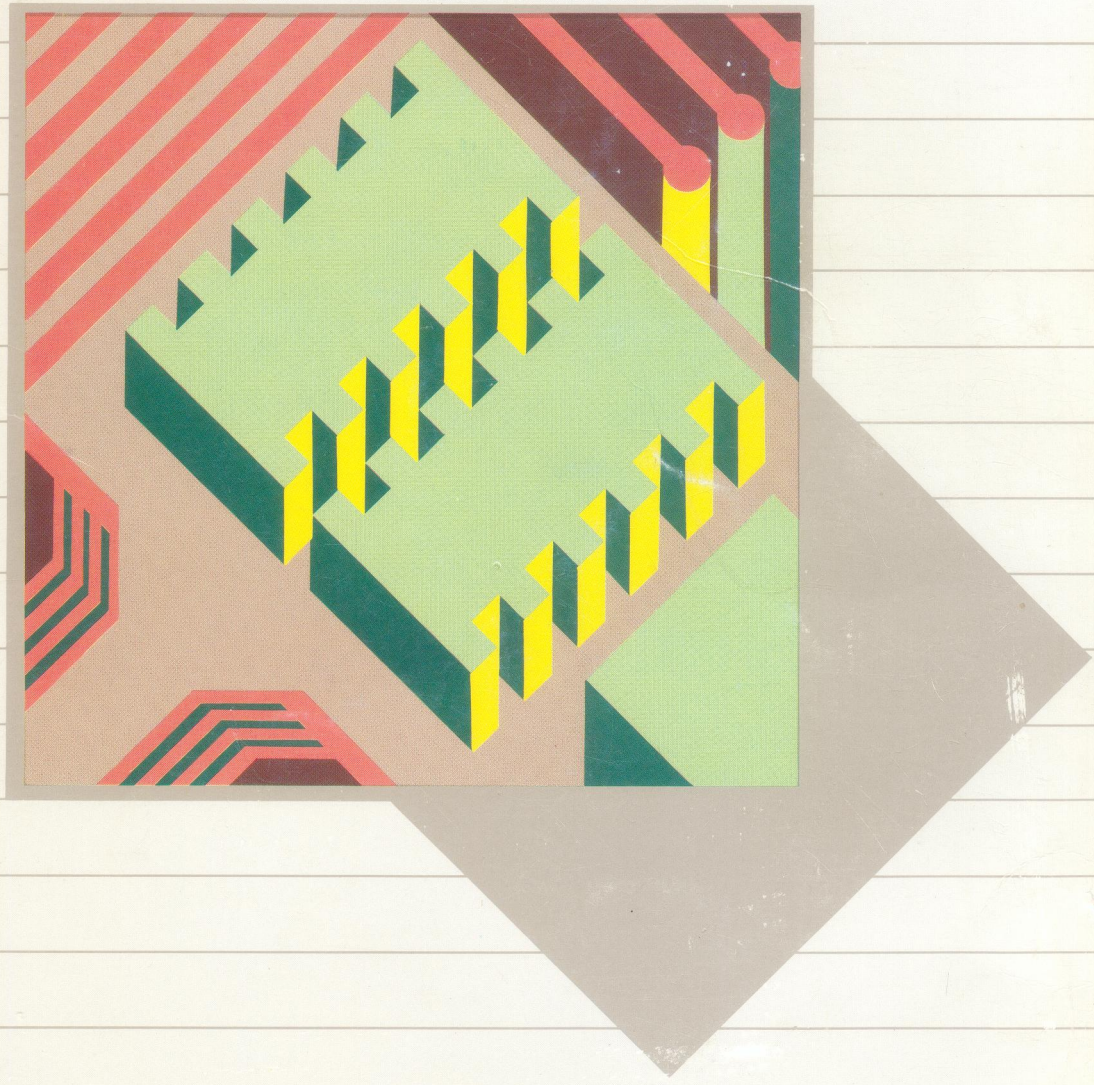
Apple //e

Manuel de Référence

Pour //e seulement



Avertissement

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

Dégagement de toutes garanties et responsabilités

Apple Computer, Inc. n'apporte aucune garantie, explicite ou implicite, en ce qui concerne le présent manuel ou le logiciel décrit dans ce manuel, sa qualité, ses performances, sa négociabilité, ou son aptitude à un usage particulier. Le logiciel d'Apple Computer, Inc. est vendu ou cédé en licence « en l'état ». L'intégralité du risque, quant à sa qualité et sa performance, est à la charge de l'acheteur. Si les programmes se révélaient défectueux à la suite de leur achat, l'acheteur (et non pas Apple Computer, Inc., son distributeur ou son détaillant) assumerait la totalité des frais des services nécessaires, réparation ou correction de tout dommage incident ou conséquent. En aucun cas Apple Computer, Inc. ne sera responsable de dommages directs, indirects, accidentels ou par voie de conséquence résultant d'un défaut quelconque du logiciel, même si Apple Computer Inc. a été avisé de la possibilité de tels dommages. Certains pays n'acceptent pas l'exclusion ou la limitation des garanties implicites ou des responsabilités pour des dommages accidentels ou par voie de conséquence, de telle sorte que la limitation ou exclusion ci-dessus ne pourront vous être appliquées.

20525 Mariani Avenue
Cupertino, California 95014
(408) 996-1010

Avertissement :

Cet équipement est certifié conforme aux restrictions des machines de calcul de la classe B, partie de Subpart J de la Part 15 des règles de la FCC. Les seuls périphériques (dispositifs d'entrée/sortie, ordinateurs, terminaux, imprimantes, etc.) certifiés conformes aux restrictions de la classe B peuvent être raccordés à cet ordinateur. Le fonctionnement en conjonction avec des périphériques non agréés peut être générateur d'interférences radio et télévision.

Rédigé par Joe Meyers, du Service Publications Apple PCS.

Numéro de produit Apple : A2L 2005F

Apple //e

Manuel de Référence

Pour //e seulement

Traduction et adaptation
Nicole BREAUD-POULIQUEN

Interférences dans la réception de la radio et de la télévision

L'équipement décrit dans ce manuel produit des fréquences radio. Des interférences de réception peuvent se produire, si l'équipement n'est pas installé et utilisé dans le strict respect de nos instructions.

Cet équipement a été testé et se conforme aux limitations de la Classe B de l'appareillage informatique et aux spécifications incluses dans le paragraphe J de l'article 15 des règlements du Bureau de Communications des États-Unis (Federal Communications Commission). Ces règlements sont conçus pour fournir une protection judicieuse contre des interférences dans une installation résidentielle. Cependant, il n'y a pas de garantie que des interférences n'aient pas lieu dans une installation particulière, spécialement si vous utilisez une antenne de télévision de type « rabbit ear ». (Une antenne « rabbit ear » est le modèle télescopique couramment utilisé des récepteurs T.V.).

Vous pouvez déterminer si votre ordinateur cause des interférences en éteignant l'appareil. Si les interférences s'arrêtent, votre ordinateur ou ses périphériques en sont probablement la cause. Si votre ordinateur provoque des interférences à la réception radio et télévision, vous pouvez essayer de les corriger grâce à une ou plusieurs des mesures suivantes :

- Tournez l'antenne de la radio ou de la T.V. jusqu'à l'arrêt de l'interférence.
- Placez l'ordinateur d'un côté ou de l'autre du téléviseur ou du poste de radio.
- Éloignez l'ordinateur le plus possible du téléviseur ou du poste de radio.
- Branchez l'ordinateur à une prise de courant située dans un circuit différent de celui de la radio ou de la T.V. (En fait, s'assurer que l'ordinateur, la télévision et la radio sont branchés sur des circuits contrôlés par des disjoncteurs ou des fusibles distincts.)
- Débranchez les périphériques et leurs câbles d'entrée/sortie un par un. Si les interférences cessent, c'est qu'elles étaient causées par le périphérique ou son câble. Ces dispositifs nécessitent normalement l'utilisation de câbles blindés. Votre fournisseur peut vous commander ces câbles pour les périphériques Apple. Pour les matériels d'autres fabricants, les contacter directement.
- Envisagez d'installer une antenne de toiture avec une rallonge en câble coaxial entre l'antenne et la T.V.

Si nécessaire, vous devriez consulter votre distributeur ou un technicien expérimenté en radio et télévision pour les suggestions supplémentaires.

Table des matières

Utilisation du moniteur 89

Liste des figures

Liste des Tableaux

Avant-propos

Voici le manuel de référence de l'ordinateur personnel Apple //e. Il contient des descriptions détaillées de tout le hardware et le logiciel de base dont est constitué l'Apple //e et fournit l'information technique dont les concepteurs de cartes d'interface et les programmeurs ont besoin. Il y a Addendum, relié séparément, qui contient les listings-sources des logiciels de base implémentés dans le système.

Ce manuel contient de nombreuses informations qui expliquent comment fonctionne l'Apple //e, mais il ne vous indique pas comment utiliser l'Apple //e. Pour savoir l'utiliser, vous devriez lire les autres manuels Apple //e, en particulier ceux-ci :

- *Le manuel de l'utilisateur de l'Apple //e*
- *Les travaux pratiques Applesoft*

Ce manuel est conçu pour répondre à la question, qu'est-ce qu'il y a dans la boîte ? Il décrit le fonctionnement interne de l'Apple //e aussi complètement que possible dans un seul volume.

Contenu de ce manuel

Les éléments de ce manuel sont présentés en suivant à peu près un ordre d'approfondissement croissant du matériel ; plus vous allez loin dans le manuel, plus la matière devient technique. Les domaines principaux traités sont :

- Introduction : avant-propos et chapitre I
- Utilisation des fonctions de base : chapitres 2 et 3
- Comment est organisée la mémoire : chapitre 4
- Informations pour les programmeurs : chapitre 5 et 6

- Implantation du hardware : Chapitre 7
- Informations supplémentaires : Annexes et Addendum

Le chapitre 1 permet d'identifier les éléments principaux de l'Apple //e et indique où chaque élément est décrit dans ce manuel.

Les deux chapitres suivants décrivent les caractéristiques des entrées et des sorties de base de l'Apple //e. Cette partie du manuel inclut les informations nécessaires à une programmation de l'Apple //e dans un langage moins évolué.
Le chapitre 2 décrit les caractéristiques des E/S de base du système et le chapitre 3 vous indique comment utiliser les sous-programmes qui les assurent.

Le chapitre 4 décrit de quelle manière l'espace mémoire est organisé, y compris l'allocation des mémoires programmables réservées aux zones tampons de l'affichage vidéo.

Le chapitre 5 est un manuel d'utilisation du programme Moniteur qui fait partie du logiciel-système. Le moniteur est un programme du système que vous pouvez utiliser pour de la mise au point de programme en langage Machine.

Le chapitre 6 décrit les caractéristiques des connecteurs périphériques et guide à leur utilisation.

Le chapitre 7 est une description détaillée du hardware qui réalise les fonctions décrites dans les chapitres précédents. Cette documentation s'adresse en priorité aux programmeurs et aux concepteurs de cadres périphériques, mais elle vous aidera si vous voulez seulement mieux comprendre comment l'Apple //e marche.

Des informations de référence supplémentaires sont données en annexe. L'annexe A est la description de l'ensemble des instructions du 6502, fournie par le fabricant.

L'annexe B contient des copies supplémentaires de quelques unes des tables qui sont dans le corps du manuel. Celles auxquelles vous aurez souvent besoin de vous référer sont recopiées ici pour en faciliter l'accès.

L'annexe C est un répertoire des sous-programmes d'E/S de base du Système, avec leur fonction et leur adresse de départ respectives.

L'annexe D énumère les différences entre l'Apple //e et les précédents modèles Apple II et Apple II Plus et vous indique à quels endroits aller chercher plus d'information dans le manuel.

A la suite de l'Annexe D, se trouve un glossaire définissant de nombreux termes techniques utilisés dans ce manuel. Quelques termes décrivant l'utilisation de l'Apple //e sont définis dans les glossaires des autres manuels cités plus haut.

Finalement, il y a une bibliographie sélective de sources d'informations supplémentaires.

L'Addendum à ce manuel contient un listing source du Moniteur. Vous pouvez vous y reporter pour mieux explorer le fonctionnement des sous-programmes du Moniteur dont la liste est à l'annexe C.

Symboles utilisés dans ce manuel

Des textes spéciaux seront imprimés de différentes manières, comme le montrent ces exemples :

Des informations qui apparaissent sur l'écran sont imprimées sur un fond gris et signalées par un rectangle noir sur le côté gauche du texte.

Avertissement
Les avertissements importants sont encadrés de cette façon

Les légendes, les définitions et les courtes notes apparaissent dans la marge comme ici

Une information qui est utile mais qui est donnée incidemment dans le texte apparaît dans un cadre gris comme celui-ci. Vous pourriez sauter ces encadrements et y revenir plus tard.

Introduction

Introduction

Ce premier chapitre vous présente l'Apple //e lui-même. Il vous montre à quoi ressemble l'intérieur, donne un nom aux composants les plus importants dont la machine est constituée, et vous indique où ils se trouvent. Ces parties principales sont illustrées dans le diagramme de la figure 1-1.

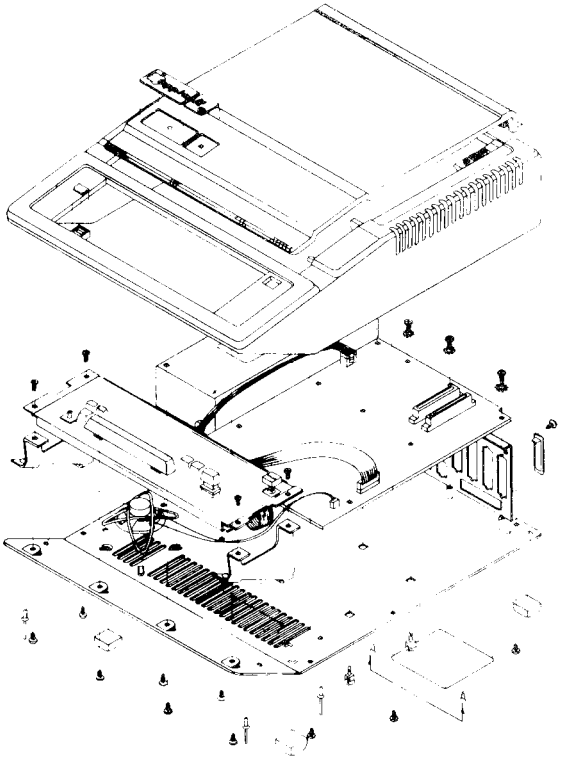


Figure 1-1. Diagramme explosé de l'Apple //e

Enlever le couvercle

Enlevez le couvercle de l'Apple //e en tirant vers le haut par les pattes qui font saillie à l'arrière du couvercle, jusqu'à ce que les fermetures à pression sautent. Puis glissez le couvercle en arrière en l'éloignant du clavier afin de pouvoir le lever et le détacher, comme l'indique la figure 1-2. Ce que vous allez voir est sur la figure 1-3.

Figure 1-2. Enlever le couvercle

Figure 1-3. L'Apple //e sans son couvercle

Avertissement

Il y a une diode émettrice de lumière rouge à l'intérieur de l'Apple //e, dans le coin gauche arrière de la carte-mère. **Si la diode est allumée, cela signifie que le système est sous tension et vous devez le mettre hors-tension avant d'insérer ou d'enlever quelque chose.** Pour éviter d'endommager l'Apple //e, ne **PENSEZ** même pas à modifier quoique ce soit à l'intérieur sans d'abord couper le courant.

Le clavier

Le clavier est le dispositif d'entrée primaire de l'Apple //e. Comme le montre la figure 1-4, il a une disposition de machine à écrire en standard ISO, avec des majuscules et des minuscules, et tous les caractères spéciaux de l'ensemble des caractères ASCII. (ASCII est l'abréviation de American Standard Code for Information Interchange, code américain standard pour l'échange d'informations). Le clavier est complètement intégré dans la machine ; son fonctionnement est décrit dans la première partie du chapitre 2. Les sous-programmes de base implantés en MEM pour lire le clavier sont décrits au chapitre 3.

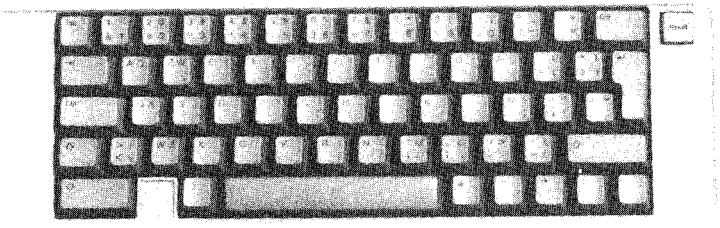


Figure 1-4. Le clavier de l'Apple //e

Le haut-parleur

L'Apple //e a un petit haut-parleur dans le coin avant gauche comme le montre la figure1-1. Le haut-parleur permet à des programmes de produire une variété de sons qui les rendent plus attrayants et intéressants. La façon dont les programmes contrôlent le haut-parleur est expliquée au chapitre 2.

Le boîtier d'alimentation

L'alimentation électrique est à l'intérieur d'un boîtier métallique situé sur le côté gauche à l'intérieur de l'Apple //e. Elle fournit du courant pour la carte-mère et pour toutes les cartes d'interfaces de périphériques installées dans l'Apple //e.

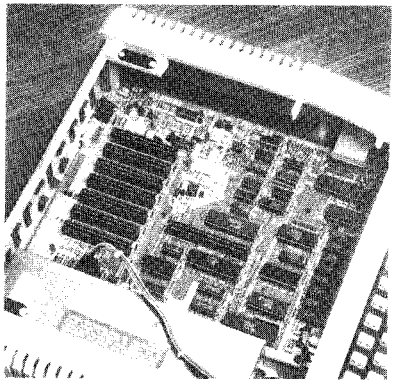
Le boîtier d'alimentation produit quatre tensions électriques continues : +5V, −5V, +12V et −12V. C'est une alimentation à commutation de haute performance qui contient des circuits spéciaux qui la protège, ainsi que le reste de l'Apple //e, contre les courts circuits et autres aléas. Les spécifications complètes du boîtier d'alimentation sont données au chapitre 7.

L'interrupteur d'alimentation et le support du cordon d'alimentation sont montés directement sur l'arrière du boîtier métallique de l'alimentation. Ce montage assure que tous les circuits qui transportent des voltages dangereux restent à l'intérieur du boîtier. N'essayez pas d'ouvrir le boîtier car vous risquez d'éliminer cette caractéristique de conception.

La carte-mère

Tous les composants électroniques de l'Apple *//e* sont connectés sur la carte-mère, qui est montée à plat dans le fond de l'appareil.

Figure 1-5. La carte-mère



La figure 1-5 montre les principaux circuits intégrés (CI) de l'Apple *//e*. Ce sont l'unité centrale (UC), le codeur et la mémoire morte (MEM) du clavier, les deux mémoires mortes de l'interpreteur, et les circuits conçus spécialement : l'unité d'entrée-sortie (IOU) et l'unité de gestion de mémoire (MMU).

L'UC est un microprocesseur 6502A. Le 6502A est une version très rapide du 6502, qui est un microprocesseur à 8 bits avec un bus d'adresses à 16 bits. Il utilise l'interpreteur des instructions qui rend l'exécution plus rapide qu'avec des microprocesseurs comparables. Dans l'Apple *//e*, le 6502A fonctionne à 1MHz et exécute jusqu'à 500 000 opérations à 8 bits par seconde. Les spécifications du 6502A sont données au chapitre 7 ; l'ensemble des instructions du 6502 est donné dans l'annexe A.

Le clavier est décodé par un circuit intégré de type AY-3600 et une mémoire morte (MEM). Ces dispositifs sont décrits au chapitre 7.

Les MEM de l'interpreteur sont des circuits intégrés qui contiennent l'interpreteur BASIC Applesoft. Les MEM sont décrites dans le chapitre 7. Le langage Applesoft est décrit dans les *Travaux pratiques d'Applesoft* et le *Manuel de référence de l'Applesoft.*

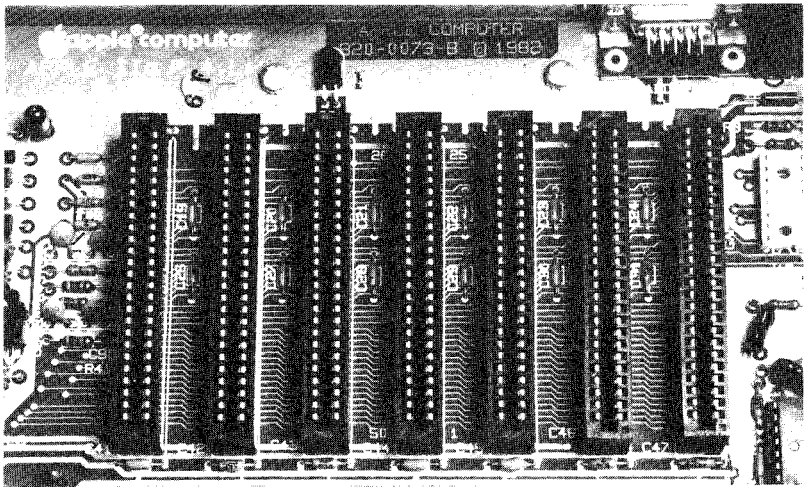
Deux des grands CI sont faits sur-mesure pour l'Apple *//e* : la MMU et l'IOU. Le CI de la MMU contient l'essentiel de la logique de contrôle de l'adressage de la mémoire de l'Apple *//e*. L'organisation de la mémoire est décrite au chapitre 4 : les circuits de la MMU elle-même sont décrits dans le chapitre 7.

Le CI de l'IOU contient l'essentiel de la logique qui contrôle les caractéristiques des entrées/sorties de base du système. Ces caractéristiques sont décrites au chapitre 2 et au chapitre 3 ; les circuits de l'IOU sont décrits dans le chapitre 7.

Les connecteurs sur la carte-mère

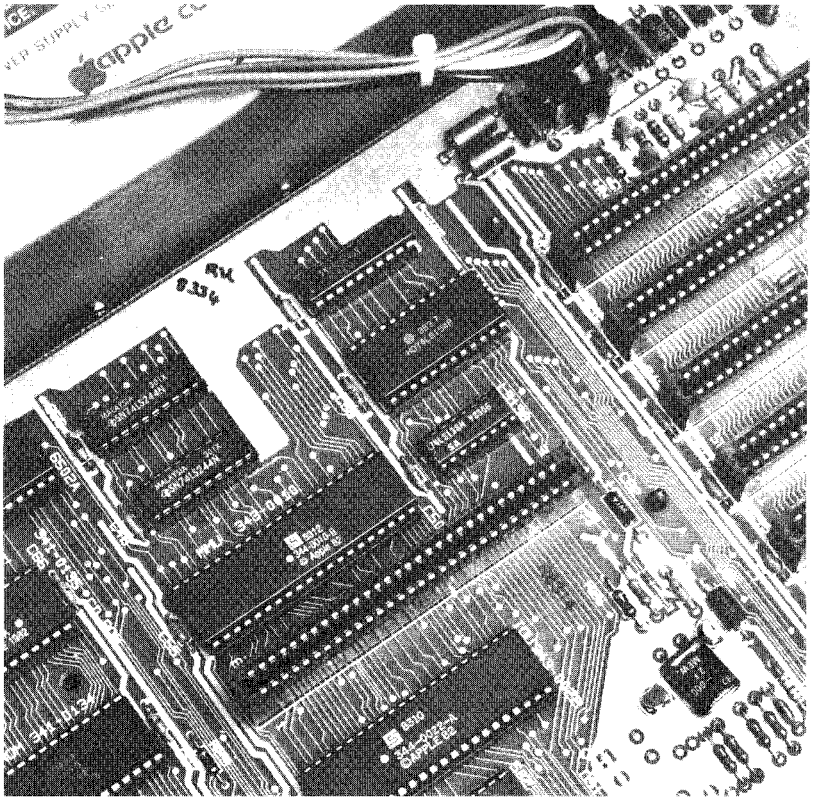
Les sept fentes alignées à l'arrière de la carte-mère de l'Apple //e sont les connecteurs d'extension, quelquefois appelés connecteurs périphériques (voir figure 1-6). Ces connecteurs permettent d'ajouter du hardware à l'Apple //e. Le chapitre 6 vous indique comment vos programmes dialoguent avec les dispositifs branchés dans ces connecteurs ; le chapitre 7 décrit les circuits pour les connecteurs eux-mêmes.

Figure 1-6. Les connecteurs d'extension



Le grand connecteur au milieu de la carte-mère aligné sur le connecteur n° 3 est le connecteur auxiliaire (voir figure 1-7). Si votre Apple //e a une carte de texte en 80 colonnes, elle sera installée dans ce connecteur. L'option d'affichage en 80 colonnes est complètement intégrée dans l'Apple //e ; elle est décrite ainsi,que les autres fonctions caractéristiques d'affichage, dans le chapitre 2. Les interfaces matérielles et logicielles sont décrites dans le chapitre 7.

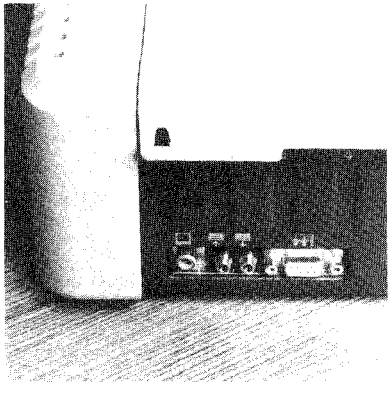
Figure 1-7. Le connecteur auxiliaire



Il y a aussi des connecteurs plus petits pour les manettes de jeu E/S et pour un modulateur interne RF (radio-fréquence). Ces connecteurs sont décrits au chapitre 7.

Connecteurs du panneau arrière

Figure 1-8. Les connecteurs du panneau arrière



Comme le montre la figure 1-8, l'arrière de l'Apple //e a deux petites prises pour connecter un magnétophone à cassettes, une prise de type RCA pour un moniteur vidéo, un petit connecteur de type D à 9 broches pour les manettes de jeu. En plus de ceux-ci, il y a des ouvertures pour des connecteurs supplémentaires utilisés avec les cartes d'interface installées dans l'Apple //e. Les manuels d'installation des cartes d'interfaces contiennent des instructions pour installer les connecteurs de périphériques.

Dispositifs d'E/S de base

Dispositifs d'E/S de base

Ce chapitre décrit les dispositifs d'entrée et de sortie (E/S) installés dans l'Apple //e selon leurs fonctions et leur utilisation par des programmes. Les dispositifs d'E/S de base sont

- le clavier
- le générateur d'affichage vidéo
- le haut-parleur
- les entrée et sortie sur cassette
- les entrée et sortie de jeu

Au plus bas niveau, les programmes utilisent les dispositifs d'E/S de base en lisant et en écrivant dans des adresses de mémoire réservées. Ce chapitre énumère ces adresses pour chaque dispositif d'E/S. Il donne aussi les adresses des commutateurs logiques internes qui sélectionnent les différents modes d'affichage de l'Apple //e. Pour les descriptions matérielles des E/S de base se reporter au chapitre 7.

Cette méthode d'entrée et de sortie — en chargeant et en mémorisant directement à des adresses spécifiques en mémoire — n'est pas la seule méthode que vous puissiez utiliser. Pour plusieurs de vos programmes, il peut s'avérer plus commode d'appeler les sous-programmes d'E/S de base implantés dans la MEM de l'Apple //e. Ces sous-programmes sont décrits au chapitre 3.

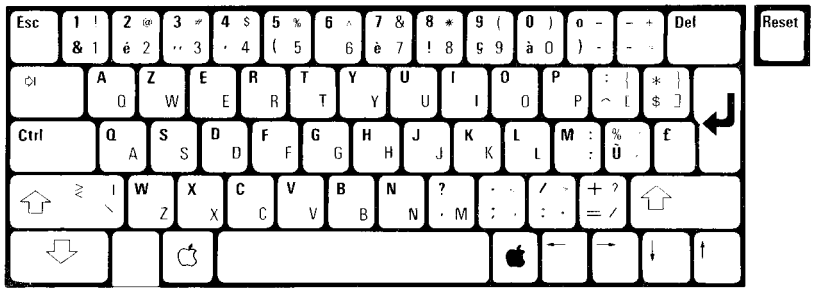
Le clavier

Le dispositif d'entrée primaire de l'Apple //e est son clavier intégré. Le clavier a 63 touches et il est similaire au clavier standard ISO d'une machine à écrire. Le clavier de l'Apple //e a le système de répétition automatique sur toutes les touches : on garde le doigt appuyé sur la touche pour la répéter. Il vous permet aussi de garder vos doigts appuyés sur un nombre quelconque de touches pendant

que vous tapez sur une autre touche (« N-Key-rollover »). Évidemment, si vous appuyez sur les touches plus longtemps qu'il ne faut normalement, la fonction de répétition automatique répétera la dernière touche sur laquelle vous avez appuyé.

La disposition des touches du clavier montré dans la figure 2-1 est celle utilisée dans certains pays européens. Les spécifications du clavier sont données au tableau 2-1. Votre Apple //e a une disposition de clavier conforme au standard ISO et comprend un inverseur pour choisir entre le jeu des caractères français et le jeu des caractères nord-américains. Le jeu des caractères nord-américains permet à des programmes écrits pour l'Apple //e nord-américain de fonctionner avec votre Apple //e.

