3.4)

● Organisation des variables en mémoire

• Variables simples :

Une variable simple est représentée sur 7 octets. Les deux premiers d'entre eux contiennent le nom de la variable et son type. Celui-ci est codé sur le bit de poids fort des 2 octets, selon le choix suivant :

réel $\begin{cases} 1^{\text{er}} & \text{octet positif} \\ 2^{\text{nd}} & \text{octet positif} \end{cases}$ entier $\begin{cases} 1^{\text{er}} & \text{octet négatif} \\ 2^{\text{nd}} & \text{octet négatif} \end{cases}$ chaîne $\begin{cases} 1^{\text{er}} & \text{octet négatif} \\ 2^{\text{nd}} & \text{octet positif} \end{cases}$

Les cinq derniers octets ont un format différent selon le type de la variable.

* VARIABLE ENTIÈRE

Numéro d'octet	Signification	Codage
0	nom	bits 0 à 6 : code ASCII du 1 ^{er} caractère 7 : type de la variable (cf. ci-dessus)
1		idem pour le second caractère
2	valeur numérique	octet de poids fort
3		octet de poids faible
4	0	
5	0	
6	0	

* VARIABLES RÉELLES

Numéro d'octet	Signification	Codage
0	idem avec un type différent	
1		
2	exposant	octet signé (bit 7 = signe)
3	mantisse	octet signé de poids fort
4	mantisse	
5	mantisse	
6	mantisse	octet de poids faible

La longueur des chaînes de caractères pouvant croître jusqu'à 255, la zone située sous HIMEM (cf. § 2.2.16) contient les « valeurs » des chaînes.

* CHAÎNE DE CARACTÈRES

Numéro d'octet	Signification	Codage
0	idem avec un codage différent	
1		
2	longueur de la chaîne adresse de la chaîne	poids fort poids faible
3		
4		
5	0	
6	0	

• VARIABLES TABLEAUX :

Les deux premiers octets ont le même format que ceux des variables simples (nom et codage du type). Est ensuite indiqué le déplacement à ajouter à l'adresse de la variable pour passer à la variable suivante.

Sont ensuite placées les informations relatives au nombre de dimensions, à leurs largeurs selon le format suivant :

Numéro d'octet	
3	nombre N de dimensions
4	octet de poids fort de la largeur de la dimension 1 octet de poids faible
5	
.	
.	
.	
4+(n-1)*2	octet de poids fort de la largeur de la dimension N
5+(n-1)*2	octet de poids faible

Selon le type de la variable, les octets suivants sont ensuite répétés :

- * réel : 5 octets de même signification que les octets 3 à 6 d'une variable simple réelle (cf. ci-dessus),
- * entier : octets 3 et 4 d'une variable simple entière (valeur numérique),
- * chaîne : octets 3, 4, 5 d'une variable simple chaîne (longueur et adresse).

• **Conseils pour réduire la durée d'un programme :**

A l'exécution d'un programme, le Basic remplit la zone mémoire des variables au fur et à mesure de leur rencontre. Par ailleurs, pour travailler sur une variable (simple ou tableau), l'APPLESOFT parcourt la zone des variables appropriée (simples ou tableaux) jusqu'à ce qu'il trouve la variable. Il écrit alors son adresse en \$81-\$82.

Pour améliorer la vitesse d'un programme, vous devrez utiliser les variables rencontrées en premier dans le programme comme variables de travail.

Le fait d'utiliser des variables à la place de constantes peut diminuer jusqu'à un facteur dix la durée d'exécution d'un logiciel.

Exemple : Si vous utilisez le nombre $\pi = 3.14159$, ne le répétez pas dans votre programme; créez plutôt la variable PI contenant 3.14159.

● **Organisation des instructions**

L'APPLESOFT code les mots-clés reconnus sur un octet. Les autres parties des instructions sont représentées selon leur code ASCII.

Le codage des mots-clés est le suivant :

Code	Mot-clé
128	END
129	FOR
130	NEXT
131	DATA
132	INPUT
133	DEL
134	DIM
135	READ
136	GR
137	TEXT

Code	Mot-clé
138	PR #
139	IN #
140	CALL
141	PLOT
142	HLIN
143	VLIN
144	HGR2
145	HGR
146	HCOLOR =
147	HPlot

Code	Mot-clé
148	DRAW
149	XDRAW
150	HTAB
151	HOME
152	ROT =
153	SCALE =
154	SHLOD
155	TRACE
156	NOTRACE
157	NORMAL

Code	Mot-clé
158	INVERSE
159	FLASH
160	COLOR =
161	POP
162	VTAB
163	HIMEM:
164	LOMEM:
165	ONERR
166	RESUME
167	RECALL
168	STORE
169	SPEED =
170	LET
171	GOTO
172	RUN
173	IF
174	RESTORE
175	&
176	GOSUB
177	RETURN
178	REM
179	STOP
180	ON
181	WAIT
182	LOAD
183	SAVE

Code	Mot-clé
184	DEF
185	POKE
186	PRINT
187	CONT
188	LIST
189	CLEAR
190	GET
191	NEW
192	TAB(
193	TO
194	FN
195	SPC(
196	THEN
197	AT
198	NOT
199	STEP
200	+
201	-
202	*
203	/
204	^
205	AND
206	OR
207	>
208	=
209	<

Code	Mot-clé
210	SGN
211	INT
212	ABS
213	USR
214	FRE
215	SCRN(
216	PDL
217	POS
218	SQR
219	RND
220	LOG
221	EXP
222	COS
223	SIN
224	TAN
225	ATN
226	PEEK
227	LEN
228	STR\$
229	VAL
230	ASC
231	CHR\$
232	LEFT\$
233	RIGHT\$
234	MID\$

Pour atteindre une instruction, ou une ligne, l'APPLESOFT parcourt séquentiellement le programme jusqu'à ce qu'il la rencontre. Nous vous conseillons, pour accroître la vitesse (non la lisibilité) de vos programmes, de placer vos sous-programmes en tête selon le format suivant :

GOTO programme principal

sous-programmes

programme principal

Pour une meilleure compréhension nous n'adopterons pas ce format dans les exemples de ce livre.

Supprimer les instructions REM réduit la taille et la durée de votre programme. Mais ne procédez pas à cette opération avant d'avoir terminé la mise au point de votre programme.

● Modification de l'adresse de chargement d'un programme Basic

Pour cela, il suffit de positionner le pointeur situé à l'adresse 103-104 avec la nouvelle adresse de début. La suite des opérations est identique. ATTENTION à ne rien écraser en mémoire, et aux valeurs de LOMEN: et de HIMEN:

2.3 UTILISATION D'UNE IMPRIMANTE

2.3.1 Introduction

L'APPLE II peut être muni d'une imprimante. Pour cela, il est nécessaire de posséder une carte d'extension. Celles-ci sont de deux types :

- Interface parallèle (les 8 bits d'un octet sont émis en parallèle sur 8 fils) insérée dans le connecteur numéro un.
- Interface série asynchrone (les 8 bits sont émis successivement sur le même fil) insérée dans le connecteur numéro deux.

Le choix d'une imprimante se fait selon le type de connexion et selon la qualité nécessaire. Il existe pour cela deux types d'imprimantes :

- Imprimante à marguerite, de qualité professionnelle, nécessaire pour faire du traitement de texte.
- Imprimante à aiguilles, de qualité moindre pour les lettres, mais permettant de définir des possibilités graphiques et de réaliser des copies d'écran.

2.3.2 Présentation de l'EPSON MX82FT

L'EPSON MX82FT peut être reliée à l'APPLE II soit par une carte d'interface parallèle, soit par une carte d'extension spécialement conçue par EPSON pour faciliter la gestion des impressions (copie d'écran, nombre de caractères par ligne, impression des lettres minuscules, des caractères gras, en mode vidéo inversé...).

L'entraînement peut se faire par traction ou friction. La vitesse d'impression est de 80 caractères par seconde.

1. Les caractéristiques indiquées sont relatives au type III MX82FT.
2. Nous supposons, dans la suite de ce paragraphe, que la carte d'interface est insérée dans le connecteur numéro un.

● Impression de textes

• Caractères spéciaux

Les divers choix possibles sont effectués par l'envoi de codes spéciaux ou par l'écriture de valeurs numériques dans des cases mémoires.

- Nombre de caractères par ligne POKE 1657, n
- Minuscules CTRL-W sert de délimiteur
PRINT "A" + CHR\$(23) + "B" + CHR\$(23) + "C"
donne Ab C.

Il est possible de changer le délimiteur des minuscules en mettant le nouveau code ASCII à l'adresse 1401.

- Double impression { début POKE 1529, 255
fin POKE 1529, 0

Les deux impressions des lignes sont séparées par un espacement vertical de 1/256 de pouce.

- Caractères condensés { début PRINT CHR\$(15) ou envoi du code SI(CTRL-O)
fin PRINT CHR\$(18) ou envoi du code DC2(CTRL-R)
- Caractères agrandis { début PRINT CHR\$(14) ou envoi du code SO(CTRL-N)
fin PRINT CHR\$(20) ou envoi du code DC4(CTRL-T)

Pour ces deux derniers types de caractères, toute fin de ligne provoque leur abandon.

- Caractères gras $\left\{ \begin{array}{l} \text{début} \left\{ \text{PRINT CHR}\$(27) + "E" \text{ ou envoi des codes ESC E} \right. \\ \text{fin} \left\{ \text{PRINT CHR}\$(27) + "F" \text{ ou envoi des codes ESC F} \right. \end{array} \right.$
- Sous-lignage $\text{début et fin PRINT CHR}\$(27) + "-" \text{ ou envoi des codes ESC -}$

• Choix du jeu de caractères :

Les jeux de caractères existent en :

- * anglais (0),
- * français (1),
- * allemand (2),
- * etc.

Ceci permet, en particulier, de disposer d'un jeu de caractères accentués.

• Modification de l'espacement des lignes :

- 1/8 POUCE = ESC 0
- 7/72 POUCE = ESC 1
- 1/6 POUCE = ESC 2
- n/216 POUCE = ESC 3 n

● Graphisme haute-résolution

La carte d'extension contient un logiciel de copie d'écran. Pour l'utiliser, il suffit de taper CTRL-Q lorsque l'imprimante est active. La copie a lieu alors selon la valeur de l'octet situé à l'adresse 1913.

N	D	I	E	O	A	S	P
---	---	---	---	---	---	---	---

Le rôle des différents bits est le suivant :

- N=0 $\left\{ \begin{array}{l} \text{image complète} \\ 128 \text{ ligne actuelle du curseur} \end{array} \right.$
- D=0 $\left\{ \begin{array}{l} \text{taille de l'image standard} \\ 64 \text{ taille de l'image double} \end{array} \right.$
- I=0 $\left\{ \begin{array}{l} \text{mode normal} \\ 32 \text{ mode inverse} \end{array} \right.$

E, A, O opérations bit à bit entre les deux pages graphiques

- $\left\{ \begin{array}{l} E = 16 \text{ ou exclusif} \\ A = 8 \text{ et logique} \\ O = 4 \text{ ou logique} \end{array} \right.$

- S=0 $\left\{ \begin{array}{l} \text{non impression de la page 2} \\ 2 \text{ impression de la page 2} \end{array} \right.$

- P=0 $\left\{ \begin{array}{l} \text{non impression de la page 1} \\ 1 \text{ impression de la page 1.} \end{array} \right.$

Il est possible d'obtenir une meilleure définition graphique en faisant

POKE 1145,DD où DD = 76 pour une résolution meilleure
75 pour une résolution normale

De plus, vous pouvez programmer directement les images bit par bit. Les exemples des figures 3.3 à 3.6 ont été obtenues avec ce programme de copie d'écran.

2.4 EXEMPLES DE SAISIES DE DONNÉES

2.4.1 Introduction

Voici des exemples classiques de sous-programmes Basic pour une application de gestion d'un carnet d'adresses, sur fichier à accès direct.

Ci-dessous, nous avons fait figurer un exemple de menu, un exemple de saisie de données, un exemple d'impression des données saisies, de modification de données et un dernier exemple de recherche de données, dans les inscriptions déjà réalisées.

Nous avons supprimé la gestion du fichier, en précisant les parties où elle a lieu.

2.4.2 Menu

```

430 REM
440 REM PROPOSITION DU MENU
450 REM *****
460 REM
470 HOME : INVERSE
480 PRINT "    *** CARNET D'ADRESSES ***": PRINT : PRINT : PRINT
490 PRINT "1- CONSULTATION AVEC/SANS MODIFICATION"
```



```

500 PRINT "2- INSCRIPTION (NOUVELLE ADRESSE )"
510 PRINT "3- LISTE DU FICHIER"
520 PRINT "4- FIN DU PROGRAMME"
530 PRINT
540 INPUT "ENTREZ VOTRE CHOIX (1,2,3,4) ":A
550 IF A < 1 OR A > 4 THEN 470
560 NORMAL
570 ON A GOSUB 640,1410,1790,2000
580 GOTO 470

```

2.4.3 Saisie de données

Les données à saisir sont les suivantes :

- nom de la personne (NOM\$)
- prénom (PRENOM\$)
- genre de la voie (GENREVOIES)
- nom de la voie (NOMVOIES)
- numéro (NUMEROS)
- ville (VILLE\$)
- code postal (CDPOST\$)
- indicatif (INDTELS)
- et numéro de téléphone (tel\$)

Les données pouvant être introduites lors d'une nouvelle inscription ou lors d'une modification, un sous-programme de saisie a été créé pour chacune d'entre elles.

• ajout d'une nouvelle inscription :

```

1390 REM
1400 REM *****
1410 REM AJOUT D'INSCRIPTIONS (2)
1420 REM *****
1430 REM
1440 INVERSE
1450 HOME : PRINT "NOUVELLE ADRESSE": PRINT : PRINT
1460 NORMAL
1470 PRINT "SI UNE REPONSE NE PEUT ETRE FOURNIE"
1480 PRINT "TAPEZ SEULEMENT RETURN"
1490 PRINT
1500 CODE$ = "1"
1510 GOSUB 2770: REM SAISIE DU NOM
1520 B$ = NOM$

```

```

1530 GOSUB 2870: REM SAISIE DU PRENOM
1540 GOSUB 2970: REM SAISIE DU GENRE DE VOIE
1550 GOSUB 3070: REM SAISIE DU NOM DE LA VOIE
1560 GOSUB 3170: REM SAISIE DU NUMERO
1570 GOSUB 3270: REM SAISIE DE LA VILLE
1580 GOSUB 3370: REM SAISIE DU CODE POSTAL
1590 GOSUB 3470: REM SAISIE DU TEL
1600 HOME
1610 INVERSE
1620 PRINT "INFORMATIONS SAISIES"
1630 NORMAL : PRINT : PRINT
1640 GOSUB 3770: REM AFFICHAGE BUFFER
1650 PRINT : INVERSE
1660 INPUT "VALIDEZ-VOUS CES INFORMATIONS (O/N)":A$
1670 IF A$ < > "N" AND A$ < > "O" THEN 1660
1680 IF A$ = "N" THEN 1410
1690 REM
1700 REM AJOUT DE L'ARTICLE DANS LE FICHIER
1710 REM

```

écriture dans le fichier

• sous-programmes de saisie de chaque donnée :

```

2770 REM
2780 REM SAISIE DU NOM
2790 REM *****
2800 REM
2810 INPUT "NOM ":A$
2820 N = 15: IF LEN (A$) < = N THEN 2850
2830 GOSUB 3690: REM AFFICHAGE MESSAGE D'ERREUR
2840 GOTO 2810
2850 GOSUB 3840:NOM$ = A$
2860 RETURN
2870 REM
2880 REM SAISIE DU PRENOM
2890 REM *****
2900 REM
2910 INPUT "PRENOM ":A$
2920 N = 15: IF LEN (A$) < = N THEN 2950
2930 GOSUB 3690: REM AFFICHAGE MESSAGE D'ERREUR
2940 GOTO 2910
2950 GOSUB 3840:PRENOM$ = A$
2960 RETURN
2970 REM
2980 REM SAISIE DU GENRE DE VOIE
2990 REM *****
3000 REM
3010 INPUT "GENRE DE VOIE(RUE,AV,BLD,...) ":A$
3020 N = 3: IF LEN (A$) < = N THEN 3050
3030 GOSUB 3690: REM AFFICHAGE MESSAGE D'ERREUR
3040 GOTO 3010
3050 GOSUB 3840:GENREVOIE$ = A$

```

```

3060 RETURN
3070 REM
3080 REM SAISIE DU NOM DE LA RUE
3090 REM *****
3100 REM
3110 INPUT "NOM DE LA VOIE ";A$
3120 N = 15: IF LEN (A$) < = N THEN 3150
3130 GOSUB 3690: REM AFFICHAGE MESSAGE D'ERREUR
3140 GOTO 3110
3150 GOSUB 3840:NVOIE$ = A$
3160 RETURN
3170 REM
3180 REM SAISIE DU NUMERO
3190 REM *****
3200 REM
3210 INPUT "NUMERO ";A$
3220 N = 3: GOSUB 3840:NUMERO$ = A$
3230 RETURN
3240 REM
3250 REM SAISIE DE LA VILLE
3260 REM *****
3270 REM
3280 INPUT "VILLE ";A$
3290 N = 15: IF LEN (A$) < = N THEN 3320
3300 GOSUB 3640: REM AFFICHAGE MESSAGE D'ERREUR
3310 GOTO 3280
3320 GOSUB 3840:VILLE$ = A$
3330 RETURN
3340 REM
3350 REM SAISIE DU CODE POSTAL
3360 REM *****
3370 REM
3380 INPUT "CODE POSTAL ";A$
3390 N = 5: IF LEN (A$) < = N THEN 3420
3400 PRINT "PAS PLUS DE CINQ CHIFFRES SVP"
3410 GOTO 3380
3420 GOSUB 3840:CDPST$ = A$
3430 RETURN
3440 REM
3450 REM SAISIE DE L'INDICATIF TELEPHONIQUE
3460 REM *****
3470 REM
3480 INPUT "INDICATIF TELEPHONIQUE ";A$
3490 N = 2: IF LEN (A$) < = N THEN 3520
3500 PRINT "PAS PLUS DE DEUX CHIFFRES SVP"
3510 GOTO 3480
3520 N = 2: GOSUB 3840:INDTEL$ = A$
3530 REM
3540 REM SAISIE DU NUMERO DE TELEPHONE
3550 REM *****
3560 REM
3570 INPUT "NUMERO DE TELEPHONE ";A$
3580 N = 7: IF LEN (A$) < = N THEN 3610
3590 PRINT "PAS PLUS DE SEPT CHIFFRES SVP"
3600 GOTO 3570
3610 GOSUB 3840:TEL$ = A$
3620 RETURN

```

```

3840 REM
3850 REM CADRAGE DES DONNEES SAISIES
3860 REM *****
3870 REM
3880 REM DANS UNE CHAINE DE N CARACTERES
3890 REM *****
3900 N = N - LEN (A$)
3910 IF N < = 0 THEN RETURN
3920 A$ = A$ + LEFT$ (C$,N)
3930 RETURN

```

Ces sous-programmes font appel à celui situé à la ligne 3960 pour l'affichage du message d'erreur « CHAINE TROP LONGUE »

```

3630 REM
3640 REM
3650 REM AFFICHAGE MESSAGE D'ERREUR
3660 REM CHAINE TROP LONGUE
3670 REM *****
3680 REM
3690 PRINT : INVERSE
3700 PRINT "PAS PLUS DE ";N;" CARACTERES, SVP"
3710 NORMAL
3720 RETURN

```

2.4.4 Affichage des données

• sur l'écran

```

3730 REM
3740 REM AFFICHAGE DES DONNEES SAISIES
3750 REM *****
3760 REM
3770 PRINT
3780 PRINT "NOM"; TAB( 20);NOM$
3790 PRINT "PRENOM"; TAB( 20);PRENOM$
3800 PRINT "ADRESSE"; TAB( 20);NUMERO$;" ";GENREVOIE$;" ";NVOIE$
3810 PRINT "VILLE"; TAB( 20);CDPST$;" ";VILLE$
3820 PRINT "TELEPHONE"; TAB( 20);"(";INDTEL$;")";TEL$
3830 RETURN

```

• sur imprimante

```

3730 REM
3740 REM IMPRESSION DES DONNEES SAISIES
3750 REM *****
3760 REM

```



```

3770 PR#1
3780 PRINT "NOM"; TAB( 20);NOM$
3790 PRINT "PRENOM"; TAB( 20);PRENOM$
3800 PRINT "ADRESSE"; TAB( 20);NUMERO$;" ";GENREVOIE$;" ";NVOIE$
3810 PRINT "VILLE"; TAB( 20);CDPST$;" ";VILLE$
3820 PRINT "TELEPHONE"; TAB( 20);"(";"INDTEL$;"");TEL$
3830 RETURN

```

2.4.5 Recherche d'une donnée

Trois étapes successives sont rencontrées :

1. choix de la façon de procéder
2. recherche dans le fichier
3. test de concordance
4. si le test n'est pas concluant, reprendre à l'étape 2.

```

650 REM *****
660 REM RECHERCHE DANS LE FICHIER (1)
670 REM *****
680 REM
690 HOME : INVERSE : PRINT " *** CONSULTATION ***": NORMAL
700 PRINT : PRINT : PRINT
710 PRINT "A PARTIR DE QUOI VA SE FAIRE LA RECHERCHE"
720 PRINT
730 PRINT "1-LE NOM"
740 PRINT "2-LE PRENOM"
750 PRINT "3-L'ADRESSE"
760 PRINT "4-LA VILLE"
770 PRINT "5-LE CODE POSTAL"
780 PRINT "6-LE NUMERO DE TELEPHONE"
790 PRINT : PRINT : INPUT "VOTRE CHOIX (ACCOLER LES DIVERS CHIFFRES) ";B$
800 RECH$(1,2) = "NOM A RECHERCHER"
810 RECH$(2,2) = "PRENOM A RECHERCHER"
820 RECH$(3,2) = "ADRESSE A RECHERCHER"
830 RECH$(4,2) = "VILLE A RECHERCHER"
840 RECH$(5,2) = "CODE POSTAL A RECHERCHER"
850 RECH$(6,2) = "NUMERO DE TELEPHONE A RECHERCHER"
860 RECH(1,2) = 15:RECH(2,2) = 15:RECH(3,2) = 15:RECH(4,2) = 15:
    RECH(5,2) = 5:RECH(6,2) = 7
870 B = LEN (B$)
880 FOR I = 1 TO 6
890 RECH(I,1) = 0
900 FOR J = 1 TO B
910 IF VAL ( MID$ (B$,J,1)) < > I THEN 950
920 RECH(I,1) = 1
930 PRINT RECH$(I,2): INPUT " ";A$
940 N = RECH(I,2): GOSUB 3840:RECH$(I,1) = A$
950 NEXT J

```

```

960 NEXT I
970 REM
980 REM RECHERCHE PROPREMENT DITE
990 REM *****
1080 REM
1090 REM TEST DE CONCORDANCE
1100 REM *****
1110 REM
1120 REM ARTICLE LU-DONNEES RECHERCHEES
1130 REM *****
1140 REM
1150 OK = 1
1160 IF RECH(1,1) = 0 THEN 1180
1170 OK = OK AND (NOM$ = RECH$(1,1))
1180 IF RECH(2,1) = 0 THEN 1200
1190 OK = OK AND (PRENOM$ = RECH$(2,1))
1200 IF RECH(3,1) = 0 THEN 1230
1210 A$ = NUMERO$ + " " + GENREVOIE$ + " " + NVOIE$
1220 OK = OK AND A$ = RECH$(3,1))
1230 IF RECH(4,1) = 0 THEN 1250
1240 OK = OK AND (VILLE$ = RECH$(4,1))
1250 IF RECH(5,1) = 0 THEN 1270
1260 OK = OK AND (CDPST$ = RECH$(5,1))
1270 IF RECH(6,1) = 0 THEN 1290
1280 OK = OK AND (TEL$ = RECH$(6,1))
1290 IF OK = 0 THEN 1370
1300 PRINT "1375"
1310 HOME : INVERSE : PRINT "DONNEES POSSIBLES": NORMAL
1320 PRINT : PRINT : PRINT : GOSUB 3770: PRINT
1330 PRINT "DESIREZ-VOUS MODIFIER OU SUPPRIMER"
1340 INPUT "CET ENREGISTREMENT (D/N) ";A$
1350 IF A$ < > "N" AND A$ < > "D" THEN 1330
1360 IF A$ = "D" THEN GOSUB 2140
1370 NEXT I
1380 RETURN

```

2.4.6 Modification/Suppression d'un article

```

2140 REM
2150 REM MODIFICATION/SUPPRESSION
2160 REM *****
2170 REM
2180 REM D'UN ENREGISTREMENT
2190 REM *****
2200 REM
2210 PRINT : PRINT "SOUHAITEZ-VOUS"
2220 PRINT "MODIFIER L'ARTICLE (1)"
2230 PRINT "SUPPRIMER L'ARTICLE (2)"
2240 INPUT "VOTRE CHOIX";A
2250 IF A < 1 OR A > 2 THEN 2210

```

```

2260 ON A GOTO 2270,2530
2270 REM
2280 REM MODIFIER UN ARTICLE
2290 REM *****
2300 REM
2310 PRINT : PRINT "QUE SOUHAITEZ-VOUS MODIFIER"
2320 PRINT : PRINT "1-LE NOM": PRINT "2-LE PRENOM": PRINT "3-L'ADRESSE":
PRINT "4-LA VILLE": PRINT "5-LE CODE POSTAL": PRINT "6-LE NUMERO DE
TELEPHONE"
2330 PRINT : INPUT "VOTRE CHOIX (1,2,3,4,5,6) ";A
2340 IF A < > 3 THEN 2390
2350 GOSUB 2970: REM GENREVOIE
2360 GOSUB 3070: REM NOM DE LA VOIE
2370 GOSUB 3170: REM NUMERO
2380 GOTO 2400
2390 ON A GOSUB 2770,2870,,3240,3340,3440.
2400 HOME : INVERSE : PRINT "INFORMATIONS SAISIES": NORMAL
2410 GOSUB 3830
2420 PRINT : PRINT "SOUHAITEZ-VOUS"
2430 PRINT "1-MODIFIER UNE AUTRE DONNEE DE L'ARTICLE"
2440 PRINT "2-ENREGISTRER LES MODIFICATIONS"
2450 PRINT "3-ABANDONNER LES MODIFICATIONS"
2460 INPUT "VOTRE CHOIX (1,2,3)";A
2470 ON A GOTO 2270,2480,2520
2480 REM
2490 REM ENREGISTREMENT DES MODIFICATIONS
2500 REM
2520 RETURN
2530 REM
2540 REM SUPPRESSION D'UN ARTICLE
2550 REM *****
2560 REM

```

CHAPITRE 3

Possibilités graphiques et sonores

3.1 INTRODUCTION

Votre système APPLE II offre de nombreuses possibilités graphiques et sonores, qui ajoutent un grand intérêt aux programmes dont vous disposez. Vous pouvez créer des jeux aussi divers que les échecs, le bridge, la guerre des étoiles, la simulation d'aéronef, etc.

Les deux modes graphiques suivants existent sur l'APPLE II :

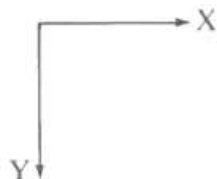
— Mode graphique basse résolution, où l'écran est divisé en 48 lignes de 40 colonnes et où les divers points apparaissent sous forme de rectangles sur l'écran (demi-caractère).

— Mode graphique haute résolution, où la définition est de 280 points (0 à 279) dans le sens horizontal et de 192 points (0 à 191) dans le sens vertical.

Avec une carte 80 colonnes plus 64 K octets d'extension mémoire, la résolution horizontale est portée à 560 points horizontalement et en noir et blanc.

Dans ces deux modes graphiques, l'axe des abscisses X est situé sur la partie supérieure de l'écran, et l'axe des ordonnées Y est situé sur la partie gauche de l'écran et dirigé vers le bas. Cette disposition est commune à tous les Basic réalisés par Microsoft.

De plus, l'APPLESOFT vous permet de créer des figures graphiques haute résolution préenregistrées, que vous pouvez ensuite facilement manipuler



3.2 LE MODE GRAPHIQUE BASSE RÉOLUTION

3.2.1 Entrée dans le mode

Pour passer du mode « texte » au mode graphique basse résolution, l'instruction GR efface l'écran, le positionne en mode mixte texte — graphisme (40 lignes graphiques et 4 lignes de texte en bas de l'écran) et place le curseur au début du texte.

Vous pouvez supprimer ces 4 lignes de texte avec l'instruction

`POKE - 16302,0`

L'écran obtenu comporte alors 48 lignes graphiques.

Vous pourrez recréer la fenêtre des 4 lignes du bas de l'écran en utilisant l'instruction

`POKE - 16301,0`

Pour revenir en mode texte, il vous suffira d'utiliser l'instruction TEXT.

Sur les vingt (ou les vingt-quatre) premières lignes de l'écran, apparaissent alors des arabesques. Ceci est tout à fait normal, l'APPLE II utilisant la même zone mémoire pour le texte et le mode graphique basse résolution.