Figure 2-1. Le clavier



- Les caractères en usage en France sont gravés sur la partie gauche des touches et en plus foncé.
- Les caractères américains sont gravés sur la partie droite des touches et en plus clair.

Tableau 2-1. Les spécifications du clavier de l'Apple //e

Nombre de touches :	63
Codification des caractères :	ASCII
Nombres de codes :	128
Touches spéciales :	Ctrl, Esc, Reset, flèches, pomme
Rollover :	Touche N
Interface électrique :	AY-5-3600 encodeur de clavier
Inverseur du clavier :	

En plus des touches normalement utilisées pour taper des caractères, il y a 4 touches de contrôle du curseur avec des flèches : à gauche, à droite, en bas, en haut. Les touches de contrôle du curseur peuvent être lues comme les autres touches ; leurs codes sont $08, $15, $0A et $0B (voir les tableaux 2-3 à 2-7).

Quatre touches spéciales, Ctrl, ⇧ SHIFT, ⇩ CAPS LOCK et Esc ESCAPE changent les codes générés par les autres touches. La touche Ctrl est similaire à la touche CTRL en ASCII. Trois autres touches ont des fonctions spéciales : la touche Reset et deux touches marquées avec des pommes, une avec son contour seulement, ⌘ et l'autre coloriée en noir ou pleine, ou ⌘. En appuyant sur la touche Reset avec la touche Ctrl maintenue enfoncée, on réinitialise l'Apple *//e*, comme cela est décrit au chapitre 4. Les touches POMME sont connectées aux entrées de jeu logiques, décrites plus loin dans ce chapitre.

L'interface électrique entre l'Apple *//e* et le clavier est un câble en ruban avec un connecteur à 26 broches. Ce câble fait passer les signaux du clavier vers le circuit de codage de la carte-mère. Une description complète de l'interface électrique vers le clavier est donnée au chapitre 7.

La lecture du clavier

Le codeur et la MEM du clavier génèrent tous les 128 codes ASCII. Donc tous les codes des caractères spéciaux de l'ensemble des caractères ASCII sont accessibles depuis le clavier. Les programmes en langage machine obtiennent les codes des caractères tapés au clavier en lisant un octet dans la position de mémoire réservée aux données du clavier indiquée dans le tableau 2-2.

Tableau 2-2. Adresses de mémoire du clavier

Adresse Hex	Décimale	Description
$C000	49152 – 16384	Données du clavier et échantillonnage
$C010	49168 – 16368	Indicateur qu'une touche est enfoncée et commutateur de remise à zéro de l'échantillonneur du clavier.

Vos programmes peuvent récupérer le code de la dernière touche appuyée en lisant à l'adresse des données du clavier. Le tableau 2-2 donne cette adresse en trois formulations différentes : la valeur *hexadécimale* utilisée en langage assembleur, indiquée par un signe dollar ($) la précédant ; la valeur décimale utilisée en BASIC Applesoft, et la valeur décimale complémentaire utilisée en BASIC Entier Apple. (Le BASIC Entier demande que toute valeur supérieure à 32767 soit écrite comme un nombre obtenu en soustrayant 65536 de la valeur. Ce sont les valeurs décimales

Hexadécimale fait référence au système de numération de base 16, qui utilise les 10 chiffres de 0 à 9 et les 6 lettres de A à F pour représenter les valeurs de 0 à 15.

indiquées comme négatives dans les tableaux ; se référer au *Manuel de programmation en BASIC* de l'Apple *//*e). Les sept bits de poids faibles de l'octet à l'adresse du clavier contiennent le code du caractère ; le bit de poids fort de cet octet est le bit d'échantillonnage, décrit plus loin.

Votre programme peut trouver si une touche est enfoncée, sauf les touches POMME, en lisant à l'adresse 49168 (héxadécimale $C010 ou décimale complémentaire—16368). Le bit de poids fort (bit 7) de l'octet que vous lisez à cette adresse est appelé une touche enfoncée ; il vaut 1 si une touche est enfoncée et vaut 0 si aucune touche n'est enfoncée. Le poids de ce bit est 128 ; si un programme BASIC saisit cette information avec un PEEK, la valeur est supérieure ou égale à 128 si une touche est enfoncée, et inférieure à 128 si aucune touche n'est enfoncée.

Le bit d'échantillonnage est le bit de poids fort de l'octet de données du clavier. Après qu'une touche ait été tapée, le bit d'échantillonnage est à 1. Il reste à 1 jusqu'à ce que vous le remettiez à 0 en lisant ou en écrivant à l'adresse de remise à zéro de l'échantillonneur. Cette adresse est à la fois un indicateur et un commutateur ; l'indicateur informe si on a appuyé sur une touche et le commutateur met à zéro le bit d'échantillonnage. La fonction de commutation de cette adresse de mémoire est appelée un *commutateur logiciel* car il est contrôlable par programme. Dans ce cas, peu importe que le programme lise ou écrive et peu importe quelle donnée le programme écrit : la seule action qui se produise est la remise à zéro du bit d'échantillonnage. Des commutateurs logiciels similaires, décrits plus loin, sont utilisés pour contrôler d'autres fonctions dans l'Apple *//*e.

> Chaque fois que vous lisez l'indicateur d'une touche enfoncée, vous mettez aussi à zéro l'échantillonneur du clavier. Si votre programme a besoin de lire l'indicateur et le bit d'échantillonnage, il doit lire le bit d'échantillonnage en premier.

Après avoir mis à zéro l'échantillonneur, il y reste jusqu'à ce qu'on appuie sur une autre touche. Même après avoir mis à zéro le bit d'échantillonnage, vous pouvez encore lire le code du caractère à l'adresse de données du clavier. L'octet a une valeur différente, car le bit de poids fort n'est plus à 1, mais le code ASCII dans les bits de poids faibles est le même jusqu'à ce qu'une autre touche soit enfoncée. Les tableaux 2-3 et 2-4 montrent les codes ASCII de la plupart des touches du clavier de l'Apple *//*e.

Tableau 2-3. Les touches et les codes ASCII, jeu de caractères français

Touche	Normal	Ctrl	Shift	Crtl et SHIFT
Del	7F	7F	7F	7F
←	08	08	08	08
→|	09	09	09	09
↓	0A	0A	0A	0A
↑	0B	0B	0B	0B
↵	0D	0D	0D	0D
→	15	15	15	15
Esc	1B	1B	1B	1B
Espace	20	20	20	20
! 8	21	21	38	38
" 3	22	22	33	33
$ *	24	24	2A	2A
& 1	26	26	31	31
' 4	27	27	34	34
(5	28	28	35	35
) °	29	29	5B	1B
, ?	2C	2C	3F	3F
- _	2D	2D	5F	1F
: /	3A	3A	2F	2F
; .	3B	3B	2E	2E
< >	3C	3C	3E	3E
= +	3D	3D	2B	2B
à 0	40	00	30	00
ç 9	5C	1C	39	39
§ 6	5D	1D	36	1D
^ ¨	5E	1E	7E	1E
` £	60	60	23	23
a A	61	01	41	01
b B	62	02	42	02
c C	63	03	43	03
d D	64	04	44	04
e E	65	05	45	05
f F	66	06	46	06
g G	67	07	47	07
h H	68	08	48	08
i I	69	09	49	09
j J	6A	0A	4A	0A
k K	6B	0B	4B	0B
l L	6C	0C	4C	0C
m M	6D	0D	4D	0D
n N	6E	0E	4E	0E
o O	6F	0F	4F	0F
p P	70	10	50	10
q Q	71	11	51	11
r R	72	12	52	12

Touche	Normal	Ctrl	Shift	Crtl et SHIFT
s S	73	13	53	13
t T	74	14	54	14
u U	75	15	55	15
v V	76	16	56	16
w W	77	17	57	17
x X	78	18	58	18
y Y	79	19	59	19
z Z	7A	1A	5A	1A
é 2	7B	7B	32	32
ù %	7C	7C	25	25
è 7	7D	7D	37	37

Tableau 2-4 a. Les touches et les codes ASCII, jeu de caractères nord-américains

Key	Normal	Control	Shift	Both
DELETE	7F	7F	7F	7F
LEFT-ARROW	08	08	08	08
TAB	09	09	09	09
DOWN-ARROW	0A	0A	0A	0A
UP-ARROW	0B	0B	0B	0B
RETURN	0D	0D	0D	0D
RIGHT-ARROW	15	15	15	15
ESC	1B	1B	1B	1B
space	20	20	20	20
' "	27	27	22	22
, <	2C	2C	3C	3C
- _	2D	2D	5F	1F
. >	2E	2E	3E	3E
/?	2F	2F	3F	3F
0)	30	30	29	29
1!	31	31	21	21
2@	32	00	40	00
3#	33	33	23	23
4$	34	34	24	24
5%	35	35	25	25
6 ^	36	1E	5E	1E
7&	37	37	26	26
8*	38	38	2A	2A
9(	39	39	28	28
: ;	3B	3B	3A	3A
= +	3D	3D	2B	2B

Tableau 2-4 b. Les touches et les codes ASCII, jeu de caractères nord-américains

Key	Normal	Control	Shift	Both
[{	5B	1B	7B	1B
|	5C	1C	7C	1C
]}	5D	1D	7D	1D
`~	60	60	7E	7E
A	61	01	41	01
B	62	02	42	02
C	63	03	43	03
D	64	04	44	04
E	65	05	45	05
F	66	06	46	06
G	67	07	47	07
H	68	08	48	08
I	69	09	49	09
J	6A	0A	4A	0A
K	6B	0B	4B	0B
L	6C	0C	4C	0C
M	6D	0D	4D	0D
N	6E	0E	4E	0E
O	6F	0F	4F	0F
P	70	10	50	10
Q	71	11	51	11
R	72	12	52	12
S	73	13	53	13
T	74	14	54	14
U	75	15	55	15
V	76	16	56	16
W	77	17	57	17
X	78	18	58	18
Y	79	19	59	19
Z	7A	1A	5A	1A

Il y a plusieurs touches à fonctions spéciales qui ne génèrent pas de codes ASCII. Par exemple, vous ne pouvez pas lire directement Ctrl, ⇧, ⇩ (blocage des lettres majuscules), mais le fait d'appuyer sur ces touches modifie les codes des caractères produits par les autres touches.

Une autre touche qui ne génère pas de code est la touche Reset, située dans le coin en haut et à droite du clavier ; elle est connectée directement aux circuits de l'Apple *//e*. Appuyer sur la touche Reset avec la touche Ctrl enfoncée normalement, provoque l'arrêt du système quelque soit le programme en cours et son redémarrage. Ce processus de réinitialisation est appelé la procédure Reset et il est décrit au chapitre 4.

Deux autres touches spéciales sont gravées avec deux versions du logo Apple, sont situées de part et d'autre de la barre d'espacement : la touche ⌘ et la touche ⌘. Ces touches sont connectées aux entrées de jeu à un bit, qui sont décrites plus loin dans ce chapitre. Appuyez dessus tout en maintenant enfoncées les touches Ctrl et Reset, conduit à l'exécution de cycles de réinitialisation spéciale et d'auto-test effectués par des sous-programmes intégrés en MEM du système, décrits avec la procédure Reset dans le chapitre 4.

Tableau 2-5. L'ensemble des caractères ASCII

Dec	Hex	Char	Dec	Hex	Char	Dec	Hex	Char	Dec	Hex	Char
0	00	NUL	32	20	SP	64	40	à	96	60	`
1	01	SOH	33	21	!	65	41	A	97	61	a
2	02	STX	34	22	"	66	42	B	98	62	b
3	03	ETX	35	23	a	67	43	C	99	63	c
4	04	EOT	36	24	$	68	44	D	100	64	d
5	05	ENQ	37	25	%	69	45	E	101	65	e
6	06	ACK	38	26	&	70	46	F	102	66	f
7	07	BEL	39	27	'	71	47	G	103	67	g
8	08	BS	40	28	(	72	48	H	104	68	h
9	09	HT	41	29	)	73	49	I	105	69	i
10	0A	LF	42	2A	*	74	4A	J	106	6A	j
11	0B	VT	43	2B	+	75	4B	K	107	6B	k
12	0C	FF	44	2C	,	76	4C	L	108	6C	l
13	0D	CR	45	2D	-	77	4D	M	109	6D	m
14	0E	SO	46	2E	.	78	4E	N	110	6E	n
15	0F	SI	47	2F	/	79	4F	O	111	6F	o
16	10	DLE	48	30	0	80	50	P	112	70	p
17	11	DC1	49	31	1	81	51	Q	113	71	q
18	12	DC2	50	32	2	82	52	R	114	72	r
19	13	DC3	51	33	3	83	53	S	115	73	s
20	14	DC4	52	34	4	84	54	T	116	74	t
21	15	NAK	53	35	5	85	55	U	117	75	u
22	16	SYN	54	36	6	86	56	V	118	76	v
23	17	ETB	55	37	7	87	57	W	119	77	w
24	18	CAN	56	38	8	88	58	X	120	78	x
25	19	EM	57	39	9	89	59	Y	121	79	y
26	1A	SUB	58	3A	:	90	5A	Z	122	7A	z
27	1B	ESC	59	3B	;	91	5B	°	123	7B	é
28	1C	FS	60	3C	<	92	5C	ç	124	7C	ù
29	1D	GS	61	3D	=	93	5D	§	125	7D	è
30	1E	RS	62	3E	>	94	5E	^	126	7E	¨
31	1F	US	63	3F	?	95	5F	_	127	7F	DEL

Le générateur d'affichage vidéo

Le dispositif de sortie primaire de l'Apple //e est l'écran vidéo. Vous avez plusieurs possibilités de raccordement vidéo :

- La prise arrière « MONITEUR » repérée par l'image d'une télévision qui est la sortie VIDÉO normale. Le signal vidéo disponible est un signal dit « VIDÉO compatible » aux normes CCIR, standard couleur P.A.L.
- Une carte d'interface spécialisée couleur, appelée « Carte RVB étendue » ou « Carte EVE » ou « Carte Chat Mauve spéciale Apple //e ». Cette carte permet un raccordement direct sur tout téléviseur ou moniteur couleur équipé d'une prise péritélévision. Elle possède également d'autres caractéristiques comme le mode 80 colonnes et l'extension de mémoire de 64 kilo-octets.

Un premier cas d'utilisation est l'affichage en noir et blanc (texte ou graphique) : n'importe quel moniteur vidéo (entrée directe en vidéo composite) est connectable à la prise arrière « MONITEUR » de l'Apple //e. Le raccordement à un téléviseur noir et blanc ou couleur est possible à condition d'intercaler un modulateur vidéo entre la sortie de l'Apple //e et l'entrée dite « UHF » ou « VHF » du téléviseur (ce type de modulateur est en vente chez les détaillants radio-télévision ou les magasins de composants électroniques). La meilleure solution est l'utilisation de la « Carte RVB étendue » avec un moniteur ou un téléviseur couleur avec prise péritélévision permettant l'affichage de 80 colonnes et une très haute résolution graphique.

Pour l'utilisation en couleur (texte ou graphique) il faut :

- soit un moniteur couleur au standard P.A.L. relié directement à la sortie moniteur de l'Apple //e,
- soit un téléviseur aux normes couleur P.A.L. relié à l'Apple //e par l'intermédiaire d'un modulateur vidéo P.A.L. dont la sortie est branchée à l'entrée « UHF » ou « VHF » du téléviseur,
- soit la « Carte RVB étendue » avec un moniteur ou un téléviseur couleur équipé d'une prise péritélévision. Pour connaître les caractéristiques de cette carte, se reporter au *Manuel de référence de « Eve ». RVB 80 col 64K-Le Chat Mauve.*

Avec une carte 80 colonnes de texte, l'Apple //e peut produire un affichage sur 80 colonnes. Cependant, si vous utilisez un moniteur ordinaire en couleur ou en noir et blanc, la lecture des caractères 80 colonnes n'est pas très satisfaisante. Pour un affichage net en 80 colonnes, vous devez utiliser un moniteur vidéo monochrome de haute-résolution ayant une bande passante de 14MHz ou plus. Avec la « Carte RVB étendue », vous aurez une meilleure lisibilité en 80 colonnes en faisant POKE −16205,0 et POKE −16203,0.

Les spécifications de l'écran vidéo sont résumées dans le tableau 2-6.

Tableau 2-6. Spécifications d'écran vidéo

Modes d'affichage :	Texte sur 40 colonnes
	Texte sur 80 colonnes avec carte en option
	Graphique couleur basse-résolution
	Graphique couleur haute-résolution
Capacité de texte :	24 lignes par 40 colonnes
	24 lignes par 80 colonnes avec carte en option
Ensemble de caractères :	96 caractères ASCII (majuscules et minuscules)
Modes spéciaux d'affichage :	Normal, Inverse, Clignotant
Graphique basse-résolution :	16 couleurs, résolution de 40 par 48
Graphique haute-résolution :	6 couleurs, résolution de 280 par 192

Le signal vidéo produit par l'Apple //e est un signal couleur composite compatible P.A.L. Il est disponible à trois endroits : la prise de type RCA située à l'arrière de l'Apple //e, la broche simple de type Molex sur la carte-mère près de l'arrière sur le côté droit, et une des broches du groupe des quatre broches de type Molex dans la même zone de la carte-mère. Utilisez la prise de type RCA pour brancher un moniteur vidéo ou un modulateur vidéo externe ; utilisez les broches Molex pour connecter le type de modulateur vidéo qui s'adapte à l'intérieur de l'Apple //e.

L'interrupteur interne situé sur la carte-mère de l'Apple //e permet de supprimer la porteuse chrominance P.A.L. et donc d'améliorer nettement l'affichage des caractères sur un moniteur monochrome (noir et blanc par exemple). Il doit être sur la position ON dans ce cas, bien entendu la sortie « moniteur » devient une sortie purement noir et blanc, le signal couleur P.A.L. n'étant plus disponible. Contrairement à certains Apple II et Apple II Plus, il n'existe plus de sortie couleur NTSC (standard couleur américain) au niveau de la sortie vidéo couleur.

Pour une description complète du signal vidéo et des branchements aux broches de type Molex, référez-vous au paragraphe « Signaux de sortie vidéo » du chapitre 7.

L'Apple *//e* peut produire quatre sortes d'affichage vidéo :

- Texte, 24 lignes de 40 caractères
- Texte, 24 lignes de 80 caractères (avec une carte en option)
- Graphique basse-résolution, 40 par 48, en 16 couleurs
- Graphique haute-résolution, 280 par 192, en 6 couleurs

La carte « Eve » donne accès à des modes de visualisation supplémentaires :

- Texte 80 colonnes, noir et vert
- Texte 40 colonnes, couleur, en 16 couleurs de fond et de caractères
- Graphique 140 par 192, en 16 couleurs
- Graphique 280 par 192, en 4 couleurs (noir, orange, vert et blanc, ou bien, noir, bleu clair, rose et jaune)
- Graphique double-résolution, 560 par 192 en noir et blanc
- Graphique 280 par 192, en 16 couleurs avec une limitation à deux couleurs indépendantes sur 7 points consécutifs horizontaux.

Les deux modes de texte peuvent afficher tous les 96 caractères ASCII : les lettres majuscules et minuscules, les nombres et les symboles.

Les deux modes graphiques peuvent avoir quatre lignes de texte, en 40 colonnes ou en 80 colonnes, dans le bas de l'écran. Les écrans graphiques ayant du texte dans le bas sont appelés des *affichages en mode mixte.*

L'écran graphique en basse-résolution est une matrice de blocs colorés (seize couleurs), 40 en largeur par 48 en hauteur.En mode mixte, les quatre lignes de texte remplacent les huit rangées de blocs du bas, laissant 40 lignes de 40 blocs chacune.

L'écran graphique en haute-résolution est une matrice de points, 280 en largeur par 192 en hauteur. Il y a six couleurs disponibles en affichage haute-résolution, mais un point donné ne peut utiliser que quatre des six couleurs. En mode mixte, les quatre lignes de texte remplacent les 32 lignes de points du bas, laissant 160 lignes de 280 points chacune.

Les modes texte

Les caractères affichables de texte incluent les lettres majuscules et minuscules, les dix chiffres, les symboles de ponctuation et les caractères spéciaux. Chaque caractère est affiché dans une zone de l'écran qui a sept points de large et huit points de haut. Les caractères sont dessinés dans une matrice de cinq points de large, laissant deux colonnes vides de points entre deux caractères voisins d'une rangée. Hormis les lettres minuscules ayant des jambages, les caractères n'ont que sept points de haut, laissant une ligne vide de points entre les rangées de caractères.

L'affichage normal présente des points blancs (ou une autre couleur seule) sur un fond noir. Les caractères peuvent aussi être affichés en points noirs sur un fond blanc ; cela s'appelle le *mode inverse.*

Les ensembles de caractères de texte

L'Apple *//*e peut afficher l'un des deux ensembles de caractères suivants : l'ensemble Primaire et un ensemble Alternatif. La forme des caractères dans les deux ensembles est la même, mais les modes spéciaux disponibles d'affichage sont différents. Les modes spéciaux sont

- le mode *normal,* avec des points blancs sur un écran noir ;
- le mode *inverse,* avec des points noirs sur un écran blanc ; et
- le mode *clignotant,* alternativement normal et inverse.

Avec l'ensemble primaire de caractères, l'Apple *//*e peut afficher les lettres majuscules dans les trois modes : normal, inverse et clignotant. Les lettres minuscules ne peuvent être affichées qu'en mode normal. L'ensemble primaire de caractères est compatible avec la plupart des logiciels écrits pour l'Apple II et l'Apple II Plus, qui peuvent afficher du texte en mode clignotant mais n'ont pas de lettres minuscules.

L'ensemble alternatif de caractères sacrifie le mode clignotant pour un mode inverse complet. Avec l'ensemble alternatif de caractères, l'Apple *//*e peut afficher les lettres majuscules, les lettres minuscules, les nombres et les caractères spéciaux dans l'un ou l'autre des modes normal et inverse.

Vous sélectionnez l'ensemble des caractères au moyen d'un commutateur logiciel de texte alternatif, décrit plus loin dans le paragraphe « Commutation de modes d'affichage ». Le Tableau 2-7 donne les codes des caractères en décimal et en hexadécimal pour les ensembles primaire et alternatif des caractères de l'Apple *//*e en modes normal, inverse et clignotant.

Tableau 2-7. Les ensembles de caractères affichables

Pour identifier des caractères particuliers et des valeurs, se reporter au tableau 2-5.

Inv. : Inverse, Cli. : Clignotant
Nor. : Normal

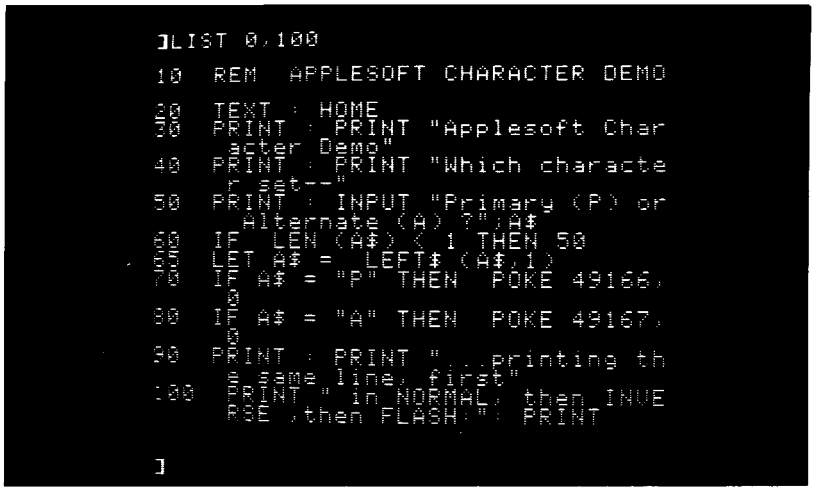
Valeurs Hexa.	Ensemble Primaire : Type de caractères	Mode	Ensemble Alternatif : Type de caractères	Mode
$00-$1F	Lettres majuscules	Inv.	Lettres minuscules	Inv.
$20-$3F	Caractères spéciaux	Inv.	Caractères spéciaux	Inv.
$40-$5F	Lettres majuscules	Cli.	Lettres majuscules	Inv.
$60-$7F	Caractères spéciaux	Cli.	Lettres minuscules	Inv.
$80-$9F	Lettres majuscules	Nor.	Lettres majuscules	Nor.
$A0-$BF	Caractères spéciaux	Nor.	Caractères spciaux	Nor.
$C0-$DF	Lettres majuscules	Nor.	Lettres majuscules	Nor.
$E0-$FF	Lettres minuscules	Nor.	Lettres minuscules	Nor.

Chaque caractère sur l'écran est mémorisé dans un octet de données à afficher. Les six bits de poids faibles constituent le code ASCII du caractère affiché. Les deux bits de poids forts restants sélectionnent le mode inverse ou clignotant et les caractères majuscules ou minuscules. Dans l'ensemble primaire des caractères, le bit 7 sélectionne le mode inverse ou normal et le bit 6 contrôle le clignotement du caractère. Dans l'ensemble alternatif des caractères, le bit 5 fait la sélection entre majuscules et minuscules, suivant les codes ASCII, et le mode clignotant n'est pas accessible.

Texte en 40 colonnes ou en 80 colonnes

L'Apple //e a deux formats d'affichage : 40 colonnes et 80 colonnes. (Ce format de 80 colonnes décrit dans ce manuel est celui que vous obtenez avec une Carte Texte en 80 colonnes ou une autre carte de mémoire auxiliaire installée dans le connecteur auxiliaire. Le nombre de points dans chaque caractère ne change pas, mais les caractères en 80 colonnes sont seulement de la moitié de la largeur des caractères en format 40 colonnes. Comparez les figures 2-2 et 2-3. Sur un téléviseur ordinaire couleur ou noir et blanc, les caractères étroits de l'écran en 80 colonnes sont difficiles à séparer les uns des autres à la lecture ; vous devez utilisez le format en 40 colonnes pour afficher du texte sur un téléviseur.

Figure 2-2. Affichage de texte en 40 colonnes



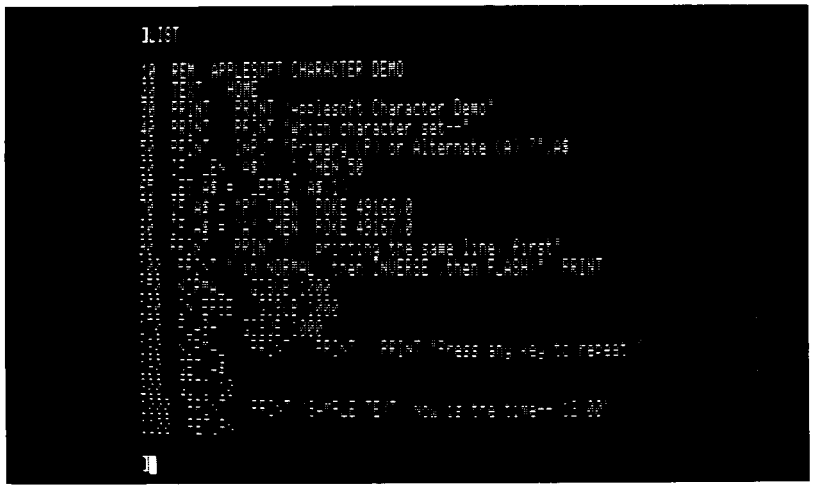
```
]LIST 0,100

10  REM  APPLESOFT CHARACTER DEMO

20  TEXT : HOME
30  PRINT : PRINT "Applesoft Char
    acter Demo"
40  PRINT : PRINT "Which characte
    r set--"
50  PRINT : INPUT "Primary (P) or
    Alternate (A) ?";A$
60  IF  LEN (A$) < 1 THEN 50
65  LET A$ =  LEFT$ (A$,1)
70  IF A$ = "P" THEN  POKE 49166,
    0
80  IF A$ = "A" THEN  POKE 49167,
    0
90  PRINT : PRINT "...printing th
    e same line, first"
100 PRINT " in  NORMAL, then INVE
    RSE ,then FLASH.": PRINT

]
```

Figure 2-3. Affichage de texte en 80 colonnes



Les modes graphiques

L'Apple *//*e peut produire des graphiques vidéo dans un des deux différents modes. Les deux modes graphiques considèrent l'écran comme une matrice rectangulaire de points lumineux. Normalement,vos programmes utiliseront les fonctions spécifiques de langages évolués pour dessiner des points, des lignes, des formes dans ces matrices ; ce paragraphe décrit la façon dont les données graphiques résultantes sont mémorisées dans la mémoire de l'Apple *//*e.

Graphiques en basse-résolution

Dans ce mode graphique basse-résolution, l'Apple //e affiche une matrice de 48 lignes par 40 colonnes de blocs colorés. Chaque bloc peut être d'une couleur parmi seize, y compris le noir et le blanc. Sur un moniteur noir et blanc ou un téléviseur, ces couleurs apparaissent en noir, blanc et trois nuances de gris. Il n'y a pas de points non affichés entre les blocs ; les blocs adjacents de la même couleur se rejoignent pour produire une forme plus large.