3.2.2 Instructions graphiques

Les deux Basic de l'APPLE II acceptent les instructions graphiques basse résolution suivantes :

COLOR = expression

Sélection de la couleur dans laquelle seront faits les prochains affichages parmi les seize couleurs suivantes :

0. Noir
1. Magenta
2. Bleu foncé
3. Pourpre
4. Vert foncé
5. Gris
6. Bleu
7. Bleu clair
8. Brun
9. Orange
10. Gris
11. Rose
12. Vert
13. Jaune
14. Transparent
15. Blanc.

La valeur de l'expression doit être comprise entre 0 et 255 et est traitée modulo 16. Par exemple, l'instruction `COLOR = 40` sélectionnera la couleur jaune.

La couleur active est remise à zéro par l'instruction GR.

PLOT expression x, expression y

(0, 0) ×

+ → Affichage d'un « point » à l'intersection de la colonne × (0, 39)
et de la ligne y (0 à 47).

y

Si les coordonnées ne sont pas dans les intervalles indiqués, le Basic affichera le message d'erreur suivant :

ILLEGAL QUANTITY ERROR

et arrêtera l'exécution du programme.

Voici un exemple traçant une ligne diagonale sur l'écran

```
GR:COLOR=12:FOR X=0 TO 39:PLT X,39-X:NEXT X
```

HLIN X1,X2 AT Y

Tracé d'une ligne horizontale sur l'écran entre les « points » (X1,Y) et (X2,Y).

VLIN Y1,Y2 AT X

Tracé d'une ligne verticale sur l'écran entre les « points » (X,Y1) et (X,Y2).

L'exemple suivant vous montre le tracé d'un carré

```
10 REM TRACE D'UN CARRE
20 GR
30 COLOR=12
40 HLIN 20,30 AT 20
50 VLIN 20,30 AT 20
60 HLIN 20,30 AT 30
70 VLIN 20,30 AT 30
80 END
```

Vous remarquerez que les côtés du carré n'ont pas les mêmes épaisseurs (cf. Fig. 3.1). Cela est dû au fait qu'un « point » en mode graphique basse résolution est un « caractère » coupé en deux. Il en résulte qu'il forme un rectangle dont les côtés horizontaux sont doubles par rapport aux verticaux.

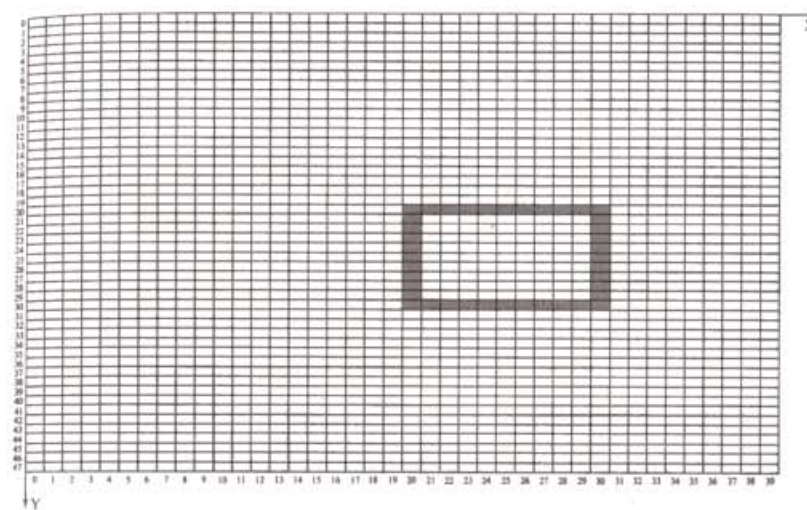


Figure 3.1 : Tracé d'un carré de coordonnées égales (abscisses et ordonnées)

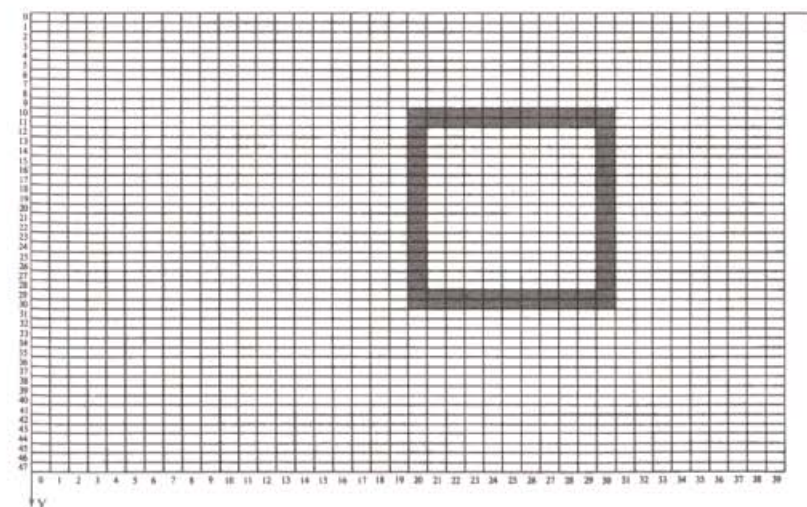


Figure 3.2 : Tracé réel d'un carré en mode basse résolution

Le programme suivant dessine un vrai carré sur l'écran :

```
10 REM TRACE D'UN CARRE
20 GR
30 COLOR=12
40 HLIN 20,30 AT 10
50 HLIN 20,30 AT 11
60 HLIN 20,30 AT 29
70 HLIN 20,30 AT 30
80 VLIN 10,30 AT 20
90 VLIN 10,30 AT 30
100 END
```

Essayez ! Le carré obtenu est plus normal (cf. fig. 3.2).

Fonction SCRN(X,Y)

Obtention du numéro de la couleur du point de coordonnées (X,Y). Remplacez la ligne 100 par la suivante :

```
100 COLOR=SCRN(20,30)+1:FOR I=1 TO 1000:
NEXT I:GO TO 40
```

Faites exécuter le programme. Vous voyez alors changer périodiquement la couleur du carré sur l'écran.

3.2.3 Utilisation de la mémoire

Deux pages (ou zones de mémoire) graphiques basse résolution existent sur l'APPLE II. La première page est celle utilisée par le Basic et le moniteur. La seconde n'est gérée ni par le moniteur, ni par le Basic. Il vous appartient, si vous désirez l'utiliser, de calculer l'adresse en mémoire des divers points. Vous pourrez vous reporter à cet effet au chapitre 4, sur le moniteur, pour connaître les adresses et les sous-programmes utilisables.

3.2.4 Exemple

Voici un programme Basic qui lit les manettes de jeux et affiche sur l'écran le point de coordonnées égal à la valeur lue sur les poignées, modulo 40.

```
10 REM AFFICHAGE DE POINTS DONT LES
20 REM COORDONNEES SONT LUES SUR LES MANETTES DE JEUX
30 REM
40 REM COPYRIGHT DE MERLY -
50 REM
60 DIM XY(1)
70 GR
80 VTAB 21:HTAB 1:CALL -958
90 COLOR=14
100 FOR I=0 TO 1
110 FOR J=0 TO 10:NEXT
120 XY(I)=PDL(I)
130 IF XY(I)>39 THEN XY(I)=39
140 NEXT
150 VTAB 21:PRINT "X= ";XY(0),"Y= ";XY(1)
160 PLOT XY(0),XY(1)
170 GOTO 100
```

Le fonctionnement de la fonction PDL est expliqué plus en détail au paragraphe 3.7.

3.3 LE MODE GRAPHIQUE HAUTE RÉOLUTION

3.3.1 Introduction

Dans ce mode graphique, le plus intéressant de l'APPLE II, l'écran est décomposé en points et non plus en rectangles et, comme en mode basse résolution, vous disposez de deux pages graphiques.

Les instructions graphiques sont uniquement incorporées dans le Basic APPLESOFT. Les utilisateurs du Basic Entier devront gérer eux-mêmes le mode haute résolution.

Les deux pages mémoires haute résolution ne sont pas protégées par le Basic APPLESOFT. Vous devez donc positionner les variables HIMEM et LOMEM. Les deux solutions suivantes sont possibles lorsque la taille de votre programme ne dépasse pas 12 K octets :

— Placer les variables de son programme utilisateur avant les pages graphiques en positionnant HIMEM à 8191 (16383) si vous désirez travailler sur la page un (deux).

— Placer les variables de son programme utilisateur après les pages graphiques en positionnant LOMEM à 16384 (24576) pour utiliser la page un (deux).

Si votre programme dépasse 12 K octets, vous devrez le placer après les pages graphiques (cf. § 2.2.16).

Selon la taille mémoire de votre système, vous ne pourrez pas toujours utiliser les pages graphiques. 16 K octets sont nécessaires pour la page un, 24 K octets pour la page deux. Si vous désirez conjointement disposer du DOS, vous devrez ajouter 12 K octets à votre calcul.

3.3.2 Entrée dans le mode

Pour positionner l'écran en mode graphique haute résolution sur la page un, l'instruction HGR efface l'écran, place 160 lignes graphiques en haut de celui-ci et 4 lignes de texte en dessous. Pour supprimer (réintroduire) celles-ci, vous devrez utiliser l'instruction :

POKE - 16302,0. (POKE - 16301,0)

En réalité, l'instruction HGR ne modifie pas la page mémoire contenant le texte, mais laisse seulement visibles les 4 lignes du bas. Le curseur ne sera visible que s'il est introduit dans ces lignes.

Il est possible de se placer dans le mode graphique haute résolution sans effacer l'écran, en utilisant les instructions suivantes :

POKE - 16304,0 passage en mode graphique
POKE - 16297,0 mode haute résolution
POKE - 16300,0 sélection de la page un.

Pour vous positionner sur la page deux, vous devrez utiliser l'instruction HGR2 au lieu de HGR. Pour passer en très haute résolution, on emploie l'instruction HGR3.

Lorsque l'on travaille sur la page deux haute résolution et que le programme rencontre l'instruction GR, il se produit le phénomène suivant : la page deux passe en basse résolution et la page un s'efface.

3.3.3 Instructions graphiques haute résolution

HCOLOR = expression

Les huit codes couleurs suivants existent en mode haute résolution :

- 0. Noir 1
- 1. Vert

- 2. Violet
- 3. Blanc 1
- 4. Noir 2
- 5. Orange
- 6. Bleu
- 7. Blanc 2

Nous avons volontairement isolé deux groupes de couleurs. En effet, afficher la couleur Blanc 1 au point de coordonnées (x, y) fera apparaître un point vert si x est pair, violet si x est impair, et blanc si les deux points (x, y) et (x + 1, y) sont allumés. Il en est de même pour le second groupe de couleurs. Ce phénomène est dû au mode de fonctionnement des télévisions couleurs.

HPlot x1, y1 TO x2, y2 To ...

Le graphisme haute résolution présente l'avantage de permettre le tracé de droites faisant un angle quelconque avec l'horizontale.

Plusieurs formats sont possibles :

HPlot x, y	éclairage du point de coordonnées x, y
HPlot x1, y1 TO x2, y2 (TO ...)	tracé des droites situées entre les points (x1, y1) et (x2, y2), (x2, y2) et...
HPlot TO x2, y2 (TO ...)	tracé de la droite située entre le dernier point éclairé et le point x2, y2.

Si aucun point n'a été éclairé auparavant, l'origine (0, 0) est prise par défaut. Les coordonnées du dernier point éclairé sont situées aux adresses \$E0-\$E2 (224-226).

Exemple :

```
HPlot 10, 10 TO 10, 20 TO 20, 20 TO 20, 10 TO 10, 10 TO
20, 20 TO 20, 10 TO 10, 20
```

Cette instruction trace un carré et ses diagonales.

3.3.4 Exemple

Voici un programme qui trace une spirale triangulaire sur l'écran. Il augmente les dimensions de cette spirale jusqu'à ce qu'elle occupe l'écran complètement, en changeant sa couleur à chaque tracé.


```

10 REM PROGRAMME DE DEMONSTRATION
20 REM DES POSSIBILITES GRAPHIQUES HAUTE RESOLUTION
30 REM DU BASIC APPLESOFT DE L'APPLE II
40 REM
50 REM COPYRIGHT DE MERLY
60 REM TRACE D'UNE SPIRALE TRIANGULAIRE
70 REM
80 COLOR= 5
90 XC = 140:YC = 96: REM CENTRE DE L'ECRAN
100 HGR : HCOLOR= 5: HPLLOT XC,YC
110 LP = 0:L = 5:ECART = 5
120 REM DEBUT DE LA BOUCLE
130 HCOLOR= COULEUR
140 HPLLOT TO XC + L,YC + LP
150 HPLLOT TO XC,YC - L
160 HPLLOT TO XC - L - ECART,YC + L
170 LP = L:L = L + ECART
180 COULEUR = COULEUR + 1
190 IF COULEUR = 8 THEN COULEUR = 0
200 IF YC + L < 193 THEN 130
210 END

```

Le résultat obtenu est présenté sur la figure 3.3.

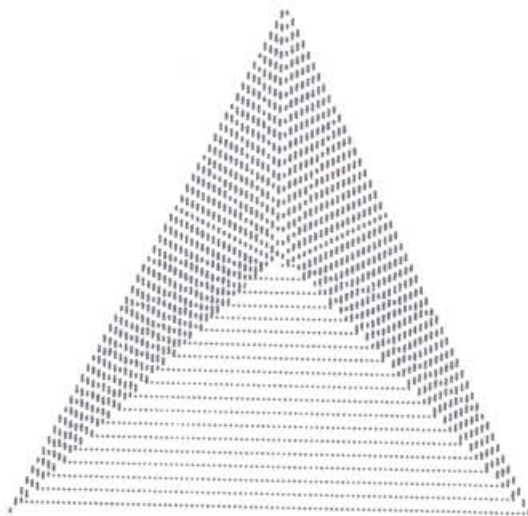


Figure 3.3

3.3.5 Tracé de courbes sur l'écran

Le tracé de courbes sur un écran pose principalement les deux problèmes suivants :

1. Détermination du facteur d'échelle nécessaire pour faire tenir la courbe dans l'écran.

2. Translation éventuelle des coordonnées pour obtenir un bon cadrage.

L'exemple ci-dessous réalise le tracé d'une cycloïde, courbe d'équations :

$$X = K(3 \cos(B) + \cos(3B))$$

$$Y = K(3 \sin(B) - \cos(3B))$$

```

10 REM TRACE D'UNE HYPO-CYCLOIDE A 4 REBOUSSEMENTS SUR L'ECRAN
20 REM
30 REM COPYRIGHT DE MERLY
40 REM
50 HGR
60 REM DIMENSIONNEMENT DE LA FIGURE
70 ECHELLE = 20
80 XC = 140
90 YC = 80
100 HCOLOR= 3
110 HPLLOT XC + 4 * ECHELLE,YC
120 FOR TETA = 0 TO 6.28 STEP 0.02
130 X = XC + ECHELLE * (3 * COS (TETA) + COS (3 * TETA))
140 Y = YC + ECHELLE * (3 * SIN (TETA) - SIN (3 * TETA))
150 HPLLOT TO X,Y
155 VTAB 21: CALL - 958: PRINT "X= "; INT (X),"Y= "; INT (Y)
160 NEXT
170 END

```

Le résultat obtenu est visible sur la figure 3.4.

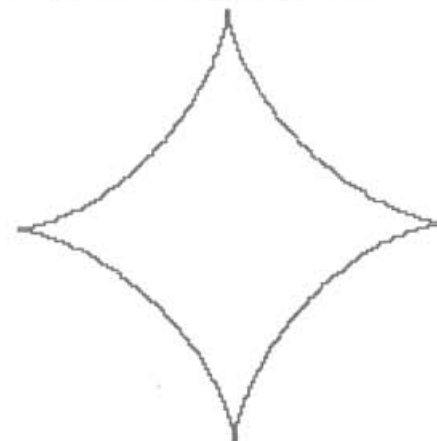


Figure 3.4

Le tracé d'un cercle peut être fait selon le même principe. Le listing est reproduit ci-dessous et la figure 3.5 représente le résultat obtenu.

```

10 REM TRACE D'UN CERCLE
20 REM
30 REM COPYRIGHT DE MERLY
40 REM
50 XC = 140:YC = 80:ECHELLE = 80
60 HGR : HCOLOR= 3
70 HPLLOT XC + ECHELLE,YC
80 FOR TETA = 0.02 TO 6.28 STEP 0.02
90 X = XC + ECHELLE * COS (TETA)
100 Y = YC + ECHELLE * SIN (TETA)
110 HPLLOT TO X,Y
120 VTAB 21: CALL - 958: VTAB 22
130 PRINT TAB( 14);"X="; INT (X); TAB( 23);"Y="; INT (Y)
140 NEXT
150 END

```

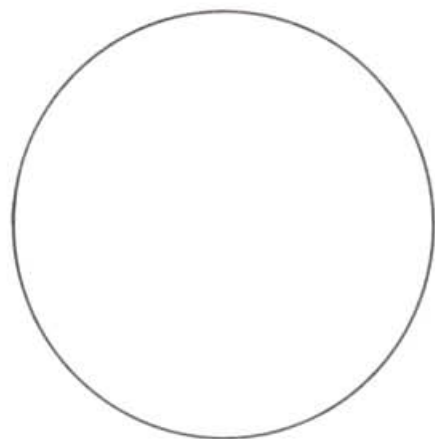


Figure 3.5

Les trois figures 3.3, 3.4 et 3.5 ont été obtenues en utilisant le programme de recopie d'écran résidant dans la carte d'interface de l'EPSON MXFT82.

Voici un exemple de tracé de courbe en coordonnées cartésiennes à la place des coordonnées polaires.

```

10 REM PROGRAMME DE TRACE DE COURBES
20 REM
30 REM COPYRIGHT DE MERLY
40 REM
50 TEXT : HOME
60 PRINT "CE PROGRAMME TRACE TOUTE COURBE"
70 PRINT "ACCESSIBLE AVEC LES FONCTIONS"
80 PRINT "MATHÉMATIQUES DE L'APPLESOFT."
90 PRINT "POUR CELA, TAPEZ VOTRE EXPRESSION"
100 PRINT "SELON LE FORMAT:"
110 PRINT " 1170: Y=F(X)"
120 PRINT
130 PRINT " EXEMPLE: 1010 Y=EXP(X)"
140 PRINT : PRINT "PUIS TAPEZ RUN 160"
150 END
160 PRINT "ENTREZ L'EQUATION SELON LE MEME FORMAT "
170 INPUT E$
180 DEF FN I(X) = INT (X + (1 - SGN (X)) / 2.00000001#)
190 REM
200 REM PARAMETRES DE LA COURBE
210 REM
220 HOME
230 PRINT E$
240 PRINT : PRINT ""
250 INPUT "X INITIAL -->";XD
260 PRINT
270 INPUT "X FINAL -->";XF
280 ONERR GOTO 1230
290 REM
300 REM CALCUL DES ECHELLES ET DES AXES
310 REM
320 IF XF < XD THEN T = XF:XF = XD:XD = T
330 IF XD < = 0 AND XF > 0 THEN XL = XF - XD:XC = - XD
340 IF XD < 0 AND XF < = 0 THEN XL = - XD:XC = - XD
350 IF XD > 0 AND XF > 0 THEN XL = XF:XC = 0
360 XI = XL / 279
370 XZ = XC / XI
380 PRINT : PRINT "TRAITEMENT EN COURS"
390 YD = 0:YF = 0
400 FOR X = XD TO XF STEP XI
410 GOSUB 1170
420 IF Y > YF THEN YF = Y
430 IF Y < YD THEN YD = Y
440 NEXT X
450 IF YD < = 0 AND YF > 0 THEN YL = YF - YD:YC = - YD
460 IF YD < 0 AND YF < = 0 THEN YL = - YD:YC = - YD
470 IF YD > 0 AND YF > 0 THEN YL = YF:YC = 0
480 YI = YL / 159

```



```

490 YZ = YC / YI
500 YZ = 159 - INT (YZ)
510 REM
520 REM AFFICHAGE DE L'EQUATION
530 REM
540 HGR
550 VTAB 22
560 IF LEN (E$) < 40 THEN HTAB (40 - LEN (E$)) / 2
570 PRINT E$
580 REM
590 REM TRACE DES AXES
600 REM
610 HCOLOR= 3
620 HPLLOT 0,YZ TO 279,YZ
630 HPLLOT XZ,0 TO XZ,159
640 REM
650 REM POINT DE DEPART
660 REM
670 X = XD
680 GOSUB 1170
690 Y = 159 - (Y + YC) / YI
700 HPLLOT (X + XC) / XI,Y
710 REM
720 REM TRACE DE LA COURBE
730 REM
740 FOR X = XD + XI TO XF STEP XI
750 GOSUB 1170
760 Y = 159 - (YC + Y) / YI
770 HPLLOT TO (X + XC) / XI,Y
780 NEXT X
790 REM
800 REM TRACE DE L'ECHELLE DE L'AXE DES X
810 REM
820 R = XL
830 GOSUB 1040
840 HCOLOR= 0
850 IF XD < 0 AND XF < 0 THEN XF = 0
860 IF XD > 0 AND XF > 0 THEN XD = 0
870 FOR X = FN I(XD / DI) * DI TO FN I(XF / DI) * DI STEP DI
880 HPLLOT XZ + X / XI,YZ
890 NEXT X
900 VTAB 24: PRINT DI: "/ DIVISION SUR X";
910 REM
920 REM TRACE DE L'ECHELLE DE L'AXE DES Y
930 REM
940 R = YL
950 GOSUB 1040
960 FOR Y = FN I(YD / DI) * DI TO FN I(YF / DI) * DI STEP DI
970 HPLLOT XZ,YZ - Y / YI
980 NEXT Y
990 PRINT SPC( 2);DI: "/DIVISION SUR Y";
1000 GOTO 1380
1010 REM

```

```

1020 REM DETERMINATION DES ECHELLES
1030 REM
1040 K = 0
1050 IF R < 1 THEN 1110
1060 IF R < 10 THEN 1140
1070 K = K + 1
1080 R = R / 10
1090 GOTO 1060
1100 IF R > 1 THEN 1140
1110 K = K - 1
1120 R = R * 10
1130 GOTO 1100
1140 DI = INT (R) * 10 ^ (K - 1)
1150 RETURN
1160 REM *****
1170 Y = 30
1180 REM *****
1190 RETURN
1200 REM
1210 REM TRAITEMENT DES ERREURS
1220 REM
1230 POKE 216,0: REM ANNULATION ONERR
1240 EA = PEEK (218) + PEEK (219) * 256
1250 E2$ = "ERREUR"
1260 N$ = "":E1$ = ""
1270 IF EA < > 1010 THEN N$ = "N'EST PAS ":E1$ = " A LA LIGNE " + STR$ (EA)
1280 HOME : VTAB 21
1290 ER = PEEK (222)
1300 IF E$ = "" THEN PRINT "ERREUR PAS D'EQUATION.": END
1310 PRINT "ERREUR DE SYNTAXE ";N$;"DANS L'EQUATION."
1320 PRINT E$
1330 IF EA < > 1010 THEN 1350
1340 IF ER = 16 OR ER = 163 OR ER = 224 THEN E2$ = "EQUATION FAUSSE"
1350 IF ER = 53 OR ER = 69 OR ER = 133 THEN E2$ = "DONES FAUSSES"
1360 IF ER = 255 THEN E2$ = "CONTROL-C"
1370 PRINT E2$ + E1$
1380 VTAB (15): PRINT CHR$ (7): GET N$: TEXT
1390 END

```

3.4 FIGURES GRAPHIQUES HAUTE RÉOLUTION

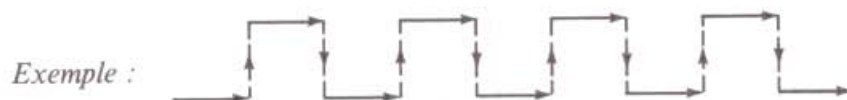
En mode graphique basse ou haute résolution, vous devez retracer toute figure dès que vous désirez la faire tourner ou que vous voulez modifier son échelle. Les figures graphiques haute résolution offrent ces possibilités.

3.4.1 Définition des figures graphiques

Une figure graphique haute résolution consiste en un ensemble de huit vecteurs de déplacement :

- Déplacement du curseur d'une position vers le haut sans tracé.
- Déplacement du curseur d'une position vers le bas sans tracé.
- Déplacement du curseur d'une position vers la droite sans tracé.
- Déplacement du curseur d'une position vers la gauche sans tracé.
- Déplacement du curseur d'une position vers le haut avec tracé du vecteur sur l'écran.
- Déplacement du curseur d'une position vers le bas avec tracé du vecteur sur l'écran.
- Déplacement du curseur d'une position vers la gauche avec tracé du vecteur sur l'écran.
- Déplacement du curseur d'une position vers la droite avec tracé du vecteur sur l'écran.

Dans la suite de ce paragraphe nous désignerons les « vecteurs sans tracé » par des flèches en pointillés et les « vecteurs avec tracé » par des flèches en trait plein.



3.4.2 Codage des figures graphiques

Le dessin précisé, vous devez le coder en machine. A cet effet, les vecteurs étant numérotés de zéro à sept, le codage nécessite trois bits par vecteur. Le premier bit représente le choix du tracé du vecteur ; les deux autres bits précisent le déplacement comme suit :

Vecteur	Code
→	01
↑	00
←	11
↓	10

Le codage des 8 vecteurs est le suivant :

Symbole	Action	Code binaire		Code décimal
↑	déplacement vers le haut sans tracé	0	00	0
→	déplacement vers la droite sans tracé	0	01	1
↓	déplacement vers le bas sans tracé	0	10	2
←	déplacement vers la gauche sans tracé	0	11	3
↑	déplacement vers le haut avec tracé	1	00	4
→	déplacement vers la droite avec tracé	1	01	5
↓	déplacement vers le bas avec tracé	1	10	6
←	déplacement vers la gauche avec tracé	1	11	7

Après avoir décomposé la figure en vecteurs et avoir codé ces derniers, vous devez regrouper les différents codes dans des octets. Un octet étant composé de 8 bits peut contenir deux codes de vecteur (avec/sans tracé) et un code de vecteur sans tracé (gauche, droite, bas) selon le format suivant :

Bit	7	6	5	4	3	2	1	0
	D	D	T	D	D	T	D	D

avec D étant un bit de déplacement

T étant un bit de tracé.

Examinons de plus près la gestion des bits 6 et 7. L'APPLESOFT ignore ces deux bits s'ils sont à zéro et considère ce vecteur comme tracé si un

au moins des deux bits est à un. Vous pouvez donc utiliser ces deux bits pour coder les vecteurs un à trois.

La fin d'une figure graphique est indiquée par un octet nul.

3.4.3 Assemblage d'une table de figures graphiques

Pour permettre le fonctionnement des instructions de l'APPLESOFT, les figures graphiques haute résolution doivent être regroupées dans une table qui pourra être stockée sur cassette ou disquette. L'organisation d'une table suit le format suivant :

Numéro d'octet	Signification
0	nombre n de figures dans la table (0 à 255)
1	inutilisé
2	poids faible déplacement de la figure n° 1 dans la table (numéro
3	poids fort du premier octet de la figure 1)
4	déplacement de la figure 2
5	
⋮	
2i+ 0	déplacement de la figure i
2i+ 1	
⋮	
2n+ 1	déplacement de la figure n
2n+ 2	définition de la figure graphique n° 1
	00 (fin de définition)
	⋮
	définition de la figure n
	00 (fin de la figure n)

Au cas où la table ne comporterait qu'une seule figure, elle aurait le format suivant :

Octet	Valeur
0	1
1	0
2	4
3	0
4	définition de la figure
⋮	00
⋮	

Pour éviter que l'APPLESOFT ne détruise la table créée, nous vous conseillons de la situer en mémoire au-dessus de HIMEM.