Les données d'un affichage graphique basse-résolution sont mémorisées dans la même partie de mémoire que les données de l'affichage de texte en 40 colonnes. Chaque octet contient les données de deux blocs graphiques basse-résolution. Les deux blocs sont affichés l'un au-dessus de l'autre dans un espace d'écran de la même taille que celle d'un caractère d'un texte à 40 colonnes, sept points de large et huit points de haut.

La moitié d'un octet — quatre bits ou un demi-octet — est affectée à chaque bloc graphique. Chaque demi-octet peut prendre une valeur entre 0 et 15, et cette valeur détermine quelle sera, parmi les seize couleurs, celle qui apparaîtra sur l'écran.

Les couleurs et leur valeur de demi-octet correspondante sont données dans le tableau 2-8. Dans chaque octet, le demi-octet de plus faible poids définit la couleur du bloc du haut de la paire, et le demi-octet de poids fort définit la couleur du bloc du bas. Donc, un octet contenant la valeur héxadécimale $D8 produit sur l'écran, un bloc marron au dessus d'un bloc jaune.

Tableau 2-8. Couleurs des graphiques en basse-résolution.

Les couleurs peuvent varier, suivant les contrôles du moniteur ou du téléviseur

Valeur du demi-octet			Valeur du demi-octet		
Dec	**Hex**	**Couleur**	**Dec**	**Hec**	**Couleur**
0	$0	Noir	8	$8	Marron
1	$1	Magenta	8	$9	Orange
1	$2	Bleu foncé	10	$A	Gris 2
3	$3	Violet	11	$B	Rose
4	$4	Vert foncé	12	$C	Vert
5	$5	Gris 1	13	$D	Jaune
6	$6	Bleu	14	$E	Bleu turquoise
7	$7	Bleu clair	15	$F	Blanc

Comme cela vous sera expliqué dans le paragraphe « Pages d'affichage », l'affichage de texte et l'affichage de graphiques basse-résolution utilisent la même zone de mémoire. La plupart des programmes qui génèrent du texte et des graphiques effacent cette partie de mémoire lorsqu'ils changent de modes d'affichage, mais il est possible de mémoriser des données en texte et de les afficher en graphique, ou vice-versa. Tout ce que vous avez à faire est de

changer le commutateur de mode, décrit dans le paragraphe « Commutation de mode d'affichage », sans changer les données affichées. Ceci produit sur l'écran des tableaux abstraits sans aucune signification, mais certains programmes ont utilisé cette technique à bon escient pour produire des affichages rapides de graphiques en basse-résolution.

Graphiques en haute-résolution

Dans le mode graphique haute-résolution, l'Apple //e affiche une matrice de points colorés de 192 lignes et de 280 colonnes. Les couleurs disponibles sont le noir, le blanc, le violet, le vert, l'orange et le bleu, bien que les couleurs des points isolés soient limitées, comme il sera décrit plus loin. Les points adjacents de la même couleur se rejoignent pour former une zone colorée plus large.

Les données des affichages graphiques en haute-résolution sont mémorisées dans l'une des deux zones de 8 192 octets en mémoire. Ces zones sont appelées la Page 1 et la Page 2 en haute-résolution graphique ; imaginez-les comme des mémoires-tampons dans lesquelles vous pouvez mettre des données à afficher. Normalement, vos programmes utiliseront les fonctions des langages évolués pour dessiner des graphiques, des points, des lignes, et des formes à afficher ; ce paragraphe décrit la façon dont les données graphiques résultantes sont enregistrées dans la mémoire de l'Apple //e.

L'affichage des graphiques haute-résolution de l'Apple //e est une application bit à bit : chaque point de l'écran correspond à un bit de la mémoire de l'Apple //e. Les sept bits de faible poids de chaque octet à afficher contrôlent une rangée de sept points adjacents sur l'écran, et quarante octets adjacents en mémoire contrôlent une rangée de 280 points (7 fois 40). Le bit de plus faible poids de chaque octet est affiché comme le point le plus à gauche dans une rangée de sept, suivi du deuxième bit de poids le plus faible, et ainsi de suite, comme le montre la figure 2-4. Le huitième bit (de plus fort poids) de chaque octet n'est pas affiché ; il sélectionne un des deux ensembles de couleurs, comme il sera expliqué plus loin.

Sur un moniteur noir et blanc, il existe une correspondance simple entre bits de la mémoire et points sur l'écran. Un point est blanc si le bit le contrôlant est à un (1), et le point est noir si le bit est à zéro (0). Sur un téléviseur noir et blanc, des paires de points se confondent ; des points alternativement noirs et blancs se fondent en un gris continu.

Sur un moniteur couleur ou une télévision en couleur, à un bit à zéro est associé un point noir. Si le bit est à un, le point sera blanc ou de couleur, suivant sa position, suivant la couleur des points de part et d'autre, et suivant l'état du bit de plus fort poids de l'octet. En appelant colonne zéro la colonne la plus à gauche d'une rangée de points, et en supposant (pour l'instant) que les bits de plus fort poids de tous les octets de données sont à zéro, si les bits qui les contrôlent sont à un, les points dans les colonnes paires, 0, 2, 4 et ainsi de suite, seront violets, et les points dans les colonnes de numéros impairs seront verts — mais seulement si les points de chaque côté sont noirs. Si deux points adjacents sont tous les deux allumés, ils seront tous les deux blancs.

Vous sélectionnez les deux autres couleurs, bleue et orange, en mettant le bit de plus fort poids (bit 7) à un. Les points colorés contrôlés par un octet dont le bit de plus fort poids est à un, sont soit de couleur bleue soit de couleur orange : les points dans des colonnes de numéros pairs sont bleus, et les points dans les colonnes impaires sont orange — et à nouveau, seulement si les points de part et d'autre sont noirs. Dans chaque ligne horizontale de sept points contrôlés par un seul octet, vous pouvez avoir du noir, du blanc, et une paire de couleurs.

Pour changer la couleur d'un point en l'une des couleurs de l'autre paire, vous devez changer le bit de poids le plus fort de son octet, ce qui affectera les couleurs de tous les sept points contrôlés par l'octet.

En d'autres termes, les graphiques affichés en haute-résolution sur un moniteur ou un téléviseur sont faits de points colorés, et respectent les règles suivantes :

- Les points situés sur des colonnes paires peuvent être noirs, violets ou bleus.
- Les points situés sur des colonnes impaires peuvent être noirs, verts ou orange.
- Si deux points adjacents d'une ligne sont allumés, ils sont tous les deux blancs.
- Les couleurs dans chaque rangée de sept points contrôlés par un seul octet sont, soit le violet et le vert, soit le bleu et l'orange, suivant que le bit de poids le plus fort est à zéro ou à un.

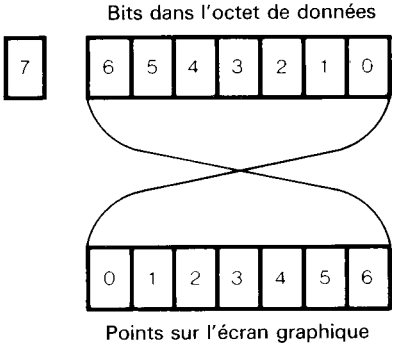
Ces règles sont résumées dans le tableau 2-9. Les noirs et blancs sont numérotés pour vous rappeler que le bit de poids fort a une valeur différente.

Tableau 2-9. Couleurs des graphiques haute-résolution

Les couleurs peuvent varier avec le réglage du moniteur ou du téléviseur.

Bits 0-6	Bit 7 à zéro	Bit 7 à un
Colonnes adjacentes à zéro	Noir 1	Noir 2
Colonnes paires à un	Violet	Bleu
Colonnes impaires à un	Vert	Orange
Colonnes adjacentes à un	Blanc 1	Blanc 2

Figure 2-4. Les bits d'affichage en haute-résolution



Le comportement particulier des couleurs en haute-résolution reflète le principe de fonctionnement de la couleur. Les points qui constituent le signal vidéo de l'Apple *//e* sont espacés pour coïncider avec la fréquence de la sous-porteuse de couleur à 3,58 MHz utilisée dans le système NTSC. En alternant des points noir et blanc avec cet espacement, on fait produire de la couleur par le téléviseur ou le moniteur, mais pas avec deux points blancs ensemble ou plus. Des circuits additionnels sont implémentés dans l'Apple *//e* International (français en l'occurrence) pour produire la sous-porteuse de couleur P.A.L. Pour plus de détails sur la manière de produire des couleurs de l'Apple *//e*, voir le chapitre 7, et éventuellement le *Manuel de référence de la carte « Ève »,* pour les caractéristiques des options graphiques supplémentaires offertes par cette carte.

Pages d'affichage sur écran

L'Apple *//e* génère ses affichages vidéo en utilisant des données stockées dans des zones spécifiques en mémoire. Ces zones, appelées pages d'affichage, servent de mémoires-tampons où vos programmes peuvent mettre des données à afficher. Chaque octet dans une mémoire-tampon d'affichage contrôle un objet localisé sur l'écran. En mode texte, l'objet est un caractère ; en basse-résolution, l'objet est fait de deux blocs colorés empilés ; en mode haute-résolution, c'est une ligne de sept points adjacents.

Les modes texte en 40 colonnes et graphique en basse-résolution utilisent deux pages d'affichage de 1 024 octets chacune. On les appelle la Page Texte 1 et la Page Texte 2, et elles sont localisées aux adresses 1024-2047 (en héxédécimal $400-$7FF) et 2048-3071 ($800-$BFF) en mémoire centrale. Normalement, seule la Page 1 est utilisée, mais vous pouvez mettre des données en texte ou en graphique dans la Page 2 et changer instantanément d'affichage par commutation. L'une ou l'autre des deux pages peut être affichée en texte 40 colonnes, en graphique basse-résolution, ou en mode mixte (quatre lignes de texte dans le bas de l'affichage graphique).

Le mode texte en 80 colonnes affiche deux fois plus de données que le mode en 40 colonnes — 1 920 octets — mais il ne permet pas de commuter les pages. L'affichage de texte en 80 colonnes utilise une page combinée faite de la Page Texte 1 en mémoire centrale plus une autre page dans la mémoire auxiliaire localisée sur la carte de texte en 80 colonnes. Cette mémoire additionnelle *n'est pas* la même que la Page Texte 2 — en fait, elle occupe le même

espace d'adresses que la Page Texte 1, et il y a un commutateur logiciel spécial qui vous autorise à y stocker des données (voir le paragraphe « Commutation des modes d'affichage » plus loin). Les sous-programmes de base de gestion des E/S décrits au chapitre 3 prennent soin automatiquement de cet adressage supplémentaire ; c'est une raison pour vous servir de ces sous-programmes pour toute sortie normale de texte.

Le mode graphique haute-résolution a aussi deux pages d'affichage, mais chaque page a 8 192 octets de long. Dans les modes texte en 40 colonnes et graphique basse-résolution, chaque octet contrôle une zone d'affichage de sept points de large sur huit points de haut. Dans le mode graphique haute-résolution chaque octet contrôle une zone de sept points de large sur un point de haut. Donc, un affichage en haute-résolution nécessite huit fois plus de mémorisation de données, comme le montre le tableau 2-10.

Tableau 2-10. Adresses des pages d'affichage vidéo

Mode d'affichage	Page	Adresse la plus basse		Adresse la plus haute	
Texte en 40 colonnes, Graphiques en basse-résolution	1	$400	1024	$7FF	2047
	2	$800	2048	$BFF	3071
Texte en 80 colonnes	1*	$400	1024	$7FF	2047
Graphiques en haute-résolution	1	$2000	8192	$3FFF	16383
	2	$4000	16384	$5FFF	24575

* Note : Le mode 80 colonnes utilise les adresses des 1 024 octets de la Page Texte 1 dans les deux mémoires principale et auxiliaire. Le commutateur PAGE2 est utilisé pour sélectionner l'une ou l'autre pour enregistrer des données (voir le paragraphe « Commutations des modes d'affichage »).

Commutation des modes d'affichage

Vous sélectionnez le mode d'affichage approprié à votre application en lisant ou en écrivant à l'adresse de mémoire réservée appelée un commutateur logiciel. Dans l'Apple *//e*, la plupart des commutateurs logiciels ont trois adresses de mémoire qui leur sont réservées : une pour mettre le commutateur à un, une pour le mettre à zéro, et une pour lire l'état actuel du commutateur.

Le tableau 2-11 montre quelles sont les adresses réservées à ces commutateurs logiciels qui contrôlent les différents modes d'affichage. Par exemple, pour passer du mode mixte au mode plein écran graphique dans un programme en assembleur, vous pourriez utiliser l'instruction :

```
STA $C052
```

Pour faire cela dans un programme en BASIC, vous auriez à utiliser l'instruction :

```
POKE 49234,0
```

Le générateur d'affichage vidéo

La table donne les adresses des commutateurs sous trois formes : héxadécimale, décimale et décimale négative. Vous pouvez utiliser les valeurs héxadécimales dans vos programmes en langage machine. Utiliser les valeurs décimales dans les commandes PEEK ou POKE du BASIC Applesoft ; les valeurs négatives sont pour le BASIC Entier.

Vous pourriez ne pas avoir besoin de manipuler ces fonctions de lecture et d'écriture directes dans les adresses de mémoire de ce tableau. Plusieurs des fonctions montrées ici sont sélectionnées automatiquement si vous utilisez les sous-programmes d'affichage des divers langages évolués sur l'Apple //e.

Quelques uns des commutateurs logiciels du tableau 2-11 sont marqués lire ou écrire. Ces commutateurs logiciels partagent leurs adresses avec celles des données du clavier et des fonctions d'échantillonnage du clavier. Dans l'Apple //e original, les adresses de mémoire de $C000 à $C01F (49152 à 49183) étaient utilisées seulement pour les données et les fonctions d'échantillonnage du clavier. Dans l'Apple //e, ces adresses sont utilisées de la même manière, mais seulement si vous *lisez* pour récupérer les données et si vous *écrivez* pour remettre à zéro l'échantillonneur. Pour réaliser la fonction indiquée dans le tableau, utilisez l'opération inscrite en note. Les commutateurs logiciels qui ne sont pas marqués avec les notes lire ou écrire peuvent être stimulés, soit par une lecture, soit par une écriture. En écrivant à l'adresse d'un commutateur logiciel, peu importe la valeur que vous y écrivez ; l'action se produit quand vous vous adressez à cette adresse, et la valeur est ignorée.

Chaque fois que vous lisez la valeur d'un commutateur logiciel, vous obtenez un octet de donnée. Cependant, la seule information contenue dans l'octet est l'état du commutateur, et celle-ci n'occupe qu'un seul bit — le bit 7, le bit de poids le plus élevé. Les autres bits de l'octet ne sont pas prévisibles. Si vous programmez en langage machine, la valeur du commutateur est le bit de signe ; dès que vous lirez l'octet, vous pourrez faire un BPL (branchement si positif) si le commutateur est à zéro, ou un BMI (branchement si négatif) si le commutateur est à un.

Si vous lisez un commutateur logiciel dans un programme en BASIC, vous obtenez une valeur entre 0 et 255. Le bit 7 a un poids de 128, donc si le commutateur est à un, la valeur sera égale ou supérieure à 128 ; si le commutateur est à zéro, la valeur sera inférieure à 128.

Tableau 2-11. Commutateurs logiciels d'affichage

(1) Ce mode n'est opérationnel que si le commutateur de mode graphique est à UN.
(2) Ce commutateur a une fonction différente si la page auxiliaire de texte d'une carte de texte en 80 colonnes a été autorisée en écriture. Se reporter au prochain paragraphe « Adressage direct des pages d'affichage ».
(3) Ce commutateur change la fonction du commutateur PAGE2 pour permettre l'adressage de la mémoire auxiliaire de texte sur une carte de texte en 80 colonnes. Le prochain paragraphe décrit comment le faire.
(4) En lisant à cette adresse, on obtient l'état du signal VBL de suppression verticale. La fonction de VBL est décrite au chapitre 7 dans le paragraphe « Signaux de sortie vidéo ».

Nom	Fonction	Adresse Hex	Adresse Décimale		Notes
ALTCHARSET	Ens. Alternatif à un	$C00F	49167	−16369	Écrire
	Ens. Alternatif à zéro	$C00E	49166	−16370	Écrire
	Lire le comm. ALTCHARSET	$C010	49182	−16354	Lire
TEXT	Mode Texte à un	$C051	49233	−16303	
	Mode Texte à zéro (GR)	$C050	49232	−16304	
	Lire le comm. TEXT	$C01A	49178	−16358	Lire
MIXED	Mode Mixte à un	$C053	49235	−16301	1
	Mode Mixte à zéro	$C052	49234	−16302	1
	Lire le comm. MIXED	$C01B	49179	−16357	Lire
PAGE2	Page 2 à un	$C055	49237	−16299	2
	Page 2 à zéro (Page 1)	$C054	49236	−16300	2
	Lire le comm. PAGE2	$C01C	49180	−16356	Lire
HIRES	Mode haute-res à un	$C057	49239	−16297	1
	Mode haute-res à zéro	$C056	49238	−16298	1
	Lire le comm. HIRES	$C01D	49181	−16355	Lire
80COL	Affi. 80 colonnes à un	$C00D	49165	−16371	Écrire
	Aff. 80 colonnes à zéro	$C00C	49164	−16372	Écrire
	Lire le comm. 80COL	$C01F	49183	−16353	Lire
80STORE	Enrgt en mémoire aux.	$C001	49153	−16383	Écrire, 3
	Enrgt en mém. centrale	$C000	49152	−16384	Écrire, 3
	Lire le comm. 80STORE	$C018	49176	−16360	Lire
VBL	Lire la suppression verticale	$C019	49177	−16359	Lire, 4

Adressage direct des pages d'affichage

Avant de vous décider à utiliser les pages d'affichage directement, prenez en considération l'alternative. La plupart des langages évolués vous permettent d'écrire des instructions qui contrôlent les affichages de texte et de graphique. De la même manière, si vous programmez en langage assembleur, vous pouvez utilisez les fonctions d'affichages des sous-programmes d'E/S de base. Vous ne devriez enregistrer directement dans les mémoires réservées à l'affichage sur écran que si les programmes existants ne peuvent pas satisfaire à vos besoins.

Les cartes des mémoires d'affichage se trouvent aux figures 2-5, 2-6, 2-7 et 2-8. Tous les modes différents d'affichage utilisent le même schéma de base d'adressage : les octets de caractères ou de graphiques sont mémorisés en rangées de 40 octets contigus, mais les rangées elles-mêmes ne sont pas enregistrées à des adresses correspondant à leur localisation sur l'écran. En fait, l'adresse d'affichage est transformée de telle sorte que trois rangées qui sont

distantes de huit lignes sur l'écran, soient regroupées et mémorisées dans les 120 premières adresses de chaque bloc de 128 octets ($80 en héxadécimal). En repliant ainsi en mémoire les données à afficher sur écran, l'Apple //e, comme l'Apple II, enregistre dans un 1K octets de mémoire tous les 960 caractères d'un texte affiché. Pour avoir une description complète de la façon dont l'Apple //e gère sa mémoire d'affichage, se reporter au paragraphe « Adressage des mémoires d'affichage » dans le Chapitre 7.

L'affichage des graphiques en haute-résolution est mémorisé d'une façon fort semblable au texte, mais il y a 8 fois plus d'octets à enregistrer, puisque 8 rangées de points occupent le même espace sur l'écran qu'une seule rangée de caractères. Le sous-ensemble,constitué de toutes les premières rangées des groupes de huit, est mémorisé dans les 1 024 premiers octets de la page de graphique en haute-résolution. Le sous-ensemble, constitué de toutes les secondes rangées des groupes de huit, est mémorisé dans les 1 024 octets suivants, et ainsi de suite pour un total de 8 fois 1 024 ou 8 192 octets. En d'autres termes, chaque bloc de 1 024 octets dans la page d'affichage en haute-résolution contient une des rangées de points de chacun des groupes de huit rangées. Les rangées individuelles sont mémorisées en ensembles de trois rangées de quarante octets, de la même manière que pour l'affichage de texte.

Tous les modes d'affichage sauf le mode en 80 colonnes, peuvent se servir de l'une ou l'autre des deux pages d'affichage. Les cartes pour l'affichage ne donnent que les adresses pour la Page 1. Pour obtenir des adresses en texte et graphique basse-résolution sur la Page 2, ajouter 1024 ($400) ; pour obtenir les adresses de la Page 2 en haute-résolution, ajouter 8192 ($2000).

L'affichage sur 80 colonnes fonctionne un peu différemment. La moitié des données est enregistrée dans la mémoire Page Texte 1, et l'autre moitié est enregistrée dans la mémoire de la carte de texte en 80 colonnes en utilisant les mêmes adresses. Les circuits d'affichage recherchent les octets simultanément dans ces deux zones de mémoire et les affichent séquentiellement : en premier l'octet de la mémoire de la carte de texte en 80 colonnes, puis l'octet de la mémoire centrale. La mémoire centrale enregistre les caractères des colonnes paires de l'écran et la mémoire de la carte de texte en 80 colonnes enregistre les caractères des colonnes impaires.

Pour mémoriser des données dans une carte de texte en 80 colonnes, tout d'abord mettez à un le commutateur 80STORE en écrivant à l'adresse 49153 (en héxadécimal $C001 ou complémentaire −16383). Avec 80STORE à un, le commutateur PAGE2 de sélection de pages choisit entre la partie de l'affichage sur 80 colonnes mémorisée dans la Page 1 de la mémoire centrale et la partie mémorisée dans la mémoire de la carte de texte en 80 colonnes. Pour sélectionner la carte de texte en 80 colonnes, mettez à un le commutateur logiciel PAGE2 en lisant ou en écrivant dans l'adresse 49237. Pour plus de détails sur la façon dont les affichages sont générés, se reporter au Chapitre 7.