3.4.4 Programme de création d'une table de figures graphiques

Voici un exemple qui permet de saisir les vecteurs de déplacement, de les assembler dans une table, de stocker celle-ci sur disque, et d'afficher les figures obtenues sur l'écran :

Lignes 10 à 999 : programme principal
 100 à 250 : initialisations
 260 à 360 : saisie du nombre de figures de la table et contrôle
 370 à 530 : saisie de l'adresse de départ de la table, contrôle et conversion en base décimale
 535 à 600 : création du catalogue de la table
 605 à 640 : définition des figures
 650 à 750 : sauvegarde sur disquette si nécessaire
 800 à 999 : essai des figures graphiques.
 1 000 à 1 440 : sous-programme de saisie des vecteurs d'une figure
 1 997 à 2 040 : sous-programme de recopie de trois vecteurs en mémoire (codage d'un octet)

Lignes 3 000 à 3 080 : sous-programme de validité d'une chaîne numérique

Lignes 3 100 à 3 230 : sous-programme de validité d'une chaîne hexadécimale et conversion en base décimale

Lignes 10 000 et 10 001 : données de conversion de la base hexa en base décimale.

```

10 REM CREATION ET ESSAI D'UNE TABLE DE FIGURES
20 REM
30 REM COPYRIGHT DE MERLY
40 REM
100 HOME : VTAB 2
110 PRINT "-----"
120 PRINT
130 PRINT "      CREATION D'UNE TABLE DE FIGURES"
140 PRINT
150 PRINT "-----"
160 REM
170 REM INITIALISATIONS
180 REM
200 DIM TCHD$(16,2),CDECT(3)
210 FOR I = 1 TO 16
220   FOR J = 1 TO 2
230     READ TCHD$(I,J)
240   NEXT J
250   POKE 34,7
260 REM
270 REM LONGUEUR DE LA TABLE
280 REM
290 FOR I = 1 TO 1000: NEXT : VTAB 9: CALL - 958: VTAB 11:T1 = 0
300 INPUT "NOMBRE DE FIGURES DANS LA TABLE ";NB$
310 C$ = NB$:Y = 9: GOSUB 3000
350 ON T1 GOTO 290
360 NB = VAL (NB$)
370 REM
380 REM ADRESSE DE DEPART
390 REM
400 VTAB 13: PRINT "-----"
410 FOR I = 1 TO 1000: NEXT : VTAB 15: CALL - 958: VTAB 17:T1 = 0
420 INPUT "ADRESSE DE DEPART ";ADR$
430 REM
440 REM VALEUR DECIMALE OU HEXADECIMALE?
450 REM
460 T2 = 1
470 IF LEFT$(ADR$,1) = "$" THEN T2 = 2
480 T1 = 0
485 Y = 15:C$ = ADR$
490 ON T2 GOSUB 3000,3100
500 ON T1 GOTO 410
510 DEF FN MOD(X) = INT ((X / 256 - INT (X / 256)) * 256 + 0.05) * SGN
(X / 256)
520 ADR = S

```

```

530 VTAB 17: HTAB 19: PRINT ADR
540 POKE 232, FN MOD(ADR)
550 POKE 233, INT (ADR / 256)
560 POKE ADR,NB
570 POKE ADR + 1,0
580 POKE ADR + 2, FN MOD(2 * NB + 2)
590 POKE ADR + 3, INT ((2 * NB + 2) / 256)
600 ACR = ADR + 2 * NB + 2
610 FOR K = 1 TO NB
620   GOSUB 1000
630   IF K < > NB THEN POKE ADR + 2 * K + 2, FN MOD(ACR - ADR): POKE ADR
+ 2 * K + 3, INT ((ACR - ADR) / 256)
640 NEXT K
650 REM
660 REM SAUVEGARDE DE LA TABLE?
670 REM
680 CALL - 936: VTAB 10
690 PRINT "DESIREZ-VOUS SAUVEGARDER LA TABLE CREEE?";
700 GET T$
710 IF T$ < > "O" AND T$ < > "N" THEN 700
720 IF T$ = "N" THEN 800
730 INPUT "NOM DU FICHIER DE SAUVEGARDE:";T$
740 T$ = "BSAVE " + T$ + ".A" + STR$(ADR) + ".L" + STR$(ACR - ADR)
750 PRINT CHR$(4);T$
800 REM
810 REM ESSAI DES FIGURES
820 REM
830 VTAB 12: PRINT "-----": VTAB 14
840 PRINT "DESIREZ-VOUS ESSAYER VOS FIGURES ?";
850 GET T$
860 IF T$ < > "O" AND T$ < > "N" THEN 850
870 IF T$ = "N" THEN 950
880 HGR : HCOLOR= 3: SCALE= 1: ROT= 0
890 FOR K = 1 TO NB
900   DRAW K AT 140,80
910   FOR I = 1 TO 5000: NEXT
920   XDRAW K AT 140,80
930 NEXT
950 POKE 34,0: TEXT : HOME
999 END
1000 REM
1010 REM SAISIE DES VECTEURS D'UNE FIGURE
1020 REM
1030 REM ET CODAGE DANS LES OCTETS
1040 REM SITUES A L'ADRESS COURANTE ADR
1050 REM
1060 FOR I = 1 TO 1000: NEXT : HOME
1070 VTAB 9: PRINT "FIGURE NO ";K
1080 VTAB 11: PRINT "-----"
1090 V = 0:NO = 0: POKE 34,11: VTAB 13
1095 V = V + 1
1100 HTAB 8: PRINT "VECTEUR NO ";V;" (0-7) ?";
1120 GET T$: IF ASC (T$) < 48 OR ASC (T$) > 55 THEN 1120
1130 T = VAL (T$): PRINT T
1140 ON T GOTO 1250,1250,1250,1400,1400,1400,1400

```



```

1150 REM
1160 REM CAS DU VECTEUR NUL
1170 REM
1180 NO = NO + 1:CDVECT(NO) = 0:NZ = NZ + 1
1190 IF NO = 3 THEN GOSUB 2000:CDVECT(NO) = 0
1200 IF NZ = 3 THEN PRINT : INVERSE : POKE ACR,0:ACR = ACR + 1: PRINT "
      FIN DE LA FIGURE ";K: NORMAL : POKE 34,7: FOR I = 1 TO 1000: NEXT : RETURN

```

```

1210 GOTO 1095
1220 REM
1230 REM CAS D'UN VECTEUR SANS TRACE
1240 REM
1250 NO = NO + 1:CDVECT(NO) = T
1260 IF NO = 3 THEN GOSUB 2000
1270 NZ = 0
1280 GOTO 1095
1290 REM
1300 REM CAS D'UN VECTEUR AVEC TRACE
1310 REM
1400 NO = NO + 1
1410 IF NO = 3 THEN CDVECT(NO) = 0: GOSUB 2000
1420 CDVECT(NO) = T
1430 NZ = 0
1440 GOTO 1095

```

```

1997 REM
1998 REM RECOPIE DE TROIS VECTEURS EN MEMOIRE
1999 REM
2000 NN = CDVECT(1) + CDVECT(2) * 8 + CDVECT(3) * 64
2010 POKE ACR,NN
2020 ACR = ACR + 1
2030 NO = 1
2040 RETURN
3000 REM
3010 REM TEST DE VALIDITE D'UNE CHAINE "NUMERIQUE"
3020 REM
3030 FOR I = 1 TO LEN (C%)
3040 T = ASC ( MID$ ( C%,I,1) )
3050 IF T < 48 OR T > 57 THEN VTAB Y: PRINT CHR$ (7); TAB ( 8); : INVERSE
      : PRINT CHR$ (T); " EST UN CARACTERE ILLEGAL": NORMAL : T1 = 1:I = LEN
      (C%)
3060 NEXT
3070 S = VAL (C%)
3080 RETURN
3100 REM
3110 REM CONVERSION HEXADECIMALE --> DECIMALE
3120 REM
3130 S = 0:C% = MID$ (C%,2)
3140 FOR I = 1 TO LEN (C%)
3150 T = ASC ( MID$ (C%,I,1) )
3160 IF T < 48 OR T > 70 THEN VTAB Y: HTAB 8: INVERSE : PRINT CHR$ (T)
      : " EST UN CARACTERE ILLEGAL": NORMAL : T1 = 1:I = LEN (C%): GOTO 3220
3170 T$ = CHR$ (T)
3180 FOR J = 1 TO 16
3190 IF T$ = TCHD$(J,1) THEN T = VAL (TCHD$(J,2)):J = 16

```

```

3200 NEXT
3210 S = S + 16 ^ ( LEN (C%) - I ) * T
3220 NEXT
3230 RETURN
5000 REM
5010 REM TRAITEMENT DES ERREURS
5020 REM
5030 STOP
5999 RETURN
10000 DATA 0,0,1,1,2,2,3,3,4,4,5,5,6,6,7,7,8,8,9,9
10001 DATA "A",10,"B",11,"C",12,"D",13,"E",14,"F",15

```

3.4.5 Sauvegarde sur cassette

Pour recopier sur cassette votre table, calculez sa longueur x et tapez les instructions suivantes :

```

] CALL - 151 ← passage dans le moniteur
* 00:0x 00 ← positionnement de la longueur à l'adresse zéro
— mettez en route la cassette (PLAY)
— tapez 0.1W.

```

La table est alors stockée sur la cassette. Deux « bip » marquent la fin normale du stockage.

Si un message d'erreur s'affiche durant le stockage, reprenez les opérations au début, et si vous n'obtenez aucun succès, consultez le chapitre 4 consacré au moniteur (§ 4.3 : Les commandes du moniteur).

Pour relire la cassette et charger la table de figures en mémoire, le BASIC APPLESOFT possède la commande SHLOAD. Lors de l'utilisation de SHLOAD, HIMEM est diminuée de la longueur de la table et celle-ci est placée dans la zone mémoire ainsi libérée.

3.4.6 Sauvegarde sur disquette

La commande de sauvegarde d'une table sur disquette est la même que pour un sous-programme. Vous utiliserez l'instruction

BSAVE nom de fichier, Aa, Ll

où a est l'adresse décimale de la table en mémoire
l est la longueur de la table

pour sauvegarder la table sur disquette et

BLOAD nom de fichier, Aa

où a est l'adresse de chargement de la table

puis, POKE 232, octet de poids faible de l'adresse

POKE 233, octet de poids fort de l'adresse

Ces deux instructions POKE permettent de préciser à l'APPLESOFT l'adresse de la table des figures.

3.4.7 Instructions de travail sur les figures graphiques

L'APPLESOFT possède les quatre instructions suivantes, qui permettent de dessiner, de supprimer, d'orienter, d'agrandir ou de diminuer une figure graphique sur l'écran.

DRAW	dessin d'une figure
XDRAW	suppression d'une figure
ROT	orientation (rotation) de la figure sur l'écran
SCALE	modification de l'échelle de la figure sur l'écran

Ces instructions utilisent la page graphique haute résolution sélectionnée (1 ou 2) ainsi que la couleur sélectionnée par la dernière instruction HCOLOR.

● L'instruction DRAW

Elle permet de tracer sur l'écran une figure de la table désignée par son numéro dans la couleur, l'échelle et l'orientation choisies précédemment. Son format est le suivant :

DRAW	4	AT	125, 90
	↑		↑ ↑
	numéro		colonne ligne
	dans la table		de départ

Il est possible de ne pas préciser les coordonnées de départ de la figure ; celle-ci est alors dessinée à partir du dernier point atteint sur l'écran.

Si aucun point n'avait été atteint, le point pris par défaut est l'origine.

Si la table n'a pas été chargée correctement ou si vous avez oublié de préciser l'adresse à l'APPLESOFT, vous verrez s'afficher le message d'erreur suivant :

ILLEGAL QUANTITY ERROR

● L'instruction XDRAW

Cette instruction vous permet d'écraser une figure graphique présente à l'écran. Sa syntaxe est identique à celle de l'instruction DRAW.

XDRAW 12 AT 30, 15

XDRAW affiche en fait la figure dans la couleur complémentaire de la couleur actuelle (négatif).

Couleur	Couleur complémentaire
noir	blanc
blanc	noir
violet	vert
vert	violet
orange	bleu
bleu	orange

● L'instruction SCALE =

Elle permet de préciser l'échelle avec laquelle sera dessinée la figure graphique. SCALE doit toujours avoir été exécutée au moins une fois avant DRAW. Elle peut être utilisée en mode direct ou en mode programmé.

SCALE = 25

La valeur de l'échelle est comprise entre un (un point pour un vecteur) et deux cent cinquante-cinq (255 points pour un vecteur). La valeur zéro correspond à 256 points par vecteur.

Le mot SCALE n'est pas un mot réservé pour l'APPLESOFT. Seul est reconnu le mot SCALE =.

● L'instruction ROT =

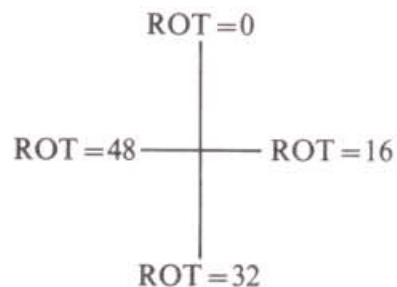
Elle permet de faire tourner les axes de la figure par rapport à l'axe vertical. Son format est le suivant :

ROT = valeur

où la valeur est un nombre compris entre 0 et 255 bien que 64 valeurs soient traitées (0 à 63). L'APPLESOFT utilise toujours la valeur maximale possible inférieure à la valeur indiquée.

Le nombre de valeurs possibles dépend en fait de l'échelle, de la façon suivante :

Echelle	Nombre de valeurs
1	4 (0, 16, 32, 48)
2	8
3	16
4	32
≥ 5	64 (0 à 63)



Comme pour l'instruction SCALE =, ROT n'est pas un mot réservé.

3.4.8 Utilisation de figures graphiques

Voici un programme qui crée un effet de zoom sur n'importe quelle figure graphique dont la taille est inférieure à celle d'un carré de 10 points par côté.

```

10 REM EFFET DE ZOOM
20 REM
30 REM COPYRIGHT DE MERLY - LAMOTIER
40 REM
50 HOME
60 INPUT "NOM DU FICHIER CONTENANT LA TABLE ";NOM$
70 PRINT
80 INPUT "ADRESSE DE CHARGEMENT (DECIMALE) ";ADR
90 PRINT CHR$(4);"BLOAD " + NOM$ + ".A" + STR$(ADR)
130 DEF FN MOD(X) = INT ((X / 256 - INT (X / 256)) * 256 + 0.05) * SGN
    (X)
140 POKE 232, FN MOD(ADR)
150 POKE 233, INT (ADR / 256)
160 HGR : POKE - 16302,0: HCOLOR= 3
170 FOR I = 1 TO PEEK (ADR)
180 FOR J = 1 TO 64
190 ROT= J
200 SCALE= 1 + J / 2
210 DRAW I AT 140,120
220 NEXT

```

La figure 3.6 représente cet effet sur un carré dont les côtés sont de longueur un (1 point), en changeant la ligne 200 comme suit :

200 SCALE=10+J/2

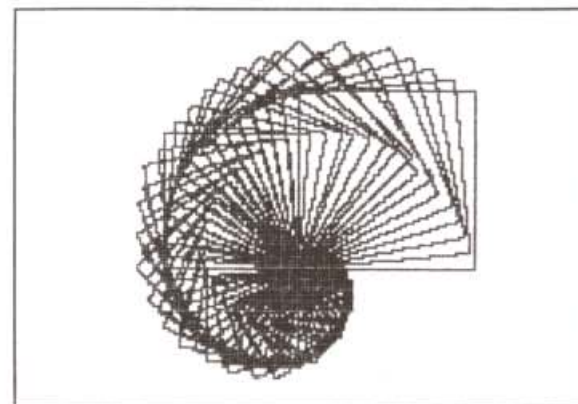


Figure 3.6

3.5 PÉRIPHÉRIQUES GRAPHIQUES

3.5.1 Périphériques de sortie

Deux types principaux de périphériques graphiques existent pour l'APPLE II. Ce sont les imprimantes graphiques et les tablettes traçantes.

La définition de l'image obtenue est de :

- 480 points par ligne, pour l'imprimante thermique SILENTYPE (APPLE),
- 480 points par ligne en mode normal ou 960 points par ligne en mode double densité, pour l'imprimante EPSON (précision de 5/10 mm ou de 2,5/10 mm).

Pour une table traçante de bas de gamme connectable à l'APPLE II *via* une interface parallèle, la précision peut atteindre 1/10 mm.

Ces cartes ne modifient pas la définition de l'écran.

3.5.2 Périphériques d'entrées

— Carte DIGISECTOR permettant de recevoir les images provenant d'une caméra vidéo, de les digitaliser, et de les traiter, par exemple, dans des applications de reconnaissance des formes.

— Tablette graphique (APPLE) pour tracer un dessin à l'aide d'un stylet magnétique, le digitaliser, et le ranger dans une page mémoire haute résolution afin de le reproduire sur l'écran.

— Système trois dimensions dont le but est de déterminer les coordonnées spatiales du point extrémité d'un stylet, de suivre des contours d'objets et de les voir représentés en perspective 3D sur l'écran de l'APPLE II (Micro control systems).

Nous reviendrons sur ces cartes au tome 2.

3.6 POSSIBILITÉS SONORES DE L'APPLE II

Sous le clavier de l'APPLE II est situé un haut-parleur géré par le moniteur. Son emploi est très simple; vous devez définir les notes émises, en faisant varier la fréquence et la durée.

3.6.1 Mise en œuvre

Le haut-parleur utilise l'adresse \$C030(-16336). Chaque accès à cette adresse provoque l'émission d'un « bip ».

En Basic, la fonction utilisable est PEEK(-16336). La fréquence obtenue est faible. Nous vous donnons ci-dessous un exemple de sous-programme assembleur de gestion du haut-parleur.

3.6.2 Emission d'une note de durée et de fréquence donnée

Plusieurs sous-programmes du moniteur gèrent le haut-parleur pour certaines fréquences. Ils sont décrits au paragraphe 4.3.4.

L'exemple proposé appelle le sous-programme d'attente du moniteur WAIT situé à l'adresse \$FCA8(-856).

0300	1	ORG \$300	
0300	2	WAIT EQU \$FCAB	; ATTENTE D'UN DELAI
0300	3	HP EQU \$C030	; HAUT-PARLEUR
0301	4	FREQ DFS #1	
0302	5	DUREE DFS #1	
0302 AC0003	6	NOTE LDY FREQ	
0305 AE0003	7	ET1 LDX FREQ	
0308 A904	8	LDA ##04	
030A 20ABFC	9	JSR WAIT	
030D AD30C0	10	LDA HP	
0310 EB	11	ET2 INX	
0311 D0FD	12	BNE ET2	
0313 8B	13	DEY	
0314 D0EF	14	BNE ET1	
0316 CE0103	15	DEC DUREE	
0319 D0E7	16	BNE NOTE	
031B 60	17	RTS	
	18	END	

Listing d'assemblage : Génération d'un son

Pour utiliser ce sous-programme en Basic, il suffit de :

- mettre aux adresses
 - \$300 fréquence (0-255)
 - \$301 durée (0-255)
- d'exécuter l'instruction
 - CALL 700 (\$302)

Exemple :

```
5 PRINT "BLOAD BSON"
10 FOR I = 1 TO 255
20 POKE 768, I
25 POKE 769, I
30 CALL 770
40 NEXT
```

3.7 MANETTES DE JEUX ET JOYSTICK

Les manettes de jeux vous permettent de fournir à l'APPLE II des données continues (analogiques) sans que vous ayez à taper au clavier.

Chaque poignée est composée d'un potentiomètre réglable (bouton central) et d'un interrupteur (bouton poussoir). Ce dernier permet par exemple de valider les données lues sur les manettes.

En Basic, la fonction PDL(I) fait la lecture de la manette numéro I (0-1) ; comme cela vous sera expliqué au chapitre 4, entre chaque lecture d'une valeur sur l'une quelconque des manettes doit s'écouler un temps minimum qui correspond approximativement à celui des instructions :

```
FOR I=1 TO 10 :NEXT I
```

Examinez l'exemple du paragraphe 3.2.4. Ces instructions ne servent apparemment à rien, mais sont en fait indispensables.

L'accès au bouton (poussoir) se fait à l'aide de la fonction PEEK aux adresses \$C061 (manette 0) et \$C062 (manette 1). Si la valeur obtenue est supérieure à 128, le bouton correspondant est enfoncé.

Le JOYSTICK est l'équivalent pour l'APPLE II de deux manettes de jeux. Pour l'utilisateur, il est composé d'un « manche » pouvant se déplacer dans toutes les directions (360 degrés). Il est très pratique dans les jeux où le mouvement est plan (modification des coordonnées X et Y simultanément).

CHAPITRE 4

La programmation en assembleur 6502 et le moniteur de l'APPLE II

4.1 INTRODUCTION

Dans les mémoires ROM de l'APPLE résident en permanence le langage Basic (Basic Entier ou APPLESOFT) et un programme de contrôle appelé le moniteur. Ce programme est écrit en langage d'assemblage et il assure les entrées-sorties standard (gestion de l'écran et du clavier). Il permet de créer des programmes en langage d'assemblage, de consulter et de modifier le contenu de la mémoire, etc...

Après avoir décrit le microprocesseur 6502 (structure interne et instructions) et les techniques de base de programmation en assembleur, nous étudierons le moniteur, ses commandes, les sous-programmes accessibles à l'utilisateur et la carte-mémoire. Le moniteur de l'APPLE II étant

performant, il est utile de le connaître en profondeur lorsque l'on veut travailler efficacement en assembleur et même en Basic.

4.2 LA PROGRAMMATION EN ASSEMBLEUR 6502

4.2.1 Introduction

Le possesseur d'un APPLE II qui veut réaliser des programmes de qualité finira nécessairement par s'intéresser au langage d'assemblage du microprocesseur 6502. En effet, le Basic est à utiliser pour de petites applications non répétitives, mais il paraît exaspérant par sa lenteur pour des calculs très répétitifs. Vous pourriez alors penser faire du Pascal, LISP, Forth. Mais pour de petits sous-programmes, vous gagnerez du temps et de l'argent en programmant en Assembleur 6502. Il vous faut pour cela un Assembleur (programme traduisant les mnémoniques en code machine). Plusieurs possibilités vous sont offertes :

- soit utiliser le miniassembleur en ROM de l'ancien moniteur. L'intérêt réside dans sa facilité de mise en œuvre, mais vous ne pourrez pas conserver de commentaires,

- soit récupérer le miniassembleur dans le fichier INTBASIC fourni avec le DØS 3.3,

- soit acheter le miniassembleur sur disquette pour l'APPLE II +,

- soit acquérir un assembleur puissant tel le LISA, qui a l'énorme avantage de détecter les erreurs de syntaxe lorsque vous tapez les instructions (comme le Basic Entier). Il peut ainsi travailler très vite.

Le but n'est pas de vous apprendre le maniement de tel ou tel assembleur. Le lecteur intéressé se reportera au paragraphe 4.3.3 pour le miniassembleur, ou à la documentation correspondante pour LISA.

Nous allons décrire la structure du microprocesseur 6502, ses spécificités par rapport aux autres microprocesseurs 8 bits existant, et ses mnémoniques.

4.2.2 Le microprocesseur 6502

Il joue dans l'APPLE II les fonctions d'unité centrale en un seul boîtier. Il comprend une unité arithmétique et logique (UAL), pour les calculs, et une unité de commande chargée de séquencer le système APPLE II ;

il communique avec les autres boîtiers par des ensembles de fils appelés bus. Ceux-ci sont au nombre de trois :

- le bus d'adresses à 16 fils qui permet de sélectionner une case mémoire parmi 65 536 cases au total (2 possibilités = 65 536),
- le bus de données à 8 fils (8 bits) qui permet de transférer les données entre les différents boîtiers,
- le bus de commande utilisé pour synchroniser le système.

En ouvrant votre APPLE II, vous verrez :

- le microprocesseur 6502,
- huit boîtiers de mémoires RAM (lecture et écriture d'informations),
- deux mémoires ROM (lecture d'informations seulement) qui contiennent le moniteur et le Basic.

- sept connecteurs pour des cartes d'extension,
- d'autres boîtiers gérant le clavier, l'écran... (cf. photo C. chap. 1).

Le microprocesseur 6502 a la structure interne suivante :

- une unité arithmétique et logique (UAL),
- un registre appelé accumulateur (A) de 8 bits,
- deux registres d'index X et Y de 8 bits,
- un pointeur de pile de 9 bits S,
- un registre d'états P,
- un pointeur de programme (PC) de 16 bits (ou compteur ordinal), vers la prochaine instruction à exécuter.

● L'accumulateur

L'UAL est relié à un registre spécial, l'accumulateur. Elle se réfère automatiquement à lui comme à l'une de ses entrées. Il y a toutefois possibilité de le court-circuiter. Dans les opérations arithmétiques ou logiques, un des opérandes sera une case mémoire, et l'autre le contenu de l'accumulateur. Le résultat est ensuite stocké dans l'accumulateur. Cette architecture

6502	Intel 8080-Zilog Z80
instructions très rapides, mais peu puissantes	instructions plus longues en taille et durée, mais puissantes

interne est classique : elle permet d'avoir des instructions courtes d'un octet, très rapides à exécuter. Par contre, l'inconvénient réside dans le fait que les données doivent être préalablement chargées.

Examinons plus en détail le rôle des autres registres, pris séparément.

● Le pointeur programme-exécution d'une instruction

Supposons qu'à la fin d'une instruction, le PC contienne sur 16 bits l'adresse de la prochaine instruction. Faisons exécuter celle-ci pas à pas en observant ce qui se passe.

Tout processeur procède en trois cycles :

● Recherche de la prochaine instruction

Le contenu du pointeur programme (ou compteur ordinal) est placé sur le bus d'adresses. La mémoire sélectionnée dépose alors l'instruction sur le bus de données.

● Décodage et exécution

Le microprocesseur examine l'instruction pour générer les signaux de commande. On dit qu'il la « décode ». Selon l'instruction, tout se déroule dans le 6502 ou des accès à la mémoire sont à faire, ce qui explique les différences de temps de traitement nécessaire.

● Recherche de la prochaine instruction

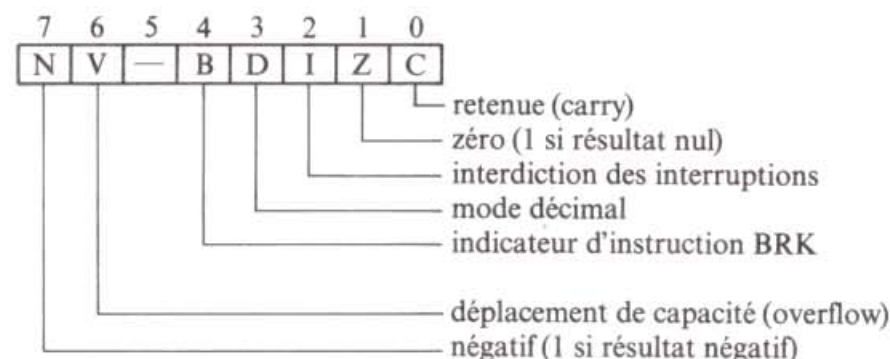
A chaque accès à la mémoire, pour chercher les octets correspondant à l'instruction (un pour le code de l'instruction, un à deux pour les opérandes), le PC est incrémenté. De cette façon, à la fin d'une instruction, il contient toujours l'adresse de la suivante.

● Le registre d'états P

Il comporte 8 bits d'indicateurs d'états. Chacun des bits est utilisé pour représenter une condition particulière (égalité ou non, nullité ou non, positif ou nul ou non, retenue ou non...).

Ces bits sont testés lors des instructions de branchement conditionnel (BMI, BPL, BVC, BCC, BCS, BEQ, BVS...).

L'instruction IF condition GOTO du Basic est obtenue en positionnant le bit correspondant à la condition et en testant ce bit par l'instruction de branchement conditionnelle correspondante.



Contenu du registre d'état

● La pile et le pointeur de pile S. La pagination

Une pile est un ensemble de registres ou de cases mémoires organisé en structure LIFO (dernier entré, premier sorti). Le dernier élément introduit est toujours au sommet de la pile (la comparaison est facile avec une pile d'assiettes par exemple). La présence d'une pile est quasiment indispensable pour gérer les sous-programmes, les interruptions, ou pour stocker un grand nombre de données de manière temporaire.

L'approche suivie par le 6502 est celle d'une pile logicielle (cases mémoires). Le registre S contient l'adresse du sommet de la pile. A la différence des microprocesseurs Intel 8080 ou Zilog Z80, la pile du 6502 est de taille finie. Le registre S ayant une taille de 9 bits, S peut adresser la mémoire comprise entre 256 et 511, avec le bit de gauche à 1. Cette zone mémoire est réservée à cet effet dans l'APPLE II.

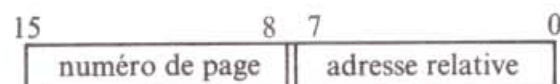
La pile correspond à la page 1 de la mémoire du 6502. Celle-ci est divisée en pages de 256 octets. C'est le concept de pagination. Nous verrons au paragraphe 4.3.4, « description de la carte mémoire », l'utilisation de ces différentes pages pour l'APPLE II +.

● Modes d'adressage du 6502. Registres d'index X et Y

Le microprocesseur 6502 admet les modes d'adressage suivant :

— adressage implicite : sélection des divers registres utilisés par les instructions ;

- adressage immédiat : chargement d'un registre avec une valeur numérique définie à l'avance ;
- adressage absolu : adresses sur 16 bits ;
- adressage en page zéro : adresses sur 8 bits réservées aux données auxquelles on veut pouvoir accéder très rapidement ;
- adressage relatif : l'adresse indiquée est relative à un début de page ;



— adressage indexé : le 6502 ne dispose pas d'un adressage indexé tout à fait général. Il possède deux registres d'index X et Y de 8 bits. Le contenu d'un registre d'index est ajouté si nécessaire à la partie adresse des instructions. Ils permettent d'accéder facilement aux divers éléments d'un tableau ; il existe pour cela des instructions d'incréméntation ou de décréméntation de ces registres, et de test de leurs valeurs par rapport à des contenus de cases mémoires (critères d'arrêt).

L'exemple suivant montre le transfert de N octets entre les adresses DEP et ARR (début de bloc).

```

LDX #N      ; chargement du nombre dans X
PROCH LDA DEP, X ; lecture
STA ARR, X ; copie
DEX          ; X = X - 1
BNE PROCH ; branchement si non terminé

```

— adressage indirect : il utilise toujours la page zéro et doit être indexé, soit avant l'indirection, soit après.

4.2.3 Tableau des instructions du 6502

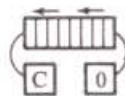
Mnémonique et description	Opération	Mode d'adressage	Forme assembleur	Code opération	Flags modifiés
ADC addition de l'accumulateur et du contenu d'une case mémoire et de la retenue (carry)	$A \leftarrow A + M + C$	immédiat page zéro page zéro, X absolu absolu, X absolu, Y (indirect, X) (indirect, Y)	ADC # OPER ADC OPER ADC OPER, X ADC OPER ADC OPER, X ADC OPER, Y ADC (OPER, X) ADC (OPER), Y	69 XX 65 XX 75 XX 6D XXXX 7D XXXX 79 XXXX 61 XX 71 XX	N Z V
AND et logique de l'accumulateur et du contenu d'une case mémoire	$A \leftarrow A \wedge M$	immédiat page zéro page zéro, X absolu absolu, X absolu, Y (indirect, X) (indirect, Y)	AND # OPER AND OPER AND OPER, X AND OPER AND OPER, X AND OPER, Y AND (OPER, X) AND (OPER), Y	29 XX 25 XX 35 XX 2D XXXX 3D XXXX 39 XXXX 21 XX 31 XX	N Z
ASL déplacement logique vers la gauche de l'accumulateur ou du contenu d'une case mémoire		accumulateur page 0 page 0, X absolu absolu, X	ASL A ASL OPER ASL OPER, X ASL OPER ASL OPER, X	0A 06 XX 16 XX 0E XXXX 14 XXXX	C N Z
BCC branchement s'il n'y a pas de retenue	branchement si C = 0	relatif	BCC OPER	90 XX	
BCS branchement sur retenue	branchement si C = 1	relatif	BCS OPER	B0 XX	
BEQ branchement sur résultat nul	branchement si Z = 1	relatif	BEQ OPER	F0 XX	
BIT tests de bits de la mémoire avec l'accumulateur	$Z \leftarrow A \wedge M$ $N \leftarrow M7$ $V \leftarrow M6$	page zéro absolu	BIT OPER BIT OPER	24 XX 2C XXXX	N = M7 V = M6 Z = A ∧ M
BMI branchement sur résultat négatif	branchement si Z = 1	relatif	BMI OPER	30 XX	
BNE branchement sur résultat non nul	branchement si Z = 0	relatif	BNE OPER	D0 XX	

Tableau des instructions du 6502 (suite)

Mnémonique et description	Opération	Mode d'adressage	Forme assembleur	Code opération	Flags modifiés
BPL branchement sur résultat positif si N=0	branchement si N=0	relatif	BPL OPER	10 XX	
BRK création d'une interruption	forçement d'une interruption	implicite	BRK	00	I←1
BVC branchement si aucun dépassement ne s'est produit	branchement si V=0	relatif	BVC OPER	50 XX	
BVS branchement si un dépassement s'est produit	branchement si V=1	relatif	BVS OPER	70 XX	
CLC remise à zéro du bit de retenue	C←0	implicite	CLC	18	C←0
CLD remise à zéro du bit décimal	D←0	implicite	CLD	D8	D←0
CLI autorisation des interruptions	I←0	implicite	CLI	58	I←0
CLV remise à zéro du bit de dépassement (overflow)	V←0	implicite	CLV	B8	V←0
CMP comparaison des contenus de l'accumulateur et d'une case mémoire. (Le 6502 fait A-M et positionne les bits N, Z, C, selon le résultat, il ne touche pas à l'accumulateur en écriture)	A-M	immédiat page zéro page zéro, X absolu absolu, X absolu, Y (indirect, X) (indirect), Y	CMP #OPER CMP OPER CMP OPER, X CMP OPER CMP OPER, X CMP OPER, Y CMP (OPER, X) CMP (OPER), Y	C9 XX C5 XX D5 XX CD XXXX DD XXXX D9 XXXX C1 XX D1 XX	N Z C
CPX comparaison des contenus de X et d'une case mémoire	X-M	immédiat page zéro absolu	CPX #OPER CPX OPER CPX OPER	E0 XX E4 XX EC XXXX	N Z C
CPY comparaison des contenus de Y et d'une case mémoire	Y-M	immédiat page zéro absolu	CPY #OPER CPY OPER CPY OPER	C0 XX C4 XX CC XXXX	N Z C

Tableau des instructions du 6502 (suite)

Mnémonique et description	Opération	Mode d'adressage	Forme assembleur	Code opération	Flags modifiés
DEC décrémenter le contenu d'une case mémoire de 1	M←M-1	page zéro page zéro, X absolu absolu, X	DEC OPER DEC OPER, X DEC OPER DEC OPER, X	C6 XX D6 XX CE XXXX DE XXXX	N Z
DEX décrémenter le registre de l	X←X-1	implicite	DEX	CA	N Z
DEY décrémenter le registre de l	Y←Y-1	implicite	DEY	88	N Z
EOR ou exclusif logique entre les contenus de l'accumulateur et d'une case mémoire	A←A⊕M	immédiat page zéro page zéro, X absolu absolu, X absolu, Y (indirect, X) (indirect), Y	EOR #OPER EOR OPER EOR OPER, X EOR OPER EOR OPER, X EOR OPER, Y EOR (OPER, X) EOR (OPER), Y	49 XX 45 XX 55 XX 4D XXXX 5D XXXX 59 XXXX 41 XX 51 XX	N Z
INC incrémenter de 1 le contenu d'une case mémoire	M←M+1	page zéro page zéro, X absolu absolu, X	INC OPER INC OPER, X INC OPER INC OPER, X	E6 XX F6 XX EE XXXX FE XXXX	N Z
INX incrémenter le registre index X de 1	X←X+1	implicite	INX	E8	N Z
INY incrémenter le registre index Y de 1	Y←Y+1	implicite	INY	C8	N Z
JMP branchement inconditionnel	OPER PC←(OPER)	absolu indirect	JMP OPER JMP (OPER)	4C XXXX 6C XXXX	
JSR appel de sous-programme : — sauvegarde PC+2 (prochaine instruction au retour) — branchement inconditionnel	PC+2 ↓ pile PC←OPER	absolu absolu	JSR OPER	20 XXXX	

Tableau des instructions du 6502 (suite)

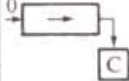
Mnémonique et description	Opération	Mode d'adressage	Forme assembleur	Code opération	Flags modifiés
LDA chargement du contenu de l'accumulateur	$A \leftarrow M$	immédiat page zéro page zéro, X absolu absolu, X absolu, Y absolu, Y (indirect, X) (indirect), Y	LDA #OPER LDA OPER LDA OPER, X LDA OPER LDA OPER, X LDA OPER, Y LDA (OPER, Y) LDA (OPER), Y	A9 XX A5 XX B5 XX AD XXXX BD XXXX B9 XXXX A1 XX B1 XX	N Z
LDX chargement du contenu du registre d'index X	$X \leftarrow M$	immédiat page zéro page zéro, Y absolu absolu, Y	LDX #OPER LDX OPER LDX OPER, Y LDX OPER LDX OPER, Y	A2 A6 B6 AE BE	N Z
LDY chargement du contenu du registre d'index Y	$Y \leftarrow M$	immédiat page zéro page zéro, X absolu absolu, X	LDY #OPER LDY OPER LDY OPER, X LDY OPER LDY OPER, X	A0 XX A4 XX B4 XX AC XXXX BC XXXX	N Z
LSR décalage d'un cran vers la droite du contenu de — l'accumulateur — d'une case mémoire		accumulateur page zéro page zéro, X absolu absolu, X	LSR A LSR OPER LSR OPER, X LSR OPER LSR OPER, X	4A XX 46 XX 56 XXXX 4E XXXX 5E XXXX	N=0 Z C
NOP instruction blanche (utilisée pour des délais par exemple)	vide	implicite	NOP	EA	
ORA ou logique des contenus de l'accumulateur et d'une case mémoire	$A \leftarrow A + M$ v	immédiat page zéro page zéro, X absolu absolu, X absolu, Y absolu, Y (indirect, X) (indirect), Y	ORA OPER ORA OPER ORA OPER, X ORA OPER ORA OPER, X ORA OPER, Y ORA (OPER, X) ORA (OPER), Y	09 XX 05 XX 15 XX 0D XXXX 1D XXXX 19 XXXX 01 XX 11 XX	N Z
PHA sauvegarde de l'accumulateur dans la pile	$A \downarrow \text{pile}$	implicite	PHA	48	

Tableau des instructions du 6502 (suite)


Mnémonique et description	Opération	Mode d'adressage	Forme assembleur	Code opération	Flags modifiés
PHP sauvegarde du registre d'états P dans la pile	$P \downarrow \text{pile}$	implicite	PHP	08	
PLA restauration de l'accumulateur à partir de la pile	$A \uparrow \text{pile}$	implicite	PLA	68	
PLP restauration du registre d'états P à partir de la pile	$P \uparrow \text{pile}$	implicite	PLP	28	P dépilé
ROL rotation circulaire d'un cran via le bit de retenue vers la gauche du contenu de l'accumulateur ou d'une case mémoire		accumulateur page zéro page zéro, X absolu absolu, X	ROL A ROL OPER ROL OPER, X ROL OPER ROL OPER	2A XX 26 XX 36 XX 2E XXXX 3E XXXX	N Z C
ROR rotation circulaire d'une position via le bit de retenue vers la droite du contenu de l'accumulateur ou d'une case mémoire		accumulateur page zéro page zéro, X absolu absolu, X	ROR A ROR OPER ROR OPER, X ROR OPER ROR OPER, X	6A XX 66 XX 76 XX 6E XXXX 7E XXXX	N Z C
RTI retour d'interruption	$P \uparrow PC \uparrow$	implicite	RTI	40	P restauré
RTS retour de sous-programme	$PC \uparrow$ $PC \leftarrow PC + 1$	implicite	RTS	60	
SBC soustraction au contenu de l'accumulateur du contenu d'une case mémoire et du bit de retenue	$A \leftarrow A - M - C$	immédiat page zéro page zéro, X absolu absolu, X absolu, Y absolu, Y (indirect, X) (indirect), Y	SBC OPER SBC OPER SBC OPER, X SBC OPER SBC OPER, X SBC OPER, Y SBC (OPER, X) SBC (OPER), Y	E9 XX E5 XX F5 XX ED XXXX FD XXXX F9 XXXX E1 XX F1 XX	
SEC positionnement à 1 du bit de retenue	$C \leftarrow 1$	implicite	SEC	38	C=1

Tableau des instructions du 6502 (suite)

Mnémonique et description	Opération	Mode d'adressage	Forme assembleur	Code opération	Flags modifiés
SED passage en mode décimal	D ← I	implicite	SED	F8	C = 1
SEI interdiction des interruptions	I ← I	implicite	SEI	78	I = 1
STA stockage du contenu de l'accumulateur dans une case mémoire	M ← A	page zéro page zéro, X absolu absolu, X absolu, Y (indirect, X) (indirect, Y)	STA OPER STA OPER, X STA OPER STA OPER, X STA OPER, Y STA (OPER, X) STA (OPER), Y	85 XX 95 XX 8D XXXX 9D XXXX 89 XXXX 81 XX 91 XX	
STX stockage du contenu du registre index X dans une case mémoire	M ← X	page zéro page zéro, Y absolu	STX OPER STX OPER, Y STX OPER	86 XX 96 XX 8E XXXX	
STY stockage du contenu du registre index Y dans une case mémoire	M ← Y	page zéro page zéro, X absolu	STY OPER STY OPER, X STY OPER	84 XX 94 XX 8C XXXX	
TAX chargement dans le registre d'index X du contenu de l'accumulateur	X ← A	implicite	TAX	AA	M Z
TAY chargement dans le registre Y du contenu de l'accumulateur	Y ← A	implicite	TAY	A8	N Z
TSX chargement dans le registre d'index X du pointeur de pile S	X ← S	implicite	TSX	BA	N Z
TXA transfert dans l'accumulateur du registre d'index X	A ← X	implicite	TXA	8A	N Z
TXS transfert dans le pointeur de pile S du registre d'index X	S ← X	implicite	TXS	9A	
TYA transfert dans l'accumulateur du contenu du registre d'index Y	A ← Y	implicite	TYA	98	N Z

4.2.4 Techniques de base de programmation en assembleur 6502

Le lecteur qui ne connaît que le Basic doit savoir maîtriser les méthodes en assembleur décrites ci-dessous :

● Branchements

● Branchement inconditionnel

L'instruction GOTO no line du Basic est remplacée en assembleur par l'instruction JMP adresse.

● Branchements conditionnels

Pour remplacer l'instruction IF...GOTO, vous devrez procéder en deux temps :

— faire la comparaison pour établir la condition à tester sur les bits du registre d'états à l'aide de l'une des trois instructions suivantes : CMP, CPX, CPY,

— tester la condition voulue et effectuer le branchement, si elle est vérifiée, à l'aide des instructions BEQ, BMI, BPL, BNE, BCC, BCS, BVC, BVS.

Exemple :

```

10 A = PEEK($1000)
20 IF A = PEEK($100) GOTO 100
100 END

```

peut s'écrire :

```

$800 LDA $1000 /
      CMP $1001
      BEQ $10
      traitement
      RTS

```

\$10 octets

Pour = on utilise BEQ.

Pour < > on utilise BNE.

Pour < on utilise BMI.

Pour > on utilise BPL.

Les autres instructions (BCC, BCS, BVC, BVS) sont liées à l'arithmétique binaire.

● Opérations logiques

- OU logique (ORA).
- ET logique (AND).
- OU exclusif logique (EOR).

Elles se font entre le contenu de l'accumulateur et soit une valeur numérique prédéfinie, soit le contenu d'une case mémoire.

OU

Bit ACC \ Bit M	0	1
0	0	1
1	1	1

ET

Bit A \ Bit M	0	1
0	0	0
1	0	1

OU EXCLUSIF

Bit A \ Bit M	0	1
0	0	1
1	1	0

Leur résultat est dans l'accumulateur et les bits d'états sont positionnés. Il est inutile donc d'utiliser CMP après AND, OR, EOR.

```

{ 10 A=PEEK(1000)
  20 IF(A MOD 256)=0 GOTO 100

```

peut s'écrire

```

{ LDA $1000
  AND $0F
  BEQ adresse voulue

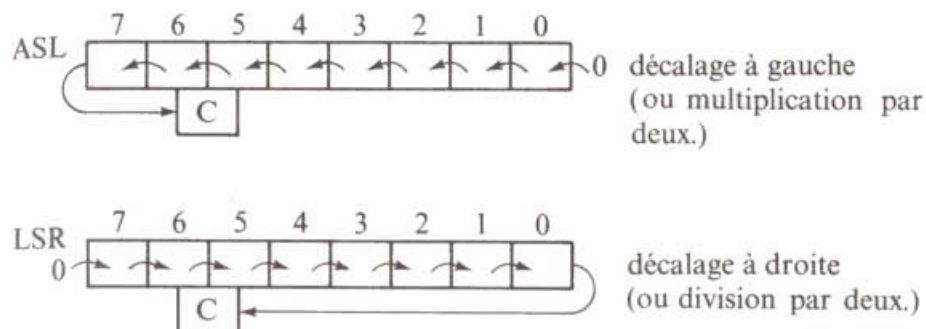
```

● Opérations de décalage et de rotation

● Décalages

Un décalage déplace le contenu de l'accumulateur (ou d'une case mémoire), d'un bit vers la gauche ou vers la droite. Le bit qui sort du

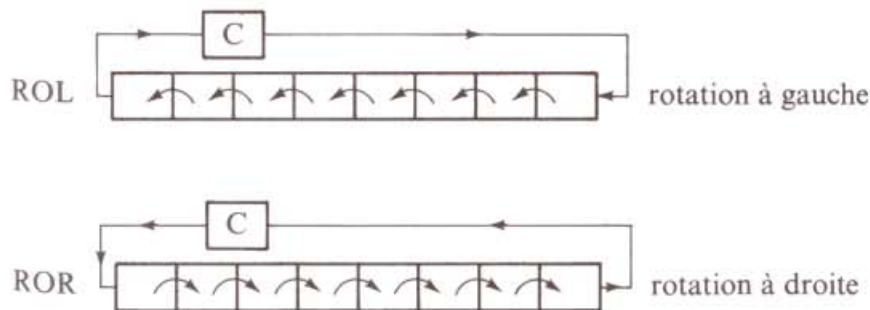
registre se retrouve dans le bit de retenue ; le bit qui entre de l'autre côté est un zéro.



Le microprocesseur 6502 est pauvre en décalages. Il lui manque par exemple des décalages arithmétiques qui conservent le bit sept, ou bit de signe et sont utiles, en arithmétique flottante notamment.

● Rotations

Une rotation réalise, de même, le déplacement du contenu de l'accumulateur (ou d'une case mémoire) d'un bit vers la gauche ou vers la droite. La différence provient du fait que le bit qui entre provient du bit de retenue.



Le microprocesseur 6502 est aussi pauvre en rotations. Il n'admet pas de rotations sur 8 bits, où celui qui sort d'un côté rentre de l'autre côté.

● Opérations arithmétiques

Les opérations arithmétiques du 6502 sont au nombre de deux. Ce sont les suivantes :

- ADC addition avec retenue,
- SBC soustraction avec retenue.

Il n'existe pas d'addition sans retenue sur le 6502. C'est un défaut, car il faut exécuter une instruction CLC avant toute addition. De plus il n'existe pas d'instructions d'addition ni de soustraction sur 16 bits comme pour la famille Intel-Zilog. Il est possible d'y arriver toutefois en faisant deux additions successives sur 8 bits, en conservant le bit de retenue.

Exemple :

OP1 à l'adresse AD1
OP2 à l'adresse AD2
RES à l'adresse ADR

le programme d'addition sur 16 bits est :

CLC		
LDA	AD1	} partie basse
ADC	AD2	
STA	ADR	
LDA	AD1+1	} partie haute
ADC	AD2+1	
STA	ADR+1	

Le principe est identique pour la soustraction 16 bits

SEC		
LDA	AD1	} partie basse
SBC	AD2	
STA	ADR	
LDA	AD1+1	} partie haute
SBC	AD2+1	
STA	ADR+1	

Le microprocesseur 6502 admet les opérations arithmétiques en « mode » hexadécimal et en « mode » DCB (décimal). Le mode de fonctionnement est différent de celui des familles Intel-Zilog.

Pour se mettre en mode hexadécimal, vous devez utiliser l'instruction CLD qui met le bit D à 0. Pour le mode DCB, c'est SED qu'il faudra.

Il n'existe pas d'instruction de multiplication ni de division. Ceci n'est pas un handicap de 6502 car les autres microprocesseurs 8 bits n'en disposent pas non plus. Voici un exemple de programme réalisant la multiplication de deux octets en mémoire, situés aux adresses AD1 et AD2, et rangeant le résultat en AD3.

MULTIP	LDA	#0	} initialisation du résultat à 0
	STA	AD3	
	LDX	#8	} X compteur de décalages
BOUC	LSR	AD2	
	BCC	VIDE	décalage du multiplicateur si le bit est à 0, pas d'addition
	CLC		} si le bit est à 1, addition du mul- tiplicande au résultat
	ADC	AD2	
VIDE	ROR	A	} dernier décalage ?
	ROR	AD3	
	DEX		
	BNE	BOUC	

Le même principe peut être utilisé sur 16 bits. Il faut gérer en plus le décalage des 16 bits entre 2 octets.

● Sous-programmes

L'instruction GOSUB du Basic est remplacée par CALL adresse.

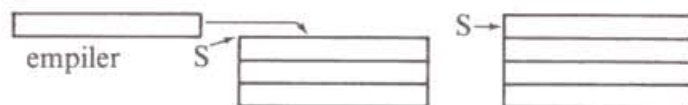
L'instruction RETURN du Basic est remplacée par RTS adresse.

L'adresse de retour du sous-programme est conservée dans la pile. Ceci implique que tout sous-programme doit laisser la pile propre au retour (on doit retrouver la même pile au retour qu'à l'appel).

● Utilisation de la pile

C'est une facilité offerte au programme assembleur pour stocker des variables locales de courte durée, la présence d'une pile étant quasi obligatoire pour avoir des sous-programmes. Tous les microprocesseurs 8 bits « célèbres » (Motorola 6800, Intel 8080, 8085, Zilog Z80) ont une pile. Il serait stupide de ne pas l'utiliser.

Deux instructions d'empilage sont disponibles : PHA et PHP.



De même deux instructions de dépilement sont disponibles : PLA et PLP.



● Entrées-sorties

Les instructions d'entrées-sorties sont des instructions spécialisées pour la gestion des unités d'entrée-sortie. Le microprocesseur 6502 utilise des entrées-sorties projetées en mémoire, ce qui signifie que les boîtiers d'interface sont connectés au bus d'adresses comme une mémoire. Ils sont vus par le programmeur comme des adresses mémoire.

Mais ceci ralentit le traitement des entrées-sorties et il est utile de disposer d'un mécanisme court dont les boîtiers résident en page zéro. Cependant celle-ci est généralement employée par de la RAM et c'est le cas pour l'APPLE II.

4.2.5 Conseils au lecteur débutant

Nous vous conseillons de procéder de la façon suivante :

- comprendre les diverses instructions et la structure interne du 6502,
- faire de petits exercices avec le miniassembleur ou LISA,
- utiliser l'assembleur pour créer des sous-programmes appelables en Basic, en exploitant au maximum le moniteur (cf. § 4.3.4),
- étudier en profondeur le moniteur, car il est très performant pour la place occupée.

4.3 LE MONITEUR

4.3.1 Présentation

Le moniteur est un programme résidant en ROM qui assure les entrées-sorties et offre à l'utilisateur des commandes permettant de mettre au point des programmes en langage machine. Nous allons étudier dans ce paragraphe les points suivants :

- accès au moniteur à partir du Basic,
- commandes du moniteur (examen des registres, de la mémoire, ...),
- miniassembleur et mise au point de programmes,
- carte mémoire de l'APPLE et structure interne du moniteur.

4.3.2 Accès au moniteur à partir du Basic

● Accès au moniteur

Il existe deux versions du moniteur : la version standard (APPLE II) et la version Autostart (APPLE II +).

Si vous avez un APPLE II standard, vous accédez au moniteur quand vous mettez en route votre système. Dans ce cas, vous voyez en bas et à gauche de l'écran une étoile suivie du curseur clignotant : le moniteur attend votre commande.

Si vous avez un APPLE II + ou si vous voulez retourner au moniteur à partir du Basic Entier ou de l'APPLESOFT, vous devez taper la commande suivante :

ou $\begin{matrix} > \\] \end{matrix}$ CALL-151

Cette commande correspond à l'appel du sous-programme situé à l'adresse \$FF69. Lorsque l'astérisque s'affiche, vous êtes dans le moniteur et il attend votre commande.

● Sortie du moniteur

Il existe plusieurs possibilités de sortir du moniteur pour retourner au Basic, ce sont les suivantes :

1. Taper CTRL-C ↵ ce qui rend la main au Basic en ROM en préservant le programme et les variables;
2. Taper CTRL-B ↵ ce qui rend la main au Basic en ROM en écrasant le programme et les variables;
3. Taper 3D0G ↵ ce qui rend la main au Basic chargé en mémoire à partir de la disquette en préservant le programme et les variables (équivalent de Ctrl-C pour le retour au Basic non situé en ROM);
4. Taper 0G ↵ même chose pour le Basic chargé à partir d'une cassette.

Les deux dernières possibilités sont utilisées lorsqu'on travaille avec l'APPLESOFT sur un APPLE II standard.

Exemples d'utilisation des possibilités d'accès et de sortie du moniteur sur un APPLE II + (APPLESOFT en ROM).

<pre>*CTRL-B]LIST]PRINT V 0</pre>	}	sortie du moniteur avec écrasement des variables
<pre>]10 V=110]20 PRINT V]LIST 10 V=110 20 PRINT V 30 RUN 110</pre>	}	entrée et exécution d'un programme Basic
<pre>] CALL-151 *CTRL-C]LIST 10 V=110 20 PRINT V]PRINT V 110]</pre>	}	passage dans le moniteur et retour par Ctrl-C : le programme et les données sont intactes.

4.3.3 Les commandes du moniteur

Les commandes du moniteur offrent à l'utilisateur des facilités de mise au point de programmes en langage machine. Parmi celles-ci :

- modification de la mémoire, copie d'une zone mémoire, sauvegarde d'une zone mémoire sur cassette et chargement à partir d'une cassette;
- création par l'utilisateur de ses commandes;
- désassemblage d'une zone mémoire, examen des registres...

Le moniteur reconnaît 22 caractères comme commandes. Il n'interprète que le premier caractère de la commande. Les adresses et données doivent être exprimées sous forme hexadécimale (0...9ABCDEF). Les adresses sont représentées par 4 chiffres hexadécimaux, les données par 2 chiffres hexadécimaux. Le moniteur complète les valeurs fournies par des zéros à gauche pour arriver au nombre de chiffres hexadécimaux nécessaires, ou il tronque à gauche si trop de nombres ont été fournis.

● Examen et modification de la mémoire

● Examen de la mémoire

Il est possible d'examiner la mémoire de deux façons :

- soit octet ou bloc de 8 octets à la fois (ce que nous appellerons au pas à pas);
- soit directement des octets d'une zone mémoire comprise entre deux adresses.

* EXAMEN PAS A PAS

Pour examiner le contenu d'un octet situé à une adresse donnée, il suffit de taper cette adresse lorsque vous avez la main : le moniteur affiche alors la valeur de l'octet.

Par exemple si vous tapez *ABCD
Le moniteur répond ABCD — D7 = contenu de l'octet d'adresse ABCD.

*

Le moniteur a alors initialisé un pointeur sur la mémoire à la valeur ABCD.

Pour examiner les octets suivants, il suffit de taper Return. Le moniteur affiche alors les octets suivants, jusqu'à la fin du bloc de 8 octets dans lequel on est situé.

Par exemple si vous tapez *ABCD
 ABCD-D7
 Puis *RETURN
 Le moniteur affiche ASD6
 *

Si vous appuyez sur Return, alors que le pointeur est en début d'un bloc de 8 octets, il affichera l'adresse du début du bloc et les valeurs des 8 octets.

Dans l'exemple ci-dessus, si vous tapez encore Return, le moniteur affichera :

ABD0-30 40 50 60 20 10 00 70
 *

Pour continuer l'affichage des blocs suivants, il suffira de taper Return autant de fois que cela sera nécessaire.

* EXAMEN D'UNE ZONE MÉMOIRE

Il est possible d'examiner une zone mémoire de taille supérieure à celle d'un bloc de 8 octets. Pour cela, il suffit :

- soit de donner les adresses de début et de fin de zone ;
- soit de donner l'adresse de fin de zone, l'adresse de début de zone étant prise par défaut égale au pointeur.

Si vous tapez par exemple : *E015.E025
 Le moniteur vous répondra :

E015-4C ED FD fin du bloc de 8 octets courants
 E018-A9 20 C5 24 B0 0C A9 8D
 E020-A0 07 20 ED FD A9 adresse de fin E025
 *

Le moniteur affiche donc :

- l'adresse de départ suivi du nombre d'octets nécessaires pour arriver au prochain bloc de 8 octets,
- tous les blocs de 8 octets dans lesquels ne se situe pas l'adresse de fin de la zone,

— l'adresse de début du dernier bloc et le nombre d'octets nécessaires pour arriver à l'adresse de fin de la zone.

Si vous voulez continuer à examiner la suite de la mémoire, il vous suffit de taper

.adresse de fin.

Dans l'exemple ci-dessous :

si vous tapez : *.E028
 Le moniteur affichera : E026 AA AB AC
 *

Il est souvent indispensable d'arrêter le défilement sur l'écran. Pour cela, il suffit de taper CTRL-S. Pour reprendre le défilement, il suffit de taper un blanc.

• Modification de la mémoire

Il est possible de modifier la mémoire de deux façons :

- soit octet par octet,
- soit plusieurs octets consécutifs en mémoire.

* MODIFICATION D'UN OCTET PAS À PAS

Si vous voulez modifier le contenu d'une adresse mémoire, il faut initialiser le pointeur vers la mémoire, et ensuite modifier l'octet situé à cette adresse.

Par exemple si vous voulez mettre la valeur AA à l'adresse 0, vous devez taper

*0 adresse à initialiser
 0000-XX
 *:AA valeur à mettre

Vérifiez ! tapez 0. Le moniteur affichera la valeur et vous constaterez que la valeur a bien été modifiée

*0
 0000-AA
 *

Si vous voulez maintenant modifier la valeur de l'adresse suivante (ici égale à 1), il vous suffit de taper : et la valeur à positionner (le pointeur

a déjà été initialisé à 0 et incrémenté lorsque l'on a mis AA à l'adresse 0; il est égal à 1).

*:AB

De nouveau le pointeur a été incrémenté, et il suffit de taper les nouvelles valeurs les unes après les autres.

Vous pouvez faire les deux opérations d'initialisation du pointeur et de modification de la valeur de l'octet en une commande.

Dans l'exemple ci-dessus, au lieu de taper les deux commandes :

*0

et

*:AA

Vous auriez pu taper seulement la commande suivante :

*0:AA

Le résultat obtenu est le même.