Dispositifs d'E/S de base

Figure 2-5. Carte d'affichage de texte en 40 colonnes

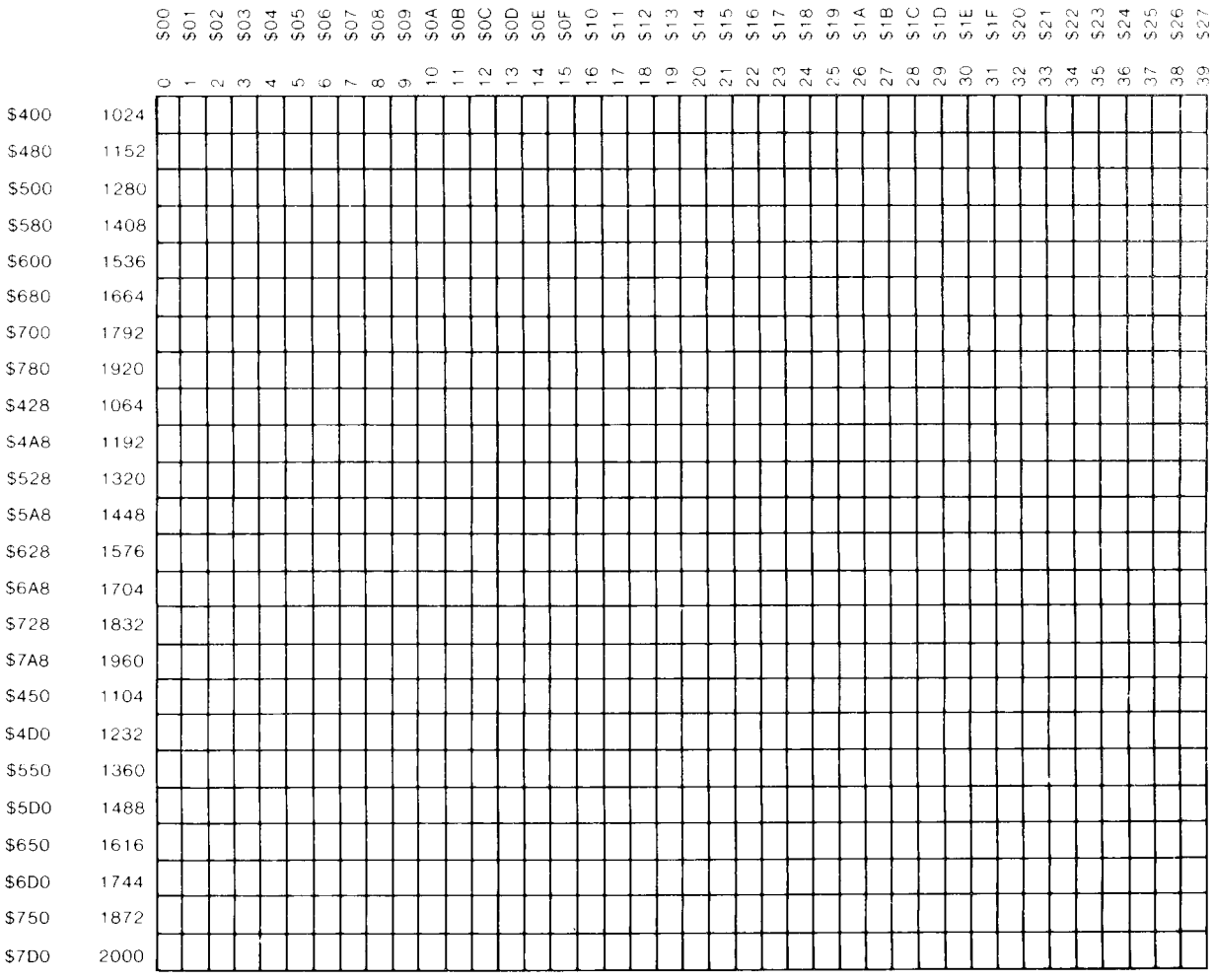


Figure 2-6. Carte d'affichage de texte en 80 colonnes

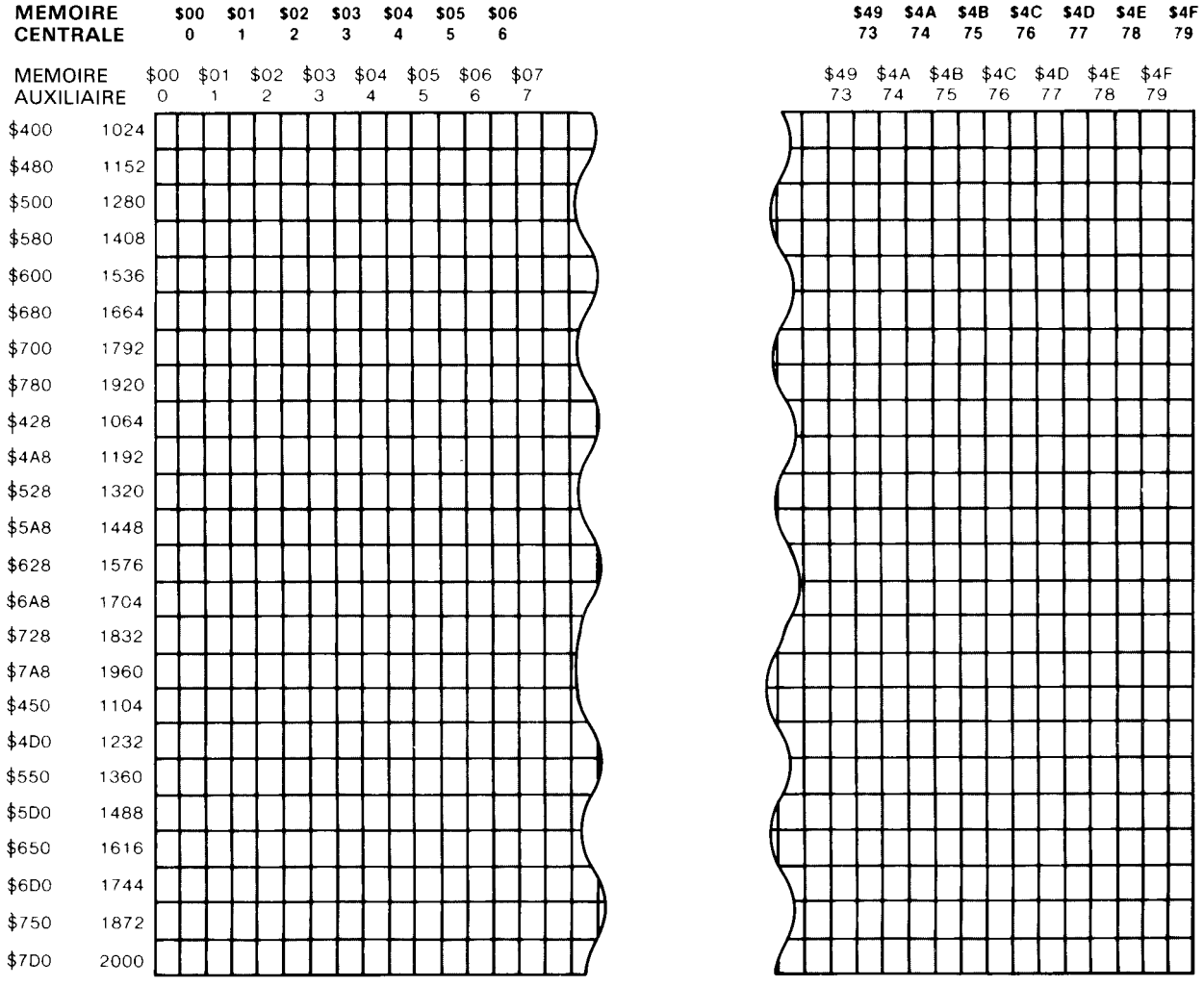


Figure 2-7. Carte d'affichage des graphiques en basse-résolution

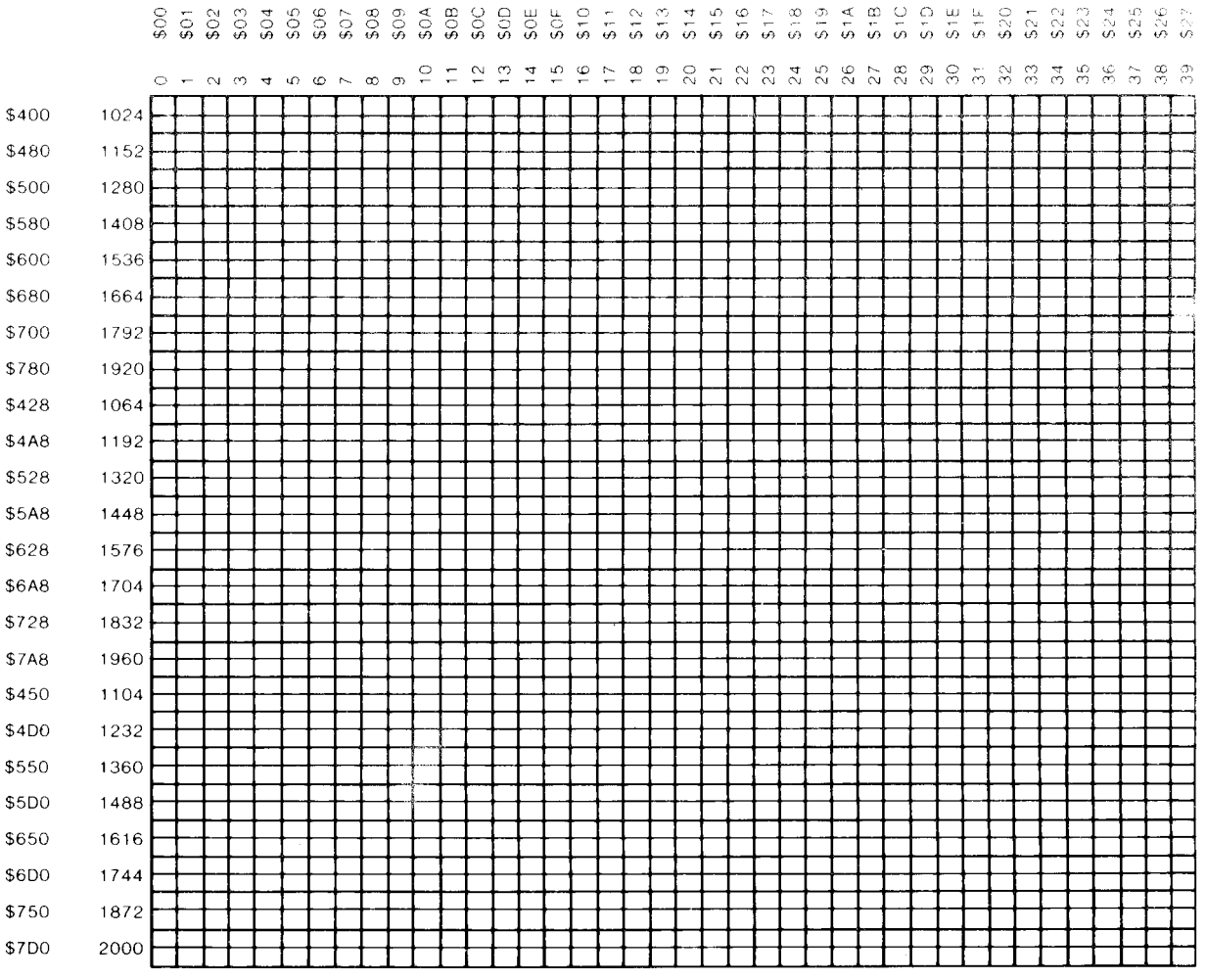
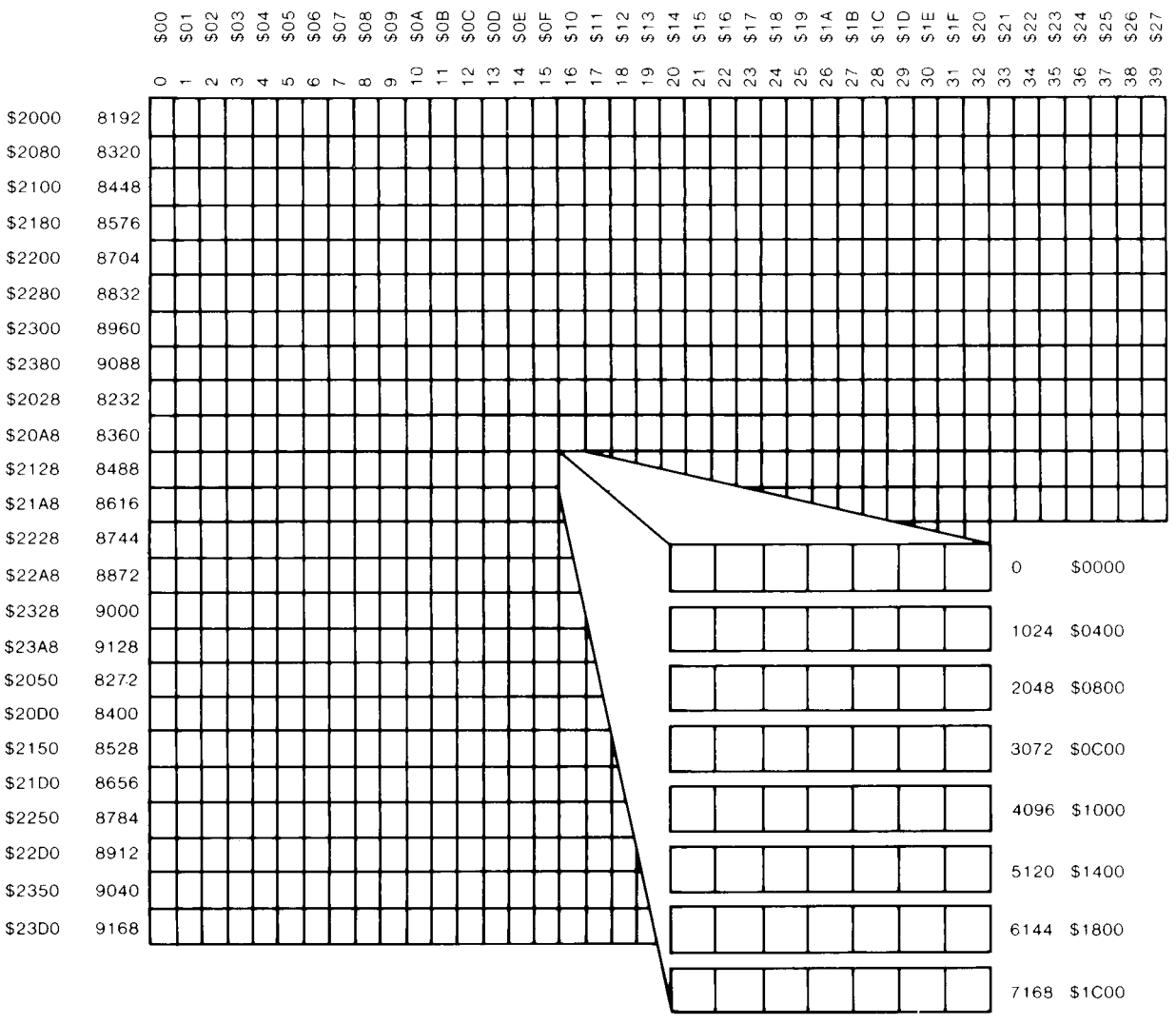


Figure 2-8. Carte d'affichage des graphiques en haute-résolution



Les entrées et les sorties secondaires

En plus des dispositifs primaires d'E/S — le clavier et l'écran — il y a plusieurs dispositifs d'entrée et de sortie dans l'Apple //e. Ces dispositifs sont :

- Le haut-parleur (sortie).
- L'entrée et la sortie sur cassette.
- Les sorties logiques ou annonciateurs.
- La sortie d'échantillonnage.
- Les entrées logiques ou commutateurs.
- Les entrées analogiques ou à commande manuelle.

On opère avec ces dispositifs comme avec les commutateurs logiciels décrits dans le paragraphe précédent : vous les contrôlez en lisant ou en écrivant dans des adresses de mémoire réservées à cet usage. L'action a lieu chaque fois que votre programme lit ou écrit à une de ces adresses ; l'information écrite est ignorée.

Quelques uns de ces dispositifs basculent — changent d'état — à chaque fois qu'on y accède. Si vous écrivez en utilisant une opération avec adressage indexé, le microprocesseur 6502 de l'Apple //e activera le bus d'adresses deux fois durant des cycles d'horloge, donc le dispositif finira par revenir dans son état initial puisqu'il change d'état chaque fois qu'il est adressé. Pour cette raison, vous devez lire, plutôt que d'écrire dans de tels dispositifs.

Le haut-parleur

L'Apple //e a un petit haut-parleur monté vers l'avant de la plaque du bas. Le haut-parleur est connecté à un commutateur logiciel qui bascule : il a deux états, zéro et un, et il passe de l'un à l'autre chaque fois qu'on y accède. Les spécifications électriques du circuit du haut-parleur sont décrites au Chapitre 7.

Si vous commutez le haut-parleur une fois, il émet un petit bruit (« click ») ; pour produire des bruits plus durables, vous répéterez l'accès au haut-parleur. Vous devez toujours utiliser une opération de lecture pour faire basculer le haut-parleur. Si vous écrivez à ce commutateur logiciel, il commute deux fois en succession rapide. L'impulsion résultante est si courte que le haut-parleur n'a pas le temps de répondre ; il n'émet aucun son.

Le commutateur logiciel pour le haut-parleur utilise l'adresse de mémoire 49200 (héxadécimal $C030). Depuis le BASIC Entier, utilisez l'adresse complémentaire —16336. Vous pouvez produire des sons variés et des bruits (« buzzes ») avec le haut-parleur en utilisant des combinaisons de boucles de temporisation dans votre programme. Il y a aussi un sous-programme de base qui émet un signal sonore à travers le haut-parleur. Ce sous-programme est appelé BELL1 ; il est décrit dans l'annexe C.

L'entrée et la sortie sur cassette

Il y a deux petites prises de type phone jack sur le panneau arrière de l'Apple //e. Vous pouvez utiliser une paire de câbles standards avec des fiches de type jack pour brancher un magnétophone à cassette à l'Apple //e et sauvegarder des programmes et des données sur des cassettes audio.

La prise marquée du dessin d'une flèche pointant sur la cassette est la prise de sortie. Elle est connectée à un commutateur logiciel de type bascule comme le commutateur du haut-parleur décrit plus haut. Le signal sur cette prise commute de zéro à 25 millivolts ou de 25 millivolts à zéro chaque fois que vous accédez au commutateur logiciel. Des spécifications électriques détaillées pour l'entrée et la sortie sur cassette sont données au Chapitre 7.

Si vous branchez un câble de cette prise à l'entrée microphone du magnétophone à cassette et si vous mettez le magnétophone en mode enregistrement («rec»), les changements de signal que vous produisez en accédant au commutateur logiciel, seront enregistrés sur la cassette. Le commutateur de sortie sur cassette utilise l'adresse de mémoire 49184 (héxadécimal $C020 ; valeur complémentaire −16352). Comme pour le haut-parleur, cette sortie va basculer deux fois si vous écrivez dessus, donc vous ne devez utiliser que des opérations de lecture pour contrôler la sortie sur cassette.

La méthode standard pour enregistrer des données d'ordinateur sur cassette utilise des tonalités de fréquences différentes pour représenter les états zéro et un. Pour mémoriser des données, vous convertissez les données en une série de bits, et vous convertissez les bits en tonalités. Pour vous épargnez cette angoisse de programmer les tonalités, et pour assurer une homogénéité parmi toutes les cassettes d'Apple //e, il existe un sous-programme du logiciel de base pour produire une sortie de données sur cassette. Ce sous-programme, appelé WRITE, est décrit dans l'annexe C.

La prise marquée du dessin d'une flèche venant d'une cassette est la prise d'entrée. Elle reçoit un câble venant de la prise écouteur («earphone») du magnétophone à cassette. Le signal de la cassette est un signal audio de 1 volt peak-to-peak. Chaque fois que la valeur instantanée de ce signal audio passe du positif au négatif, ou vice-versa, l'état du circuit d'entrée de cassette passe de zéro à un ou vice-versa. Vous pouvez lire l'état de ce circuit à la mémoire d'adresse 49248 (en héxadécimal $C060, ou décimal complémentaire −16288).

Quand vous lisez à cette adresse, vous obtenez un octet, mais seul le bit de poids le plus fort (bit 7) est valide. Si vous programmez en langage machine, c'est le bit de signe, donc vous pouvez effectuer un BPL (branchement si positif) ou un BMI (banchement si négatif) immédiatement après avoir lu l'octet. BASIC est trop lent pour gérer les tonalités audio utilisées pour l'enregistrement de données sur cassette, mais vous n'avez pas besoin d'écrire un programme : il y a un sous-programme de base pour lire des données sur cassette. Il s'appelle READ et il est décrit dans l'annexe C.

Les signaux du connecteur des manettes de jeu

Plusieurs entrées sont disponibles sur un connecteur miniature à 9 broches de type D à l'arrière de l'Apple *//e* : trois entrées à un bit, ou entrées logiques, et quatre entrées analogiques. Ces signaux sont aussi disponibles sur un connecteur de CI à seize broches sur la carte-mère, ainsi que quatre sorties à un bit et un échantilloneur de données. Vous avez accès à tous ces signaux depuis vos programmes.

Ordinairement vous branchez une paire de manettes de jeu au connecteur à 9 broches. Les potentiomètres rotatifs utilisent deux entrées analogiques, et les boutons-poussoirs utilisent les deux entrées à un bit. Cependant, vous pouvez aussi utiliser ces entrées et ces sorties à bien d'autres fins. Par exemple, deux entrées analogiques peuvent être utilisées avec un levier de jeu à deux axes («joystick»). Les spécifications électriques complètes de ces entrées et sorties sont fournies au Chapitre 7 ; le tableau 7-18 montre les numéros des broches du connecteur.

Sorties logiques ou annonciateurs

Les quatre sorties à un bit sont appelées annonciateurs. Chaque annonciateur peut être utilisé pour allumer ou éteindre une lampe, fermer ou ouvrir un relais, ou mettre tout dispositif électronique similaire à un ou à zéro. Pour les spécifications électriques de ces sorties logiques, se reporter au Chapitre 7.

Chaque annonciateur est contrôlé par un commutateur logiciel, et chaque commutateur utilise une paire d'adresses de mémoire. Ces adresses de mémoire sont indiquées dans le Tableau 2-12. Tout accès à la première adresse d'une paire met à zéro l'annonciateur correspondant ; un accès à la seconde adresse met à un l'annonciateur. Il n'est pas possible de lire l'état de cet annonciateur.

Table 2-12. Adresses des mémoires des annonciateurs

*les numéros de broches sont ceux du connecteur de CI à 16 broches sur la carte mère

Annonciateur			Adresse	
N°	Broche*	Etat	Décimale	Hex
0	15	zéro	49240 —16296	$C058
		un	49241 —16295	$C059
1	14	zéro	49242 —16294	$C05A
		un	49243 —16293	$C05B
2	13	zéro	49244 —16292	$C05C
		un	49245 —16291	$C05D
3	12	zéro	49246 —16290	$C05E
		un	49247 —16289	$C05F

La sortie d'échantillonnage («strobe»)

La sortie d'échantillonnage est normalement à 5 volts, mais elle retombe à zéro pendant environ une demi-micro seconde chaque fois qu'on accède à son adresse de mémoire réservée. Vous pouvez utiliser ce signal pour commander une fonction telle que le blocage de données («data latching») sur des dispositifs externes. Si vous utilisez ce signal, rappelez-vous que la mémoire est adressée deux fois si vous y écrivez ; si vous n'avez besoin que d'une seule impulsion, utilisez une opération de lecture pour activer l'échantillonneur. L'adresse de mémoire du signal d'échantillonnage est 49216 (héxadécimal $C040 ou complémentaire —16320).

Les entrées logiques ou commutateurs d'entrée

Les trois entrées logiques peuvent être connectées à la sortie d'un autre dispositif électronique ou à un bouton-poussoir. Quand vous lirez un octet d'une de ces mémoires, seul le bit de poids le plus fort — bit 7 — donne une bonne information ; le reste de l'octet est indéfini. Dans un programme en langage-machine, vous pouvez faire un BPL (branchement si positif) ou un BMI (branchement si négatif) sur l'état du bit 7. Dans un programme en BASIC, vous lirez le commutateur avec un PEEK et vous le comparerez avec la valeur 128. Si la valeur est 128 ou plus grande, le commutateur est à un.

Les adresses de mémoire de ces entrées logiques sont 49249 à 49251 (héxadécimal $C061 à $C063 ou complémentaires —16287 à —16285), comme indiqué dans le tableau 2-13. Le commutateur 0

et le commutateur 1 sont en permanence connectés aux touches ⌘ et ⌘ du clavier ; ils sont normalement branchés aux boutons des manettes de jeu. Certains programmes écrits pour les plus anciens modèles de l'Apple II utilisent le troisième commutateur, le commutateur 2, comme moyen de détecter la touche SHIFT ⇧ . Cette technique nécessite une modification matérielle connue sous le nom de modification de la touche SHIFT par un seul fil (« single-wire-key mod »).

Pour réaliser cette modification sur votre Apple //e, il suffit de souder sur le cercle cassé dénommé X6 sur la carte mère. Les premiers modèles d'Apple //e, identifiables par un numéro de série terminant par —A, ont la modification de la touche SHIFT opérationnelle ; vous pouvez la supprimer en ouvrant le circuit en X6. Rappelez-vous qu'il faut éteindre l'alimentation du système avant de changer quelque chose à l'intérieur de l'Apple //e. Rappelez-vous aussi que de tels changements sont à votre propre risque et peuvent annuler la garantie.

Avertissement
Si vous réalisez la modification de la touche SHIFT et si vous connectez une manette de jeu ou une autre commande manuelle qui utilise le commutateur 2, vous devez faire attention à ne pas mettre le commutateur à un et enfoncer la touche SHIFT en même temps : en faisant cela, on produit un court circuit qui oblige l'alimentation stabilisée à s'éteindre. Quand ceci arrive, tous les programmes et données en mémoire centrale sont perdus.

Les entrées analogiques

Les quatre entrées analogiques sont conçues pour utiliser des potentiomètres ou des résistances variables de 150k ohm. La résistance variable est branchée entre la tension + 5 volt et chaque entrée, de telle sorte qu'elle constitue une partie d'un circuit de temporisation (se reporter au chapitre 7 pour des détails). Le circuit change d'état lorsque la constante de temps s'est écoulée, et la constante de temps varie avec la résistance. Votre programme peut mesurer cette durée en comptant dans une boucle jusqu'à ce que le circuit change d'état ou que le temps soit écoulé.

Avant qu'un programme puisse lire les entrées analogiques, il doit remettre à zéro les circuits de temporisation. L'accès à l'adresse 49264 (hexadécimal $C070 ou complémentaire —16272) le permet. Dès que vous remettez à zéro les circuits de temporisation, les bits de plus fort poids des octets d'adresses 49252 à 49255 (héxadécimal $C064 à $C067 ou complémentaires —16284 à —16281) sont mis à un. Si vous lisez dans ces adresses par un PEEK en BASIC, les valeurs sont égales ou supérieures à 128. Dans les 3 millisecondes suivantes, ces bits vont repasser par zéro — valeurs des octets plus petites que 128 — et y rester jusqu'à une nouvelle remise à zéro des circuits de temporisation. Le temps exact pendant lequel chacun des quatre bits reste à un est directement proportionnel à la résistance branchée sur l'entrée correspondante. Si ces entrées sont

en circuit ouvert — aucune résistance n'est branchée — les bits correspondants peuvent rester indéfiniment à un.

Pour lire les entrées analogiques en langage machine, vous pouvez utiliser une boucle de programme qui remet à zéro les temporisateurs et ensuite incrémente un compteur jusqu'à ce que le bit de l'adresse de mémoire appropriée passe à zéro, ou bien vous pouvez utilisez le sous-programme microprogrammé dans le système. Il s'appelle PREAD, et il est décrit dans l'annexe C. Le BASIC et d'autres langages évolués disposent aussi de moyens adaptés à la lecture des entrées analogiques ; se reporter aux manuels de ces langages.

Résumé des adresses des E/S secondaires

Le tableau 2-13 indique les adresses de mémoire de tous les dispositifs d'E/S de base sauf le clavier et l'écran. Comme indiqué plus haut, certains commutateurs logiciels ne doivent être atteints que par des opérations de lecture ; ces commutateurs sont marqués en notes.

Tableau 2-13. Adresses des mémoires d'E/S secondaires

Pour l'identification du connecteur et des numéros de broches, se reporter aux Tableaux 7-17 et 7-18

Fonction	Adresse Décimale		Hex	Notes
Haut-parleur	49200	−16336	$C030	Lire
Sortie de cassette	49184	−16352	$C020	Lire
Entrée de cassette	49248	−16288	$C060	Lire
Annonciateur 0 à Un	49241	−16295	$C059	
Annonciateur 0 à Zéro	49240	−16296	$C058	
Annonciateur 1 à Un	49243	−16293	$C05B	
Annonciateur 1 à Zéro	49242	−16294	$C05A	
Annonciateur 2 à Un	49245	−16291	$C05D	
Annonciateur 2 à Zéro	49244	−16292	$C05C	
Annonciateur 3 à Un	49247	−16289	$C05F	
Annonciateur 3 à Zéro	49246	−16290	$C05E	
Sortie d'échantillonnage	49216	−16320	$C040	Lire
Commutateur d'entrée 0 (touche ⓐ)	49249	−16287	$C061	Lire
Commutateur d'entrée 1 (touche ⓐ)	49250	−16286	$C062	Lire
Commutateur d'entrée 2	49251	−16285	$C063	Lire
Remise à zéro sur les entrées analogiques	49264	−16272	$C070	
Entrée analogique 0	49252	−16284	$C064	Lire
Entrée analogique 1	49253	−16283	$C065	Lire
Entrée analogique 2	49254	−16282	$C066	Lire
Entrée analogique 3	49255	−16281	$C067	Lire

Sous-programmes d'E/S de base

Sous-programmes d'E/S de base

Le **Moniteur**, ou Moniteur du Système, est un programme d'ordinateur qui est utilisé pour faire fonctionner l'ordinateur au niveau de la machine.

Presque tous les programmes sur l'Apple *//e* se servent du clavier pour entrer des données et envoient des sorties à afficher sur écran. Le *Moniteur* et les BASIC Applesoft et Entier le font par des sous-programmes d'E/S standards qui sont figés dans la mémoire morte contenant les sous-programmes de base de l'Apple *//e*. De nombreux programmes d'application utilisent aussi les sous-programmes d'E/S standards, mais les programmes écrits en Pascal ne le font pas ; Pascal a ses propres sous-programmes d'E/S.

Ce chapitre décrit les caractéristiques de ces sous-programmes,telles qu'elles sont utilisées par le Monitor et par les interpreteurs BASIC, et vous indique comment utiliser ces sous-programmes standards dans vos programmes en langage assembleur.

Les langages évolués contiennent déjà des méthodes adaptées pour gérer la plupart des fonctions décrites dans ce chapitre. Vous n'avez pas besoin d'utiliser les sous-programmes E/S standards dans vos programmes sauf si vous programmez en langage assembleur.

Tableau 3-1. Sous-programmes d'E/S standards

Nom du sous-programme	Adresse	Description
COUT	$FDED	Sortie de caractère : envoie un caractère sur l'écran
RDKEY	$FD0C	Lit une touche : affiche le curseur clignotant ; va au sous-programme standard d'entrée, normalement KEYIN.
KEYIN	$FD1B	Entrée de touche : avec les sous-programmes pour 80 colonnes, affiche un curseur de type damier. Accepte un caractère du clavier.
GETLN	$FD6A	Entrée d'une ligne : affiche le caractère de sollicitation ; accepte une séquence de caractères au moyen de RDKEY.

Les sous-programmes d'E/S standards énumérés dans le tableau 3-1 sont complètement décrits dans ce chapitre. La mémoire morte d'Apple //e contient aussi de nombreux autres sous-programmes que vous pourriez trouver utiles. Ces sous-programmes sont décrits à l'Annexe C. Deux de ces sous-programmes du système, AUXMOVE et XFER, pourront vous aider à utiliser la mémoire auxiliaire en option ; ces sous-programmes sont décrits au Chapitre 4.

Utilisation des sous-programmes d'E/S

Avant d'utiliser les sous-programmes d'E/S standards, il vaudrait mieux que vous compreniez un peu comment ils sont utilisés. Les sous-programmes implantés dans l'Apple //e fonctionnent différemment avec différentes options comme la carte de texte en 80 colonnes. Ce paragraphe décrit des situations générales qui affectent l'opération des sous-programmes d'E/S standards. Des exemples spécifiques sont décrits dans les paragraphes réservés aux sous-programmes considérés individuellement.

Compatibilité avec l'Apple II

Comparé aux anciens modèles Apple II, l'Apple //e a des caractéristiques de clavier et d'écran supplémentaires. Pour faire exécuter des programmes écrits pour les anciens modèles, vous pouvez rendre l'Apple //e semblable à un Apple II Plus en supprimant ces caractéristiques. Les caractéristiques que vous pouvez supprimer ou non pour mettre l'Apple //e en mode Apple II ou non sont énumérées dans le tableau 3-2.

Table 3-2. Mode Apple II

	Apple *//e*	**Mode Apple II**
Clavier :	Majuscules et minuscules	Majuscules seules
Caractères affichables :	Inverse et normal seuls	Clignotant, inverse, et normal
Format d'affichage :	40 colonnes ; et avec une carte en option 80 col.	40 colonnes seulement

Si l'Apple //e n'a pas de carte de texte en 80 colonnes installée dans le connecteur auxiliaire, il est presque en mode Apple //e dès que vous l'allumez ou que vous le réinitialisez. Le clavier est l'exception, puisqu'il est à la fois majuscule et minuscule. Pour être compatible avec d'anciens logiciels, il faut mettre le clavier de l'Apple //e en mode majuscule en enfonçant la touche ⇩ .

Les instructions des BASIC Applesoft et Entier doivent être tapées en lettre majuscules. La touche ⇩ en prend soin, mais cela rend peu pratique l'utilisation des lettres minuscules dans les instructions PRINT. Si les sous-programmes de gestion des 80 colonnes (voir plus loin) sont en ligne, vous pouvez utiliser le mode majuscule restreint qui force les lettres à être majuscules sauf à l'intérieur des guillemets (voir tableau 3-6).

Une autre caractéristique qui est différente sur l'Apple //e est l'ensemble des caractères affichés. Les anciens Apple II n'affichent que les lettres majuscules mais de trois façons : en normal, en inverse, et en clignotant. L'Apple //e peut afficher les lettres majuscules de ces trois façons mais les lettres minuscules seulement en affichage normal. Cette combinaison s'appelle l'*ensemble PRIMAIRE de caractères.* Quand l'Apple //e est allumé pour la première fois ou réinitialisé, il affiche l'ensemble primaire de caractères.

L'Apple //e a un autre ensemble de caractères appelé l'*ensemble ALTERNATIF de caractères,* qui affichent un ensemble complet de caractères majuscules et minuscules en normal et en inverse, mais ne peut afficher de caractères clignotants. Les ensembles de caractères primaire et alternatif sont décrits au chapitre 2. Vous pouvez changer d'ensembles de caractères à tout moment au moyen du commutateur logiciel ALTCHARSET, décrit aussi au chapitre 2.

Les sous-programmes pour 80 colonnes

Il y a quelques caractéristiques qui ne sont normalement disponibles qu'avec l'affichage optionnel sur 80 colonnes. Ces caractéristiques sont données dans les tableaux 3-3a, 3-3b et le tableau 3-6. Les sous-programmes qui gèrent ces caractéristiques sont déjà dans l'Apple //e, mais ils ne sont en fonction que si la carte de texte en 80 colonnes est installée dans le connecteur auxiliaire.

Quand vous allumez l'Apple //e ou que vous le réinitialisez, les sous-programmes pour 80 colonnes ne sont pas en fonction et l'Apple //e affiche l'ensemble primaire de caractères, même si la carte de texte en 80 colonnes est installée. Quand vous activez les sous-programmes pour 80 colonnes, comme indiqué plus loin, l'Apple //e passe sur l'ensemble alternatif de caractères.

Les sous-programmes pour 80 colonnes en place dans le système sont implémentés comme si ils étaient installés dans le connecteur d'extension N° 3. Les programmes écrits pour les anciens AppleII avec des cartes d'affichage en 80 colonnes installées sur le connecteur N° 3 marcheront correctement sur un Apple //e avec une carte de texte en 80 colonnes.