* MODIFICATION DE PLUSIEURS OCTETS CONSÉCUTIFS EN MÉMOIRE

Il serait fastidieux d'avoir à taper pour chaque octet :

: valeur

Le moniteur vous autorise à taper plusieurs valeurs à la suite l'une de l'autre. Par exemple si vous tapez :

*3000:00 01 02 03 04 05 06 07 08 09 0A

Vous obtiendrez en mémoire après l'exécution de la commande :

```
3000 00
3001 01
3002 02
3003 03
3004 04
3005 05
3006 06
3007 07
3008 08
3009 09
300A 0A
```

Vérifiez ! tapez la commande suivante :

*3000.300A

Le moniteur vous répondra :

```
3000-00 01 02 03 04 05 06 07
3008-08 09 0A
*
```

Le moniteur accepte jusqu'à 83 valeurs si le pointeur est initialisé et jusqu'à 84 valeurs si le pointeur n'est pas initialisé. Toutefois, nous vous déconseillons fortement de taper autant de valeurs, car le risque d'erreur se trouve fortement augmenté.

● Recopie et comparaison de zones mémoire

Il est possible de traiter des zones mémoire comme des entités propres. Le moniteur vous autorise à recopier une zone mémoire, à comparer deux zones mémoires.

● Recopie d'une zone mémoire

Pour recopier une zone mémoire, le moniteur doit connaître les paramètres suivants :

- adresses de début et de fin de la zone mémoire à recopier,
- adresse destinatrice à laquelle on va recopier la zone mémoire.

Le format de la commande est le suivant :

$$\left\{ \begin{array}{c} \text{adresse} \\ \text{destinatrice} \end{array} \right\} < \left\{ \begin{array}{c} \text{adresse de} \\ \text{début} \end{array} \right\} . \left\{ \begin{array}{c} \text{adresse de} \\ \text{fin} \end{array} \right\} \text{ Move}$$

Par exemple si vous tapez la commande suivante :

*10<0.F Move

Le moniteur recopiera les 16 octets situés aux adresses 0 à F à partir de l'adresse /0.

Si vous tapez les commandes

*0.F

et

*10.1F

le résultat obtenu est le même.

Lorsque le moniteur recopie la zone, il ne modifie pas la zone qu'il recopie. Si l'adresse de fin de la zone à recopier est inférieure ou égale à l'adresse de début, le moniteur ne recopiera que le premier octet de la zone.

Si l'adresse destinatrice est comprise entre l'adresse de début de la zone à recopier et l'adresse de fin de cette même zone, le début de la zone sera recopié un certain nombre de fois, écrasant le reste de la zone (cf. § 4.3.3).

• Comparaison de deux zones mémoires

Vous pouvez utiliser le moniteur pour comparer deux zones mémoire. Cette possibilité est utile lorsque, par exemple, on sauvegarde une zone mémoire sur cassette et que l'on veut vérifier que la sauvegarde est bonne (cf. § 4.3.3).

Le format de cette commande est le même que la précédente :

$$\left\{ \begin{array}{c} \text{adresse} \\ \text{destinatrice} \\ \text{zone de} \\ \text{référence} \end{array} \right\} < \left\{ \begin{array}{c} \text{adresse de} \\ \text{début de} \\ \text{la zone à} \\ \text{comparer} \end{array} \right\} \cdot \left\{ \begin{array}{c} \text{adresse} \\ \text{de fin de la} \\ \text{zone à} \\ \text{comparer} \end{array} \right\} \text{ Verify}$$

Par exemple, si vous tapez la commande suivante :

*10<0.F Verify

Après avoir tapé

*10<0.F Move

Le moniteur doit rendre la main sans rien signaler si la recopie s'est bien passée.

Par contre, si vous tapez les commandes suivantes :

*00:0A
*10<0.FM
*1A:1A
*10<0.FV

Le moniteur affichera sur l'écran

000A-0A (1A)

Pour chaque différence obtenue durant la comparaison, le moniteur affiche :

— l'adresse située entre les adresses du début et de fin de la zone à comparer ;

— la valeur trouvée à cette adresse et, entre parenthèses, la valeur trouvée à l'adresse correspondante dans l'autre zone.

Vous devez toujours mettre l'adresse de la zone dont vous êtes sûr à gauche du chevron <. De cette façon, vous connaîtrez les adresses où se situent les erreurs et non les adresses où se situent les bonnes valeurs.

• Initialisation d'une zone mémoire

La commande Move permet d'initialiser une zone mémoire. En effet initialiser une zone mémoire correspond à recopier un octet autant de fois que nécessaire.

Supposons que nous voulions initialiser la zone mémoire comprise entre les adresses 0 et FF. Montrons que l'exécution des commandes suivantes :

*0:0
*1<0.FEM

suffit à initialiser la zone à zéro.

La première commande initialise l'octet d'adresse zéro à la valeur zéro (cf. § 4.3.3).

La deuxième commande recopie la valeur de l'octet d'adresse zéro à l'adresse un. L'octet d'adresse un passe à la valeur zéro. L'octet d'adresse un est recopié à l'adresse deux et ainsi de suite. Tous les octets d'adresse, compris entre 0 et FF, sont donc initialisés à zéro.

Il est possible de reproduire de la même façon des structures de taille supérieure à 1 octet.

Supposons que nous voulions initialiser la zone comprise entre les adresses 0 et 256 de la manière suivante :

0 1 2 3 4 5 6 7 0...7 0...7 0...7 0...7...0...7
c'est-à-dire (0...7) autant de fois que possible

Pour cela tapons les commandes suivantes :

*0:0 1 2 3 4 5 6 7
*8<0.F7M

Après les 8 premières recopies, la structure a été recopiée aux adresses 8 à F. Après les 8 recopies suivantes, la structure a été recopiée aux adresses 10 à 17; etc...

L'adresse destinatrice est égale à l'adresse du début de la zone à initialiser ajoutée de la longueur de la structure. L'adresse de fin de zone est celle de fin de la zone à initialiser moins la longueur de la structure.

D'une manière générale, pour initialiser une zone avec une structure de longueur n, il faut :

- initialiser les n premiers octets de la zone avec la structure,
- taper la commande

{ début zone + n } < { début zone } { fin zone - n } M

Il est possible avec le même principe de vérifier qu'une zone a bien été initialisée en utilisant la commande Verify.

{ début zone + n } < { début zone } { fin zone - n } V

● Sauvegarde et restauration d'une zone mémoire

Le moniteur a deux commandes spéciales qui vous permettent de sauvegarder une zone mémoire sur cassette et de restaurer pour un usage ultérieur. Cette zone mémoire peut contenir par exemple un sous-programme en langage machine 6502 ou une forme graphique haute résolution. Toutefois pour les possesseurs du DOS, il est préférable de l'utiliser pour une question de vitesse et de sécurité.

● Sauvegarde d'une zone mémoire sur cassette

La commande WRITE de sauvegarde d'une zone mémoire sur cassette a le format suivant :

{ adresse de début }, { adresse de fin } W

Par exemple la commande

*0.FFFFW

sauvegardera 64 K octets sur une cassette.

Pour faire une sauvegarde, vous devez respecter le scénario suivant :

- mettre le magnétophone en position enregistrement,
- laisser tourner la bande quelques secondes,
- taper la commande.

Le moniteur écrit alors un en-tête de 10 s sur la cassette, puis les données. Il écrit à la fin un octet qui est égal à la somme de tous les octets écrits, tronquée sur 8 bits.

Lorsqu'il a fini, le moniteur émet un top et rend la main à l'utilisateur.

Le moniteur envoie environ 1 350 bits par seconde sur la cassette, soit 4 K octets en 35 s.

Le moniteur ne sait pas si le magnétophone est prêt ou même si un magnétophone est connecté. C'est à l'utilisateur de vérifier si tout est prêt du côté magnétophone.

● Restauration d'une zone mémoire à partir d'une cassette

La commande READ permet de restaurer les données enregistrées sur cassette en mémoire. La zone de chargement n'est pas forcément située à la même adresse que la zone mémoire que l'on a sauvegardée. Le format de la commande Read est le suivant : vous devez fournir au moniteur l'adresse où doit être chargé le premier octet lu sur la cassette et l'adresse où doit être chargé le dernier octet lu. Ces deux adresses doivent être séparées par un point comme suit :

{ adresse de début } . { adresse de fin } R

Pour lire une cassette, vous devez faire les opérations suivantes :

1. Placer la cassette après le début de l'en-tête de 10 s placé par le moniteur avant les données : il suffit pour cela d'écouter la cassette, l'en-tête étant la seule partie de la cassette qui ne donne pas l'impression d'un bruit (comme les données).

2. Taper la commande : le moniteur se met en attente des informations arrivant de la disquette ; il lui faut environ 3 s d'en-tête pour se synchroniser avec la cassette.

3. Appuyer sur la touche PLAY ou : le moniteur charge alors les données en mémoire. Si le moniteur trouve une erreur il affiche ERR ; sinon il n'affiche rien. Lorsqu'il a fini, le moniteur envoie un « beep » au haut-parleur et rend la main à l'utilisateur.

Le moniteur ne sait pas si le magnétophone est prêt, ou même s'il y a un magnétophone branché sur l'APPLE. C'est à l'utilisateur de vérifier que tout est prêt du côté magnétophone. Le moniteur reste bloqué tant qu'il ne s'est pas synchronisé avec la cassette.

Si vous tapez par exemple la commande :

*200,300 R

le moniteur chargera les 257 octets à partir de la cassette.

Quelles sont les sources d'erreur pendant une lecture de la cassette ?

- longueur d'en-tête inférieure à 3 s ;
- longueur à recopier différente de la longueur sauvegardée sur la cassette ;
- cassette abîmée ou effacée ;
- erreur durant l'écriture de la cassette.

Comment sont détectées les deux premières erreurs ? Le dernier octet écrit par le moniteur est un octet calculé en fonction des octets précédemment écrits. En cas d'erreur, l'octet calculé à la lecture des octets de la cassette sera différent de l'octet, lu sauf pure coïncidence.

Dans le premier cas, vous pouvez écouter la cassette. Si vous trouvez que l'en-tête est effectivement inférieur à 3 s, il vous faudra refaire la cassette (ou juste une partie si la cassette sert plusieurs fois). Sinon réessayer. Le chargement doit bien se passer.

Pour éviter le second cas, nous vous conseillons de marquer sur la cassette les enregistrements qui y ont été faits.

Le meilleur moyen pour ne pas avoir de problème est d'essayer de faire une lecture de la cassette tant que la zone mémoire que l'on a sauvegardée n'a pas été effacée (cf. 4.3.3). Ainsi vous pourrez refaire la manipulation autant de fois que nécessaire.

• Sauvegarde d'une mémoire sur disquette

Si vous avez une unité de disquette, sauvegarder une zone mémoire est plus rapide et plus facile qu'avec une cassette. Vous devez l'utiliser chaque fois que possible.

Pour sauvegarder une zone mémoire, vous devez sortir du moniteur et vous placer dans le Basic, avec le DOS en mémoire (taper CTRL-C).

Exemple : Pour sauvegarder la zone mémoire d'adresse \$200, de longueur 200 dans le fichier SPMACH (disquette de volume 254, drive 1 du slot 6), vous devez taper la commande :

BSAVE SPMACH, A\$200, L200, S6, D1, V254

La longueur maximum qui peut être sauvegardée est de 32 K octets. BSAVE vous autorise à taper les constantes en base décimale ou hexa-

décimale. Dans ce cas, n'oubliez pas de faire précéder les constantes par un dollar \$.

Les trois derniers paramètres sont facultatifs.

• Restauration de la mémoire à partir d'une disquette

Pour charger en mémoire le contenu d'un fichier, vous devez utiliser la commande BLOAD du DOS.

Par exemple si vous tapez :

BLOAD SPMACH, A\$290, L200, S6, D1, V254

le DOS chargera à l'adresse \$290, le fichier SPMACH.

L'adresse figurant dans la commande est l'adresse à laquelle on doit charger le fichier. Si cette adresse ne figure pas dans la commande, le fichier est chargé à l'adresse qui figurait dans le BSAVE. La longueur est un paramètre optionnel, le DOS arrêtant de recopier le fichier lorsqu'il arrive à sa fin.

Lorsque vous chargez un fichier en mémoire, il écrase une zone mémoire. Faites attention à ne pas écraser le DOS, les pages graphiques, le programme et les variables Basic. Ou même l'interpréteur Basic si celui-ci ne figure pas en ROM (exemple APPLESOFT chargé à partir d'une disquette).

• Vérification de la mémoire sauvegardée sur un périphérique

Pour être sûr de la sauvegarde que vous venez de faire, il existe une méthode qui consiste à relire tout de suite les données sauvegardées, à les charger en mémoire à une adresse différente de l'adresse de la zone que l'on a sauvegardée, et à comparer les deux zones à l'aide de la commande Verify du moniteur.

Nous donnons deux exemples d'application qui peuvent s'appliquer pour un sous-programme assembleur, une structure graphique ou toute autre donnée.

* CASSETTE

Si vous utilisez les cassettes pour sauvegarder des zones mémoires, la première étape est de sauvegarder la zone mémoire avec la commande Write.

Tapez par exemple :

*500.5FF W

Le moniteur sauvegarde les octets situés entre les adresses \$500 et 45FF sur cassette. Lorsqu'il a fini, le haut-parleur émet un « beep ». Vous devez alors rembobiner la bande jusqu'au début de l'en-tête sonore, et taper une commande ordonnant au moniteur de charger la cassette en mémoire. Supposons que la zone mémoire 400.4FF soit libre. Vous pouvez taper la commande suivante

*\$400.4FFR

Nous avons alors en mémoire :

- entre les adresses \$400 et \$4FF les données lues à partir de la cassette,
- entre les adresses \$500 et \$5FF les données de référence.

Nous allons comparer les deux zones. Pour cela tapez la commande :

*500 < 400.4FFV

Le moniteur compare alors les deux zones. S'il ne trouve pas d'erreur, vous pouvez considérer la cassette comme bonne, sinon vous devrez recommencer la sauvegarde.

* DISQUETTE

Le principe est le même pour les disquettes où il faut avoir le DOS en mémoire et être sous Basic.

Tapez alors les commandes :

BSAVE MEMOIRE, A\$500, L\$FF	sauvegarde zone mémoire
BLOAD MEMOIRE, A\$400	restauration de la zone à une autre adresse
CALL - 151	passage dans le moniteur
*500 < 400.4FFV	comparaison des deux zones

Si le moniteur ne trouve pas de différences, vous pouvez alors être sûr que le fichier est la représentation exacte de la zone mémoire sauvegardée. Sinon revenez au Basic (CTRL-C (ou 3DOG)) et refaites la sauvegarde.

● Mise au point de programmes en langage machine et en assembleur

Le moniteur offre à l'utilisateur de nombreuses aides à la mise au point de sous-programmes assembleurs et en langage machine. Toutefois certaines de ces possibilités (miniassembleur, possibilité d'exécution pas à pas...) ne sont disponibles qu'avec la version standard du moniteur (APPLE II standard) et ne sont pas accessibles avec la version ROM autostart pour une question de taille. Nous allons étudier dans ce paragraphe chacune de ces possibilités en précisant si elles sont disponibles ou non avec la ROM autostart (APPLE II +).

● Examen et modification des registres du microprocesseur 6502

Le moniteur réserve 5 adresses dans la mémoire pour sauvegarder les 5 registres du microprocesseur 6502 : A, X, Y, P (état du microprocesseur), S (pointeur de pile). La commande Examine, que l'on fait exécuter au moniteur en tapant CTRL-E, ordonne au moniteur d'afficher le contenu de ces adresses et de positionner le pointeur mémoire vers la première de ces adresses.

Si vous tapez la commande suivante :

*CTRL-E

le moniteur affichera le contenu des registres selon le format suivant :

A=XX X=XX Y=XX P=XX S=XX

Le moniteur vous autorise à modifier le contenu des 5 registres du 6502. Tapez CTRL-E ; le moniteur affiche le contenu des registres et positionne le pointeur mémoire vers la première de ces adresses. Il vous suffit donc de modifier le contenu de ces adresses.

Supposons que les contenus des registres soient respectivement \$B0 \$FF \$FE/\$30 \$F8 et que vous vouliez mettre l'accumulateur A à la valeur \$FF et le registre d'index Y à la valeur \$AA. Le processus à suivre est le suivant :

*CTRL-E ↵

A=B0 X=FF Y=FE P=30 S=F8

*:FF FF AA

vérifiez

*CTRL-E

A=FF X=XX Y=AA P=30 S=F8

Le moniteur chargera les octets correspondants à ces adresses dans les registres correspondant avant de redonner la main au programme utilisateur que vous voulez exécuter ou mettre au point (commandes G, S, T ci-dessous).

Si vous vouliez seulement modifier le pointeur de pile, il vous faut tout de même redonner les valeurs des registres précédents (dans la liste). En effet, lorsque vous tapez CTRL-E, le moniteur initialise le pointeur vers la mémoire avec l'adresse correspondant à l'accumulateur. Pour accéder au pointeur de pile, qui est la cinquième adresse, il faut modifier le pointeur en modifiant les adresses précédentes.

Par exemple, si vous voulez mettre la valeur 00 dans le pointeur de pile, vous devez taper les commandes :

```
*CTRL-E
A=FF X=FF Y=AA P=30 S=F8
*:FF FF AA 30 00
valeur valeur valeur valeur nouvelle valeur
de A de X de Y de P de S
```

• Création et exécution d'un programme en langage machine

Il est facile de rentrer en mémoire un programme en langage machine. Préparez les codes à mettre en mémoire. Vous entrez ensuite les codes en mémoire en utilisant les commandes de modification de la mémoire. Nous vous conseillons fortement de sauvegarder ensuite la zone mémoire dans laquelle se situe le programme sur cassette ou disquette.

Vous voulez maintenant essayer votre programme. Nous supposons dans les explications suivantes que votre programme commence à l'adresse \$8000. Pour lancer votre programme, tapez la commande :

```
*8000G
```

Le moniteur restaure alors les registres à partir des adresses où ils sont normalement sauvegardés. Si par exemple, vous avez modifié les registres avant de commencer l'exécution de votre programme, c'est à ce moment précis qu'ils sont restaurés. Il appelle ensuite votre programme, qu'il considère comme un sous-programme, en utilisant l'instruction JSR. Votre programme prend alors le contrôle du microprocesseur et s'exécute.

Afin que le moniteur reprenne le contrôle après l'exécution de votre programme, la dernière instruction doit être l'instruction RTS (retour à partir d'un sous-programme dans le programme principal). Sinon il vous faudra relancer votre APPLE.

Le format général de l'instruction GO est le suivant :

```
{ adresse de début } G
```

Voici deux exemples de la commande GO :

Exemple 1 : Lorsque vous avez un APPLE II standard et que vous avez chargé l'APPLESOFT à partir d'une disquette, vous devez taper la commande 3DOG pour revenir à l'APPLESOFT à partir du moniteur. Cette commande correspond à un branchement vers l'adresse \$3D0 où se trouve l'instruction JMP (aller à l'adresse) vers le DOS.

Exemple 2 : Si vous tapez les commandes suivantes :

(entrée des instructions)

```
*300:A9 C1 20 ED FD 18 69 01 C9 DB D0 F6 60
```

(vérification)

```
*300.30C
```

```
0300-A9 C1 20 ED FD 18 69 01
```

```
0308-C9 DB D0 F6 60
```

(sauvegarde)

```
*300.30CW
```

(exécution)

```
*300G
```

```
ABCDEFGHIJKLMNPOQRSTUVWXYZ
```

```
*
```

vous entrez en mémoire le programme à exécuter, le vérifiez, le sauvegardez, et l'exécutez.

C'est le scénario à suivre de manière générale.

• Désassemblage d'un programme en langage machine

Le moniteur offre à l'utilisateur la possibilité d'avoir un listing du sous-programme langage machine en mémoire, le listing étant présenté sous forme d'instruction assembleur. Au lieu d'avoir à retrouver les instructions selon les codes qui comprennent un, deux ou trois octets, la commande List fait cette décomposition et recherche le mnémonique correspondant, pour 20 instructions.

Le format de la commande est le suivant :

```
{ adresse de début } L
```


Le moniteur utilise le pointeur de programme (registre interne au micro-processeur) comme pointeur vers la prochaine instruction à désassembler. Il n'est donc utile de préciser l'adresse de début que la première fois, lors du désassemblage d'une zone mémoire.

Dans l'exemple ci-dessus, si vous tapez la commande suivante *300L, le moniteur affichera :

```

0300 - A9  C1      LDA  #C1
0302 - 20  ED  FD  JSR  #FDED
0305 - 18  E       CLC
0306 - 69  01      ADC  #01
0308 - C9  DB      CMP  #$DB
030A - D0  F6      BNE  $0302
030C - 60          RTS
030D - A3  ??      fin du programme, la suite ne nous
                        intéresse pas
030E - 00          BRK

```

• Le miniassembleur

Si vous avez un APPLE II standard, ou une carte langage (utiliser INT), vous pourrez écrire des programmes en assembleur. Celui-ci réside dans la même ROM que le Basic Entier et n'est pas disponible avec l'APPLE-SOFT.

Le miniassembleur vous permet d'entrer en machine les mnémoniques des instructions. Il n'accepte pas les symboles. Vous devez donc entrer les adresses directement sous forme hexadécimale.

* ACCÈS AU MINIASSEMBLEUR

Le point d'entrée du miniassembleur est à l'adresse \$F666. Pour accéder au miniassembleur à partir du moniteur vous devez donc taper la commande :

*F666G

Pour accéder au miniassembleur à partir d'un Basic (Basic Entier en ROM ou APPLESOFT sur disque ou cassette), vous devez appeler le sous-programme situé à l'adresse -2458 (version décimale de l'adresse \$F666) et donc taper :

```

>
] CALL -2458
-

```

Le miniassembleur affichera alors un point d'exclamation !

Le miniassembleur existe maintenant sur disquette pour les utilisateurs de l'APPLE II +. Il est possible de l'obtenir à partir du fichier INT Basic fourni avec le DOS 3.3, copie des ROM basic entier.

* ACCÈS AUX COMMANDES DU MONITEUR A PARTIR DU MINIASSEMBLEUR

Il est possible d'accéder aux commandes du moniteur à partir du miniassembleur. Pour cela il vous suffit de taper le caractère \$, avant de taper votre commande.

Si vous voulez par exemple désassembler les instructions se situant à l'adresse \$300, il vous faudra taper la commande :

!\$300L

* SORTIE DU MINIASSEMBLEUR

Pour quitter le miniassembleur, utilisez les commandes du moniteur.

Pour revenir au Basic, tapez selon le cas l'une des quatre commandes suivantes :

!\$CTRL-B

!\$CTRL-C

!\$3DOG APPLESOFT chargé à partir d'une disquette

!\$OG APPLESOFT chargé à partir d'une cassette

Pour revenir au moniteur, utilisez la commande !\$FF69G ou alors appuyez sur la touche RESET.

* INSTRUCTIONS

Le miniassembleur gère un pointeur vers la prochaine instruction à assembler qui est différent de celui du moniteur. Vous aurez donc à l'initialiser lorsque vous introduirez la première instruction à assembler.

Par exemple si vous voulez mettre l'instruction BRK à l'adresse \$300, vous devrez utiliser la commande suivante :

!300 : BRK

Si vous voulez faire suivre cette instruction par une autre, il est inutile de calculer l'adresse de la nouvelle instruction, le miniassembleur le faisant automatiquement. Vous devez par contre laisser un blanc entre le point

d'exclamation et l'instruction à assembler. Le format des instructions acceptées par le miniassembleur est le suivant :

! mnémonique sur 3 lettres opérandes.

Lorsque la ligne est tapée, le miniassembleur l'analyse. S'il ne trouve pas d'erreur, il incrémente le pointeur vers la prochaine instruction et rend la main à l'utilisateur en affichant un point d'exclamation. Sinon le miniassembleur émet un « beep » et affiche un accent circonflexe sous le premier caractère incriminé. Puis il attend que l'utilisateur retape l'instruction.

Le format de (ou des) opérandes est imposé selon l'instruction et le mode d'adressage. Le microprocesseur 6502 possède onze modes d'adressage possibles.

Il n'y a toutefois que six formats acceptés par le miniassembleur pour le ou les opérandes.

Ce sont les suivants :

Mode d'adressage	Format	N° de format
Accumulateur implicite		
immédiat	#\$ valeur	1
absolu	\$ valeur	2
absolu indexé	\$ valeur, X	3
	\$ valeur, Y	4
page zéro	\$ valeur	2
page zéro indexé	\$ valeur, X	3
	\$ valeur, Y	4
relatif	\$ valeur	2
indexé indirect	(\$ valeur, X)	5
indirect indexé	(\$ valeur), Y	6

Le choix entre le mode absolu et le mode page zéro se fait d'après la valeur de l'opérande. Si elle est inférieure, le mode d'adressage page 0 est choisi, sinon le mode absolu est choisi.

• Mise au point des programmes

Parallèlement au miniassembleur, la version standard du moniteur offre à l'utilisateur deux commandes pour faire exécuter ses programmes pas à pas, en gardant une trace des registres.

— La commande Step qui permet d'exécuter un programme en gardant une trace des registres.

— La commande Trace qui permet d'exécuter un programme en gardant une trace des registres.

Ces deux commandes ne sont pas offertes avec le ROM autostart.

* EXÉCUTION D'UN PROGRAMME PAS À PAS

Pour tester un programme assembleur, il est souvent utile d'exécuter une partie du programme instruction par instruction, et de regarder le résultat après chaque instruction. C'est le rôle rempli par la commande Step dont le format est le suivant :

{ adresse de l'instruction à exécuter } S

Pour chaque commande S, le moniteur affiche l'instruction à exécuter, sous ses formes langage machine et mnémonique assembleur. Il l'exécute ensuite et incrémente le pointeur de programme pour qu'il pointe sur l'instruction suivante. Vous n'avez donc à taper l'adresse de l'instruction que pour la première instruction à exécuter dans la série. L'enchaînement se fait automatiquement.

Lorsque l'instruction a été traitée par le 6502, le moniteur affiche les registres du 6502 et rend la main à l'utilisateur. Celui-ci peut alors modifier les registres. Ils seront pris en compte lors de la prochaine commande S, GO, ou Trace.

Si par exemple vous voulez tester le programme suivant :

```

300 - A2 02   LDX #102
302 - B5 00   LDA 00,X
304 - 95 10   STA 10,X
306 - CA      DEX
307 - BD 30 C0 STA C030
30A - 10 F6   BPL 03C2
30C - 60      RTS

```


vous devez taper la commande suivante :

*300S

le moniteur affiche alors :

```
0300-A2    02    LDX ≠ $02
A=0A    X=02    Y=D8    P=30    S=F8
```

Si vous voulez continuer à exécuter les instructions suivantes, vous devez taper la commande S autant de fois que d'instructions à traiter.

Il est possible entre deux commandes S d'utiliser les autres commandes du moniteur pour examiner et modifier la mémoire ou les registres, activer une imprimante, passer à l'écran en mode inverse ou normal.

* TRACE DURANT L'EXÉCUTION D'UN PROGRAMME

Pour les programmes qui sont trop longs à traiter pas à pas, le moniteur offre à l'utilisateur la possibilité d'avoir une trace de l'exécution d'un programme. Le format de la commande Trace est le suivant :

{ adresse de début du programme à « tracer » } T

L'adresse de début n'a pas à être fournie au moniteur si elle a été positionnée par la commande Step par exemple. La Trace du programme s'arrêtera seulement si le moniteur rencontre une instruction BRK (code 00) ou si vous avez appuyé sur la touche Reset.

Le format de l'affichage est le même que pour la commande Step, l'instruction en code machine avec mnémonique et opérandes correspondants, et les registres à la fin de l'instruction.

Les commandes du moniteur S, L, T, G utilisent le même pointeur.

● Exemple de création d'un programme assembleur avec le moniteur

Nous vous donnons ci-dessous un exemple de mise au point d'un sous-programme assembleur pour l'affichage d'un point (instruction PLOT en Basic). Les coordonnées sont données par la lecture des manettes de jeu, à l'aide du miniassembleur et des commandes du moniteur dans sa version standard.

Entrons d'abord dans le miniassembleur.

Pour cela, tapez CALL-2458 si vous travaillez en Basic Entier, INT et CALL-2458 si vous avez la carte langage. Lorsque le point d'excla-

mation est affiché, le miniassembleur attend vos commandes. Tapez alors les suivantes :

```
0304: LDA    $C050          (Return)
0307: LDA    $C053
030A: LDA    $C054
030D: LDA    $C056
0310: JSR    $FB46
0313: LDA    #$99
0315: STA    $30
0317: LDX    #$00
0319: JSR    $FB1E
031C: CPY    #$27
031E: BCC    $0322
0320: LDY    #$27
0322: STY    $0300
0325: LDA    #$50
0327: JSR    $FCAB
032A: LDX    #$01
032C: JSR    $FB1E
032F: CPY    #$27
0331: BCC    $0335
0333: LDY    #$27
0335: STY    $0301
0338: LDA    #$50
033A: JSR    $FCAB
033D: LDA    $C061
0340: CMP    #$00
0342: BPL    $0317
0344: LDA    $0300
0347: LDY    $0301
034A: JSR    $FB00
034D: JMP    $0317
0350: BRK
```

Retournez au Basic (\$CTRL-C) et sauvegardez votre programme

BSAVE BPLOT, A\$300, L\$50

Passons maintenant dans le moniteur et tapez

*304L

Le listing obtenu est le suivant :

```
0304- AD 50 C0    LDA    $C050
0307- AD 53 C0    LDA    $C053
030A- AD 54 C0    LDA    $C054
030D- AD 56 C0    LDA    $C056
0310- 20 46 FB    JSR    $FB46
0313- A9 99      LDA    #$99
```

```

0315- B5 30      STA  $30
0317- A2 00      LDX  #$00
0319- 20 1E FB   JSR  $FB1E
031C- C0 27      CPY  #$27
031E- 90 02      BCC  $0322
0320- A0 27      LDY  #$27
0322- 8C 00 03   STY  $0300
0325- A9 50      LDA  #$50
0327- 20 AB FC   JSR  $FCAB
032A- A2 01      LDX  #$01
032C- 20 1E FB   JSR  $FB1E
032F- C0 27      CPY  #$27
0331- 90 02      BCC  $0335
0333- A0 27      LDY  #$27
0335- 8C 01 03   STY  $0301
0338- A9 50      LDA  #$50
033A- 20 AB FC   JSR  $FCAB
033D- AD 61 C0   LDA  $C061
0340- C9 00      CMP  #$00
0342- 10 D3      BPL  $0317
0344- AD 00 03   LDA  $0300
0347- AC 01 03   LDY  $0301
034A- 20 00 FB   JSR  $FB00
034D- 4C 17 03   JMP  $0317
0350- 00        BRK

```

Vous pouvez maintenant exécuter votre programme. Pour cela, tapez la commande

*304G

L'écran passe en mode graphique basse résolution, et le point de coordonnées PDL(0), PDL(1) s'affiche lorsque vous appuyez sur le bouton-poussoir de la manette 0.

La dernière instruction faisant un GO TO vers le début du programme après les initialisations, le seul moyen d'arrêter votre programme est d'appuyer sur Reset.

A titre de comparaison, voici le listing du programme avec commentaires obtenu sur le LISA assembleur version 1.5.

```

0300      1      ORG $300
0300      2      ;
0300      3      ; DEFINITION DES SOUS-PROGRAMMES
0300      4      ;
0300      5      WAIT EQU $FCAB
0300      6      PDL  EQU $FB1E
0300      7      PLOT EQU $FB00
0300      8      ;

```

```

0300      9      ; DEFINITION DES ADRESSES MEMOIRES
0300     10      ;
0300     11      GRAPH EQU $C050
0300     12      MIXTE EQU $C053
0300     13      PAGE1 EQU $C054
0300     14      BASRES EQU $C056
0300     15      BOUT  EQU $C061
0300     16      ORANGE EPZ $99
0300     17      ACOLOR EPZ $30
0300     18      ;
0300     19      ; DEFINITION DES VARIABLES
0300     20      ;
0301     21      LIGNE DFS $1
0302     22      COLON DFS $1
0302     23      ;
0302     24      ; INITIALISATIONS
0302     25      ;
0304     26      ORG $304
0304 AD50C0 27      LDA GRAPH
0307 AD53C0 28      LDA MIXTE
030A AD54C0 29      LDA PAGE1
030D AD56C0 30      LDA BASRES
0310 2046FB 31      JSR $FB46
0313 A999   32      LDA #ORANGE
0315 8530   33      STA ACOLOR
0317        34      ;
0317        35      ; LECTURE DE LA LIGNE PDL(0)
0317        36      ;
0317 A200   37      BOUCLE LDX #$00
0319 201EFB 38      JSR PDL
031C C027   39      CPY #$27
031E 9002   40      BCC OK1
0320 A027   41      LDY #$27
0322 8C0003 42      OK1 STY LIGNE
0325        43      ;
0325        44      ; ATTENTE DE TEMPORISATION
0325        45      ;
0325 A950   46      LDA #$50
0327 20ABFC 47      JSR WAIT
032A        48      ;
032A        49      ; LECTURE DE LA COLONNE PDL(1)
032A        50      ;
032A A201   51      LDX #$01
032C 201EFB 52      JSR PDL
032F C027   53      CPY #$27
0331 9002   54      BCC OK2
0333 A027   55      LDY #$27
0335 8C0103 56      OK2 STY COLON
0338        57      ;
0338        58      ; ATTENTE DE TEMPORISATION
0338        59      ;
0338 A950   60      LDA #$50
033A 20ABFC 61      JSR WAIT
033D        62      ;
033D        63      ; TEST DE PRESSION DU BOUTON POUSSOIR
033D        64      ;

```



```

033D AD61C0 65 LDA BOUT
0340 C900 66 CMP #00
0342 10D3 67 BPL BOUCLE
0344 68 ;
0344 69 ; AFFICHAGE DU POINT
0344 70 ;
0344 AD0003 71 LDA LIGNE
0347 AC0103 72 LDY COLON
034A 2000F8 73 JSR PLOT
034D 4C1703 74 JMP BOUCLE
0350 00 75 BRK
0350 76 END

```

```
***** END OF ASSEMBLY
```

```

*****
*
* SYMBOL TABLE -- V 1.5 *
*
*****

```

```
LABEL. LOC. LABEL. LOC. LABEL. LOC.
```

```
** ZERO PAGE VARIABLES:
```

```
ORANGE 0099 ACOLOR 0030
```

```
** ABSOLUTE VARIABLES/LABELS
```

```

WAIT FCAB PDL FB1E PLOT FB00 GRAPH C050
MIXTE C053 PAGE1 C054 BASRES C056 BOUT C061 LIGNE 0300 COLON 0301
BOUCLE 0317 OK1 0322 OK2 0335

```

```

SYMBOL TABLE STARTING ADDRESS: 6000
SYMBOL TABLE LENGTH: 008A

```

● Commandes permettant de modifier les entrées-sorties

● Modes de l'écran

Les commandes I/N permettent de faire passer l'écran en mode inverse/normal. Toutefois, l'écho des caractères saisis au clavier restent affichés en mode normal.

● Utilisation sur une imprimante

L'imprimante est utilisée en tapant la commande CTRL-P. L'affichage sur l'écran est alors arrêté, les caractères à afficher étant alors envoyés à l'imprimante.

Le format exact de la commande est le suivant :

numéro du connecteur CTRL-P

où le numéro du connecteur est compris entre 1 et 7 et où le connecteur indiqué contient une carte d'interface pour imprimante.

Si le connecteur que vous indiquez ne contient pas de carte d'extension, vous perdrez le contrôle de votre APPLE.

Pour revenir au fonctionnement normal de l'APPLE, vous devrez taper la commande CTRL-P.

● Utilisation d'un autre clavier que celui de l'APPLE (ou de tout autre périphérique permettant d'entrer des données)

Il est possible d'utiliser une autre source de données pour remplacer le clavier de l'APPLE. Pour cela, il vous suffit d'avoir une carte d'interface et de taper la commande :

numéro du connecteur de la carte CTRL-K

Pour revenir au fonctionnement normal de l'APPLE, vous devrez taper la commande zéro CTRL-K.

Si vous tapez un mauvais numéro de connecteur, vous perdrez le contrôle de l'APPLE.

● Création de votre commande utilisateur

La commande CTRL-Y entraîne le branchement du moniteur à l'adresse \$3F8. Vous pouvez y mettre une instruction Jump vers notre programme.

Vous pouvez par exemple décider que vous voulez utiliser cette possibilité pour passer dans le miniassembleur. Pour cela, il vous suffit de taper la commande

3F8:4C 66 F6 (i.e. JMP \$F666)

Lorsque vous taperez CTRL-Y, vous vous retrouverez placés dans le miniassembleur.

Il vous est aussi permis d'utiliser cette possibilité pour retourner au Basic chargé en mémoire à l'aide d'une disquette. Pour cela, il vous suffit de taper

3F8:4C D0 03 (i.e. JMP 3D0 ou 3DOG).

4.3.4 Adresses mémoire et sous-programme du moniteur :

Ce paragraphe décrit : la carte mémoire de l'APPLE II, les sous-programmes d'entrées-sorties avec le clavier et la sortie vidéo de l'APPLE, les sous-programmes de gestion des interruptions, l'utilisation des commandes du moniteur comme sous-programmes et la gestion des autres entrées-sorties de l'APPLE II.

● Description de la carte mémoire

● Présentation générale

La mémoire est divisée en segments de 256 octets, appelés « pages », numérotés à partir de 0. La page 0 est réservée au moniteur, au DOS et au Basic; la page 1 est utilisée comme pile par le microprocesseur 6502 (nous la réservons uniquement pour cet usage). La page 2 sert de « buffer » de stockage des données saisies au clavier. Viennent ensuite les pages 4 à 7 utilisées par la page graphique numéro 1 basse résolution, les pages 8 à B utilisées pour la seconde page graphique basse résolution. Les 2 pages graphiques haute résolution sont situées respectivement aux adresses \$2000-\$3FFF et \$4000-\$7FFF. La mémoire ROM contenant le moniteur est située entre les adresses \$D000-\$FFFF.

Le tableau ci-dessous récapitule ces diverses adresses.

Carte mémoire de l'APPLE II

0000-00FF	Page 0
0100-01FF	Pile du 6502
0200-02FF	Buffer d'entrée du clavier
0300-03CF 03D0-03EF 03F0-03FF	Vecteurs de données du DOS
0400-07FF	Page 2 du graphisme basse résolution

0800-0BFF	Programme utilisateur et données	Page 2 du graphisme basse résolution	RAM utilisée par l'interpréteur APPLESOFT chargé à partir d'une disquette
0C00-2000	Programme utilisateur du basic APPLESOFT en ROM (APPLE II +)		
2000-2FFF	Données du basic entier	Page 1 du graphisme	
3000-3FFF	Fin 16K mémoire	Haute résolution	Programme utilisateur RAM APPLESOFT
4000-5FFF		Page 2 du graphisme Haute résolution	
-7FFF	Fin 32K	Mémoire	
-BFFF	Fin RAM 48K		
C000-CFFF	Zone des cartes d'entrées-sorties		
D000-FFFF	ROM(s) { Basic Moniteur		

● Utilisation de la page zéro (\$0000, 00FF)

Le moniteur utilise dans la page 0 les adresses comprises entre \$20 et \$49 pour les fonctions générales, entre \$4E et \$4F pour lire le clavier, et pour la version Basic Entier (ancien moniteur) les adresses \$50-\$55.

Nous décrivons ci-dessous d'une manière plus précise l'utilisation de ces adresses, par le moniteur de l'APPLE II +.

Adresse décimale	Adresse hexa-décimale	Etiquettes	Utilisation
00-01	\$00-\$01	L0C0 L0C1	Ces adresses sont utilisées par la ROM auto-start au démarrage de la machine pour déterminer la présence ou non d'un contrôleur et pour charger le DOS.
32	\$20	WNDLFT	Colonne de début (gauche de la fenêtre sur l'écran). Elle est utilisée par le sous-programme VTABZ qui positionne BASL, H à partir de WNDLFT et CV. Après modification du contenu de cette adresse, le résultat sera effectif au prochain affichage.
33	\$21	WNDWDTH	Largeur de la fenêtre sur l'écran. Elle est utilisée par le sous-programme COUT pour déterminer le passage à la ligne suivante.
34	\$22	WNDTOP	Ligne la plus haute de la fenêtre sur l'écran. 0 à 22 pour du texte seulement. 20 à 22 pour un mélange texte graphisme basse résolution. Elle est utilisée pour indiquer la ligne à supprimer et les lignes à déplacer lorsque l'on doit ajouter une ligne en bas de l'écran. C'est aussi la ligne à laquelle on revient après l'instruction HOME.
35	\$23	WNCBTM	C'est la dernière ligne de la fenêtre sur l'écran. En fait WNCBTM contient le numéro de la ligne suivante. <div style="border: 1px solid black; padding: 5px; width: fit-content; margin: 10px auto;">fenêtre</div> ←WNCBTM→ Sa valeur est comprise entre (WNTOP) + 1 et 24. Elle est testée lors de l'affichage de CR (retour chariot-return) et de LF (passage à la ligne suivante).

Adresse décimale	Adresse hexa-décimale	Etiquettes	Utilisation
36	\$24	CH	Positionnement horizontal du curseur relativement au début de la fenêtre. C'est la position où l'on affichera le prochain caractère. Elle doit être comprise entre 0 et (WNDWDTH) - 1. Un passage à la ligne suivante sera automatiquement généré si l'on débord d'une ligne de la fenêtre. Cette adresse est utilisée par les sous-programmes d'affichage de caractères sur l'écran.
37	\$25	CV	Position du curseur par rapport au haut de l'écran (et non par rapport au haut de la fenêtre). Cette position est utilisée uniquement pour déterminer BASL, H. Si la valeur contenue à cette adresse indique une ligne inférieure à la limite WNCBTM de la fenêtre, toutes les lignes de celle-ci seront déplacées d'une ligne vers le haut lorsque la ligne courante sera remplie.
38-39	\$26-\$27	GBASL GBASH	Adresse mémoire où l'on doit ranger le prochain point à afficher en basse résolution.
40-41	\$28-\$29	BASL BASH	Adresse mémoire où l'on doit ranger le prochain caractère à afficher. C'est une fonction de CV et de WNDLFT. Elle est positionnée par BASCALC.
42-43	\$2A-\$2B	BAS2L BAS2H	Zone de travail lors du déplacement vers le haut des lignes de la fenêtre.
44-45	\$2C-\$2D	H2 V2 LMNEM RMNEM RTNL RTNH	Plusieurs utilisations. Dernier point à afficher sur la ligne tracée par HLINE (0 à 39). Pointeur utilisé par la partie désassembleur sur la table des mnémoniques 6502. Utilisées par la commande Trace pour conserver le mode de l'écran. 0-39 texte et basse résolution. 0-47 haute résolution. On garde la dernière ligne atteinte par VLINE (tracé vertical).

Adresse décimale	Adresse hexa- décimale	Etiquettes	Utilisation
46	\$2E	FORMAT CHKSUM MASK	Plusieurs utilisations. Utilisée par le désassembleur pour sauvegarder un code caractérisant l'instruction traitée. Utilisée pour calculer le code de vérification lors de la lecture d'information à partir de la cassette. Cette adresse est utilisée par PLOT pour déterminer si le point est sur la ligne du haut ou du bas de l'adresse pointée par GBASL, H MASK = \$OF MASK = \$FO
47	\$2F	LENGTH SIGN LASTIN	Plusieurs utilisations. Longueur de l'instruction traitée par le désassembleur. Le bit 1 est positionné à 1 si le résultat de MULPM ou DIVPM (multiplication ou division 16 bits, ancien moniteur) est à compléter par le programme appelant. Utilisée par RDBIT comme zone de travail lorsque l'on utilise la cassette.
48	\$30	COLOR	Code de la couleur positionné par SET COL, COLOR. Sa valeur est 17* code indiqué dans le Basic.
49	\$31	MODE	
50	\$32	INVFLG	Mode de l'écran : \$FF mode normal (noir sur blanc). \$3F mode inverse (blanc sur noir). \$7F mode clignotant. Il est utilisé par COUT1 et peut être positionné par SETNORM, RESET, SETINV. Il n'existe pas de sous-programme positionnant le mode clignotant.
51	\$33	PROMPT	Cet octet contient le caractère affiché par le sous-programme GETLN de saisie de données au clavier.

Adresse décimale	Adresse hexa- décimale	Etiquettes	Utilisation
52 53	\$34 \$35	YSAV YSAV1	Zones de sauvegardes du registre Y par le moniteur et par COUT.
54 55	\$36 \$37	CSWL CSXH	Adresse du sous-programme d'affichage des caractères sur l'écran. Elle est positionnée par défaut à l'adresse de COUT1. Vous utiliserez cette adresse si vous avez un autre écran-clavier où afficher des données. La commande n CTRL-P positionne cette adresse à la valeur Cnoo (carte d'extension numéro n).
56 57	\$38 \$39	KSWL KSWH	Même chose pour la saisie de données. La commande du moniteur utilisée est n CTRL-K Par défaut la valeur de cette adresse est celle du sous-programme KEYIN.
58 59	\$3A \$3B	PCL PCH	Zone utilisée pour sauvegarder le pointeur sur la prochaine instruction à exécuter (PC). Elle est utilisée par le miniassembleur pour savoir où placer la prochaine instruction. Les commandes L, G, S, T positionnent ou utilisent cette adresse. Ces deux octets contiennent aussi lors d'une instruction BRK l'adresse de l'instruction qui suit BRK.
60-67	\$3C \$3D-\$43	XQT XQTNZ	Zone de 8 octets de travail réservée au moniteur. Elle est utilisée notamment par les commandes : S } ancien T } moniteur SUBTRACT MOVE VERIFY READ WRITE STORE examen mémoire

Adresse décimale	Adresse hexa-décimale	Etiquettes	Utilisation
69	\$45	ACC	Sauvegarde de l'accumulateur.
70	\$46	XREG	Sauvegarde du registre X.
71	\$47	YREG	Sauvegarde du registre Y.
72	\$48	STATUS	Sauvegarde du registre d'état (flags).
73	\$49	SPNT	Sauvegarde du pointeur de pile. Le sous-programme SAVE réalise ces sauvegardes. Le sous-programme RESTORE les reprend.
74-77	\$4A-\$4D		Zone inutilisée.
78-79	\$4E-\$4F	RNDL-RNDH	Compteur sur 16 bits incrémenté par KEYIN pour tester les touches du clavier.
80	\$50	ACL	Zone utilisée uniquement par les sous-programmes de multiplication et de division de l'ancien moniteur.
8	\$51	ACH	
82	\$52	XTNDL	
83	\$53	XTNDH	
84	\$54	AUXL	
85	\$55	AUXH	

• Pages un à trois

* PAGE 1 (\$0100-\$01FF)

La page 1 est la zone de la pile 6502. Le moniteur utilise cette zone uniquement par le biais des instructions 6502 utilisant la pile (PHA, JSR, RIS...). Le moniteur n'initialise jamais ni la zone, ni le pointeur de pile (registre S).

* PAGE 2 (\$0200-\$02FF)

La page 2 est le buffer où sont rangées les informations saisies au clavier par le sous-programme GETLN. Celui-ci initialise le registre X comme index, puis à l'étiquette ADDINP le caractère saisi est rangé à l'adresse \$0200+X. Au retour de GETLN, le dernier caractère dans le buffer est return (code \$8D=\$80+\$0D).

* PAGE 3 (\$0300-\$03FF)

La page 3 contient certains vecteurs d'adresses utilisés pour des interruptions. Ces vecteurs sont situés en haut de la page 3.

Les adresses \$03D0 à \$03EF sont utilisées par le DOS. Le reste de la page est utilisable par des programmes assembleurs.

Le tableau ci-dessous décrit les vecteurs d'interruptions.

Adresse hexadécimale	Adresse décimale	Utilisation
\$0300-\$03CF	768-995	Zone libre
\$03D0-\$03EF	995-1007	DOS (cf. tome 2, § 1)
\$03F0-\$03F1	1008-1009	Vecteur d'interruption BRK (adresse)
\$03F2-\$03F3	1010-1011	Vecteur d'interruption RESET utilisé par l'autostart (adresse)
\$03F4	1012	Indicateur de chargement du système : — Si (\$A5) ou ((\$03F3))=(\$03F4) contenu de 3F3 On recharge complètement le système, y compris le DOS. — Sinon on arrête le programme en cours
\$03F5-\$03F7	1013-1015	Réservé par l'APPLESOFT (instruction)
\$03F8-\$03FA	1016-1018	Lorsque l'utilisateur tape CIRC-Y
\$03FB-\$03FD	1019-1021	Vecteur des interruptions non masquables (instruction à exécuter) (NMI)
\$03FE-\$03FF	1022-1023	Vecteur d'interruption IRQ (adresse de branchement)

• Pages de textes et graphisme basse résolution

Les adresses comprises entre \$0400 et \$07FF constituent la première page de texte et de graphisme basse résolution. C'est à ces adresses que sont copiées les informations à afficher sur l'écran. Celles-ci seront affichées par le biais d'une logique matérielle de gestion d'écran.

A l'adresse \$0800 commence généralement le programme utilisateur. Toutefois pour les systèmes équipés en RAM, la zone comprise entre les adresses \$0800 et \$0BFF peut être utilisée comme seconde page de texte

et de graphisme basse résolution. Cette page est acceptée par le matériel, mais non par le moniteur.

Pour passer de la page 1 à la page 2, il suffit de faire

POKE - 16299,0

Inversement, il suffit de faire

POKE - 16300,0

La table ci-dessous indique les adresses où sont situées les lignes d'écran.

Ligne	Page 1		Page 2	
	Décimal	Hexadécimal	Décimal	Hexadécimal
0	1024	\$0400	2048	\$0800
1	1152	\$0480	2176	\$0880
2	1280	\$0500	2304	\$0900
3	1408	\$0580	2432	\$0980
4	1536	\$0600	2560	\$0A00
5	1664	\$0680	2688	\$0A80
6	1792	\$0700	2816	\$0B00
7	1920	\$0780	2944	\$0B80
8	1064	\$0428	2088	\$0828
9	1192	\$04A8	2216	\$08A8
10	1320	\$0528	2344	\$0928
11	1448	\$05A8	2472	\$09A8
12	1576	\$0628	2600	\$0A28
13	1704	\$06A8	2728	\$0AA8
14	1832	\$0728	2856	\$0B28
15	1960	\$07A8	2984	\$0BA8
16	1104	\$0450	2128	\$0850
17	1232	\$04D0	2256	\$08D0
18	1360	\$0550	2384	\$0950
19	1488	\$05D0	2512	\$09D0
20	1616	\$0650	2640	\$0A50
21	1744	\$06D0	2768	\$0AD0
22	1872	\$0750	2896	\$0B50
23	2000	\$07D0	3024	\$0BD0

Les 40 caractères d'une ligne sont groupés en mémoire. La ligne numéro L de la page P est située à l'adresse :

$$\text{ADDRL} = 1024 * P + 256((L/2) \bmod 4) + (128 * (L \bmod 2)) + 40((L/8) \bmod 4)$$

Nous pouvons noter que 24 lignes de 40 caractères occupent 960 octets. Or 1 024 octets sont réservés en mémoire pour simplifier l'organisation matérielle. 64 octets sont donc disponibles par page en 8 groupes de 8 octets.

Ces octets ne restent pas disponibles dans la page 1. Ils sont utilisés par les cartes d'extension de la manière suivante :

Commun	Plot 1	Plot 2	Plot 3	Plot 4	Plot 5	Plot 6	Plot 7
\$0478	\$0479	\$047A	\$047B	\$047C	\$047D	\$047E	\$047F
\$04F8	\$04F9	\$04FA	\$04FB	\$04FC	\$04FD	\$04FE	\$04FF
\$0578	\$0579	\$057A	\$057B	\$057C	\$057D	\$057E	\$057F
\$05F8	\$05F9	\$05FA	\$05FB	\$05FC	\$05FD	\$05FE	\$05FF
\$0678	\$0679	\$067A	\$067B	\$067C	\$067D	\$067E	\$067F
\$06F8	\$06F9	\$06FA	\$06FB	\$06FC	\$06FD	\$06FE	\$06FF
\$0778	\$0779	\$077A	\$077B	\$077C	\$077D	\$077E	\$077F
\$07F8	\$07F9	\$07FA	\$07FB	\$07FC	\$07FD	\$07FE	\$07FF

L'adresse 2040 (\$07F8) est réservée et doit contenir Cn où n est le numéro du plot du périphérique actif. Ceci permet de gérer les interruptions.

Nous vous déconseillons de sauvegarder en bloc les adresses \$0400 et \$07FF car lorsque vous voudrez réutiliser cette zone, les adresses réservées aux périphériques seront écrasées et cela peut créer des problèmes.

• Mémoire réservée par les entrées-sorties

En plus des 64 octets définis précédemment, une grande zone de mémoire est réservée pour les entrées-sorties entre les adresses \$C000 et \$CFFF. Nous allons la décrire dans ce paragraphe.

* ADRESSES SPÉCIALES

Nous examinons ici les octets utilisés par le clavier, la cassette, le haut-parleur, les manettes de jeux et pour le choix du graphisme.

CLAVIER :

Le clavier utilise 2 octets situés aux adresses \$C000 et \$C010 dont les rôles sont les suivants :

\$C000(−16384) 1 bit d'état (caractère reçu)
7 bits de données (codes ASCII standards).

Ceci fait que les codes traités par l'APPLE II sont compris entre 128 et 255 si le bit d'état n'est pas remis à 0.

\$C010(−16368) Ecrire à cette adresse signifie que le caractère a été lu. Le bit d'état est alors remis à 0.

CASSETTE :

La gestion de la cassette utilise 2 octets en mémoire situés aux adresses :

\$C060(−16288) : IN entrée
\$C020(−16352) : OUT sortie

Décrivons le fonctionnement de la cassette :

Pour écrire des informations binaires (0, 1) sur une cassette, on utilise l'octet d'adresse \$C020. A cette adresse se trouve une bascule (logique conservant un état) qui change l'état chaque fois qu'on la référence dans une instruction. Une lecture de la valeur contenue à cette adresse provoque un changement d'état, une écriture deux changements d'états. En effet une écriture mémoire par le 6502 est toujours précédée par une lecture. Il existe entre la bascule et la prise OUT une logique matérielle qui transforme les changements d'états logiques (0, 1) en modification de la tension de sortie (0/V-25 mV). De cette manière des informations sont stockées sur cassette. Comment l'APPLE II les relit-il ? Il existe pour cela une logique matérielle décodant les modifications de tension en entrée pour créer des 0 et des 1. Un programme peut tester la valeur générée en lisant l'adresse \$C060. Si la valeur obtenue est supérieure à 128, l'état est 1. Sinon il est nul.

La lecture des données doit être faite en assembleur, le Basic étant dépassé par la rapidité.

HAUT-PARLEUR :

Chaque référence à l'adresse \$C030(−16336) provoquera l'émission d'un « beep ». Comme pour la cassette, une écriture changera deux fois l'état de la bascule à cette adresse et il ne se passera rien.

MANETTES DE JEUX :

Le connecteur des manettes de jeux fournit :

1. trois entrées numériques sur un bit 0/1
2. trois sorties numériques sur un bit 0/1
3. quatre entrées analogiques
4. un signal de validation des données.

Vous pouvez utiliser ce connecteur pour tout autre objet que les manettes de jeu (relais, haut-parleurs...) et le contrôler par programme.

1. Entrées numériques :

Trois entrées digitales sont acceptées. Leurs états peuvent être lus (de la même manière qu'une cassette) aux adresses \$C061, \$C062, \$C063 (i.e. de −16287 à −16285). S'ils sont négatifs, le bouton est appuyé.

2. Sorties numériques :

Elles sont gérées par des « bascules logicielles » utilisant 2 octets de mémoire. Lire un octet mettra la « bascule » dans un état, lire l'autre octet mettra la « bascule » dans l'autre état.

Les adresses utilisées sont les suivantes :

0	{	OFF	\$C058	(−16296)
		ON	\$C059	(−16295)
1	{	OFF	\$C05A	(−16294)
		ON	\$C05B	(−16293)
2	{	OFF	\$C05C	(−16292)
		ON	\$C05D	(−16291)
3	{	OFF	\$C05E	(−16290)
		ON	\$C05F	(−16289)

OFF correspond à $\begin{cases} 0 & \text{logique} \\ 0 \text{ V} & \text{tension de sortie} \end{cases}$

ON correspond à $\begin{cases} 1 & \text{logique} \\ 5 \text{ V} & \text{tension de sorties} \end{cases}$

3. Entrées analogiques :

Les quatre entrées analogiques peuvent être connectées à des potentiomètres de 150 K ohms. Pour lire les entrées analogiques, un programme assembleur doit procéder de la manière suivante :

— remettre à 0 (RESET) les circuits compteurs en lisant le contenu de l'adresse \$C070 (-16272). Les valeurs des contenus des adresses \$C064 à \$C067 (-16284 à -16281) sont alors supérieures à 128 ;

— observer le temps qu'il faut pour que ces valeurs redescendent en-dessous de 128. Ce temps est proportionnel à la valeur prise par le potentiomètre placé entre 5 V et l'entrée. Le temps maximal est de 3,060 ms.

S'il n'y a pas de potentiomètre à l'entrée ou si la valeur de la résistance est trop forte, les valeurs lues n'atteindront jamais 0.

4. Signal de validation Strobe en sortie :

Ce signal permet de valider les données sorties de l'APPLE. Il est contrôlé en référençant l'adresse \$C040. Ceci provoque le passage du signal de 5 V à 0 V pendant 1/2 s.

GRAPHISMES :

Trois types d'information peuvent être affichées sur l'écran :

- texte 24 lignes de 40 caractères,
- basse résolution en graphisme
48 lignes de 40 carrés de diverses couleurs,
- haute résolution en graphisme
192 (hauteur) × 280 (largeur) points sur l'écran.

Comment passer d'un type d'informations à un autre ?