Si l'Apple //e a une carte de texte en 80 colonnes et que vous vouliez utiliser l'affichage de texte en 80 colonnes, vous pouvez mettre en fonction les sous-programmes depuis le BASIC en tapant

PR≠3

Pour mettre en fonction les sous-programmes pour 80 colonnes depuis le moniteur, tapez 3 et appuyez sur (Ctrl)-P. Remarquez que c'est la même procédure que vous utilisez pour activez une carte dans le connecteur d'extension 3. Toute carte auxiliaire installée dans le connecteur auxiliaire a priorité sur une carte installée dans le connecteur d'extension N° 3 : voir le paragraphe « Commutation des mémoires d'E/S » dans le chapitre 6 pour plus de détails.

• Bien que vous ayez activé les sous-programmes pour 80 colonnes en tapant PR≠3, vous ne devez jamais les désactiver en tapant PR≠0, parce que cela ne déconnecte que les sous-programmes, en laissant plusieurs commutateurs logiciels toujours à un pour le fonctionnement en 80 colonnes. A la place, tapez la séquence (Esc)(Ctrl)-Q (voir le tableau 3-6).

Si il n'y a pas de carte de texte en 80 colonnes dans votre Apple //e, vous pouvez toujours activer les sous-programmes pour 80 colonnes et les utiliser avec un format en 40 colonnes. D'abord, mettez à un le commutateur logiciel INTC3ROM situé en $C00A (49192) ; ce commutateur est décrit au chapitre 6 dans le paragraphe « Commutation des mémoire d'E/S ». Puis tapez PR≠3 pour transférer le contrôle aux sous-programmes.

Quand les sous-programmes pour 80 colonnes sont en fonction,sans qu'il y ait une carte dans le connecteur auxiliaire, cela ne marche pas tout-à-fait de la même façon qu'avec une carte. Les fonctions qui effacent l'écran (CLREOL, CLEOLZ, CLREOP et HOME) fonctionnent comme si les sous-programmes étaient hors-fonction ; elles effacent toujours en noir, même en mode inverse. De plus, les interruptions sont verrouillées pendant des opérations longues comme l'effacement d'écran. Alors qu'avec une carte installée, les sous-programmes autorisent périodiquement des interruptions pendant ces longues opérations.

Avertissement

Si vous n'avez pas de carte de texte en 80 colonnes dans le connecteur auxiliaire, ni de carte de type terminal dans le connecteur N° 3, n'essayez pas d'activer les sous-programmes en tapant simplement PR≠3. En tapant PR≠3 si aucune carte n'est installée, vous transférez le contrôle vers un connecteur vide avec des résultats imprévisibles.

Les programmes activent les sous-programmes pour 80 colonnes en transférant le contrôle à l'adresse $C300. S'il n'y a pas de carte dans le connecteur auxiliaire, vous devez mettre à un le commutateur logiciel INTC3ROM en premier lieu. Pour désactiver les sous-programmes pour 80 colonnes depuis un programme, écrivez un caractère (Ctrl)-U via le sous-programme COUT.

Sous-programmes d'E/S de base

L'ancien moniteur

Les anciens modèles Apple II et Apple II Plus contenaient une version différente du moniteur système. Il contenait les sous-programmes standards d'E/S, mais quelques unes des caractéristiques étaient différentes ; par exemple, il n'y avait pas de touches flèches pour le déplacement du curseur. Quand vous démarrez l'Apple //e avec une disquette DOS ou BASIC et que le BASIC Entier est chargé dans la zone de mémoire vive à banc-commuté, l'ancien moniteur est chargé en même temps (il est appelé quelquefois Autostart). Quand vous tapez INT depuis l'Applesoft pour activer le BASIC entier, vous activez aussi cette copie de l'ancien moniteur, qui reste actif jusqu'à ce que vous tapiez FP pour revenir en Applesoft, qui lui utilise le nouveau moniteur en mémoire morte, ou bien si vous tapez

PR≠3

pour mettre en fonction les sous-programmes pour 80 colonnes. Une partie de la procédure d'initialisation implantée en mémoire morte vérifie quelle version du moniteur est en MEV. Si elle trouve l'ancien moniteur, elle le remplace par une copie du nouveau moniteur contenu en MEM. Après avoir été recopié en MEV par la procédure, le nouveau moniteur y reste jusqu'à ce que vous réinitialisez le système.

Les liaisons d'E/S standards

Quand vous appelez un des sous-programmes d'E/S (COUT et RDKEY), la première chose qui se produit est un saut indirect à une adresse mémorisée dans la mémoire programmable. Les adresses de mémoire utilisées pour transférer le contrôle aux autres sous-programmes sont quelquefois appelées vecteurs ; dans ce manuel, les adresses utilisées pour transférer le contrôle aux sous-programmes d'E/S sont appelées les *liaisons d'E/S.* Dans l'Apple //e fonctionnant sans le système d'exploitation de disquettes (DOS), chaque liaison d'E/S est normalement l'adresse du corps du sous-programme (COUT 1 ou KEYIN). Si un système d'exploitation des disquettes est en ligne, une ou deux de ces liaisons contiennent les adresses des sous-programmes correspondants d'E/S du DOS. (Le DOS maintient ses propres liaisons aux sous-programmes d'E/S standards).

En appelant les sous-programmes d'E/S qui sautent aux adresses de liaison, au lieu d'appeler directement les sous-programmes standards, vous êtes sûr que votre programme fonctionnera correctement en conjonction avec un autre logiciel, tel que le DOS ou un contrôleur d'imprimante, qui change une ou deux des deux liaisons d'E/S. Dans le cadre de ce chapitre, nous supposerons que les liaisons d'E/S contiennent les adresses des sous-programmes d'E/S standards COUT1 et KEYIN. Pour de plus amples informations sur les liaisons d'E/S, voir le paragraphe « Changer les liaisons d'E/S » du chapitre 6.

Les caractéristiques de la fonction de sortie standard

Le sous-programme de sortie standard s'appelle COUT, à prononcer C-out, qui veut dire sortie d'un caractère. COUT appelle COUT1, qui envoie un caractère sur l'écran, avance le curseur d'une position, et fait défiler l'écran si nécessaire. COUT1 restreint son usage de l'écran à une zone active dénommée fenêtre de texte, décrite plus loin.

Le sous-programme de sortie COUT

Votre programme fait un appel de sous-programme à COUT à l'adresse $FDED avec un caractère dans l'accumulateur. COUT passe alors le contrôle à la sortie courante par la liaison de sortie CSW. La sortie courante, normalement COUT1, prend le

Tableau 3-3a. Les caractères de contrôle avec COUT1.

(1) N'est disponible que si les sous-programmes pour 80 colonnes sont en fonction.
(2) Ne fonctionne que depuis le clavier.
(3) Ne fonctionne pas depuis le clavier.

Caractère de contrôle	Nom ASCII	Nom Apple *//e*	Action prise par COUT1	Notes
Ctrl-G	(BEL)	bell	Produit une tonalité de 1000 Hz pendant 0,1 seconde	
Ctrl-H	(BS)	backspace	Déplace le curseur d'une position vers la gauche ; du bord gauche de la fenêtre, déplace vers l'extrême droite de la ligne du dessus.	
Ctrl-J	(LF)	line feed	Descend le curseur d'une ligne en le laissant dans la fenêtre ; fait défiler vers le haut si nécessaire.	
Ctrl-K	(VT)	clear EOS	Efface de la position du curseur 1 jusqu'au bas de la fenêtre.	1
Ctrl-L	(FF)	clear	Remonte le curseur au coin en haut et à gauche de la fenêtre et efface la fenêtre.	1
Ctrl-M	(CR)	return	Descend le curseur au début de la ligne suivante dans la fenêtre ; fait défiler si nécessaire.	
Ctrl-N	(SO)	normal	Met l'affichage en mode normal.	1,3
Ctrl-O	(SI)	inverse	Met l'affichage en mode inverse.	1,3
Ctrl-Q	(DC1)	40 col.	Met l'affichage en 40 colonnes	1
Ctrl-R	(DC2)	80 col.	Met l'affichage en 80 colonnes	1

caractère dans l'accumulateur et l'affiche. Si l'accumulateur contient un caractère de contrôle associé à une lettre majuscule ou minuscule, à un nombre, ou à un caractère spécial, COUT1 effectue ou bien une des fonctions spéciales décrites ci-dessus ou bien ignore le caractère.

Chaque fois que vous envoyez un caractère à COUT1, il affiche le caractère à la position courante du curseur, en remplaçant ce qu'il y avait, puis avance le curseur d'une position vers la droite. Si la position du curseur est déjà le bord droit de la fenêtre, COUT1 le déplace à la position la plus à gauche de la ligne suivante. Si ceci doit déplacer le curseur au-delà de la dernière ligne de la fenêtre, COUT1 fait défiler l'écran d'une ligne vers le haut et met le curseur à l'extrême gauche de la nouvelle ligne du bas.

Tableau 3-3b. Les caractères de contrôle avec COUT1, suite.

(1) N'est disponible que si les sous-programmes pour 80 colonnes sont en fonction.
(2) gotoXY n'est pas implémenté en BASIC : voir le *Manuel de référence du système d'exploitation en Pascal sur Apple IIe*.

Caractère de contrôle	Nom ASCII	Nom Apple *IIe*	Action prise par COUT1	Notes
Ctrl-S	(DS3)	stop-list	Arrête d'envoyer des caractères sur l'écran, jusqu'à ce qu'une touche soit enfoncée.	
Ctrl-U	(NAK)	quit	Met les sous-programmes pour 80 colonnes hors-fonction, remonte le curseur en haut et à gauche et efface l'écran.	1
Ctrl-V	(SYN)	scroll	Fait descendre l'affichage d'une ligne, en laissant le curseur dans sa position courante.	1
Ctrl-W	(ETB)	scroll-up	Fait défiler l'affichage d'une ligne vers le haut en laissant le curseur dans sa position courante.	1
Ctrl-Y	(EM)	home	Met le curseur dans le coin en haut et à gauche de la fenêtre (mais n'efface pas).	1
Ctrl-Z	(SUB)	clear line	Efface la ligne sur laquelle se trouve le curseur.	1
Ctrl-\	(FS)	fwd. space	Déplace le curseur d'une position vers la droite ; du bord droit de la fenêtre, le déplace à l'extrême gauche de la ligne du dessous.	1
Ctrl-]	(GS)	clear EOL	Efface la ligne depuis la position du curseur jusqu'au bord droit de la fenêtre	1
Ctrl-^	(RS)	gotoXY	En utilisant les deux caractères moins 32, comme valeurs de X et Y sur un octet, déplace le curseur en CH = X, CV = Y.	1,2

Les caractéristiques de la fonction de sortie standard

La position du curseur est contrôlée par les valeurs contenues dans les adresses de mémoire 36 et 37 (hexadécimal $24 et $25). Ces adresses sont dénommées CH, pour curseur et horizontal, et CV, pour curseur et vertical. COUT1 n'affiche pas un curseur, mais les sous-programmes d'entrée décrits ci-dessous le font et ils utilisent la position du curseur. Si un autre sous-programme affiche le curseur, il ne le mettra pas nécessairement dans la position de curseur utilisée par COUT1.

Les caractères de contrôle avec COUT1

COUT1 n'affiche pas les caractères de contrôle. De fait, les caractères de contrôle dont la liste figure dans les tableaux 3-3a et 3-3b sont utilisés pour que les sous-programmes provoquent une certaine action. Les autres caractères de contrôle sont ignorés. La plupart des fonctions énumérées peuvent être aussi produites depuis le clavier, soit en tapant le caractère de contrôle de la liste soit en utilisant le code d'évasion approprié, comme cela est décrit dans le paragraphe « Codes d'évasion avec KEYIN ». La fonction de suspension d'affichage, décrite séparemment, ne peut être provoquée qu'à partir du clavier.

La suspension d'affichage

Quand vous utilisez un programme qui affiche du texte par COUT1, vous pouvez ne pas mettre à jour l'affichage en maintenant enfoncée la touche Ctrl tout en appuyant sur la touche S. Chaque fois que COUT1 reçoit un retour chariot du programme, il regarde si vous avez appuyé sur Ctrl-S. Si oui, COUT1 arrête d'afficher et attend que vous tapiez sur une autre touche. Une touche étant appuyée, COUT1 va alors envoyer le retour chariot qu'il avait reçu juste avant et continuera normalement. Le code du caractère de la touche sur laquelle vous avez appuyé pour interrompre l'affichage est ignoré sauf s'il s'agit de Ctrl-C. COUT1 renvoie Ctrl-C au programme ; si c'est un programme en BASIC, ceci vous permet d'arrêter l'exécution du programme durant le mode de suspension d'affichage.

La fenêtre d'écran

Après avoir démarré l'ordinateur ou après une réinitialisation, COUT1 utilise tout l'écran. Cependant, vous pouvez restreindre l'activité de COUT1 à une portion rectangulaire de l'écran que vous désirez. La partie active de l'écran est appelée la *fenêtre de texte.* COUT1 n'affiche les caractères que dans la fenêtre ; quand il atteint la fin de la dernière ligne de la fenêtre, il ne fait défiler que le contenu de la fenêtre.

Vous pouvez fixer le haut, le bas, le côté gauche et la largeur de la fenêtre de texte en mémorisant les valeurs appropriées dans quatre adresses de mémoire. Ceci permet à vos programmes de contrôler la place du texte dans l'écran et de protéger les autres portions de l'écran contre un nouveau texte qui pourrait les modifier à tort.

La position de mémoire d'adresse 32 (en hexadécimal $20) contient le numéro de la colonne la plus à gauche de la fenêtre de texte. Ce numéro est normalement égal à 0, le numéro de la colonne la plus à gauche dans l'écran. Dans un affichage en 40 colonnes, la valeur maximale de ce numéro est 39 (hexadécimal $27) ; dans l'affichage à 80 colonnes, la valeur maximale est 79 (hexadécimal $4F).

La position de mémoire d'adresse 33 (en hexadécimal $21) contient la largeur de la fenêtre de texte. Pour un affichage sur 40 colonnes, elle est normalement égale à 40 (hexadécimal $28) ; pour un affichage en 80 colonnes, elle vaut normalement 80 (hexadécimal $50). COUT1 arrondit la largeur à une valeur paire.

Avertissement

Soyez prudent en ne laissant pas la somme de la largeur de la fenêtre et de la position la plus à gauche dans la fenêtre dépasser la largeur de l'écran que vous utilisez (40 ou 80). Si cela se produisait, il est possible que COUT1 mette des caractères dans des adresses de mémoire en dehors de la page d'affichage, détruisant probablement des programmes ou des données.

La position de mémoire d'adresse 34 (en hexadécimal $22) contient le numéro de la ligne du haut de la fenêtre d'écran. Il est égal normalement à 0, la ligne la plus haute de l'écran. Sa valeur maximale est 23 (hexadécimal $17).

La position de mémoire d'adresse 35 (en hexadécimal $23) contient le numéro de la ligne du bas de l'écran, plus 1. Il est égal normalement à 24 (hexadécimal $18) pour la ligne du bas de l'affichage. Sa valeur minimale est 1.

Avertissement

Chaque fois que vous changerez les limites de la fenêtre d'écran, vous devrez vous assurer que la position courante du curseur (mémorisée en CH et CV) soit située à l'intérieur de la nouvelle fenêtre. Si elle est en dehors, il est possible que COUT1 mette des caractères dans des adresses de mémoire en dehors de la page d'affichage, détruisant probablement des programmes ou des données.

Tableau 3-4. Les adresses de mémoire de la fenêtre de texte.

Paramètre de la fenêtre	Adresse		Valeurs minimales		Valeurs normales : 40 col.		Valeurs normales : 80 col.		Valeurs maximales : 40 col.		Valeurs maximales : 80 col.	
	Dec	Hex	Dec	Hex	Dec	Hex	Dec	Hex	Dec	Hex	Dec	Hex
Bord gauche	32	$20	0	$0	0	$0	0	$0	39	$27	79	$4F
Largeur	33	$21	0	$0	40	$28	80	$50	40	$28	80	$50
Haut	34	$22	0	$0	0	$0	0	$0	23	$17	23	$17
Bas	35	$23	1	$1	24	$18	24	$18	24	$18	24	$18

Les textes en modes inverse et clignotant

Le sous-programme COUT1 peut afficher du texte en mode normal, en mode inverse, avec quelques restrictions, en mode clignotant. Le mode d'affichage d'un caractère quelconque sur l'écran dépend de deux choses : l'ensemble des caractères utilisés à ce moment et les valeurs des deux bits de plus fort poids de l'octet du caractère dans la mémoire d'affichage.

Tout en envoyant vos caractères de texte sur l'écran, COUT1 met les bits de plus fort poids à des valeurs dépendantes de la valeur contenue dans la mémoire d'adresse 50 (hexadécimal $32). Si cette valeur est de 255 (hexadécimal $FF), COUT1 met les caractères à afficher en mode normal ; si la valeur est 63 (hexadécimal $3F), COUT1 affiche les caractères en inverse. Si la valeur est 127 (hexadécimal $7F) et si vous sélectionnez l'ensemble primaire de caractères, les caractères seront affichés clignotants. Remarquez que le mode clignotant n'est pas accessible dans l'ensemble alternatif de caractères.

Pour commander le mode d'affichage des caractères, le sous-programme COUT1 utilise la valeur de la mémoire d'adresse 50 comme un masque logique pour forcer les valeurs des deux bits de poids fort de chaque octet de caractère qu'il met dans la page d'affichage. Il le fait en effectuant un ET logique sur l'octet de donnée et l'octet de masquage. L'octet résultant contient un 0 dans tout bit qui valait 0 dans le masque. La version de COUT1 dans les sous-programmes pour 80 colonnes ne change que le bit de poids fort de la donnée.

Tableau 3-5. Valeurs de contrôle du mode d'affichage des caractères de texte

Note : ces valeurs de masque ne s'appliquent que sur l'ensemble primaire de caractères (voir le texte).

Valeur du masque Dec	Hex	Mode d'affichage
255	$FF	Normal, majuscules et minuscules
127	$7F	Clignotant, majuscules et symboles
63	$3F	Inverse, majuscules et minuscules

Si les sous-programmes pour 80 colonnes ne sont pas en fonction et si vous mettez des zéros dans les bits de poids faible à l'adresse 50, COUT1 va masquer ces bits dans votre texte. Comme résultat, certains caractères seront transformés en d'autres caractères. Vous ne devez donner au masque que les valeurs indiquées au tableau 3-5.

Si vous donnez une valeur de masque de 127 (hexadécimal $7F) à la mémoire d'adresse 50, le bit de poids fort de chaque octet résultant vaudra 0, et les caractères seront affichés, soit en minuscules soit clignotants, suivant l'ensemble de caractères que vous aurez sélectionné. Reportez-vous aux tableaux des ensembles de caractères affichés dans le chapitre 2. Dans l'ensemble primaire de caractères, le second bit de poids le plus fort, le bit 6, sélectionne l'affichage clignotant avec les caractères majuscules. Avec l'ensemble primaire des caractères en ligne, vous pouvez afficher des minuscules en mode normal et des majuscules en normal, inverse et clignotant. Avec l'ensemble alternatif de caractères choisi, vous pouvez afficher des caractères majuscules et minuscules en normal et en inverse. La commutation entre les ensembles de caractères est décrite dans le paragraphe « Commutation des modes d'affichage » dans le chapitre 2.

Les caractéristiques de l'Entrée standard

Les sous-programmes intégrés en mémoire morte dans l'Apple //e comprennent deux sous-programmes différents pour lire le clavier. L'un s'appelle RDKEY, qui veut dire lire une touche. Il appelle le sous-programme standard d'entrée de caractère KEYIN, qui accepte un caractère à la fois, venant du clavier. L'autre s'appelle GETLN, qui veut dire capter une ligne. En faisant des appels répétés à RDKEY, GETLN accepte une séquence de caractères se terminant par un retour chariot. GETLN fournit aussi des fonctions d'édition sur l'écran : voir le paragraphe « Édition avec GETLN ».

Le sous-programme d'entrée RDKEY

Un programme peut capter un caractère du clavier en faisant appel au sous-programme RDKEY à l'adresse $FD0C. RDKEY rend clignotant le caractère situé à la position du curseur, puis passe le contrôle, à travers la liaison d'entrée KSW, au sous-programme d'entrée, qui est normalement KEYIN.

RDKEY affiche le curseur à sa position courante, qui est juste à droite du dernier caractère, que vous avez envoyé sur l'écran, quel qu'il soit, (normalement en utilisant le sous-programme COUT, décrit plus haut). Le curseur affiché par RDKEY est une version clignotante du caractère qui apparaît à cette position sur l'écran. C'est habituellement un caractère « espace », donc le curseur apparaît comme un rectangle clignotant.

La méthode qu'utilise RDKEY pour afficher un curseur est celle utilisée dans les anciens modèles d'Apple II, qui n'affichaient pas de lettres minuscules. Avec les lettres minuscules ou l'ensemble alternatif de caractères, cette méthode d'affichage du curseur n'est plus satisfaisante.

Le sous-programme d'entrée KEYIN

KEYIN est le sous-programme d'entrée standard. Quand il est appelé, il attend que l'utilisateur appuie sur une touche, puis retourne au programme appelant avec le code de la touche dans l'accumulateur.

Le problème de l'affichage d'un curseur sans utiliser le mode clignotant est résolu par KEYIN. Si les sous-programmes pour 80 colonnes ne sont pas en fonction, KEYIN affiche un curseur en mémorisant alternativement un bloc de type damier à l'adresse du curseur, puis en mémorisant le caractère original, puis le damier à nouveau. Si les sous-programmes sont en fonction, KEYIN affiche un espace (rectangle) stable en blanc sur noir, à moins que vous ne soyez en mode d'évasion, pendant lequel est affiché un signe plus (+) en noir sur blanc. (Le mode d'évasion est décrit au paragraphe suivant).

KEYIN génère aussi un nombre aléatoire. Pendant qu'il attend qu'on appuie sur une touche, KEYIN répète l'incrémentation d'un nombre de 16 bits dans les mémoires d'adresses 78 et 79 (hexadécimal $4E et $4F). Ce nombre continue à augmenter de 0 à 65535, puis repart à nouveau de 0. La valeur de ce nombre change si vite qu'il n'y a aucun moyen de prévoir combien elle vaudra après qu'une touche ait été enfoncée. Un programme qui lit des données du clavier peut utiliser cette valeur comme nombre aléatoire ou comme amorce d'un sous-programme de génération d'un nombre pseudo-aléatoire.

Quand on appuie sur une touche, KEYIN accepte le caractère, arrête d'afficher le curseur, et revient au programme appelant avec le caractère dans l'accumulateur.

Les codes d'évasion avec KEYIN

KEYIN a de nombreuses fonctions spéciales que vous pouvez provoquer en tapant des codes d'évasion sur le clavier. Un code d'évasion est obtenu en appuyant sur [Esc] puis en le relachant, et en appuyant alors sur une autre touche, comme le tableau 3-6 le montre. La notation [Esc] suivi d'un espace dans la table veut dire d'appuyer sur la touche [Esc], de la relacher, puis d'appuyer sur le caractère qui suit.

Tableau 3-6. Les codes d'évasion

(1) Touche de contrôle du curseur dans l'ancien style : voir le texte
(2) Touche de contrôle du curseur : voir le texte
(3) Ce code ne fonctionne que si les sous-programmes pour 80 colonnes sont en fonction.

Code d'évasion	Fonction	Notes
[Esc] @	Efface la fenêtre et remonte le curseur dans le coin en haut à gauche	
[Esc] A	Remonte le curseur d'une ligne	1
[Esc] B	Déplace le curseur d'une position à droite	1
[Esc] C	Déplace le curseur d'une position à gauche	1
[Esc] D	Descend le curseur d'une ligne	1
[Esc] E	Efface la fin de la ligne	
[Esc] F	Efface le bas de la fenêtre	
[Esc] I [Esc] ↑	Remonte le curseur d'une ligne et met en mode d'évasion	2
[Esc] J [Esc] ←	Déplace le curseur d'une position à gauche et met en moded'évasion	2
[Esc] K [Esc] →	Déplace le curseur d'une position à droite et met en moded'évasion	2
[Esc] M [Esc] ↓	Descend le curseur d'une ligne et met en mode d'évasion	2
[Esc] R	Entre dans le mode majuscule restreint	3
[Esc] T	Sort du mode majuscule restreint	3
[Esc] 4	Commute en mode à 40 colonnes, met le curseur dans le coin en haut à gauche,et efface l'écran	3
[Esc] 8	Commute en mode à 80 colonnes, met le curseur dans le coin en haut à gauche,et efface l'écran	3
[Esc] [Ctrl] – Q	Met les sous-programmes pour 80 colonnes hors-fonction	3

Le tableau 3-6 comprend trois ensembles de touches de contrôle du curseur. Le premier ensemble est composé de la touche (Esc) suivie de A, B, C, et D. Les touches des lettres peuvent être minuscules ou majuscules. Ces touches sont les touches standards de déplacement du curseur dans les anciens modèles d'Apple II ; elles existent sur l'Apple //e principalement pour la compatibilité avec les programmes écrits pour les anciennes machines.

Le déplacement du curseur en mode d'évasion

Le second et le troisième ensembles de touches de contrôle du curseur sont énumérés ensemble parce qu'ils activent le mode d'évasion. En mode d'évasion, vous pouvez continuer à utiliser les touches de déplacement du curseur sans avoir à appuyer à nouveau sur la touche (Esc). Ceci vous permet de répéter des déplacements de curseur en maintenant enfoncée la touche appropriée.

Quand les sous-programmes pour 80 colonnes sont en fonction, vous pouvez savoir si vous êtes en mode d'évasion : un signe plus (+) est affiché en mode inverse en guide de curseur. Vous quitterez le mode d'évasion en tapant sur n'importe quelle touche sauf les touches de déplacement du curseur.

Les codes d'évasion avec les touches portant les flèches directionnelles sont les touches normales de déplacement du curseur sur l'Apple //e. Les codes d'évasion avec les touches I, J, K, et M sont les touches normales de déplacement du curseur sur l'Apple II Plus, et sont présentes sur l'Apple //e pour assurer une compatibilité avec l'Apple II Plus. Sur l'Apple //, les codes d'évasion avec les touches I, J, K et M fonctionnent avec les lettres majuscules ou minuscules

Le sous-programme d'entrée GETLN

Les programmes ont souvent besoin de chaînes de caractères comme entrée. Bien qu'il soit possible de répéter à RDKEY pour entrer plusieurs caractères à partir du clavier, il existe un sous-programme plus puissant que vous pouvez utiliser. Ce sous-programme s'appelle GETLN, qui veut dire capter une ligne, et il débute à l'adresse $FD6A. En répétant des appels à RDKEY, GETLN accepte les caractères du sous-programme standard d'entrée — habituellement KEYIN — et les met dans la zone tampon d'entrée située dans la page mémoire d'adresses $200 à $2FF. GETLN fournit aussi à l'utilisateur des moyens d'édition et de contrôle sur l'écran, décrits plus bas dans le paragraphe « Édition avec GETLN ».

La première chose que fait GETLN quand vous l'appelez est d'afficher un caractère de sollicitation (le prompting character) appelé simplement *solliciteur* (« prompt »). Le solliciteur indique à l'utilisateur que le programme attend une entrée de données. Des programmes différents utilisent des solliciteurs différents, aidant l'utilisateur à se rappler du programme qui attend des entrées. Par

exemple, une instruction INPUT d'un programme BASIC affiche le point d'interrogation (?) comme solliciteur.

Les caractères de sollicitation utilisés par les différents programmes de l'Apple //e sont donnés dans le tableau 3-7.

GETLN utilise le caractère mémorisé à l'adresse 51 (hexadécimal $33) comme solliciteur. Dans un programme en langage assembleur, vous pouvez changer le solliciteur en n'importe quel autre caractère. En BASIC, le changement du solliciteur n'a aucun effet, parce que les deux interpréteurs BASIC et le moniteur le restaurent chaque fois qu'ils demandent une entrée de la part de l'utilisateur.

Tableau 3-7. Les caractères de sollicitation
* note : le Mini-Assembleur n'est disponible que lorsque le BASIC Entier est en fonction.

Caractère de sollicitation	Programme demandant une entrée
?	Un programme d'utilisateur écrit en BASIC (instruction INPUT)
>	L'interpréteur BASIC Entier
]	L'interpréteur BASIC Applesoft
*	Le Moniteur intégré en mémoire morte
!	Le Mini-Assembleur*

Pendant que l'utilisateur tape la chaîne de caractères, GETLN envoie chaque caractère au sous-programme standard de sortie — normalement COUT1 — qui l'affiche à la position précédente du curseur et met le curseur à la prochaine position disponible sur l'écran, habituellement immédiatement à droite. Au fur et à mesure que le curseur traverse l'écran, il indique la position où le prochain caractère sera affiché.