Il existe 4 « bascules logicielles » à 2 positions permettant de sélectionner les types d'information. Elles correspondent aux adresses suivantes :

Adresse hexadécimale	Adresse décimale	Description
\$C050 \$C051	-16304 -16303	type graphique type texte
\$C052 \$C053	-16302 -16301	texte ou graphisme non mélangés texte et graphisme mélangés
\$C054 \$C055	-16300 -16299	page 1 page 2
\$C056 \$C057	-16298 -16297	basse résolution haute résolution

Il existe 10 combinaisons de ces « bascules logicielles » :

Page 1

Page 2

Ecran	Adresses		Ecran	Adresses	
Texte	\$C054		texte	\$C055	
	\$C051			\$C051	
Basse résolution	\$C054	\$C056	basse résolution	\$C055	\$C056
	\$C052	\$C050		\$C052	\$C050
Haute résolution	\$C054	\$C057	haute résolution	\$C055	\$C057
	\$C052	\$C050		\$C052	\$C050
Texte et basse résolution	\$C054	\$C056	texte et basse résolution	\$C055	\$C056
	\$C053	\$C050		\$C053	\$C050
Texte et haute résolution	\$C054	\$C057	texte et haute résolution	\$C055	\$C057
	\$C053	\$C050		\$C053	\$C050

Tapez le programme suivant :

```
10 A=PEEK(-16299)
20 A=PEEK(-16297)
30 A=PEEK(-16301)
40 A=PEEK(-16304)
```

puis RUN.

L'écran affiche la page 2 en mixte texte et haute résolution.

Tapez ensuite :

```
A=PEEK(-16302)
```

les lignes de texte sont alors supprimées.

Puis :

```
A=PEEK(-16303)
```

Le graphisme haute résolution laisse l'écran pour le texte.

Tapez enfin :

```
A=PEEK(-16300)
```

Vous repassez sur la page 1. Vous voyez alors ce que vous avez tapé précédemment.

* CARTES D'EXTENSION

En plus des 8 groupes de 8 octets définis dans la page 1, les cartes d'extension peuvent disposer chacune de :

- 16 adresses réservées pour les entrées-sorties (buffer...);
- 256 octets de ROM contenant les drivers (contrôleurs de périphériques). Ces programmes sont de préférence indépendants du plot dans lequel on met la carte. Pour déterminer le plot « actif » on pourra utiliser le programme suivant :

```
20 4A FF JSR $FF4A
78      SEI
20 58 FF JSR $FF58
BA      TSX
BD 00 01 LDA $0100,X
8D F8 07 STA $07F8
29 OF    AND $0F
A8      TAY
```

Le numéro du connecteur est dans le registre Y.

Les tableaux ci-dessous résument ces adresses.

	\$0	\$1	\$2	\$3	\$4	\$5	\$6	\$7	\$8	\$9	\$A	\$B	\$C	\$D	\$E	\$F
\$C080				0												
\$C090				1												
\$C0A0				2												
\$C0B0				3												
\$C0C0				4												
\$C0D0				5												
\$C0E0				6												
\$C0F0				7												

	\$00	\$10	\$20	\$30	\$40	\$50	\$60	\$70	\$80	\$90	\$A0	\$B0	\$C0	\$D0	\$E0	\$F0
\$C100				1												
\$C200				2												
\$C300				3												
\$C400				4												
\$C500				5												
\$C600				6												
\$C700				7												

Il existe une zone comprise entre les adresses \$C800 et \$CFFF réservée à une ROM (ou PROM) de 2 K octets. Sur chaque carte d'extension peut se trouver cette ROM, qui sera connectée sur le bus d'adresse lorsque la carte d'extension sera sélectionnée. Ceci permet d'avoir des contrôleurs de périphériques plus complets.

Une carte d'extension voulant utiliser cette zone doit indiquer aux autres cartes de se connecter. Pour cela, référencer l'adresse \$CFFF.

● Sous-programmes d'entrées-sorties du moniteur

Nous allons examiner successivement les sous-programmes gérant le clavier, l'écran et le graphisme, la cassette, les manettes de jeu et le haut-parleur, pour l'APPLE II +.

● Gestion du clavier

Les sous-programmes de gestion du clavier réalisent les opérations suivantes :

- saisie d'un caractère,
- écho du caractère,
- stockage du caractère dans un buffer.

Le tableau ci-dessous décrit les différents sous-programmes (adresse, paramètres d'appel, de retour, rôle, structure interne, ...).

Sous-programmes de gestion du clavier

Nom	Adresse		Paramètres d'appel	Paramètres de retour	Structure interne	Sémantique
	décimale	hexa-décimale				
GETLNZ	-665	\$FD67	CV = ligne à saisir	A = CR(\$8D) X = nombre de caractères saisis avant CR Y = (WINDWDTH) CH = 0 CV = numéro ligne	Emission de retour chariot (via CROUT) Branchement à l'étiquette GETLN	Passage à la ligne suivante et saisie d'une nouvelle ligne
GETLN	-662	\$FD6A	CH = colonne du caractère d'en-tête CV = n° ligne	idem GETLNZ	GETLN affiche le caractère d'en-tête et initialise le registre X comme index de stockage dans le buffer Le contrôle est passé à NXTCHAR	Affichage du caractère d'en-tête (*moniteur, JAPPLE-SOFT, > Basic Entier, ?INPUT, ...) et saisie d'une ligne
NXTCHAR	-651	\$FD75	X = \$0200 CV, CH	idem GETLNZ	Appel de RDCHAR pour saisir le prochain caractère : — si le caractère est CTRL-V (i.e. →), il recherche dans la mémoire d'écran, le caractère que l'on conserve — si le caractère est une lettre minuscule, transformation en majuscule — si le caractère est un retour chariot, branchement en CLREOL et au retour en COUT (remise du reste de la ligne à blanc et affichage de retour chariot) — sinon branchement en NOTCR	Saisie d'une ligne sans affichage préliminaire

Nom	Adresse		Paramètres d'appel *	Paramètres de retour	Structure interne	Sémantique
	décimale	hexa-décimale				
NOTCR	-707	\$FD3D			Sauvegarde du mode écran (normal, inverse, ...) Affichage du caractère en mode normal Restauration du mode écran Test du caractère : * Backspace (+-) branchement en BCKSPC * Line cancel (CTRL-X) branchement en CANCEL — sinon test du remplissage de la ligne. Pour plus de 247 caractères saisis, appel de BELL pour prévenir Passage à NOTCRI	
NOTCRI	-673	\$FD5F			Incréméntation du registre X et : — si débordement de ligne, appel de CANCEL — sinon retour à NXTCHAR pour le traitement du caractère suivant	
CANCEL	-670	\$FD62			Affichage/via COUT Retour à GETLNZ pour réinitialiser la ligne	Effacement ligne courante
BCKSPC	-655	\$FD71			A l'appel, Backspace a déjà été affiché : — si le contenu du registre X est nul, retour à GETLNZ pour réinitialiser la ligne ; — sinon décrémenter le registre X	

Nom	Adresse		Paramètres d'appel	Paramètres de retour	Structure interne	Sémantique
	décimale	hexa-décimale				
RDCHAR	-715	\$FD35	CH, CV, BASL, H (cf. page 0)	A = caractère saisi Y=CH X non affecté CV, CH, BASL, H non modifiés sauf par les fonctions A B C D ESC	Appel de RDKEY pour saisir le caractère : — si le caractère est ESC (ESCAPE) : * transfert du contrôle à ESC1 (ancien moniteur) ESCNEW (Autostart)	Saisie d'un caractère au clavier et réalisation fonctions ESC(modification curseur)
RDKEY	-756	\$FD0C	CH, CV, BASL, H (cf. page 0)	A = caractère saisi	Supprime de l'affichage le caractère si-tué sous le curseur, appelle le sous-programme qui saisit physiquement le caractère et restaure le caractère sous le curseur (s.p. à l'adresse (KSWL, H))	Saisie d'un caractère
KEYIN	-741	\$FD1B	A = caractère à restaurer	A = caractère lu	Lit le caractère au clavier et restaure le caractère situé sous le curseur	

• Gestion de l'écran

De multiples fonctions de gestion de l'écran sont offertes par le moniteur. Nous allons examiner successivement les points suivants :

- affichage de texte dans la fenêtre définie sur l'écran,
- contrôle du mode d'écran, du type d'informations (texte, graphisme),
- fonctions de manipulation des données de la fenêtre (effacement, mouvement...),
- affichage de texte à l'extérieur de la fenêtre,
- utilisation des secondes pages graphiques,
- utilisation du graphisme basse résolution,
- etc...

* AFFICHAGE DE TEXTE DANS LA FENETRE

C'est le mode de fonctionnement standard de l'APPLE II.

L'affichage d'un caractère se fait par appel du sous-programme COUT, qui se branche à l'adresse contenue dans CSWL, H. Cette adresse est positionnée par défaut au RESET avec l'adresse de COUT1 qui est le sous-programme d'affichage d'un caractère sur l'écran.

L'utilisateur peut changer de périphérique avec les commandes n CTRL-P (moniteur) ou PR n (Basic) qui changent CSWL, H ≠.

Les octets suivants gèrent la position du curseur sur l'écran :

CV	ligne du curseur
CH	colonne du curseur-colonne de début de la fenêtre à gauche
BASL, H	adresse du curseur dans la mémoire d'écran
WNDLFT	colonne de début de la fenêtre à gauche
WNDWDTH	largeur de la fenêtre
WNDTOP	ligne du haut de la fenêtre
WNBDM	ligne située juste dessous la fenêtre.

Tableau des sous-programmes utilisés

Fonction	Adresse hexa-décimale	Adresse décimale	Nom	Registres détruits
Branchement de l'adresse contenue dans CSWL,H	\$FDED	-531	COUT	aucun
Affichage d'un caractère utilisant INVFLG et acceptant les mouvements de curseur	\$FDF0	-528	COUTI	aucun
Affichage d'un caractère avec mouvement du curseur	\$FDF6	-522	COUTZ	aucun
Passage à la ligne suivante (affichage CR)	\$FD8E	-626	CROUT	A
Affichage de ERR via COUT et émission d'un 'beep'	\$FF2D	-211	PRERR	A
Emission d'un 'beep' par COUT	\$FF3A	-198	BELL	A
Positionnement de BASCL,H en fonction de CV et WNDLFT	\$FC22	-990	VTAB	A
Positionnement de BASL,H en fonction de (A) et WNDLFT	\$FC24	-998	VTABZ	A
Positionnement du curseur au début de la ligne (hors fenêtre)	\$FBCI	-1087	BASCALC	A

* GESTION DES MODES DE L'ÉCRAN

Les modes de l'écran sont le texte, le graphisme basse résolution, le graphisme haute résolution, les modes inverses, clignotants.

* MANIPULATIONS DES DONNÉES DE L'ÉCRAN

Trois types de manipulation des données de l'écran existent :

- fonctions graphiques ESC,
- effacement de parties d'écran et déplacement du curseur,
- déplacement des lignes de la fenêtre vers le haut avec suppression de la première ligne et création d'une nouvelle ligne en bas de la fenêtre.

Tableau des sous-programmes de gestion des modes de l'écran

Fonction	Adresse hexa-décimale	Adresse décimale	Nom	Registres détruits
Remise à zéro de la haute résolution	\$FB33	-1229	SETTXT	A
Passage à la première page de l'écran	\$FB36	-1226		A
Passage en mode texte	\$FB39	-1223		A
Chargement de A = 0 pour WNDTOP et branchement en SETWND	\$FB3C	-1220		A
Passage en mode graphique	\$FB40	-1216	SETGR	A, Y
Passage en mode mixte texte/graphique	\$FB43	-1213		A, Y
CALL CLRTOP (remise à zéro du graphisme)	\$FB46	-1210		A, Y
Chargement de A = 20 pour WNDTOP	\$FB49	-1207	SETND	A
Chargement de A dans WNDTOP	\$FB4B	-1205		A
A = 0	\$FB4D	-1203		A
Chargement de A dans WNDLEFT	\$FBEF	-1201		A
A = 40	\$FB51	-1199	TABV	A
Chargement de A dans WNDWDTH	\$FB53	-1197		A
A = 24	\$FB55	-1195		A
Chargement de A dans WNDBTM	\$FB57	-1193		A
A = 23	\$FB59	-1191		A
Chargement de A dans CV	\$FB5B	-1189		A
Branchement en VTAB				
Passage en mode normal	\$FE84	-380	SETNORM	Y
Passage en mode inverse	\$FE80	-384	SETINV	Y
Stockage de Y dans INVFLG	\$FE86	-374	SETIFLG	aucun

* FONCTIONS GRAPHIQUES ESC

Ce sont les suivantes :

ESC	HOME
ESC A	déplacement du curseur vers la droite
ESC B	BS
ESC C	LF
ESC D	VP

ESC E appel de CLREOL
 ESC F appel de CLREOL
 ESC autre touche retour à l'appelant

Le sous-programme traitant ces fonctions est

ESC I (ancien moniteur)
 ESC NEW (autostart)

* EFFACEMENT DE PARTIES D'ÉCRAN ET MANIPULATIONS DU CURSEUR

La table d'adresses suivante indique les sous-programmes utiles pour effacer certaines parties d'écran ou pour déplacer le curseur.

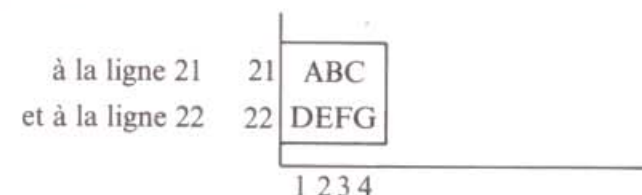
Fonction	Nom	Adresse hexa-décimale	Adresse décimale	Registres détruits
Effacement des caractères sur l'écran entre la position CV,CH jusqu'au bas de la fenêtre	CLREOF	\$FC42	-958	A, Y
Effacement de tout l'écran et positionnement du curseur en haut et à gauche de la fenêtre (utile en Basic Entier)	HOME	\$FC58	-936	A, Y
Effacement des caractères situés après le curseur dans la ligne	CLREOL	\$FC9C	-868	A, Y
Retour au début de la ligne (CH = 0)	CR	\$FC62	-926	A, Y
Passage à la ligne suivante (CV = CV + 1) et appel de SCROLL si nécessaire	LF	\$FC66	-922	A, Y
Retour à la ligne précédente (CV = CV - 1)	VP	\$FCIA	-998	A
Déplacement du curseur vers la droite (CH = CH + 1)	ADVANCE	\$FBF4	-1036	A
Déplacement du curseur vers la gauche (CH = CH - 1)	BS	\$FC10	-1008	A

* DÉPLACEMENT DES LIGNES DE LA FENÊTRE A UNE POSITION VERS LE HAUT (SCROLL)

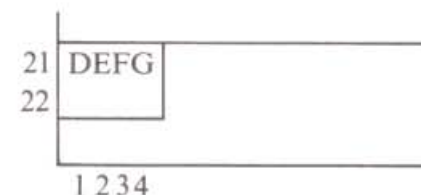
Le sous-programme SCROLL réalise cette fonction. Il est situé à l'adresse \$FC70 (-912).

Exemple : si WNDLFT = 1 (colonne de gauche)
 WNDWDTH = 4 (largeur)
 WNDTOP = 21 (ligne du haut)
 WNDBTM = 23 (ligne sous la fenêtre)

et si l'on a sur l'écran



et si l'on appelle SCROLL, on obtiendra



* AFFICHAGE DE TEXTE A L'EXTÉRIEUR DE LA FENÊTRE ET UTILISATION DE LA SECONDE MÉMOIRE D'ÉCRAN

Dans les deux cas, le moniteur n'effectue aucun travail. C'est à l'utilisateur de mettre les caractères à afficher dans les mémoires d'écran.

Si vous désirez avoir un texte dans une page mémoire d'écran, et dans l'autre du graphisme basse résolution, il est hautement préférable de réserver la première page pour le graphisme et la seconde pour le texte, celui-ci étant plus facile à gérer.

* GESTION DU MODE GRAPHISME BASSE RÉOLUTION

Dans le mode graphisme basse résolution standard, le moniteur utilise les 20 premières lignes de texte pour 40 lignes graphiques, et laisse 4 lignes de texte en bas de l'écran.

Ceci est dû au fait que la zone mémoire occupée par un caractère est utilisée pour deux points basse résolution. Les informations relatives à ceux-ci sont stockées sur deux fois quatre bits et indiquent la couleur. La distinction des points se fait avec l'octet MASK.

Le moniteur offre des sous-programmes permettant :

- d'effacer la partie graphique de l'écran,
- de positionner des couleurs,
- de placer un point sur l'écran,
- de tracer des lignes verticales ou horizontales,
- de déterminer la couleur d'un point de l'écran.

Le tableau ci-dessous donne les différentes adresses des sous-programmes utilisables.

Fonction	Nom	Adresse hexa-décimale	Adresse décimale	Registres détruits
Affichage sur l'écran d'un point de coordonnées ((A),(Y))	PLOT	\$F800	-2048	A
Tracé d'une ligne horizontale sur l'écran ((A)=n° de ligne, (Y)=1 ^{re} colonne, (44)=dernière colonne)	HLINE	\$F819	-2023	A, Y
Tracé d'une ligne verticale sur l'écran ((Y)=n° de colonne, (A)=1 ^{re} ligne, (45)=dernière ligne)	VLINE	\$F828	-2008	A
Effacement de l'écran (48 lignes graphiques)	CLRSCR	\$F832	-1998	A, Y
Effacement du graphisme (40 lignes graphiques)	CLRTOP	\$F836	-1994	A, Y
Positionnement de la couleur indiquée dans l'accumulateur	COLOR	\$F85F	-1953	A
Détermination de la couleur du point de coordonnées ((A),(Y))	SCRN	\$F864	-1948	A = couleur

- (A) désigne le contenu du registre A
 (X) désigne le contenu du registre X
 (Y) désigne le contenu du registre Y
 (nn) désigne le contenu de l'adresse nn

• Gestion du haut-parleur par le moniteur

Il existe plusieurs manières d'utiliser le haut-parleur de l'APPLE II. La plus utilisée correspond à signaler des événements de programmes. Il existe par exemple un sous-programme qui fait vibrer le haut-parleur à la fréquence de 1 kHz pendant 0,1 s. Celui-ci est utilisé par RESET, par la touche CTRL-G.

Table d'adresses, précisant les sous-programmes du moniteur qui utilisent le haut-parleur

Fonction	Adresse hexa-décimale	Adresse décimale	Etiquette	Registres détruits
Si le contenu de l'accumulateur est \$87(CTRL-G, Bell, code ASCII = 7), ce sous-programme attend 0,01s et émet un "beep"	\$FBD9	-1063	BELL1	A, Y
Attente de 0,01 s et émission d'un "beep"	\$FBDD	-1059		A, Y
Charger le registre Y avec la valeur 192 pour 0,1 s de "beep"	\$FBE2	-1054		A, Y
Fait résonner ("beep") le haut-parleur à la fréquence de 1 kHz sur le nombre de fois contenu dans le registre Y	\$FBE4	-1052	BELL2	A, Y
Affiche "ERR" et émet un "beep"	\$FF2D	-211	PRERR	A
Emission d'un "beep"	\$FF3A	-198	BELL	A

• Gestion de la cassette

Il existe deux sous-programmes principaux pour la gestion de la cassette qui permettent de la lire (READ) et d'y écrire (WRITE). D'autres sous-programmes sont utilisés de manière interne par le moniteur. Un fichier stocké sur cassette est décomposé en un ou plusieurs enregistrements. Un enregistrement est constitué d'une chaîne d'octets stockés en « même temps » à partir d'une zone physique, c'est une entité physique sur la cassette. Un fichier comporte plusieurs enregistrements, il est appelé entité logique.

Un programme Basic contient deux enregistrements. Etudions la structure de ces enregistrements pour les deux Basic de l'APPLE II.

***BASIC ENTIER:**

Le premier enregistrement contient uniquement 2 octets qui servent à stocker la longueur du second enregistrement. Celle-ci est utilisée par le Basic pour déterminer la fin de la zone mémoire occupée par le second enregistrement, là où le programme Basic est complet.

***APPLESOFT**

Le premier enregistrement contient dans ce cas 3 octets. Les deux premiers octets sont aussi la longueur du second enregistrement; le troisième octet indique la version de l'APPLESOFT (\$55 pour la version standard de l'APPLESOFT).

Fonction	Etiquette	Adresse hexadécimale	Adresse décimale
Ecrit sur la cassette : — un en-tête de 10 s — les données dont l'adresse de début est contenue dans A1L,H (\$3C,3D) et dont l'adresse de fin est contenue dans A2L,H (\$3E,\$3F) — un octet égal au OU EXCLUSIF de tous les octets écrits à titre de vérification lorsque l'on reliera les données	WRITE	\$FECD	-307
Lit la cassette. Il place le premier octet à l'adresse contenue dans A1L,H, le dernier à celle contenue dans A2L,H. Il lit ensuite un octet qui doit être égal au OU EXCLUSIF de tous les octets lus Dans le cas contraire ERR est affiché sur l'écran et CH	READ	\$FEFD	-259
Ce sous-programme écrit l'en-tête de synchronisation au début de chaque enregistrement de 10 s. Il est aussi utilisé pour indiquer le début réel des données lors de la lecture	HEADR	\$FCC9	-823
Recherche de deux transitions sur la cassette	RD2BIT	\$FCFA	-774
Lecture d'un bit	RDBIT	\$FCFD	-771
Lecture d'un octet	RDBYTE	\$FCEC	-788
Ecriture d'un bit	WRBIT	\$FCD6	-810
Ecriture d'un octet	WRBYTE	\$FEED	-275

Vous pouvez utiliser de même un ou plusieurs enregistrements pour vos fichiers. Chaque enregistrement est l'entité physique créée à l'appel du sous-programme WRITE. Il est lu par la fonction READ.

Nous donnons ci-dessus (p. précédente) la description des sous-programmes d'utilisation de la cassette par le moniteur.

• Gestion des manettes de jeux

L'APPLE II offre à l'utilisateur 4 entrées digitales, 4 sorties digitales, 4 entrées analogiques et 1 bit de validation (voir Adresses spéciales. Manettes de jeux). Les quatre entrées analogiques sont utilisées pour les manettes de jeux. Le moniteur lit ces entrées, en déclenchant des compteurs, et en mesurant le temps nécessaire pour qu'elles reviennent en dessous d'une certaine valeur. Le sous-programme utilisé est

PREAD à l'adresse \$FBIE(-1250)

le numéro (0-3) de la manette (entrée analogique) doit être dans le registre X avant l'appel de PREAD; la valeur lue est, au retour, dans le registre Y.

Deux lectures des entrées analogiques ne doivent pas être faites consécutivement, car lorsque le moniteur lit une valeur, il active les compteurs correspondant aux quatre entrées et ceux-ci peuvent ne pas être revenus à la bonne valeur lors d'un nouvel appel de PREAD. Pour éviter les erreurs, il convient donc de laisser s'écouler un certain délai entre deux lectures.

Il n'existe pas de sous-programmes du moniteur utilisant les quatre entrées et les quatre sorties digitales accessibles. L'accès à ces entrées/sorties se fera en utilisant POKE, PEEK en BASIC ou LDx, STx en assembleur. Les adresses correspondantes sont indiquées au paragraphe Adresses spéciales. Manettes de jeux. Ces sorties sont toutefois initialisées lors d'une interruption RESET par la version AUTOSTART du moniteur.

ANO et ANI sont initialisées à zéro (sorties 0 et 1 à l'état TTL bas), AN2 et AN3 sont initialisées à un (sorties 2 et 3 à l'état TTL haut).

• Sous-programme d'attente de délai WAIT

Le sous-programme WAIT consiste en un ensemble de boucles imbriquées de telle manière que le délai d'attente dépende du contenu de l'accumulateur.

WAIT est situé à l'adresse \$FCA8(-856).

Il peut être utilisé pour gérer une fréquence du haut-parleur, ou un périphérique lent tel qu'une imprimante ou un télétype.

Le temps qui s'écoule avec l'appel de WAIT peut se calculer par la formule suivante :

$$(2,5A2 + 13,5 * A + 13) * 1,023 \text{ ms.}$$

Le tableau ci-dessous indique quelques valeurs obtenues :

Contenu de l'accumulateur	Temps écoulé en secondes
1	0,000 029
5	0,000 14
9	0,000 34
18	0,001 09
26	0,002 10
32	0,003 07
37	0,004 02
42	0,005 10
46	0,006 06
50	0,007 09
60	0,010 05
75	0,015 43
105	0,029 66
150	0,059 63
205	0,110 32
240	0,150 64
255	0,169 83

● Gestion des interruptions sur l'APPLE II +

● Généralités

Gérer des interruptions est nécessaire, pour permettre à des événements d'être traités lorsqu'ils se produisent et à un programme de se dérouler dans l'intervalle séparant deux événements.

Lorsqu'une interruption (événement) se produit, le programme en cours est arrêté, son contexte sauvegardé et un programme spécifique est lancé. A la fin de celui-ci, le contexte est restauré et le programme interrompu relancé.

Le microprocesseur 6502 admet trois catégories d'interruptions. Ce sont les suivantes :

- *RESET
- *NMI (interruption non masquable)
- *IRQ (et instruction BRK #).

Lors d'une interruption, le processus suivant se produit :

— si l'interruption est NMI ou IRQ (BRK), le contenu du compteur de programme (PC) et le registre P (registre d'état) ainsi que les autres interruptions sont bloquées,

— sinon (RESET) la mémoire est mise en lecture jusqu'à ce que le sous-programme RESET soit actif,

— le microprocesseur 6502 transfère alors au sous-programme de gestion le contrôle de l'interruption à traiter en utilisant les vecteurs d'interruption situés aux adresses \$FFFA-\$FFFF.

Adresses des sous-programmes de gestion des interruptions

Interruption	Vecteur d'interruption	Nom	Ancien moniteur	Autostart
NMI	\$FFFA-B	NMI	\$03FB	\$03FB
RESET	\$FFFC-D	RESET	\$FF59	\$FA62
IRQ/BRK	\$FFFE-F	IRQ	\$FA86	\$FA40

Nous étudions plus précisément dans la suite de ce paragraphe, la gestion des différents types d'interruption.

● Interruption NMI

Le moniteur de l'APPLE II ne gère pas ces interruptions. Il assure seulement le transfert du contrôle du 6502 à l'instruction située à l'adresse \$03FB. Vous devrez donc y placer une instruction JUMP vers votre sous-programme de traitement s'il y a lieu.

● Interruption RESET

Celle-ci se produit lorsque vous appuyez sur la touche RESET au clavier ou lorsque vous mettez en route votre système, sauf pour les toutes premières versions. Son traitement se fait de manière très différente pour les deux versions du moniteur.

* ANCIEN MONITEUR

Quand une interruption RESET se produit, l'ancien moniteur établit la configuration matérielle suivante :

- clavier comme périphérique d'entrée de données,
- écran comme périphérique de sortie de données,
- mode normal de l'écran (blanc sur fonc noir), curseur en haut et à gauche (HOME), fenêtre de texte occupant tout l'écran,
- première mémoire d'écran sélectionnée.

Le contrôle est alors transféré à l'étiquette MON(\$FF65), un "beep" est émis et le moniteur est prêt à prendre une commande.

* ROM AUTOSTART

L'interruption RESET est ici traitée en plusieurs étapes :

1. Etablissement d'une configuration matérielle fixée : écran, clavier, sorties analogiques, mode texte, affichage d'APPLE II.

2. Si le contenu de la mémoire (adresses \$03F2—\$03F4, voir ci-dessous) indique que l'on vient de mettre en route le système, le moniteur initialise les vecteurs d'interruptions de BRK aux adresses \$03F0—\$03F1, de RESET aux adresses \$3F2—\$3F3, avec l'octet de validation situé à l'adresse \$3F4. Le rôle de cet octet est d'indiquer l'initialisation à faire : 2. ou 3.

Le moniteur Autostart recherche alors s'il existe une carte contrôleur de disquettes dans les plots 7 à 1. Dans le cas favorable, un branchement est effectué à l'adresse Cnoo (n numéro du plot du contrôleur) pour charger le DOS. Dans l'autre cas, la commande CTRL-B est simulée.

3. Si le système n'est pas à réinitialiser complètement, le moniteur teste le contenu du vecteur d'interruption (\$3F2—3F3).

S'il contient \$E000, le moniteur le positionne à l'adresse \$E003 (point d'entrée de CTRL-C) et se branche au début du Basic (\$E000). Sinon, il se branche à l'adresse indiquée.

Test réalisé pour savoir quelle initialisation opérer :

Ce test est basé sur l'octet de validation situé à l'adresse \$03F4. Si la valeur du OU EXCLUSIF entre les valeurs des octets des adresses \$3F3 et \$3F4 est \$A5, l'initialisation complète (2.) n'est pas à réaliser.

● Interruption IRQ/BRK

Quand une interruption IRQ se produit ou qu'une instruction BREAK est exécutée, le microprocesseur 6502 suit une séquence d'interruption. Celle-ci est la suivante :

— sur IRQ

- *interdiction d'autres interruptions,
- *branchement indirect ou sous-programme dont l'adresse est située à \$03FE—\$03FF,
- *autorisation d'interruptions;

— sur BRK

la séquence est identique, mais on se branche à l'étiquette BRK au lieu du sous-programme de traitement d'IRQ. Avec l'ancien moniteur l'instruction qui suit BRK est affichée ainsi que le contenu des registres. Avec la ROM autostart, le moniteur se branche à l'adresse contenue aux adresses \$03F0—\$03F1. Par défaut, ces adresses contiennent celles du même sous-programme que pour l'ancien moniteur.