GETLN enregistre les caractères dans la mémoire-tampon, en commençant à la position de mémoire $200 et en utilisant le registre X pour indexer la zone tampon. GETLN continue à accepter et à afficher des caractères jusqu'à ce qu'on appuie sur la touche ⏎; alors il efface le reste de la ligne sur laquelle se trouvait le curseur, enregistre le code du retour-chariot dans la mémoire-tampon, envoie le code de retour-chariot sur l'écran, et revient au programme qui l'avait appelé.

La longueur maximale de ligne que GETLN peut gérer est de 255 caractères. Si on en tape plus, GETLN envoie une barre oblique arrière (\) et un retour-chariot sur l'écran, annule la ligne qu'il avait acceptée jusqu'alors, et recommence. Pour avertir l'utilisateur que la ligne va devenir pleine, GETLN émet une tonalité à chaque fois qu'une touche est pressée à partir du 248e caractère.

Les caractéristiques d'entrée standard

Dans l'Apple II et l'Apple II Plus, le sous-programme GETLN convertit toutes les entrées en majuscules. GETLN dans l'Apple *//e* ne fait pas cela, même en mode Apple II. Pour avoir des entrées majuscules pour BASIC, utilisez la touche blocage de majuscules ou le commutateur pour obtenir le mode majuscule restreint des lettres grâce à la séquence d'évasion indiquée dans la Table 3-6. Si le mode restreint des lettres est actif, les lettres sont automatiquement transformées en majuscules sauf à l'intérieur des guillemets.

Edition avec GETLN

Le sous-programme GETLN fournit les moyens d'édition standards sur écran utilisés par les interpréteurs BASIC et le programme Moniteur. Pour une introduction à ces moyens d'édition, reportez-vous au manuel *Travaux pratiques en Applesoft*. Tout programme utilisant GETLN pour lire le clavier dispose de ces moyens.

Annuler une ligne

Chaque fois que vous tapez une ligne, le fait d'appuyer sur Ctrl – X conduit GETLN à annuler la ligne. GETLN affiche une barre oblique arrière (/) et envoie un retour-chariot, puis affiche le solliciteur et attend que vous tapiez une nouvelle ligne. GETLN fait la même chose lorsque vous tapez plus de 255 caractères, comme cela a été expliqué ci-dessus.

Reculer

Si vous appuyez sur la touche ←, GETLN déplace le pointeur d'une place en arrière dans la zone-tampon, effaçant effectivement le dernier caractère de sa mémoire-tampon. Il envoie aussi un caractère de recul au sous-programme COUT, qui déplace la position d'affichage et le curseur d'une position vers l'arrière. Si vous tapez alors un autre caractère, il remplacera le caractère sur lequel vous avez reculé, tant sur l'écran que dans la mémoire-tampon. Chaque fois que vous appuyez sur la touche ←, il déplace le curseur vers la gauche et efface un autre caractère, jusqu'à ce que vous atteigniez le début de la ligne. Si alors vous appuyez sur la touche ← une fois encore, vous avez effectivement annulé la ligne, et GETLN enverra un retour-chariot et affichera le solliciteur.

Recopier

La touche → a une fonction complémentaire de celle de la fonction de recul. Si vous appuyez sur la touche →, GETLN va chercher le caractère qui se trouve à la position du curseur comme si ce caractère avait été tapé au clavier. Vous pouvez utiliser cette procédure pour aller rechercher des caractères que vous veniez juste d'effacer en reculant sur eux. Vous pouvez utiliser les fonctions de recul et de recopie avec les fonctions de déplacement du curseur pour modifier des données sur l'écran (voir le paragraphe précédent « Déplacement du curseur en mode d'évasion »).

Sous-programmes d'E/S de base

Organisation de la mémoire

Organisation de la mémoire

Le microprocesseur 6502 de l'Apple //e peut adresser 65536 (64K) positions de mémoire (K veut dire 1024 ; voir le glossaire). L'ensemble de la mémoire programmable de l'Apple //e (MEV, pour mémoire vive), de la mémoire en lecture seulement (MEM, pour mémoire morte) et des dispositifs d'entrée et de sortie ont des positions allouées dans cet espace de 64K d'adresses. Plusieurs fonctions partagent les mêmes adresses — mais pas en même temps —. Pour avoir des informations sur ces espaces d'adresses partagées, voir le paragraphe « Mémoire à banc-commuté » de ce chapitre et les paragraphes « Autres utilisations de l'espace de mémoire d'E/S » et « Espace d'expansion de mémoire morte » dans le chapitre 6.

Toutes les entrées et sorties de l'Apple //e ont des adresses projetées en mémoire (« memory mapped »). Dans ce chapitre, les espaces de mémoire d'E/S sont décrits simplement comme des blocs de mémoire. Pour des détails sur les caractéristiques des E/S de base, se reporter aux descriptions des chapitres 2 et 3. Pour s'informer sur les opérations d'E/S avec des cartes périphériques, se reporter au chapitre 6.

On fait souvent référence à la mémoire de l'Apple //e par des blocs de 256 octets appelés pages. Une raison à cela est qu'un compteur d'adresse sur un octet ou un registre d'index peut faire référence à l'une des 256 différentes adresses. Donc, la page 0 comprend les adresses de mémoire de 0 à 255 (héxadécimal $0 à $FF), inclus. La page 1 comprend les positions d'adresses 256 à 511 (héxadécimal $100 à $1FF — remarquez que le numéro de page est la partie de poids fort de l'adresse héxadécimale). Ne confondez pas ce type de page avec les zones-tampons d'affichage sur écran dans l'Apple //e, que l'on appelle quelquefois la page 1 et la page 2.

Carte de la mémoire principale

La carte de l'espace d'adresses de la mémoire principale en figure 4-1 montre les fonctions des zones les plus importantes de la mémoire. Pour plus de détails sur l'espace des E/S de 48K à 52K

($C000 à $CFFF), se reporter au chapitre 2 et au chapitre 6 ; la mémoire à banc-commuté dans l'espace de mémoire de 52K à 64K ($D000 à $FFFF)) est décrite plus loin.

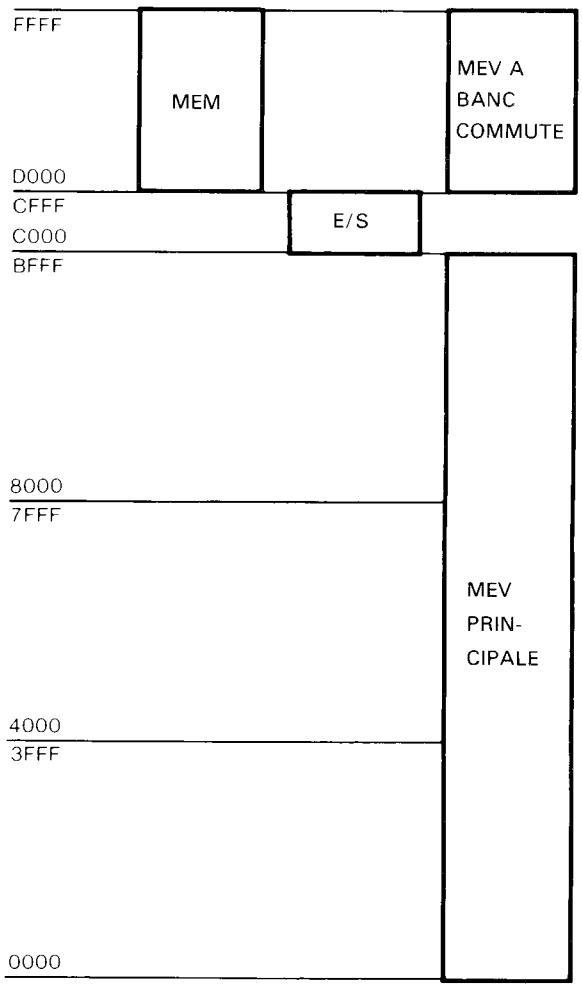
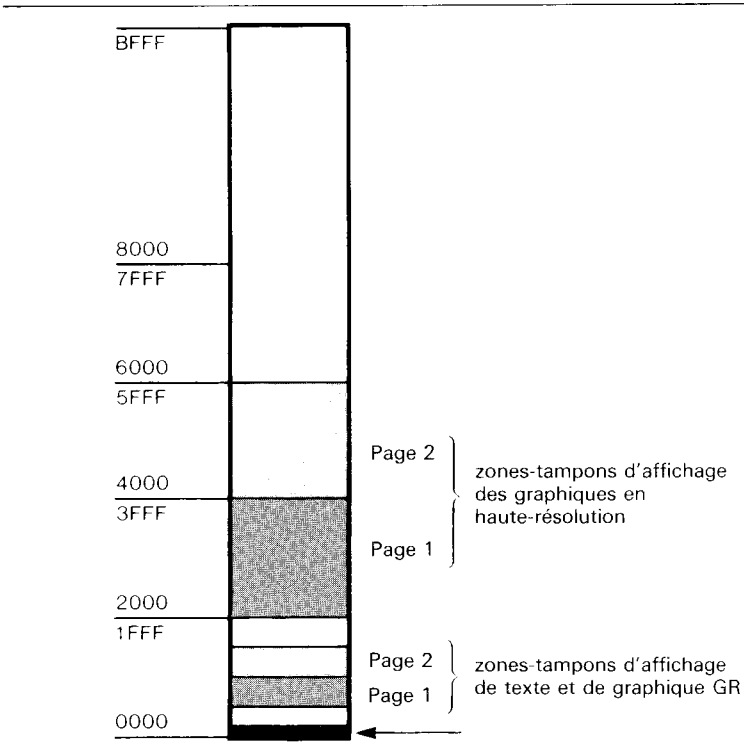


Figure 4-1. Carte de la mémoire du système

Figure 4-2. Carte d'allocation de la MEV



L'allocation de la mémoire vive

Comme le montre la figure 4-1, la plus grande partie de l'espace de mémoire de l'Apple //e est allouée à de la mémoire programmable (MEV). La figure 4-2 montre quelles sont les zones allouées à la MEV. La mémoire vive principale s'étend de l'adresse 0 à l'adresse 49151 (hex $BFFF), et occupe les pages 0 à 191 (héxadécimal $BF). Il y a aussi de la mémoire vive dans l'espace à banc — commuté de 53248 à 65535 (héxadécimal $D000 à $FFFF), décrit dans une partie séparée de ce chapitre, et de la MEV auxiliaire sur la carte de texte en 80 colonnes, ou la carte de texte en 80 colonnes étendue, décrites au chapitre 6.

Les pages de mémoire réservées

La plus grande partie de la MEV de l'Apple //e est disponible pour enregistrer vos programmes et vos données. Pourtant quelques pages de MEV sont réservées à l'usage des sous-programmes du moniteur et des interpréteurs BASIC. Les pages réservées sont décrites plus bas.

Le système ne vous empêche pas d'utiliser ces pages, mais si vous les utilisez, vous devez faire attention à ne pas déranger les données du système que ces pages contiennent, ou sinon le système fonctionnera mal.

La page Zéro

Plusieurs des modes d'adressage du microprocesseur 6502 demandent d'utiliser des adresses de la page zéro, appelées aussi page zéro. Le programme Moniteur, les interpréteurs BASIC, et le DOS se servent abondamment de la page zéro.

Pour se servir de l'adressage indirect dans vos programmes en langage assembleur, vous devez mémoriser les adresses de base en page zéro. Au même moment, vous devez éviter d'interférer avec les autres programmes qui utilisent la page zéro — le programme Moniteur, les interpréteurs BASIC, et le système d'exploitation des disquettes. Une façon d'éviter ces conflits consiste à n'utiliser que les adresses de la page zéro qui ne sont pas déjà utilisées par d'autres programmes. Les tableaux 4-1, 4-2, 4-3 et 4-4 indiquent les adresses de la page zéro utilisées par le Moniteur, le BASIC Applesoft, le BASIC Entier et le DOS 3.3.

Comme vous pouvez le constater sur ces tableaux, la page zéro est largement utilisée, sauf pour quelques octets ici et là. C'est difficile de trouver plus de un ou deux octets qui ne soient pas utilisés soit par BASIC ou le Moniteur ou le DOS. Plutôt que d'essayer de coincer vos données dans un coin inutilisé, il est préférable de choisir une alternative plus sûre : sauvegarder le contenu d'une partie de la page zéro, utiliser cette partie, puis restaurer le contenu précédent avant de passer le contrôle à un autre programme.

La pile du 6502

Le microprocesseur 6502 utilise la page 1 comme pile — l'endroit où sont enregistrées les adresses de retour des sous-programmes, dans l'ordre « first-in, first-out », la première à entrer est la dernière à sortir. De nombreux programmes utilisent aussi la pile comme mémoire temporaire des registres (grâce à des opérations d'empilement et de désempilement). Vous pouvez faire pareil, mais vous devez utilisez la pile avec parcimonie. Le pointeur du haut de la pile (« stack pointer ») a huit bits de longueur, donc la pile peut enregistrer seulement 256 octets d'information à chaque fois. Si vous mémorisez le 257^{e} octet sur la pile, le pointeur de pile se recopie lui-même, ou bien se remet à zéro, de telle sorte que le nouvel octet remplace le premier octet enregistré, qui est maintenant perdu. Cette écriture sur d'anciennes données est appelée le débordement de la pile (« stack overflow »), et quand il se produit, le programme se poursuit normalement jusqu'à ce qu'il ait besoin de l'information perdue, d'où une fin catastrophique du programme.

La zone de mémoire-tampon d'entrée

Le sous-programme d'entrée GETLN, qui est utilisé par le moniteur et les interpréteurs BASIC, utilise la page 2 comme mémoire-tampon d'entrée par le clavier. La taille de cette zone fixe la

longueur maximale des chaînes d'entrée. (Note : Applesoft n'utilise que les 237 premiers octets, bien qu'il vous soit permis d'en taper jusqu'à 256). Si vous savez que vous n'aurez jamais d'aussi longues chaînes de caractères à taper, vous pouvez mémoriser des données temporaires dans la partie la plus haute de la page 2.

La mémorisation des adresses de liaison

Le moniteur et le DOS 3.3 utilisent tous les deux la partie la plus haute de la page 3 pour des adresses de liaison ou des vecteurs. Le tableau 4-10 indique la partie de la page 3 dont le moniteur se sert ; se reporter au *Manuel du DOS* pour voir comment le DOS utilise la page 3.

Les zones de mémoire-tampon d'affichage

La zone-tampon primaire d'affichage de texte et de graphiques basse-résolution occupe les pages de mémoire 4 à 7 (adresses 1024 à 2047, en hexadécimal $0400 à $07FF). Toute cette zone de 1024 octets est appelée la page 1 d'affichage, et ne peut être utilisée pour mémoriser des programmes et des données. Il y a 64 positions de mémoire de cette zone qui ne sont pas affichées sur l'écran ; ces adresses sont réservées à l'usage des cartes périphériques (voir le chapitre 6).

La page 2 d'affichage, la zone de mémoire-tampon alternative d'affichage du texte et de graphique en basse-résolution, occupe les pages mémoire 8 à 11 (adresses 2048 à 3071, en héxadécimal $0800 à $0BFF). La plupart des programmes n'utilisent pas la page 2 pour l'affichage, donc ils peuvent se servir de cette zone pour mémoriser des programmes et des données.

La mémoire-tampon primaire d'affichage des graphiques en haute-résolution, appelée page 1 haute-résolution, occupe les pages de mémoire 32 à 63 (adresses 8192 à 16383, en héxadécimal $2000 à $3FFF). Si votre programme n'utilise pas de graphiques en haute-résolution, cette zone est utilisable pour des programmes ou des données.

La page 2 de haute-résolution graphique occupe les pages de mémoire 64 à 95 (adresses 16384 à 24575, en héxadécimal $4000 à $5FFF). La plupart des programmes utilise cette zone pour mémoriser des programmes et des données.

Pour d'autres informations sur les zones de mémoire-tampon d'affichage, voir le chapitre 2.

L'allocation de la mémoire vive

Tableau 4-1. Utilisation de la page zéro par le Moniteur

Poids forts de l'adresse	Poids faibles de l'adresse $0	$1	$2	$3	$4	$5	$6	$7	$8	$9	$A	$B	$C	$D	$E	$F
$00																
$10																•
$20	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•
$30	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•
$40	•	•	•	•	•	•	•	•	•	•					•	•
$50	•	•	•	•	•	•										
$60																
$70																
$80																
$90																
$A0																
$B0																
$C0																
$D0																
$E0																
$F0																

Tableau 4-2. Utilisation de la page zéro par Applesoft

Poids forts de l'adresse	Poids faibles de l'adresse $0	$1	$2	$3	$4	$5	$6	$7	$8	$9	$A	$B	$C	$D	$E	$F
$00	•	•	•	•	•	•					•	•	•	•	•	•
$10	•	•	•	•	•	•	•	•	•	•				•		
$20																
$30																
$40																
$50	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•
$60	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•
$70	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•
$80	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•
$90	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•
$A0	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•
$B0	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•
$C0	•	•	•	•	•	•	•	•	•	•	•	•	•	•		
$D0	•	•	•	•	•	•			•	•	•	•	•	•	•	•
$E0	•	•	•	•	•	•	•	•	•	•	•					
$F0	•	•	•	•	•	•	•	•	•							

Tableau 4-3. Utilisation de la page zéro par le BASIC Entier

Poids forts de l'adresse	Poids faibles de l'adresse															
	$0	$1	$2	$3	$4	$5	$6	$7	$8	$9	$A	$B	$C	$D	$E	$F
$00														•		
$10																
$20																
$30																
$40											•	•	•	•		
$50						•	•	•	•	•	•	•	•	•	•	•
$60	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•
$70	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•
$80	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•
$90	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•
$A0	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•
$B0	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•
$C0	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•
$D0	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•
$E0																
$F0															•	•

Tableau 4-4. Utilisation de la page zéro par le DOS. 3.3

Poids forts de l'adresse	Poids faibles de l'adresse															
	$0	$1	$2	$3	$4	$5	$6	$7	$8	$9	$A	$B	$C	$D	$E	$F
$00																
$10																
$20							•	•			•	•	•	•	•	•
$30						•	•	•	•	•					•	•
$40	•	•	•	•	•	•	•	•	•		•	•	•	•		
$50																
$60								•	•	•	•					•
$70	•															
$80																
$90																
$A0																•
$B0	•															
$C0											•	•	•	•		
$D0									•							
$E0																
$F0																

L'allocation de la mémoire vive

La mémoire à banc-commuté

L'espace d'adresse de mémoire de 52K à 64K (hexadécimal $D000 à $FFFF) est deux fois alloué : il est utilisé à la fois par de la MEM et de la MEV. Les 12K octets de MEM (mémoire morte) dans cet espace d'adresses contiennent le Moniteur et l'interpréteur BASIC. Il y a aussi 16K de MEV sur cet espace. La MEV est normalement utilisée pour mémoriser soit le BASIC entier soit une partie du système d'exploitation du Pascal (acheté séparemment).

Vous pouvez vous demander pourquoi cette partie de la mémoire a une personnalité si divisée. Certaines des raisons sont historiques : l'Apple //e est capable d'exécuter du logiciel écrit pour l'Apple II et l'Apple II Plus parce qu'il utilise cette partie de mémoire de la même manière qu'eux. Il est commode d'avoir l'interpréteur Applesoft en MEM, mais l'Apple //e, comme un Apple II équipé d'une carte-langage, est aussi capable d'utiliser l'espace d'adresses pour d'autres choses quand l'Applesoft n'est pas nécessaire.

Vous pourriez aussi vous demander comment 16K octets de MEV sont projetés en seulement 12K d'espace d'adresses. La réponse habituelle est que c'est comme un miroir, et ce n'est pas une mauvaise analogie : l'espace d'adresse de 4K octets entre 52K et 56K (hexadécimal $D000 et $DFFF) est utilisé deux fois.

La multiplication du même espace d'adresses en différents blocs de mémoire est appelée commutation de banc. Nous avons deux exemples de commutation de bancs ici : en premier, tout l'espace d'adresses entre 52K et 64K ($D000 à $FFFF) est commuté soit sur la MEM soit sur la MEV, et en second, l'espace d'adresses entre 52K et 56K ($D000 à $DFFF) est commuté entre deux blocs différents de MEV.

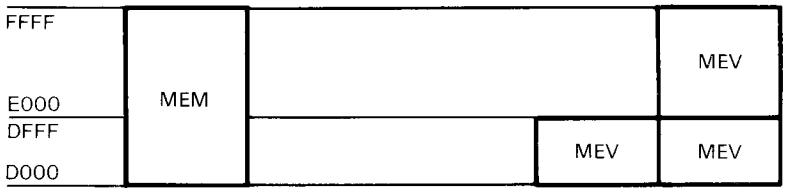


Figure 4-3. Carte de la mémoire à banc-commuté

Les commutateurs de sélection d'un banc

Vous commutez les bancs de mémoire de la même façon que vous faites commuter d'autres fonctions dans l'Apple //e : en utilisant des commutateurs logiciels. Ces commutateurs logiciels font trois choses : sélectionner soit la MEM soit la MEM dans cet espace d'adresses ; autoriser ou interdire l'écriture dans la MEV (protection d'écriture) ; et sélectionner le premier ou le second des deux bancs de 4K octets de MEV dans l'espace d'adresses $D000 à $DFFF.

Avertissement
N'utilisez pas ces commutateurs logiciels sans une planification soigneuse. Une commutation sans précaution entre la MEV et la MEM est presque certaine de provoquer des effets catastrophiques sur votre programme.

Le tableau 4-5 donnent les adresses des commutateurs logiciels autorisant toutes les combinaisons de lecture et d'écriture dans cet espace de mémoire. Toutes les valeurs hexadécimales sont de la forme $C08x. Remarquez que plusieurs adresses activent la même fonction : ceci parce que chaque commutateur logiciel est activé par un seul bit d'adresse. Par exemple, toute adresse de la forme $C08x avec un 1 dans le bit de plus faible poids permet l'écriture en MEV. De même, le bit 3 de l'adresse sélectionne l'un des deux bancs de 4K octets de MEV dans l'espace d'adresse $D000-$DFFF ; si le bit 3 vaut 0, le premier banc de MEV est utilisé, et si le bit 3 est à 1, le second banc est utilisé.

Quand la lecture n'est pas autorisée en MEV, elle est autorisée en MEM dans cet espace d'adresses. Même lorsque la lecture n'est pas autorisée en MEV, il est encore possible d'y écrire si l'écriture est autorisée.

Quand vous allumez ou redémarrez l'Apple //e, il initialise les commutateurs de bancs pour la lecture en MEM et l'écriture en MEV, en prenant le deuxième banc de MEV. Remarquez que ceci diffère du redémarrage de l'Apple II Plus, qui n'affectait pas la mémoire à banc commuté (la carte-langage). Sur l'Apple //e, vous ne pouvez pas utiliser le vecteur de réinitialisation (« reset vector ») pour redonner le contrôle à un programme situé dans la mémoire à banc-commuté, comme vous pouviez le faire sur l'Apple II Plus.

Si vous utilisez l'interprèteur BASIC Entier sur l'Apple //e, le redémarrage par Reset fonctionne correctement, réinitialisant le BASIC en laissant intact votre programme. Ceci se produit parce que le vecteur de réinitialisation transfère le contrôle au DOS, et le DOS remet les commutateurs à leurs valeurs appropriées de la version courante du BASIC.

Tableau 4-5. Commutateurs de sélection de banc

(1) Ce commutateur autorise l'écriture en MEV et la lecture en MEM.
(2) Deux lectures successives à ce commutateur autorise l'écriture et la lecture en MEV.

Adresse de commutateur	Écriture en MEV	Lecture en MEV	Lecture en MEM	Banc de 4K de MEV : Premier	Banc de 4K de MEV : Second	Notes
$C080		•			•	
$C081	•		•		•	1
$C082			•		•	
$C083	•	•			•	2
$C084		•			•	
$C085	•		•		•	1
$C086			•		•	
$C087	•	•			•	2
$C088		•		•		
$C089	•		•	•		1
$C08A			•	•		
$C08B	•	•		•		2
$C08C		•		•		
$C08D	•		•	•		1
$C08E			•	•		
$C08F	•	•		•		2

Remarquez que vous ne pouvez pas lire un banc de MEV et écrire sur l'autre ; si vous choisissez un des deux bancs en lecture, vous l'aurez choisi aussi en écriture.

Vous ne pouvez pas lire en MEM sur une partie de la mémoire à banc-commuté et lire en MEV sur le reste : spécifiquement, vous ne pouvez pas lire le Moniteur situé en MEM, tout en lisant la MEV à banc-commuté. Si vous voulez utiliser les sous-programmes du Moniteur avec un programme en MEV à banc-commuté, tout d'abord recopiez le Moniteur de la MEM (adresses $F0800 à $FFCB) dans une zone plus basse de la MEV et puis dans la MEV à banc commuté.

Pour étudier comment utiliser ces commutateurs, regardez l'extrait suivant d'un programme en langage assembleur :

```
AD 83 C0    LDA  $C083     ; SELECTIONNE LE 2e BANC DE 4K
                           EN LECTURE/ECRITURE
AD 83 C0    LDA  $C083     ; PAR DEUX LECTURES CONSECUTIVES
A9 D0 C0    LDA  #$D0      ; MISE EN PLACE...
85 01       STA  BEGIN     ; ... DES NOUVEAUX
A9 FF       LDA  #$FF      ; ... POINTEURS EN
85 02       STA  END       ; ... MEMOIRE PRINCIPALE
20 97 C9    JSR  RAMTST    ; ...POUR LE BANC DE 12K

AD 8B C0    LDA  $C08B     ; SELECTIONNE LE 1er BANC DE 4K
20 97 C9    JSR  RAMTST    ; UTILISE LES POINTEURS CI-DESSUS

AD 83 C0    LDA  $C088     ; SELECTIONNE LE 1er BANC ET
                           PROTEGE DE L'ECRITURE
```

Organisation de la mémoire

```
A9  80        LDA   #$80
E6  10        INC   TSTNUM
20  58  C9    JSR   WPTSINIT
AD  80  C0    LDA   $C080      ; SELECTIONNE LE 2e BANC ET LE
                                 PROTEGE DE L'ECRITURE
E6  10        INC   TSTNUM
A9  01        LDA   #PAT12K
20  58  C9    JSR   WPTSINIT
AD  8B  C0    LDA   $C08B      ; SELECTIONNE LE 1er BANC ET
                                 LA LECTURE/ECRITURE
AD  8B  C0    LDA   $C08B      ; PAR DEUX LECTURES CONSECUTIVES
E6  0E        INC   RWMODE     ; INDICATEUR QUE LA MEV EST EN L/E
E6  10        INC   TSTNUM
A9  08        LDA   #PAT4K
20  58  C9    JSR   WPTSINIT
```

L'instruction LDA, qui réalise une opération de lecture dans une position spécifiée de mémoire, est utilisée pour manipuler les commutateurs logiciels.

La séquence inhabituelle de deux instructions LDA consécutives réalise les deux lectures consécutives qui autorisent l'écriture dans cette zone de MEV ; dans ce cas, les données qui sont lues ne sont pas utilisées.

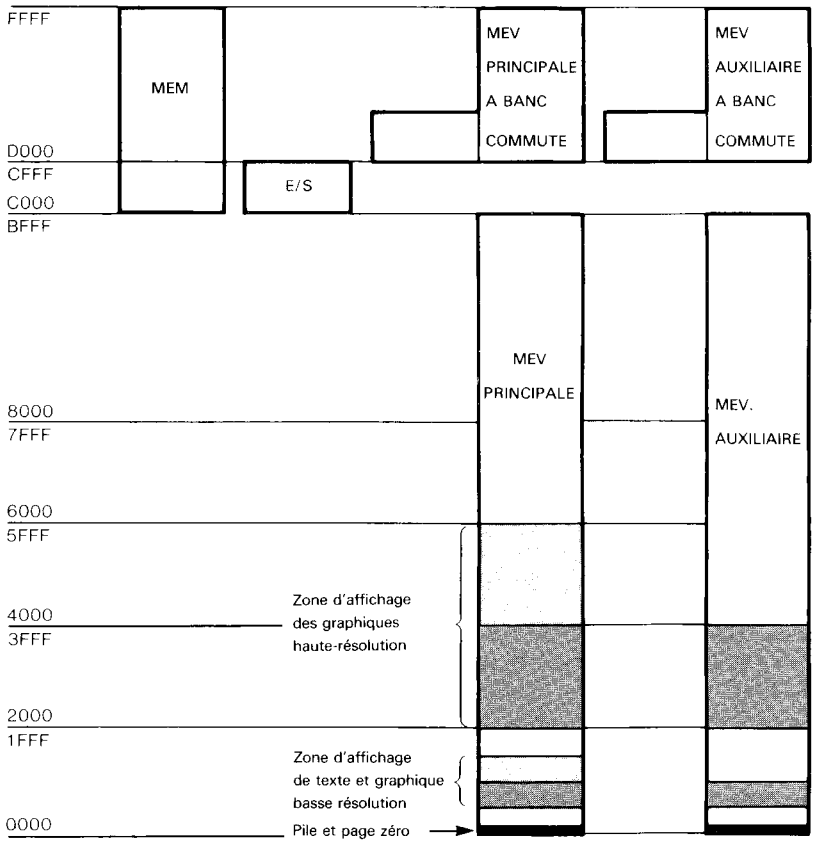
La mémoire auxiliaire et ses sous-programmes de gestion

En installant une carte en option dans le connecteur auxiliaire, vous pouvez ajouter de la mémoire à l'Apple //e. Une telle carte est la carte de texte en 80 colonnes qui a 1K octets de mémoire vive additionnelle pour élargir l'affichage de texte de 40 colonnes à 80 colonnes.

Une autre carte en option, la carte de texte en 80 colonnes étendue, a 64K de MEV additionnelle. Une zone de 1K octet sert à la même chose que la mémoire sur la carte de texte en 80 colonnes : élargir l'affichage de texte à 80 colonnes. Les autres 63K octets peuvent servir de mémoire auxiliaire de programmes et de données. Si vous n'utilisez qu'un affichage en 40 colonnes, tout l'ensemble des 64K est disponible pour des programmes et des données.

Avertissement
N'essayez pas d'utiliser la mémoire auxiliaire depuis un programme en BASIC. L'interpréteur BASIC utilise plusieurs zones de la MEV principale, y compris la pile et la page zéro. Si vous commutez la mémoire auxiliaire dans ces zones, l'interpréteur BASIC se perdra et vous devrez réinitialiser le système et redémarrer.

Figure 4-4. Carte de la mémoire avec la mémoire auxiliaire



Comme vous pouvez le constater en étudiant la carte de mémoire de la figure 4-4, la mémoire auxiliaire est divisée en deux grandes sections et une petite. La plus grande section est commutée sur l'espace d'adresse 512 à 49151 ($200 à $BFFF). Cet espace comprend les pages de mémoires-tampons d'affichage : comme cela est décrit au chapitre 2, l'espace dans la mémoire auxiliaire est utilisé pour la moitié du texte en 80 colonnes. Vous pouvez commutez la mémoire auxiliaire sur l'espace de mémoire tout entier, ou bien vous pouvez ne commutez que les pages d'affichage d'écran : voyez plus loin le paragraphe « Commutation de mode de mémoire ».

Si la seule raison pour laquelle vous utilisez la mémoire auxiliaire est un affichage sur 80 colonnes, remarquez que vous pouvez mémoriser dans les pages d'affichage de la mémoire auxiliaire grâce aux commutateurs logiciels 80STORE et PAGE2 décrits dans le paragraphe « Commutation de modes d'affichage » du chapitre 2.

L'autre grande section de mémoire auxiliaire est commutée dans l'espace d'adresses entre 52K et 64K ($D000 à $FFFF). Cet espace de mémoire et les commutateurs qui le contrôle sont décrits plus

haut dans le paragraphe « Mémoire à banc-commuté ». Si vous utilisez la MEV auxiliaire dans cet espace, les commutateurs logiciels ont le même effet sur la MEV auxiliaire que sur la MEV principale : la commutation de banc est indépendante de la commutation de la mémoire auxiliaire.

Notez que les commutateurs logiciels pour la mémoire à banc-commuté, décrits dans le paragraphe précédent, ne changent pas quand vous commutez sur la MEV auxiliaire. En particulier, si la MEM est autorisée dans l'espace de mémoire à banc-commuté avant que vous ne commutiez sur la mémoire auxiliaire, la MEM sera toujours autorisée après la commutation. Chaque fois que vous commutez la section à banc-commuté de la mémoire auxiliaire d'un côté à l'autre, vous devez aussi vous assurer que les commutateurs de bancs ont des valeurs correctes.

Quand vous commutez la mémoire auxiliaire dans l'espace de mémoire à banc commuté, vous commutez aussi les deux premières pages, de 0 à 511 ($0000 à $01FF). Cette partie de mémoire contient la page zéro, qui est utilisée pour conserver d'importantes données et adresses de base, et la page une qui est la pile du 6502. La pile et la page zéro sont commutées de telle sorte que le logiciel-système s'exécutant sur l'espace de mémoire à banc-commuté puisse maintenir sa propre pile et sa propre page zéro pendant qu'il manipule l'espace d'adresses de 48K (de $0200 à $BFFF), soit dans la mémoire principale soit dans la mémoire auxiliaire.

Commutation de modes de mémoire

La commutation de la section de 48K de mémoire est réalisée par deux commutateurs logiciels : le commutateur appelé RAMRD sélectionne la lecture dans la mémoire principale ou dans la mémoire auxiliaire, et celui appelé RAMWRT sélectionne l'écriture sur la mémoire principale ou sur la mémoire auxiliaire. Comme l'indique le tableau 4-6, chaque commutateur a une paire d'adresses de mémoire qui lui est réservée : une adresse pour sélectionner la mémoire principale et l'autre pour sélectionner la mémoire auxiliaire. En autorisant l'écriture et la lecture indépendamment, il est possible qu'un programme, dont les instructions sont lues sur un espace de mémoire, mémorise des données sur l'autre espace de mémoire.

Avertissement
N'utilisez pas ces commutateurs sans une planification soigneuse. Une commutation sans précaution entre les mémoire principale et auxiliaire provoque des effets catastrophiques sur le fonctionnement de l'Apple //e. Par exemple, si vous commutez la mémoire auxiliaire sans que la carte de mémoire auxiliaire ne soit installée, le programme qui s'exécute va s'arrêter et vous aurez à redémarrer l'Apple //e et à reprendre depuis le début.

La mémoire auxiliaire et ses sous-programmes de gestion

En écrivant dans le commutateur logiciel d'adresse $C003, on met RAMRD à un et on autorise la lecture dans la mémoire auxiliaire ; en écrivant dans l'adresse $C002, on met à zéro RAMRD et on autorise la lecture dans la mémoire principale. En écrivant dans le commutateur logiciel à l'adresse $C005, on met RAMWRT à un et on autorise l'écriture dans la mémoire auxiliaire ; en écrivant à l'adresse $C004, on met à zéro RAMWRT et on autorise l'écriture en mémoire principale. En manipulant ces commutateurs indépendamment, vous pouvez utiliser l'une des quatre combinaisons de lecture et d'écriture en mémoire principale ou en mémoire auxiliaire.

La mémoire auxiliaire correspondante à la Page 1 de texte et à la Page 1 de graphique haute-résolution peut être utilisée comme une partie de l'espace d'adresses de $0200 à $BFFF en utilisant RAMRD et RAMWRT comme on vient de le décrire plus haut. Ces zones dans la MEV auxiliaire peuvent aussi être contrôlées séparemment grâce aux commutateurs décrits dans le paragraphe « Commutation de mode d'affichage » dans le chapitre 2. Ces commutateurs sont appelés 80STORE, PAGE2, et HIRES.

Comme le montre le tableau 4-6, le commutateur 80STORE fonctionne comme un commutateur d'autorisation : si il est à un, le commutateur PAGE2 sélectionne la mémoire principale ou la mémoire auxiliaire. Avec le commutateur HIRES à zéro, l'espace de mémoire commuté par PAGE2 est la Page 1 d'affichage de texte, de $0400 à $07FF ; avec HIRES à un, PAGE2 commute à la fois la Page 1 de texte et la Page 1 de graphique haute-résolution, de $2000 à $3FFF.

Si vous utilisez à la fois les commutateurs de contrôle de la MEV auxiliaire et les commutateurs de contrôle de la page d'affichage auxiliaire, ce sont les commutateurs de contrôle de la page d'affichage qui ont priorité : si 80STORE est à zéro, RAMRD et RAMWRT fonctionnent sur l'espace de mémoire $0200 à $BFFF tout entier, mais si 80STORE est à un, RAMRD et RAMWRT n'ont aucun effet sur la page d'affichage. Spécifiquement, si 80STORE est à un et HIRES est à zéro, PAGE2 contrôle la Page 1 de texte quelles que soient les valeurs données à RAMRD et RAMWRT. Autrement dit, si 80STORE et HIRES sont tous les deux à un, PAGE2 contrôle à la fois la Page 1 de texte et la Page 1 graphique de haute-résolution, à nouveau indépendamment de RAMRD et RAMWRT.

Un commutateur logiciel unique appelé ALTPZ (qui veut dire page zéro alternative) commute la mémoire à banc-commuté et la pile associée et la page zéro entre la mémoire principale et la mémoire auxiliaire. Comme l'indique le tableau 4-6, en écrivant à l'adresse $C009 on met ALTZP à un et on sélectionne la page zéro et la pile de la mémoire auxiliaire : en écrivant dans le commutateur logiciel à l'adresse $C008 on met ALTZP à zéro et on sélectionne la pile et la page zéro de la mémoire principale en écriture et en lecture. Le paragraphe « Sous-programmes pour la mémoire auxiliaire », ci-dessous, décrit les sous-programmes intégrés au système que vous pouvez appeler pour vous aider à commuter entre la mémoire principale et la mémoire auxiliaire.

Organisation de la mémoire

Il y a encore trois autres adresses associées aux commutateurs de la mémoire auxiliaire. Les bits de poids fort des octets que vous lirez à ces adresses vous diront les valeurs des trois commutateurs logiciels décrits ci-dessus. L'octet que vous lirez à l'adresse $C013 a son bit de poids fort à 1 si RAMRD est à un (la lecture est autorisée sur la mémoire auxiliaire), à 0 si RAMRD est à zéro (la lecture est autorisée sur le bloc de 48K de la mémoire principale). L'octet d'adresse $C014 a son bit de poids fort à 1 si RAMWRT est à un (l'écriture sur la mémoire auxiliaire est autorisée), ou à 0 si RAMWRT est à zéro (l'écriture sur la mémoire principale est autorisée).L'octet d'adresse $C016 a son bit de poids fort à 1 si ALTZP est à un (la zone à banc-commuté, la pile, et la page zéro de la mémoire auxiliaire sont sélectionnées), ou à 1 si ALTZP est à zéro (ces mêmes zones mais en mémoire principale sont sélectionnées).

Quand ces commutateurs sont à un, c'est la mémoire auxiliaire qui est utilisée ; quand ils sont à zéro, c'est la mémoire principale.

Tableau 4-6. Commutateurs de sélection de la mémoire auxiliaire

(1) Si 80STORE est à un, le commutateur PAGE2 sélectionne la mémoire d'affichage, soit auxiliaire soit principale.
(2) Si 80STORE est à un, le commutateur HIRES vous autorise à utiliser le commutateur PAGE2 pour commuter la zone Page 1 en haute résolution entre la mémoire principale et la mémoire auxiliaire.

Nom	Fonction	Adresse Hex	Adresse Décimale	Notes
RAMRD	Lire en mémoire auxiliaire	$C003	49155 —16381	Écrire
	Lire en mémoire principale	$C002	49154 —16382	Écrire
	Lire l'état de RAMRD	$C013	49171 —16365	Lire
RAMWRT	Écrire en mémoire auxiliaire	$C005	49157 —16379	Écrire
	Écrire en mémoire principale	$C004	49156 —16380	Écrire
	Lire l'état de RAMWRT	$C014	49172 —16354	Lire
80STORE	A un : accès page d'affich.	$C001	49153 —16383	Écrire
	A zéro : utilisez RAMRD/WRT	$C000	3152 —16384	Écrire
	Lire l'état de 80STORE	$C018	49176 —16360	Lire
PAGE2	Page 2 à un (mémoire aux.)	$C055	49237 —16299	1
	Page 2 à zéro (mémoire princ.)	$C054	49236 —16300	1
	Lire l'état de PAGE2	$C01C	49180 —16356	Lire
HIRES	A un : accès pages hte-res.	$C057	49239 —16297	2
	A zéro : utilisez RAMRD/WRT	$C056	49238 —16298	2
	Lire l'état de HIRES	$C01D	49181 —16355	Lire
ALTZP	Pile et page zéro aux.	$C009	49161 —16373	Écrire
	Pile et page zéro princ.	$C008	49160 —16374	Écrire
	Lire l'état de ALTZP	$C016	49174 —16352	Lire

Pour avoir assez d'adresses de mémoire pour tous les commutateurs logiciels et rester compatibles avec l'Apple II et l'Apple II Plus, les commutateurs logiciels énumérés dans la table 4-6 partagent leurs adresses de mémoire avec les fonctions du clavier dont la liste est sur la table 2-2. Les opérations — écriture et lecture — indiquées à la table 4-6 pour contrôler la mémoire auxiliaire sont justement celles qui ne sont pas utilisées pour lire le clavier et mettre à zéro le bit d'échantillonnage.

Les sous-programmes de la mémoire auxiliaire

Si vous voulez écrire des programmes en langage Assembleur qui utilisent la mémoire auxiliaire mais que vous ne vouliez pas gérer la mémoire auxiliaire vous-même, vous pouvez vous servir des sous-programmes déjà implantés pour la mémoire auxiliaire. Ces sous-programmes rendent possibles l'utilisation de la mémoire auxiliaire sans avoir à manipuler les commutateurs logiciels décrits dans les paragraphes précédents.

Les sous-programmes décrits ci-dessous rendent plus faciles l'usage de la mémoire auxiliaire, mais ils ne vous protègent pas en cas d'erreurs. Vous devez toujours planifier l'utilisation de la mémoire auxiliaire pour éviter des effets catastrophiques sur votre programme.

L'utilisation de ces sous-programmes se fait de la même façon que l'utilisation des sous-programmes d'E/S décrits au chapitre 3 : en faisant des appels aux sous-programmes à leur adresse de début. Ces adresses sont données dans le tableau 4-7.

Tableau 4-7. Sous-programmes pour la mémoire auxiliaire

Nom du sous-programme	Adresse	Description
AUX MOVE	$C311	Déplacent des blocs de données entre mémoires principale et auxiliaire
XFER	$C314	Transfert le contrôle d'un programme entre mémoires principales et auxiliaire

Déplacement de données vers la mémoire auxiliaire

Dans vos programmes en langage Assembleur, vous pouvez vous servir du sous-programme appelé AUXMOVE pour copier des blocs de données de la mémoire principale à la mémoire auxiliaire ou de la mémoire auxiliaire à la mémoire principale. Avant d'appeler ce sous-programme, vous devez mettre les adresses des données dans des paires d'octets en page zéro et donner une valeur au bit de retenue (« carry bit ») pour sélectionner la direction du déplacement — principale vers auxiliaire ou auxiliaire vers principale.

Avertissement
N'essayez pas d'utiliser AUXMOVE pour copier des données dans la page zéro ou la page un (la pile du 6502) ou dans la mémoire à banc commuté ($D000 - $FFFF). AUXMOVE utilise la page zéro pendant tout le temps de la copie, donc il ne peut réaliser des déplacements dans l'espace de mémoire commuté par ALTZP.

Les paires d'octets que vous utiliserez pour transmettre des adresses à ce sous-programme sont appelés A1, A2 et A4, et elles sont utilisées par plusieurs sous-programmes de base de l'Apple *//e* pour passer des paramètres. Les adresses de ces paires d'octets sont données dans le tableau 4-8.

Tableau 4-8. Paramètres du sous-programme AUXMOVE

Nom	Adresse	Paramètre transmis
Retenue		1 = Déplacement de la mémoire principale vers la mémoire auxiliaire 0 = Déplacement de la mémoire auxiliaire vers la mémoire principale
A1L	$3C	Octet de poids faible de l'adresse de début
A1H	$3D	Octet de poids fort de l'adresse de début de la source de données
A2L	$3E	Octet de poids faible de l'adresse de fin
A2H	$3F	Octet de poids fort de l'adresse de fin de la source des données
A4L	$42	Octet de poids faible de l'adresse de début
A4H	$43	Octet de poids fort de l'adresse de début de la destination des données

Mettez les adresses du premier et du dernier octets du bloc de mémoire que vous voulez copier dans A1 et A2. Mettez l'adresse de début du bloc de mémoire dans lequel vous voulez recopier les données en A4.

Le sous-programme AUXMOVE utilise le bit de retenue pour sélectionner la direction dans laquelle sont recopiées les données. Pour copier des données de la mémoire principale à la mémoire auxiliaire, mettez le bit de retenue à un ; pour copier des données de la mémoire auxiliaire à la mémoire principale, mettez à zéro le bit de retenue.

Quand vous ferez appel au sous-programme AUXMOVE, le sous-programme copiera le bloc de données comme il est spécifié par les paires d'octets A1, A2, et A4 et le bit de retenue. Quand il aura fini, l'accumulateur et les registres X et Y seront tels qu'ils étaient quand vous l'avez appelé.

Transfert de contrôle à la mémoire auxiliaire

Vous pouvez vous servir du sous-programme déjà implémenté,appelé XFER, pour transférer le contrôle à et depuis des segments de programme en mémoire auxiliaire. Vous devez fixer trois paramètres avant d'utiliser XFER : l'adresse du sous-programme vers lequel vous voulez vous transférer, la direction du transfert (de principale à auxiliaire ou d'auxiliaire à principale), et quelles page zéro et pile vous voulez utiliser.

Tableau 4-9. Paramètres du Sous-programme XFER

Nom ou Adresse	Paramètre passé
Retenue	1 = Transfert de la mémoire principale à la mémoire auxiliaire 0 = Transfert de la mémoire auxiliaire à la mémoire principale
Dépassement	1 = Utilisation de la page zéro et de la pile de la mémoire auxiliaire 0 = Utilisation de la page zéro et de la pile de la mémoire principale
$3ED	Octet de poids faible de l'adresse de début du programme
$3EE	Octet de poids fort de l'adresse de début du programme

Mettez l'adresse de transfert dans deux octets aux adresses $3ED et $3EE, avec en premier l'octet de poids faible, comme d'habitude. La direction du transfert est contrôlée par le bit de retenue : mettez à un le bit de retenue pour transférer vers un programme en mémoire auxiliaire ; mettez à zéro le bit de retenue pour transférer vers un programme en mémoire principale. Utilisez le bit de débordement pour sélectionner quelles pages zéro et pile vous voulez utilisez : mettez à zéro le bit de débordement pour vous servir de la mémoire principale ; mettez le bit de débordement à un pour vous servir de la mémoire auxiliaire.

Après avoir donné leurs valeurs aux paramètres, passez le contrôle au sous-programme XFER par une instruction de saut, plutôt que par un appel au sous-programme. XFER sauvegarde l'accumulateur et l'adresse de transfert sur la pile, puis positionne les commutateurs logiciels pour les paramètres que vous avez sélectionnés et saute au nouveau programme.

Avertissement
C'est le programmeur qui est responsable de la sauvegarde du pointeur de pile se déplaçant quelque part dans l'espace courant de mémoire avant d'utiliser XFER ; ainsi que de sa récupération après avoir repris le contrôle. En ne respectant pas cette consigne on provoquera des erreurs de programme.

La procédure de réinitialisation (Reset)

Pour mettre l'Apple //e dans un état bien déterminé et connu lorsqu'il vient juste d'être allumé, ou après qu'un programme ait mal fonctionné, il existe une routine appelée procédure de réinitialisation ou sous-programme Reset. Le sous-programme Reset fait partie des sous-programmes intégrés en MEM dans l'Apple //e, et il est enclenché chaque fois que vous mettez sous-tension ou que vous appuyez sur la touche (**Reset**) tout en maintenant enfoncée la touche (**Ctrl**). Le sous-programme Reset met l'Apple //e dans un mode normal de fonctionnement et redémarre le programme résident.

Quand vous enclenchez une réinitialisation, le hardware dans l'Apple //e met les commutateurs logiciels de contrôle de mémoire en position normale : la mémoire vive sur la carte-mère est autorisée, et, s'il y a une carte de texte en 80 colonnes ou une carte de texte en 80 colonnes étendue dans le connecteur auxiliaire, le connecteur d'extension N° 3 est alloué aux sous-programmes pour 80 colonnes. La MEV auxiliaire est inhibée et l'espace de mémoire à banc-commuté est positionné pour lire en MEM et écrire en MEV, le second banc étant utilisé à $D000.

Le sous-programme Reset positionne les commutateurs logiciels de contrôle d'affichage pour afficher la Page 1 de texte en 40 colonnes en utilisant l'ensemble primaire de caractères, puis la fenêtre est fixée à tout l'écran en 40 colonnes, le curseur est positionné au bas de l'écran et l'affichage est en mode normal.
Le sous-programme Reset considère le clavier et l'écran comme dispositifs standards d'entrée et de sortie en chargeant les adresses de liaisons standards (voir le Chapitre 6). Il met à zéro les annonciateurs 0 et 1 et les annonciateurs 2 et 3 à un, il met à zéro le bit d'échantillonnage du clavier, déconnecte toute MEM de carte-périphérique (voir Chapitre 6) et émet un signal sonore.

L'Apple //e a trois types d'initialisation : par mise sous-tension appelé démarrage à froid ; réinitialisation de type démarrage à chaud ; et réinitialisation de type démarrage à froid forcé. La

procédure décrite ci-dessus reste valable pour ces trois sortes de réinitialisation. Ce qui va se produire ensuite dépend du vecteur de réinitialisation. Le sous-programme Reset teste le vecteur de réinitialisation pour déterminer s'il est valide ou non, comme on le décrira plus loin dans le paragraphe, « Le vecteur de réinitialisation ». Si la réinitialisation est dûe à une remise sous-tension, le vecteur n'est pas valide, et le sous-programme Reset effectue un démarrage forcé à froid. Si le vecteur est valide, le sous-programme effectue la réinitialisation de type démarrage à chaud.

La procédure de démarrage à froid

Si le vecteur de réinitialisation n'est pas valide, c'est parce que l'Apple //e vient juste d'être allumé ou bien quelque chose a fait changer le contenu de la mémoire. Le sous-programme Reset efface l'écran et met la chaîne de caractères « Apple][» en haut de l'écran. Il charge le vecteur de réinitialisation et l'octet pour tester la validité comme décrit plus loin, puis commence à tester les connecteurs d'extension pour voir s'il n'y a pas un contrôleur de lecteur de disquettes dans l'un d'entre eux, en commençant par le connecteur 7 et en continuant avec les connecteurs de numéros inférieurs. S'il trouve une carte contrôleur, il enclenche le sous-programme d'amorçage (« bootstrap ») qui réside dans la MEM de la carte-contrôleur. Le programme d'amorçage charge en MEV le Système d'Exploitation des disquettes depuis la disquette du lecteur 1. Quand le DOS a été chargé, il affiche d'autres messages sur l'écran. S'il n'y a pas de disquette dans le lecteur, le moteur du lecteur continue simplement à tourner jusqu'à ce que vous appuyez sur **Ctrl** - **Reset** .
Pour d'autres informations sur le DOS et la procédure de démarrage, voir le *Manuel du DOS*.

Si le sous-programme Reset ne trouve pas de carte contrôleur, ou si vous appuyez sur **Ctrl** - **Reset** à nouveau avant que la procédure de démarrage ne soit terminée, le sous-programme Reset se poursuivra sans utiliser de disquette, et passera le contrôle des opérations à l'interprèteur Applesoft implanté dans le système.

La procédure de démarrage à chaud

Chaque fois que vous appuyez sur **Ctrl** - **Reset** si l'Apple //e a déjà terminé une initialisation de type démarrage à froid, le vecteur de réinitialisation est encore valide et il n'est pas nécessaire de réinitialiser tout le système. Le sous-programme Reset utilise simplement le vecteur pour transférer le contrôle au programme résident, qui est normalement l'interprèteur Applesoft. Si le programme résident est en effet Applesoft, votre programme écrit en Applesoft et ses variables sont encore intacts. Si vous êtes sous le contrôle du DOS, c'est lui le programme résident et il redémarrera soit le BASIC Applesoft soit le BASIC Entier, celui que vous utilisiez au moment où vous avez appuyez sur **Ctrl** - **Reset** .

Un programme dans la MEV à banc-commuté ne peut pas se servir du vecteur de réinitialisation pour reprendre le contrôle des opérations après une réinitialisation, parce que le sous-programme Reset n'autorise la lecture qu'en MEM dans l'espace de mémoire à banc-commuté. Si vous vous servez du BASIC Entier, qui est dans la MEV à banc-commuté, c'est donc que vous utilisez aussi le DOS, et c'est le DOS qui contrôle le vecteur de réinitialisation et redémarre BASIC.

La procédure de démarrage à froid forcé

Si un programme a mis dans le vecteur de réinitialisation un pointeur au début du programme, comme on le décrira plus loin, appuyer sur (Ctrl)-(Reset) provoque une réinitialisation de type démarrage à chaud qui utilise le vecteur pour transférer le contrôle à ce programme. Si vous voulez arrêter un tel programme sans éteindre l'ordinateur et le rallumer, vous pouvez forcer une réinitialisation de type démarrage à froid en maintenant appuyées la touche (Ctrl) et la touche (), puis en appuyant sur (Reset) et en relachant (Reset).

Si vous voulez arrêter un programme inconditionnellement - par exemple, pour démarrer l'Apple //e avec un autre programme - il est préférable d'utiliser la réinitialisation de type démarrage à froid forcé, (Ctrl)-(Reset), au lieu d'éteindre et de rallumer l'ordinateur.

A chaque fois que vous appuyez sur (Ctrl)-(Reset), les sous-programmes intégrés dans l'Apple //e testent toujours pour voir si une des touches POMME est enfoncée. Si c'est la touche () , avec ou sans la touche (), le sous-programme effectue l'auto-test décrit plus bas. Si c'est seulement la touche () qui est enfoncée, le sous-programme déclenche une réinitialisation de type démarrage à froid forcé. En premier lieu, il détruit le programme et les données en mémoire en écrivant deux octets de données arbitraires dans chaque page de la MEV. Les deux octets qui vont s'inscrire sur la Page 3 sont ceux qui contiennent le vecteur de réinitialisation. Le sous-programme effectue alors une réinitialisation de type démarrage normal à froid.

Le vecteur de réinitialisation

Quand vous réinitialisez l'Apple //e, le sous-programme Reset transfère le contrôle au programme résident au moyen d'une adresse mémorisée dans la page 3 de la MEV principale. Cette adresse s'appelle un *vecteur* parce qu'il dirige le contrôle du programme vers une destination spécifiée. Il y a plusieurs autres adresses-vecteurs mémorisées en page 3, comme le montre le tableau 4-10, y compris les vecteurs d'interruption décrits dans le Chapitre 6, et les vecteurs du DOS décrits dans le *Manuel du DOS.*

La procédure de réinitialisation

La procédure de réinitialisation de type démarrage à froid a mémorisé l'adresse de début de l'interprèteur Applesoft, l'octet des poids faibles en premier, dans le vecteur de réinitialisation aux positions 1010 et 1011 (hexadécimal $3F2 et $3F3). Elle a mémorisé aussi un octet de test de validité, appelé assi l'octet de mise sous-tension, à l'adresse 1012 (hexadécimal $3F4). L'octet de test de validité est calculé en effectuant un OR-exclusif du second octet du vecteur de réinitialisation avec la valeur constante 165 ($A5). A chaque fois que vous réinitialisez l'Apple //e, le sous-programme Reset utilise cet octet pour déterminer si le vecteur de réinitialisation est toujours valide.

Vous pouvez changer le vecteur de réinitialisation pour que le sous-programme Reset transfère le contrôle à votre programme au lieu de le transférer à l'interprêteur Applesoft. Pour que cela fonctionne, vous devez aussi changer l'octet de test de validité en le rendant égal au OR-exclusif de l'octet de poids fort de votre nouveau vecteur de réinitialisation avec la constante 165 ($A5). Si vous ne faites pas cela, alors la prochaine fois que vous réinitialiserez l'Apple //e, le sous-programme Reset trouvera que le vecteur de réinitialisation est invalide et effectuera une initialisation de type démarrage à froid forcé, éventuellement en transférant le contrôle au sous-programme d'amorçage du DOS ou à l'Applesoft.

Tableau 4-10. Vecteurs de la page 3

Adresse du vecteur		Fonction du vecteur
Dec	**Hex**	
1008 1009	$3F0 $3F1	Adresse du sous-programme qui gère une demande de BRK (arrêt en langage assembleur) (normalement $59, $FA)
1010 1011	$3F2 $3F3	Vecteur de réinitialisation (Reset) (voir le texte)
1012	$3F4	Octet de mise-sous-tension (« power-up ») (voir le texte)
1013 1014 1015	$3F5 $3F6 $3F7	Instruction de saut au sous-programme qui prend en charge la commande « & » de l'Applesoft (normalement $4C, $58, $FF).
1016 1017 1018	$3F8 $3F9 $3FA	Instruction de saut au sous-programme qui prend en charge la commande (Ctrl - Y) donnée par l'utilisateur.
1019 1020 1021	$3FB $3FC $3FD	Instruction de saut au sous-programme qui gère les interruptions non masquables
1022 1023	$3FE $3FF	Vecteur d'interruption (adresse du sous-programme qui exploite les demandes d'interruption) programme qui exploite les demandes d'interruption)

Organisation de la mémoire

La procédure Reset a un sous-programme qui génère l'octet de test de validité pour le vecteur courant de réinitialisation. Vous pouvez vous servir de ce sous-programme en faisant un appel de sous-programme à l'adresse - 1169 (hexadécimal $FB6F). Quand votre programme se termine, il peut remettre l'Apple //e dans son fonctionnement normal en restaurant le vecteur original de réinitialisation et à nouveau en appelant le sous-programme qui donnera une valeur correcte à l'octet de test de validité.