● Utilisation des commandes du moniteur de l'APPLE II + comme sous-programmes

Placer dans le buffer de saisie de données (GETLN, à l'adresse \$200) les commandes à exécuter, appeler l'interpréteur de commandes du moniteur et en revenir. A cet effet, vous devrez créer un sous-programme assembleur dépilant 2 octets de la pile et rendant ensuite la main au programme appelant.

L'exemple suivant indique ce qu'il est nécessaire de faire :

```
10 REM APPEL DES COMMANDES DU MONITEUR
20 RM$ = "2FC:68 68 60 'N'2FCG"
30 GOSUB 1000
40 RM$ = "F800L'2FCG"
50 CALL - 936
60 GOSUB 1000
70 END
```

```

1000 A = 511
1010 L = LEN (RM$)
1020 FOR I=1 TO L
1030 B$ = MID$ (RM$,I,1)
1040 B = 128 + ASC (B$)
1050 POKE A+I,B
1060 NEXT
1070 CALL -144
1080 RETURN

```

Il est possible d'accéder aux commandes du moniteur d'une autre manière. Vous devrez alors créer des sous-programmes assembleur interfaçant votre programme et le moniteur.

Adresses des sous-programmes correspondant aux commandes du moniteur

Sous-programmes	Adresse décimale	Adresse hexa- décimale	Nom	Registres détruits
Point d'entrée du moniteur pour traiter CR comme un blanc	-512	\$FE00	BL1	A, X, Y
Point d'entrée du moniteur pour traiter un blanc	-508	\$FE04	BLANK	A, X, Y
Commande de modification de la mémoire	-501	\$FE0B	STOR	A
Positionnement du mode de traitement du moniteur lorsque l'on rencontre les caractères : + - .	-488	\$FE18	SETMODE	A, Y
Positionnement du mode par BLANK	-483	\$FE1D	SETMDZ	aucun
Commande < du moniteur (enregistrement des adresses dans A1L, H, A2L, H, A4L, H)	-480	\$FE20	LT	A, X
Commande MOVE du moniteur de A1L, H à A2L, H vers A4L, H	-468	\$FE2C	MOVE	A(Y=0)
Commande verify du moniteur de A1L, H à A2L, H vers A4L, H	-458	\$FE36	VFY	(AY=0)

Sous-programmes	Adresse décimale	Adresse hexa- décimale	Nom	Registres détruits
Commande de désassemblage de 20 instructions L	-418	\$FESE	LIST	A, X, Y
Positionnement de INVFLG en mode inverse \$3F	-384	\$FE80	SETINV	Y
Positionnement de INVFLG en mode normal \$FF	-380	\$FE84	SETNORM	Y
Positionnement de INVFLG avec le contenu de Y	-378	\$FE86	SETIFLG	aucun
Commande 0-CTRL-K (retour au clavier)	-375	\$FE89	SETKBD	A, X, Y
Saisie des données à partir du port dont le numéro est contenu dans l'accumulateur - A2L	-373 -371	\$FE8B \$FE8D	INPORT INPRT	A, X, Y A, X, Y
Retour de l'écran pour la sortie de données 0-CTRL-P	-365	\$FE93	SETVID	A, X, Y
Sortie des données sur le port dont le numéro est contenu dans l'accumulateur dans A2L	-363 -361	\$FE95 \$FE97	OUTPORT OUTPRT	A, X, Y A, X, Y
Commande GO : branchement de l'adresse contenue dans A1 PCL/H	-330	\$FEB6	GO	A, X, Y, P
Restauration des registres	-327	\$FEB9		
Branchement à l'adresse contenue dans PCL, H	-324	\$FEBC		
Affichage des registres (CTRL-E)	-321	\$FEBF	REGZ	
Traitement de la touche RETURN : — simulation d'un blanc — branchement au point d'entrée du moniteur MON Z	-266	\$FEF6	CRMON	
Restauration des registres STATUS et empilage :	-193 -190 -188 -186 -184	\$FF3F \$FF42 \$FF44 \$FF46 \$FF48	RESTORE RESTR1	

Sous-programmes	Adresse décimale	Adresse hexa- décimale	Nom	Registres détruits
Sauvegarde des registres : ACC à ACC \$45 X à XREG \$46 Y à YREG \$47 P à STATUS \$48 S à SPNT \$49	-182 -180 -178 -176 -172	\$FF4A \$FF4C \$FF4E \$FF50 \$FF54	SAVE SAVI	
Passage du 6502 en mode hexadécimal				
Entrée du moniteur : (AUTOSTART) sur RESET sur RESET	-167	\$FA62 \$FF59	RESET	
Appels de SETNORM mode normal de l'écran INIT écran de texte SETVID sortie sur l'écran SETKBD entrée à partir du clavier	-164 -161 -158	\$FF5C \$FF5F \$FF62		
Passage du 6502 en mode hexadécimal Emission d'un "beep"	-155 -154	\$FF65 \$FF66	MON	
Entrée du moniteur Positionnement de '*' comme en-tête Lecture d'une ligne Initialisation à vide du mode de traite- ment (sans : + - .)	-151 -149 -147 -144	\$FF69 \$FF6B \$FF6D \$FF70	MONZ	
Traitement d'une commande : — appel de GETNUM pour lire une commande (1 ou 2 octets suivis d'une lettre) — sauvegarde de la position du pro- chain caractère à traiter dans YSAV — appel du sous-programme spécifi- que à la commande désirée (appel de TOSUB) — restauration de la position du pro- chain caractère à traiter dans Y et branchement en NXTITM	-141 -126 -123	\$FF73 \$FF82 \$FF85	NXTITM	

Sous-programmes	Adresse décimale	Adresse hexa- décimale	Nom	Registres détruits
Lecture d'une commande (1 ou 2 oc- tets) que l'on range dans A2L, H et d'un caractère dont on range le code ASCII dans l'accumulateur	-89	\$FFA4	GETNOM	
Appel de la commande désirée : — mettre dans la pile l'adresse de la commande — mise dans l'accumulateur du mode du moniteur — RAZ de MODE — RTS (branchement à la com- mande)	-66	\$FFBE	TOSUB	
Remise à zéro du mode entre deux commandes	-57	\$FFC7	ZMODE	
Exécution de l'instruction dont l'a- dresse est dans (PCL, H) affichage de l'instruction et des registres obtenus : commande Trace étape de Trace	-1469 -318 -316	\$FA43 \$FEC2 \$FEC4	STEP TRACE STEPZ	} ancienne version du moniteur

● Sous-programmes utilisables en langage machine

Nous donnons ci-dessous une liste non limitative de sous-programmes du moniteur utilisables en langage machine.

Fonction	Adresse décimale	Adresse hexa- décimale	Nom	Registres détruits
Affichage d'un CR (retour chariot)	-626	\$FD8E	CROUT	A
Affichage d'un caractère sur l'écran à la position CV, CH	-531	\$FDED	COUT	A
Affichage de trois blancs sur l'écran	-1720	\$F948	PRBLNK	A, X

Fonction	Adresse décimale	Adresse hexa- décimale	Nom	Registres détruits
Affichage du nombre de blancs (contenu dans X)	-1718	\$F94A	PRBL2	A, X
Affichage du caractère dont le code ASCII est dans A et de (X-1) blancs :	-1716	\$F94C	PRBL3	A, X
— émission d'un "beep"	-198	\$FF3A	BELL	A
— affichage du message "ERR" et émission d'un "beep"	-211	\$FF2D	PRERR	A
Affichage de la valeur hexadécimale de la partie basse de l'accumulateur :	-541	\$FDE3	PRHEX	A
— affichage du contenu de l'accumulateur	-550	\$FDDA	PRBYTE	A
— affichage du contenu des registres Y, X	-1728	\$F940	PRNTYX	A
— affichage du contenu des registres A, X	-1727	\$F941	PRNTAX	A
— affichage du contenu du registre X	-1724	\$F944	PRNTX	A
Affichage de CR, du contenu des registres Y, X et d'un tiret	-618	\$FD96	PRYX2	A, Y
Affichage du contenu des registres Y, X et d'un tiret	-615	\$FD33		A, Y
Affichage de CR, de la valeur contenue dans A1H, A1L, et d'un tiret	-622	\$FD93	PRA1	A, X, Y
Affichage des contenus hexadécimaux des adresses (A1L, H) à (A2L, H)	-605	\$FDA3	XAM8	A(Y=0)
	-589	\$FDB3	XAM	A(Y=0)
Sauvegarde des registres A, X, Y, P, S aux adresses \$45-49	-182	\$FF4A	SAVE	A, X
Affichage de CR, des noms des registres et de leurs valeurs (adresses \$45-49X)	-1321	\$FAD7	REGDSP	A, X
Affichage des noms et valeurs des registres	-1318	\$FADA	RGDSP1	A, X
Restauration des registres A, X, Y, P (pas S) à partir des adresses \$45-\$49			RESTORE	A, X, Y, P
Restauration des registres (sauf S) et branchement à l'adresse contenue dans A1L, H	-330	\$FEB6	GO	A, X, Y, P

Fonction	Adresse décimale	Adresse hexa- décimale	Nom	Registres détruits
Déplacement de la zone mémoire comprise entre (A1L, H) et (A2L, H) vers la zone commençant à (A4L, H) (contenu de A4)	-468	\$FE2C	MOVE	A(Y=0)
Vérification de ce que la zone mémoire débutée à l'adresse (A4L, H) est identique à la zone mémoire comprise entre (A1L, H) et (A2L, H). Les différences sont affichées sur l'écran	-458	\$FE36	VFY	A(Y=0)
Incrémentation de (A4L, H)	-844	\$FCB4	NXTA4	A
Incrémentation de A1L, H et si A2L, H est inférieur à A1L, H, positionnement du carry	-838	\$FCBA	NXTA1	A
Positionnement de GBASL, H pour la ligne dont le numéro est dans l'accumulateur	-1977	\$F847	GBASCALC	A
Si le carry n est positionné RAZ partie haute de l'accumulateur	-1927	\$F879	SCRN2	A
Sinon on décale l'accumulateur de 4 positions vers la droite (conservation partie haute de l'accumulateur)				
Désassemblage de l'instruction située à l'adresse contenue dans PCL, H et affichage sur l'écran	-1840	\$F8D0	INSTDSP	A, X, Y
Passage à l'instruction suivante à désassembler, ajout du contenu de LENGTH à celui de PCL, H, les résultats sont dans les registres A et Y (le 6502 doit être en mode hexadécimal)	-1709	\$F953	PCADJ	A, X, Y
Lecture des manettes de jeux	-1250	\$FB1E	PREAD	A, Y
Attente de 0,1 s et émission de "beep"	-1059	\$FBDD		A, Y
Chargement de Y=192 pour 0,1 s de beep	-1054	\$FBE2		A, Y
Emission de "beep" à 1 kHz durant le nombre de cycles indiqués dans Y	-1052	\$FBE4	BELL2	A, Y

Fonction	Adresse décimale	Adresse hexa- décimale	Nom	Registres détruits
Positionnement du caractère dans la mémoire d'écran si ce n'est pas un caractère de contrôle. Sinon traitement du caractère de contrôle	-1027	\$FBFD	VIDOUT	A, Y
Effacement de l'écran et positionnement du curseur en haut et à gauche de l'écran	-936	\$FC58	HOME	A, Y
Remise à zéro de Y et affichage d'un tiret	-612	\$FD9C		
Affichage d'un tiret sur l'écran	-610	\$FD94		
Affichage d'un caractère sur l'écran avec les actions appropriées pour les caractères de contrôle. Si (A) < \$A0 branchement en COUTZ	-528	\$FDF0	COUT1	A

ANNEXE 1 Codes ASCII

ASCII Code	Character	ASCII Code	Character	ASCII Code	Character
000	NUL	036	\$	072	H
001	SOH	037	%	073	I
002	STX	038	&	074	J
003	ETX	039	'	075	K
004	EOT	040	(076	L
005	ENQ	041)	077	M
006	ACK	042	*	078	N
007	BEL	043	+	079	O
008	BS	044	,	080	P
009	HT	045	-	081	Q
010	LF	046	.	082	R
011	VT	047	/	083	S
012	FF	048	0	084	T
013	CR	049	1	085	U
014	SO	050	2	086	V
015	SI	051	3	087	W
016	DLE	052	4	088	X
017	DC1	053	5	089	Y
018	DC2	054	6	090	Z
019	DC3	055	7	091	[
020	DC4	056	8	092	\
021	NAK	057	9	093]
022	SYN	058	:	094	^
023	ETB	059	;	095	<
024	CAN	060	<	096	'
025	EM	061	=	097	a
026	SUB	062	>	098	b
027	ESCAPE	063	?	099	c
028	FS	064	@	100	d
029	GS	065	A	101	e
030	RS	066	B	102	f
031	US	067	C	103	g
032	SPACE	068	D	104	h
033	!	069	E	105	i
034	"	070	F	106	j
035	#	071	G	107	k

108	l	115	s	122	z
109	m	116	t	123	
110	n	117	u	124	
111	o	118	v	125	
112	p	119	w	126	~
113	q	120	x	127	DEL
114	r	121	y		

ASCII codes are in decimal.

LF=Line Feed, FF=Form Feed, CR=Carriage Return, DEL=Rubout

ANNEXE 2

Codes ASCII saisissables au clavier

Key	Alone	CTRL	SHIFT	Both	Key	Alone	CTRL	SHIFT	Both
space	\$A0	\$A0	\$A0	\$A0	RETURN	\$8D	\$8D	\$8D	\$8D
0	\$B0	\$B0	\$B0	\$B0	G	\$C7	\$87	\$C7	\$87
1!	\$B1	\$B1	\$A1	\$A1	H	\$C8	\$88	\$C8	\$88
2"	\$B2	\$B2	\$A2	\$A2	I	\$C9	\$89	\$C9	\$89
3#	\$B3	\$B3	\$A3	\$A3	J	\$CA	\$8A	\$CA	\$8A
4\$	\$B4	\$B4	\$A4	\$A4	K	\$CB	\$8B	\$CB	\$8B
5%	\$B5	\$B5	\$A5	\$A5	L	\$CC	\$8C	\$CC	\$8C
6&	\$B6	\$B6	\$A6	\$A6	M	\$CD	\$8D	\$DD	\$9D
7'	\$B7	\$B7	\$A7	\$A7	N	\$CE	\$8E	\$DE	\$9E
8(\$B8	\$B8	\$A8	\$A8	O	\$CF	\$8F	\$CF	\$8F
9)	\$B9	\$B9	\$A9	\$A9	P@	\$D0	\$90	\$C0	\$80
:*	\$BA	\$BA	\$AA	\$AA	Q	\$D1	\$91	\$D1	\$91
;+	\$BB	\$BB	\$AB	\$AB	R	\$D2	\$92	\$D2	\$92
,<	\$AC	\$AC	\$BC	\$BC	S	\$D3	\$93	\$D3	\$93
-=	\$AD	\$AD	\$BD	\$BD	T	\$D4	\$94	\$D4	\$94
.>	\$AE	\$AE	\$BE	\$BE	U	\$D5	\$95	\$D5	\$95
/?	\$AF	\$AF	\$BF	\$BF	V	\$D6	\$96	\$D6	\$96
A	\$C1	\$81	\$C1	\$81	W	\$D7	\$97	\$D7	\$97
B	\$C2	\$82	\$C2	\$82	X	\$D8	\$98	\$D8	\$98
C	\$C3	\$83	\$C3	\$83	Y	\$D9	\$99	\$D9	\$99
D	\$C4	\$84	\$C4	\$84	Z	\$DA	\$9A	\$DA	\$9A
E	\$C5	\$85	\$C5	\$85	-	\$88	\$88	\$88	\$88
F	\$C6	\$86	\$C6	\$86	_	\$95	\$95	\$95	\$95
					ESC	\$9B	\$9B	\$9B	\$9B

ANNEXE 3

Codes ASCII affichables à l'écran

Decimal	Inverse															
	0	16	32	48	64	80	96	112	Flushing							
Hex	500	510	520	530	540	550	560	570								
0 50	@	P	!	0	@	P	!	0	(Control)							
1 51	A	Q	"	1	A	Q	"	1	Normal							
2 52	B	R	#	2	B	R	#	2	(Lowercase)							
3 53	C	S	\$	3	C	S	\$	3								
4 54	D	T	%	4	D	T	%	4								
5 55	E	U	&	5	E	U	&	5								
6 56	F	V	'	6	F	V	'	6								
7 57	G	W	(7	G	W	(7								
8 58	H	X)	8	H	X)	8								
9 59	I	Y	*	9	I	Y	*	9								
10 5A	J	Z	+	10	J	Z	+	10								
11 5B	K	[,	11	K	[,	11								
12 5C	L	\	-	12	L	\	-	12								
13 5D	M]	.	13	M]	.	13								
14 5E	N	^	/	14	N	^	/	14								
15 5F	O	_		15	O	_		15								

ANNEXE 4

Tableau des sous-programmes assembleur utilisables classés par adresse croissante : (Moniteur)

Adresse décimale	Adresse hexa-décimale	Nom	Fonction	Instruction APPLESOFT	Page
-2048	\$F800	PLOT	affichage d'un point basse résolution	PLOT	
-2023	\$F819	HLIN	tracé d'une ligne horizontale	HLIN AT	
-2008	\$F828	VLIN	tracé d'une ligne verticale	VLIN AT	
-1998	\$F832	CLRSCR	effacement de tout l'écran		
-1994	\$F836	CLRTOP	effacement des lignes graphiques basse résolution (40 l) sans toucher au texte	GR (partie)	
-1953	\$F85F	COLOR	positionnement de la couleur	COLOR	
-1948	\$F864	SCRN	obtention de la couleur	SCRN	
-1728	\$F940	PRNTYX	affichage du contenu des registres Y et X selon le format YYXX		
-1727	\$F941	PRNTYA	idem selon le format AAXX		
-1724	\$F944	PRNTX	affichage du contenu du registre X		
-1720	\$F948	PRBLNK	affichage de trois blancs		
-1718	\$F94A	PRBL2	affichage du nombre de blancs précisé par le registre X	SPC	
-1716	\$F94C	PRBL3	affichage du caractère dont le code ASCII est dans l'accumulateur et de (X) - 1 blancs		
-1250	\$FB1E	PREAD	lecture de la manette dont le numéro est dans X	PDL	
-1223	\$FB39	SETTXT	passage en mode texte	TEXT	
-1216	\$FB40	SETGR	passage en mode graphique basse résolution	GR	
-1063	\$FBD9	BELL1	attente d'un délai d'un centième de seconde et émission d'un "beep", si le contenu de l'accumulateur vaut \$87		

ANNEXE 4 (suite)

Adresse décimale	Adresse hexo- décimale	Nom	Fonction	Instruction APPLESOFT	Page
-1052	\$FBE4	BELL2	émission à la fréquence 1 kHz du nombre de "beep" égal au contenu du registre Y		
-1036	\$FBF4	ADVANCE	déplacement du curseur d'une position vers la droite	ESC-A	
-1008	\$FC10	BS	déplacement du curseur d'une position vers la gauche	ESC-B	
-998	\$FC1A	UP	retour à la ligne précédente	ESC-D	
-958	\$FC42	CLREOP	effacement de l'écran entre la position actuelle du curseur et le bas de l'écran	ESC-F	
-936	\$FC58	HOME	effacement de l'écran et posi- tionnement du curseur dans le coin supérieur gauche	HOME ESC-@	
-926	\$FC62	CR	retour au début de la ligne sui- vante		
-922	\$FC66	LF	passage à la ligne suivante sans changer de colonne	ESC-C	
-912	\$FC70	SCROLL	déplacement des lignes de la fe- nêtre d'une position vers le haut		
-868	\$FC9C	CLREOL	effacement de l'écran entre le curseur et la fin de la ligne	ESC-E	
-756	\$FDOC	RDKEY	remplacement du caractère sous le curseur par le prochain ca- ractère saisi		
-741	\$FD1B	KEYIN	lecture d'un caractère et affi- chage		
-715	\$FD35	RDCHAR	saisie d'un caractère avec trai- tement des fonctions ESC		
-665	\$FD67	GETLNZ	saisie d'une ligne	INPUT	
-550	\$FDDA	PRBYTE	affichage du contenu de l'accu- mulateur sous le format AA		
-531	\$FDED	COUT	affichage d'un caractère sur le périphérique actif ou sur l'écran	PRINT	

ANNEXE 4 (suite)

Adresse décimale	Adresse hexo- décimale	Nom	Fonction	Instruction APPLESOFT	Page
-	\$FDFO	COUT1	affichage d'un caractère sur l'écran	PR #0:PRINT	
-384	\$FE80	SETINV	écran en mode inverse	INVERSE	
-380	\$FE84	SETNORM	écran en mode normal	NORMAL	
-336	\$FEBO		retour au Basic avec écrasement du programme	CTRL-B	
-211	\$FF2D	PRERR	affichage de 'ERR' et émission d'un "beep"		116
-198	\$FF3A	BEEP	émission d'un "beep" sur la périphérie de sortie actif		
-193	\$FF3F	RESTORE	restauration des registres à par- tir des adresses \$45-\$49		132
-182	\$FF4A	SAVE	sauvegarde des registres aux adresses \$45-\$49		
-151	\$FF69	MONZ	entrée dans le moniteur		132
-141	\$FF73	NXTITM	traitement d'une commande		132

ANNEXE 5

Tableau des adresses utiles dans la zone des E/S

Adresse décimale	Adresse hexadécimale	Signification
-16384	\$C000	entrée clavier
-16368	\$C010	validation clavier
-16352	\$C020	sortie cassette
-16336	\$C030	haut-parleur
-16304	\$C050	mode graphique
-16303	\$C051	mode texte
-16302	\$C052	graphisme seul
-16301	\$C053	texte et graphisme
-16300	\$C054	page 1
-16299	\$C055	page 2
-16298	\$C056	basse résolution
-16297	\$C057	haute résolution
-16296	\$C058	mise à 0 de la sortie numérique 0 (TTL)
-16295	\$C059	1
-16294	\$C05A	mise à 0 de la sortie numérique 1 (TTL)
-16293	\$C05B	1
-16292	\$C05C	mise à 0 de la sortie numérique 2 (TTL)
-16291	\$C05D	1
-16290	\$C05E	mise à 0 de la sortie numérique 3 (TTL)
-16289	\$C05F	1
-16288	\$C060	entrée cassette
-16287	\$C061	} entrées numériques-bouton poussoir manette {
-16286	\$C062	
-16285	\$C063	
-16284	\$C064	} entrées analogiques-manettes de jeux {
-16283	\$C065	
-16282	\$C066	
-16281	\$C067	
-16272	\$C070	

INDEX ALPHABÉTIQUE

A

accès à la mémoire (Basic), 15.
 accès à la mémoire (moniteur), 95.
 accès au miniassembleur, 110.
 accès au moniteur, 93.
 accès aux registres du 6502, 107.
 accumulateur, 77, 107.
 adressage en mémoire, 79.
 affichage de points sur l'écran (mode basse résolution), 48, 143.
 affichage de points sur l'écran (mode haute résolution), 53.
 alimentation, 3.
 AND, 13, 88.
 arrêt d'un programme (Basic), 15.
 assembleur, 16, 73, 87, 92, 107, 113.
 assemblage d'une table de figures graphiques, 62.
 attente d'un délai, 147.
 autotest, 4.

B

branchement (Basic), 14.
 branchement (assembleur), 86.

C

caractères alphanumériques, 11.
 carte d'extension, 6, 9, 134.
 carte mémoire, 26, 75, 120.
 carte Péritel, 8.
 carte R.V.B., 8.
 cassette, 3, 67, 102, 145.
 chaîne de caractères, 11, 12, 24.
 clavier, 1, 3, 130, 135.
 codage d'une table de figures graphiques, 60.
 COLOR, 47.
 comparaison de zones mémoire (moniteur), 100.
 connecteur, 3, 77.
 contrôle (instructions de) (Basic), 14.
 contrôle (instructions de) (assembleur), 87.

couleur (mode basse résolution), 46.
 couleur (mode haute résolution), 51.
 courbe (tracé de), 55.
 création de votre commande utilisateur (moniteur), 119.
 création d'un programme assembleur (moniteur), 108.
 création d'une table de figures graphiques, 63.

D

définition d'une table de figures graphiques, 60.
 disquette, 67, 104, 105, 106.
 DRAW, 68.

E

édition de texte sur l'écran, 16, 139.
 entier numérique, 11, 30.
 entrées/sorties, 4, 6, 18, 21, 92, 118, 129, 135.
 examen de la mémoire (Basic), 15.
 examen de la mémoire (moniteur), 95.
 exécution d'un programme (Basic), 15.
 exécution d'un programme (assembleur), 108.

F

figures graphiques haute-résolution, 60.
 définition, 60.
 codage, 60.
 assemblage d'une table, 62.
 création d'une table, 63.
 sauvegarde sur cassette, 67.
 sauvegarde sur disquette, 67.
 utilisation, 67.
 DRAW, 68.
 XDRAW, 69.
 SCALE =, 69.
 ROT =, 70.
 fonctions (Basic), 24.
 fréquence d'un son, 3, 73.
 GOTO, 14.
 GOSUB, 14.

G

graphisme basse résolution, 8, 45, 127, 132, 140, 143.
 passage en mode graphique basse résolution, 46.
 couleurs basse résolution, 47, 50.
 affichage de points, 48, 50.
 tracé de lignes, 48.
 gestion du mode par le moniteur, 143.
 graphisme haute résolution, 8, 36, 45, 52, 126, 132, 140.
 passage en mode graphique haute résolution, 52.
 couleurs haute résolution, 52.
 affichage de points, 53.
 tracé de lignes, 53.
 tracé de courbe, 55.
 figures graphiques haute résolution, (voir lettre F).
 GR, 46.

H

haut-parleur, 3, 131, 145.
 HGR, 52.
 HIMEM, 15.

I

imprimante, 34, 118.
 INPUT, 18.
 initialisation de la mémoire (Basic), 15.
 initialisation de la mémoire (moniteur), 101.
 instructions de la mémoire (Basic), 14.
 instructions de la mémoire (assembleur), 81.
 instructions de la mémoire (miniassembleur), 111.
 interface parallèle, 7, 34.
 interface série, 8, 34.
 interruption, 148.
 interruption IRQ/BRK, 151.
 interruption NMI, 149.
 interruption Reset, 150.

J/K

jeux, 74, 131, 147.

L

langage, 5, 6, 11, 108.
 LOMEM, 15.

M

manettes de jeux mémoire, 3, 50, 74, 131, 147.
 carte mémoire, 25, 75, 120.
 mémoire morte, 4, 75, 107.
 mémoire vive, 5, 77.
 examen de la mémoire, 15, 95.
 modification de la mémoire, 15, 97.
 initialisation de la mémoire, 101.
 recopie d'une zone mémoire, 99.
 comparaison de 2 zones mémoire, 100.
 sauvegarde sur cassette, 102.
 sauvegarde sur disquette, 104.
 vérification d'une zone mémoire, 105.
 pages mémoires, 26 à 29, 120 à 125.
 microprocesseur 6502, 3, 75, 81, 107.
 miniassembleur, 76, 110 à 114.
 accès au miniassembleur, 110.
 sortie du miniassembleur, 111.
 instructions du miniassembleur, 111.
 accès aux commandes du moniteur, 111.
 mise au point des programmes assembleur, 113.
 moniteur, 75, 93 à 158.
 commandes du moniteur, 93.
 accès au moniteur, 93.
 sortie du moniteur, 94.
 accès à la mémoire (voir mémoire), 95.
 création d'une commande utilisateur, 119.
 sous-programmes d'entrée-sortie, 137.
 utilisation des commandes comme sous-programmes, 151.
 modem, 8.
 modes d'adressage, 79.

N

NOT
 négation logique (NOT), 13, 88.

O

opérateurs logiques, 13, 88.
 opérateurs numériques, 12, 90.
 opérateurs relationnels, 13, 87.
 OR, 13, 88.
 ou logique (OR), 13, 88.

P

pile, 77, 79, 91.
 pointeur de pile, 77, 79.

pointeur de programmes, 78.
 PRINT, 18.

R

RAM (voir mémoire vive)
 registre d'état du 6502, 78.
 registres d'index du 6502, 77, 79.
 ROM (voir mémoire morte), 70.

S

SCALE = , 69.
 son, 3, 72.
 sortie vidéo, 1, 3, 8.
 sous-programmes, 14, 37, 91, 151.
 système d'exploitation, 5, 6.

T

tableaux, 12, 30.
 texte (mode de l'écran), 8, 35, 127, 132, 140.
 TEXT, 45.
 tracé de courbe, 55.
 tracé de lignes, 48, 53.
 traitement des erreurs en Basic, 22.

U

unité arithmétique et logique (UAL), 76.
 unité centrale, 4.

V

variable entière, 11, 29.
 variable réelle, 11, 29.

X

XDRAW, 69.

EDIMICRO
10, rue Henri-Pape 75013 PARIS

Imprimé en France. — JOUVE, 18, rue Saint-Denis, 75001 PARIS
Dépôt légal 1^{re} Edition : avril 1983
2^e Edition. Dépôt légal : juin 1983
N° 12312. Dépôt légal réimpression : janvier 1984