L'auto-test automatique

Si vous réinitialisez l'Apple //e en appuyant sur les touches Ctrl et ⌘ tout en appuyant sur la touche Reset et en la relachant, la procédure Reset va faire tourner l'auto-test implémenté dans le système. Une exécution avec succès de ce test vous assure que l'Apple IIe est bien opérationnel.

Avertissement
Le sous-programme d'auto-test teste la mémoire programmable de l'Apple IIe en y écrivant et en la relisant. Tous les programmes et les données dans la mémoire programmable seront détruits si vous faites exécuter l'auto-test.

L'auto-test prend plusieurs secondes à s'exécuter. Tandis qu'il tourne, l'affichage passe du noir au blanc et vice-versa deux fois. Si le test se termine normalement, l'Apple //e affiche un message « OK » et attend que vous demandiez une réinitialisation normale.

Si vous aviez fait fonctionner des programmes avant d'effectuer l'auto-test, certains commutateurs logiciels pouvaient être à un au moment d'exécuter l'auto-test. Si cela se produit, l'auto-test affichera un message d'erreur.

Eteignez l'appareil pendant plusieurs secondes, puis rallumez-le et refaites passer l'auto-test.

Utilisation du moniteur

Utilisation du moniteur

Le moniteur du système est un ensemble de sous-programmes implantés en MEM dans l'Apple //e. Le moniteur fournit une interface normale aux dispositifs d'E/S de base décrits au Chapitre 2. Les sous-programmes décrits dans le Chapitre 3 font partie du moniteur du système.

Le système d'exploitation des disquettes (DOS) et les interpréteurs BASIC utilisent ces sous-programmes par des appels directs à leur adresse de début, comme ce fut décrit au Chapitre 3 pour les sous-programmes d'E/S ; les adresses de début de tous les sous-programmes standards sont énumérées dans l'Annexe C. Si vous le désirez, vous pouvez appeler les sous-programmes standards de la même manière.

Vous pouvez faire exécuter la plupart des fonctions du moniteur depuis le clavier. Ce chapitre vous indique comment utiliser le moniteur pour

- regarder dans une ou plusieurs adresses de mémoire
- changer le contenu d'une adresse quelconque
- écrire des programmes en langage machine exécutables directement par le microprocesseur de l'Apple //e
- sauvegarder des blocs de données et des programmes sur cassettes magnétiques et les relire
- déplacer et comparer des blocs de mémoire
- appeler d'autres programmes depuis le moniteur

Appel du moniteur

Le moniteur du système débute à l'adresse $FF69 (décimal 65385 ou -151). Pour appeler le moniteur, vous faites une instruction CALL à cette adresse depuis le clavier ou depuis un programme en BASIC. Quand le moniteur est en ligne, son caractère de sollicitation, un astérisque (*), apparait sur le côté gauche de l'écran, suivi d'un curseur clignotant.

Pour utiliser le moniteur, vous tapez des commandes sur le clavier. Quand vous avez fini d'utiliser le moniteur, vous reviendrez au langage BASIC que vous utilisiez, en tapant Ctrl - Reset, ou en tapant Ctrl -C et en appuyant sur ↵, ou en tapant 3D0G, qui provoque l'exécution du programme résident — habituellement Applesoft — dont l'adresse est enregistrée dans l'instruction de saut en $3D0.

Syntaxe des commandes du moniteur

Pour donner un ordre au moniteur, vous tapez une ligne sur le clavier, puis vous appuyez sur ↵. Le moniteur acceptera la ligne en se servant du sous-programme d'E/S GETLN décrit au chapitre 3. Une commande du moniteur peut avoir jusqu'à 255 caractères de long, se terminant par un retour-chariot.

Une commande du moniteur peut inclure trois types d'information : des adresses, des valeurs de données et des caractères de commande. Vous devez taper les adresses et les valeurs de données en notation hexadécimale. La notation hexadécimale nécessite les dix chiffres décimaux (0-9) et les six premières lettres (A-F) pour représenter les seize valeurs de 0 à 15. Une paire de chiffres hexadécimaux représente des valeurs de 0 à 255, correspondant à un octet, et un groupe de 4 chiffres hexadécimaux peut représenter des valeurs de 0 à 65536, correspondant à un mot. Toute adresse dans l'Apple //e peut être représentée par quatre chiffres hexadécimaux.

Quand la commande que vous tapez contient une adresse, le moniteur accepte n'importe quel groupe de chiffres hexadécimaux. S'il y a moins de quatre chiffres dans le groupe, il ajoute des zéros en tête ; s'il y a plus de quatre chiffres hexadécimaux, le moniteur n'utilise que les quatre derniers. Il suit une procédure similaire lorsque la syntaxe d'une commande nécessite des valeurs de données à deux chiffres.

Chaque commande que vous tapez consiste en un caractère de commande, habituellement la première lettre du nom de la commande. Le moniteur reconnait 22 caractères de commande différents. Certains d'entre eux sont des symboles de ponctuation, d'autres des lettres majuscules, et d'autres des caractères de contrôle. Notez que, bien que le moniteur les reconnaisse et les interprète, les caractères de contrôle tapés sur une ligne d'entrée n'apparaissent pas sur l'écran. Voir le « Résumé des commandes du moniteur » à la fin de ce chapitre.

Ce chapitre contient de nombreux exemples d'utilisation des commandes du moniteur. Dans les exemples, les commandes et les valeurs que vous taperez sont imprimées en caractère normaux alors que les réponses du moniteur sont dans un autre caractère.

Évidemment, quand vous essayerez les exemples, tous les caractères apparaîtront sur l'écran sous la même forme. Certaines des valeurs affichées par votre Apple *//e* peuvent être différentes des valeurs imprimées dans ces exemples, puisque ce sont des variables enregistrées en mémoire programmable.

Les commandes relatives aux mémoires

Quand vous vous servez du moniteur pour examiner et changer le contenu d'une mémoire, il garde l'adresse de la dernière position de mémoire dont vous cherchiez la valeur et l'adresse de la position suivante de celle dont la valeur a changé. On les appelle la dernière adresse ouverte et la prochaine modifiable.

Examen du contenu d'une mémoire

Quand vous tapez l'adresse d'une position de mémoire et que vous appuyez sur ↵, le moniteur répond avec l'adresse que vous avez tapée, un tiret, un espace et la valeur mémorisée à cette position, comme ceci

```
*E000

E000- 20

*33

0033- AA
*
```

A chaque fois que le moniteur affiche la valeur enregistrée dans une position, il sauvegarde l'adresse de cette position comme la dernière adresse ouverte et comme la prochaine adresse modifiable.

Affichage d'une zone de mémoire (dump)

Quand vous tapez un point (.) suivi par une adresse, et puis que vous appuyez sur ↵, le moniteur affiche les contenus d'une zone de mémoire : les valeurs des données contenues dans les positions de mémoire depuis celle suivant la dernière adresse ouverte jusqu'à l'adresse que vous aviez tapée après le point. Le moniteur sauvegarde la dernière adresse affichée comme la dernière adresse ouverte et la prochaine modifiable. Dans ces exemples, la quantité de données affichées dépend de la différence entre l'adresse après le point et la dernière adresse ouverte.

```
* 20

020- 00

* .2B

0021- 28 00 18 0F 0C 00 00
0028- A8 06 D0 07

* 300

0300- 99

* .315

0301- B9 00 08 0A 0A 0A 99
0308- 00 08 C8 D0 F4 A6 2B A9
0310- 09 85 27 AD CC 03

* .32A

0316- 85 41
0318- 84 40 8A 4A 4A 4A 4A 09
0320- C0 85 3F A9 5D 85 3E 20
0328- 43 03 20
```

L'affichage d'une zone de mémoires comprend plusieurs éléments différents d'information. La première ligne de cet affichage commence avec l'adresse de la position suivant la dernière adresse ouverte ; toutes les autres lignes commencent par des adresses qui terminent alternativement par zéro et huit, et il n'y a jamais plus de huit valeurs de données affichées sur une seule ligne dans un affichage d'une zone de mémoire.

Quand le moniteur fait un « dump », il commence à l'adresse suivant immédiatement la dernière adresse ouverte et affiche cette adresse et la valeur de la donnée mémorisée là. Il affiche ensuite les valeurs des positions successives jusqu'à et y compris la position dont vous aviez tapé l'adresse, mais seulement jusqu'à concurrence de huit valeurs par ligne. Quand il atteint une position dont l'adresse est un multiple de huit - c'est-à-dire, une qui se termine par un 8 ou un 0 - il affiche cette adresse comme le début d'une nouvelle ligne, puis continue à afficher d'autres valeurs.

Après que le moniteur est affiché la valeur à la position dont vous avez spécifiée l'adresse dans la commande, il arrête l'affichage de la zone de mémoire et garde cette position comme la dernière adresse ouverte et la prochaine modifiable. Si l'adresse spécifiée sur la ligne d'entrée est plus petite que l'adresse de la dernière ouverte, le moniteur n'affiche que l'adresse et la valeur de cette position suivant la dernière adresse ouverte.

Vous pouvez combiner les deux commandes, en ouvrant une adresse et en affichant une zone de mémoire, par une simple concatenation des deux : tapez la première adresse, un point, et la seconde adresse. Cette combinaison de deux adresses séparées par un point est appelée une zone de mémoire.

```
* 300.32F

0300- 99 B9 00 08 0A 0A 0A 99
0308- 00 08 C8 D0 F4 A6 2B A9
0310- 09 85 27 AD CC 03 85 41
0318- 84 40 8A 4A 4A 4A 4A 09
0320- C0 85 3F A9 5D 85 3E 20
0328- 43 03 20 46 03 A5 3D 4D

* 30.40

0030- AA 00 FF AA 05 C2 05 C2
0038- 1B FD D0 03 3C 00 40 00
0040- 30

* E015.E025

E015- 4C ED FD
E018- A9 20 C5 24 B0 0C A9 8D
E020- A0 07 20 ED FD A9
*
```

En appuyant sur la touche ↵, on fait afficher par le moniteur une ligne de « dump » de mémoire ; c'est-à-dire un affichage d'une zone de mémoire depuis la position qui suit la dernière adresse ouverte jusqu'à la prochaine adresse multiple de huit. Le moniteur sauvegarde l'adresse de la dernière adresse affichée comme dernière adresse ouverte et prochaine adresse modifiable.

```
* 5

0005- 00

* ↵

00 00

* ↵

0008- 00 00 00 00 00 00 00 00

*32

032- FF

* ↵

AA 00 C2 05 C2

* ↵

0038- 1B FD D0 03 3C 00 3F 00
*
```

Modifications des contenus d'une mémoire

Le paragraphe précédent vous a montré comment afficher des valeurs enregistrées dans la mémoire de l'Apple //e ; ce paragraphe vous montre comment changer ces valeurs. Vous pouvez modifier n'importe quelle position en MEV (mémoire programmable) et vous pouvez aussi changer les commutateurs logiciels et les dispositifs périphériques en modifiant les positions de mémoire qu'on leur a affectées.

Avertissement
Utilisez ces commandes avec précaution. Si vous modifiez les positions de la page zéro utilisées par l'Applesoft et le DOS, vous pourriez perdre des programmes ou des données enregistrées en mémoire.

Changement d'un octet

La commande précédente a gardé l'adresse de la prochaine position modifiable ; ces commandes s'en servent. Dans l'exemple suivant, vous allez ouvrir l'adresse 0, puis tapez un deux points (:) suivi d'une valeur.

```
* 0

0000- 00

* :5F
```

Le contenu de la prochaine adresse modifiable vient juste d'être changé en la valeur que vous avez tapé, comme vous pouvez le voir en examinant le contenu à cette adresse :

```
* 0

0000- 5F
*
```

Vous pouvez aussi combiner ouverture et changement en une seule opération en tapant une adresse suivie d'un deux points et d'une valeur. Dans l'exemple, vous tapez l'adresse à nouveau pour vérifier la modification.

```
* 302:42

* 302

0302- 42
*
```

Quand vous modifiez le contenu d'une position de mémoire, la valeur qui était contenue à cette position disparait et ne pourra plus jamais être revue. La nouvelle valeur y restera tant qu'on ne la remplacera pas par une autre valeur.

Changement dans des adresses consécutives

Il n'est pas nécessaire de taper une commande séparée avec une adresse, un deux-points, une valeur, et ↵ pour chaque position que vous voulez modifier. Vous pouvez changer les contenus d'adresses consécutives jusqu'à quatre-vingt six en une

seule fois (ou plus si vous oubliez les zéros en tête des valeurs) en ne tapant que l'adresse initiale et le deux-points suivi par toutes les valeurs séparées par des espaces, et se terminant par ↵. Le moniteur enregistrera consciencieusement les valeurs consécutives dans les positions consécutives, en commençant à la position dont vous aviez tapé l'adresse. Après avoir traité la chaîne de valeurs, il prend la position suivant la dernière position modifiée comme prochaine adresse modifiable. Donc vous pouvez continuer à changer des positions consécutives sans taper une adresse sur la prochaine ligne, en tapant un autre deux-points et d'autres valeurs. Dans ces exemples, vous changez d'abord quelques positions, puis vous les examinez pour vérifier les changements.

```
* 300:69 01 20 ED FD 4C 0 3

* 300

0300- 69

* ↵

01 20 ED FD 4C 00 03

* 10:0 1 2 3

* :4 5 6 7

* 10.17

0010- 00 01 02 03 04 05 06 07
*
```

Déplacement de données en mémoire

Vous pouvez recopier un bloc de données mémorisées dans un domaine de positions de mémoire, depuis une zone en mémoire vers une autre, en utilisant la commande MOVE du moniteur. Pour déplacer un domaine de mémoire, vous devez dire au moniteur où les données sont situées en mémoire - adresses de la source - et où vous voulez les recopier - adresses de la destination. Vous donnerez ces informations au moniteur au moyen de trois adresses : l'adresse de la première position de la destination et les adresses de la première et de la dernière position de la source. Vous devez spécifier les adresses de début et de fin du domaine-source en les

séparant par un point. Vous devez séparer l'adresse de destination des adresses de la source avec un caractère plus-petit-que (<), qui peut vous faire penser à une flèche pointant dans la direction du déplacement des données. Finalement, vous dites au moniteur qu'il s'agit d'une commande de déplacement (« MOVE ») en tapant la lettre M. Le format de la commande MOVE complète ressemble à cela :

{destination} < {début} . {fin} M

Quand vous taperez la commande actuelle, les mots entre parenthèses devront être remplacés par des adresses hexadécimales, les crochets et les espaces devront être omis. Voici quelques exemples de déplacement de mémoire. D'abord vous examinez les valeurs mémorisées dans un domaine de mémoire, puis vous enregistrez plusieurs valeurs dans un autre domaine de mémoire ; les lignes de commandes présentes se terminent par M :

```
*0.F

0000-5F 00 05 07 00 00 00 00
0008-00 00 00 00 00 00 00 00

*300:09 8D 20 ED FD A9 45 20 DA FD 4C 00 03

*300.30C

0300- A9 8D 20 ED FD A9 45 20
0308- DA FD 4C 00 03

*0  300.30CM

*0.C

0000- A9 8D 20 ED FD A9 45 20
0008- DA FD 4C 00 03

*310 8.AM

*310.312

0310- DA FD 4C

*2  7.9M

*0.C
```

Les commandes relatives aux mémoires

```
0000- A9 8D 20 DA FD A9 45 20
0008- DA FD 4C 00 03
*
```

Le moniteur déplace une copie des données enregistrées dans le domaine-source vers le domaine de destination. Les valeurs dans le domaine-source ne sont pas perturbées. Le moniteur se souvient de la dernière adresse dans le domaine-source comme la dernière adresse ouverte, et de la première position du domaine-source comme la prochaine adresse modifiable. Si la seconde adresse dans la spécification du domaine-source est plus petite que la première, alors seulement une seule valeur sera déplacée (celle de la première adresse du domaine).

Si l'adresse de destination de la commande MOVE est à l'intérieur du domaine-source d'adresses, alors d'étranges choses (et quelquefois magnifiques) se produisent : les positions entre le début du domaine-source et l'adresse de destination sont traitées comme un sous-domaine et les valeurs de ce sous-domaine sont recopiées dans le domaine-source. Voir le paragraphe « Astuces d'utilisation des commandes du moniteur » pour une application intéressante de cette caractéristique.

Comparaison de données en mémoire

Vous pouvez vous servir de la commande VERIFY pour comparer deux domaines de mémoire en utilisant le même format que pour déplacer un domaine de mémoire d'un endroit à un autre. En fait, la commande VERIFY peut être utilisée immédiatement après un MOVE pour s'assurer que le déplacement a réussi. La commande VERIFY, comme la commande MOVE, a besoin d'un domaine et d'une destination. La syntaxe de la commande VERIFY est :

{destination} < {début} . {fin} V

Le moniteur compare les valeurs des positions de la source avec les valeurs des positions commençant à l'adresse de destination. Si des valeurs ne sont pas identiques, le moniteur affiche l'adresse dans laquelle une différence a été trouvée et les deux valeurs qui diffèrent. Dans l'exemple, vous allez enregistrer des données dans le domaine des positions de 0 à $D, les copier aux positions commençant en $300 avec la commande MOVE, puis les comparer en utilisant la commande VERIFY. Si vous utilisez la commande VERIFY, après avoir changé la valeur de la position 6 en $E4, le changement sera détecté.

```
*0:D7 F2 E9 F4 F4 E5 EE A0 E2 F9 A0 C3 C4 C5
*300  0.DM
*300  0.DV
*6:E4
*300  0.DV

0006-E4 (EE)
*
```

Si la commande VERIFY trouve une différence, l'adresse de la position dans le domaine-source dont la valeur diffère de sa contrepartie est affichée dans le domaine de destination. S'il n'y a aucune différence, rien n'est affiché. La commande VERIFY laisse intactes les valeurs dans les deux domaines. La dernière adresse ouverte est la dernière du domaine-source, et la prochaine adresse modifiable est la première adresse du domaine-source, exactement comme pour la commande MOVE. Si l'adresse de fin du domaine est inférieure à l'adresse de début, seules les premières positions des domaines seront comparées. Comme pour la commande MOVE, la commande VERIFY produit aussi des choses inhabituelles si l'adresse de la destination est à l'intérieur du domaine-source ; voir le paragraphe « Astuces d'utilisation de commandes du moniteur ».

Les commandes relatives aux registres

Même si les contenus des registres internes du 6502 changent quand vous utilisez le moniteur, vous pouvez examiner les valeurs que les registres contenaient quand le moniteur a pris le contrôle, soit parce que vous l'avez appelé, soit parce que le programme dont vous cherchez les erreurs s'est arrêté sur l'instruction BRK (« break » ou arrêt). Vous pouvez aussi enregistrer de nouvelles valeurs de registres qui seront utilisées quand vous demanderez l'exécution d'un programme depuis le moniteur avec la commande GO, décrite plus loin.

Examen et modification des registres

Quand vous appelez le moniteur, il sauvegarde les contenus des registres du 6502. Les registres sont mémorisés dans l'ordre A, X, Y, P (registre d'état du processeur), et S (registre du pointeur de pile), en commençant à l'adresse $45 (décimal 69). Quand vous donnez au moniteur l'ordre GO, le moniteur charge les registres à partir de ces cinq positions de mémoire avant d'exécuter la première instruction de votre programme.

En tapant Ctrl -E et en appuyant sur ↵ , on stimule la commande EXAMINE, qui affiche les valeurs mémorisées des registres et prend comme prochaine adresse modifiable celle de la position contenant la valeur du registre A. Après avoir utilisé la commande EXAMINE, vous pouvez changer les valeurs de ces mémoires en tapant un deux-points puis en tapant les nouvelles valeurs séparées par des espaces. Dans l'exemple suivant, on affiche les registres, on change les deux premiers, et ensuite on les affiche à nouveau pour vérifier les modifications.

```
* Ctrl – E

A=0A X=FF Y=D8 P=B0 S=F8

*:B0 02

* Ctrl – E

A=B0 X=02 Y=D8 P=B0 S=F8
*
```

Les commandes relatives au magnétophone à cassettes

L'Apple //e a deux prises pour brancher un magnétophone à cassettes audio. Avec le magnétophone branché, vous pouvez utiliser les commandes du moniteur décrites plus loin pour sauvegarder sur cassette les contenus d'une zone de mémoire et les rappeler pour les utiliser plus tard.

Sauvegarde de données sur cassette

La commande WRITE du moniteur sauvegarde jusqu'à 65 536 positions de mémoire sur cassette. Pour sauvegarder une zone de mémoires, vous donnez au moniteur l'adresse de début et l'adresse de fin de la zone, suivies de la lettre W (pour WRITE ou écrire), comme ceci :

{début} . {fin} W

N'appuyez pas sur ↵ tout de suite : mettez d'abord le magnétophone à cassette en mode d'enregistrement (« record ») et laissez la bande se dérouler pendant une seconde, puis appuyez sur ↵ .Le moniteur va écrire pendant dix secondes, une tonalité sur la bande et ensuite écrire les données. La tonalité sert d'en-tête : plus tard, quand le moniteur lira la bande, l'en-tête permettra au moniteur de se mettre au pas avec le signal sur la bande. Quand le moniteur a fini d'écrire la zone-mémoire spécifiée, il va émettre une sonnerie et afficher le caractère de sollicitation. Il faut que vous

rembobiniez la cassette et prépariez une étiquette décrivant la zone-mémoire enregistrée sur la bande et à quoi elle correspond.

Voici un petit exemple que vous pouvez sauvegarder et utiliser plus tard pour essayer la commande READ. Rappelez-vous que vous devez démarrez la cassette en mode enregistrement avant d'appuyez sur ↵ après avoir tapé la commande WRITE.

```
*0.FF FF AD 30 C0 88 D0 04 C6 01 F0 08 CA
D0 F6 A6 00 4C 02 00 60

*0.14


0000- FF FF AD 30 C0 88 D0 04
0008- C6 01 F0 08 CA D0 F6 A6
0010- 00 4C 02 00 60


*0.14W

*
```

Cela dure environ 35 secondes au total pour sauvegarder 4 906 positions de mémoire précédées d'un en-tête de dix secondes. Ce qui donne un taux de transfert moyen de données d'environ 1 350 bits par seconde.

La commande WRITE écrit une valeur supplémentaire après avoir écrit les valeurs de la zone-mémoire. Cette valeur est la somme de contrôle (« checksum »), qui est la somme sur 8 bits de toutes les valeurs de la zone-mémoire. Quand le moniteur lit la bande, il utilise cette valeur pour déterminer si les données ont été décrites et lues correctement (voir plus bas).

Lecture de données sur cassette

Une fois que vous avez sauvegardé une zone-mémoire sur cassette avec la commande WRITE du moniteur, vous pouvez lire cette zone-mémoire et la charger dans l'ordinateur en utilisant la commande READ. Les valeurs des données que vous avez enregistrées sur cassette ne sont pas obligées d'être rechargées dans le même domaine que celui d'où elles venaient ; vous pouvez dire au moniteur de mettre ces valeurs dans un domaine quelconque de la mémoire de l'ordinateur, pourvu qu'il ait la même taille que celui que vous avez sauvegardé. Le format de la commande READ est le même que celui de la commande WRITE, sauf que la lettre de commande est R :

{début} . {fin} R

Et à nouveau, après avoir tapé cette commande, n'appuyez pas sur [↵] . Il faut démarrez le magnétophone à cassettes en mode de lecture (« play ») et attendre quelques secondes. Bien que la commande WRITE mette un en-tête de dix secondes au début de la bande, la commande READ n'a besoin que de trois secondes de cet en-tête pour se synchroniser sur le signal de la bande. Vous devez laisser défiler quelques secondes de bande avant d'appuyer sur [↵] pour laisser le temps à la sortie du magnétophone à cassette de fournir une tonalité stable.

Cet exemple a deux parties. Tout d'abord, vous mettez à zéro tout une zone-mémoire, vous en vérifiez les contenus et, ensuite, vous tapez la commande READ, mais vous n'appuyez pas sur [↵] .

```
*0:0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
```

```
*0.14
```

```
0000- 00 00 00 00 00 00 00 00
0008- 00 00 00 00 00 00 00 00
0010- 00 00 00 00 00
```

```
*0.14R
```

Puis vous démarrez la cassette en mode lecture ou « play », vous attendez quelques secondes et vous appuyez sur [↵]. Après que le moniteur ait émis la tonalité et affiché le solliciteur, examinez la zone-mémoire pour constater que les valeurs ont été lues correctement sur la bande.

```
*0.14
```

```
0000- FF FF AD 30 C0 88 D0 04
0008- C6 01 F0 08 CA D0 F6 A6
0010- 00 4C 02 00 60
*
```

Après avoir lu les valeurs sur la bande, le moniteur lit la valeur de la somme de contrôle. Il calcule la somme de contrôle sur les données qu'il a lues et la compare avec la somme de contrôle de la bande. Si les deux sommes de contrôle diffèrent, le moniteur émet une sonnerie par le haut-parleur et affiche « ERR ». Ceci vous signale qu'il y a eu un problème de lecture de la bande et que les valeurs enregistrées en mémoire ne sont pas celles enregistrées sur la bande. Si les deux sommes sont identiques, le moniteur vous enverra juste une tonalité et affichera le caractère de sollicitation.

Les commandes diverses du moniteur

Ces commandes vous autorisent à changer l'affichage vidéo du mode normal au mode inverse et vice-versa, et d'attribuer l'entrée et la sortie à des périphériques branchés aux connecteurs d'extension.

Affichage inversé et normal

Vous pouvez contrôler depuis le moniteur, la valeur du masque inverse-normal utilisé par le sous-programme COUT (décrit au chapitre 3) de telle sorte que toutes les sorties du moniteur apparaissent en mode inverse. La commande INVERSE, I, met le masque à une valeur telle que toutes les entrées et les sorties à venir soient affichées en mode inverse. Pour revenir au mode normal d'affichage des sorties, utilisez la commande NORMAL, N.

```
*0.F

0000- 0A 0B 0C 0D 0E 0F D0 04
0008- C6 01 F0 08 CA D0 F6 A6

*I
*0.F

0000- 0A 0B 0C 0D 0E 0F D0 04
0008- C6 01 F0 08 CA D0 F6 A6

*N
*0.F

0000- 0A 0B 0C 0D 0E 0F D0 04
0008- C6 01 F0 08 CA D0 F6 A6
*
```

Retour au BASIC

Utilisez la commande BASIC, Ctrl – B, pour quitter le moniteur et entrer dans le BASIC qui était actif quand vous avez commencé à travailler avec le moniteur. Normalement, c'est le BASIC Applesoft, à moins que vous n'ayez délibérément commuté sur le BASIC Entier. Tout programme ou toute variable que vous aviez antérieurement sous BASIC sera perdu. Si vous voulez reprendre en BASIC, avec votre programme et vos variables antérieures intactes, utilisez la commande CONTINUE BASIC, Ctrl – C. Si vous êtes sous le système d'exploitation de disquettes (DOS), appuyez sur Ctrl - Reset ou tapez

```
3D0G
```

pour revenir sur le langage que vous étiez en train d'utiliser, avec vos programme et données intacts.

Si vous tapez cette dernière commande, assurez-vous que le dernier caractère tapé est bien un zéro et non la lettre O. La lettre G est la commande GO du moniteur, décrite au paragaphe « Programmes en langage-machine ».

Réaffectation de l'entrée et de la sortie

La commande PRINTER, activée par Ctrl – P, dirige toute sortie normalement destinée à l'écran vers une carte d'interface dans un connecteur d'extension spécifié de 1 à 7. Il doit y avoir une carte d'interface dans ce connecteur, ou alors vous allez perdre le contrôle de l'ordinateur et votre programme et ses variables pourront être perdus... Le format de cette commande est

{numéro du connecteur} Ctrl – P

Une commande PRINTER au connecteur 0, fera commuter le flux des caractères de sortie vers l'écran vidéo de l'Apple //e.

Avertissement
Ne donnez pas un ordre PRINTER avec le connecteur 0 pour désactiver les sous-programmes pour 80 colonnes, même si vous utilisez cet ordre pour les activer au connecteur 3. La commande Ctrl – P fonctionne, mais elle ne met hors-fonction que les sous-programmes, en laissant quelques uns des commutateurs logiciels positionnés pour l'affichage en 80 colonnes.

De même que la commande PRINTER fait commuter le flux de sortie, la commande KEYBOARD remplace le dispositif normal d'entrée de l'Apple //e par une carte d'interface dans un connecteur d'extension particulier. Le format de la commande KEYBOARD est :

numéro du connecteur (Ctrl) – K

Un numéro de connecteur 0 dans la commande KEYBOARD oriente le moniteur pour qu'il accepte des entrées du clavier faisant partie de l'Apple //e.

Les commandes PRINTER et KEYBOARD sont les équivalents exacts des commandes de BASIC PR# et IN#. Pour en savoir plus sur la façon dont ces commandes fonctionnent, se reporter au paragraphe « Les liaisons d'E/S standards » au chapitre 3.

Arithmétique hexadécimale

Le moniteur effectue aussi des opérations d'addition et de soustraction héxadécimales sur un octet. Vous tapez simplement une ligne dans un de ces formats :

valeur + valeur
valeur – valeur

L'Apple //e effectue les opérations arithmétiques et affiche le résultat, comme le montre ces exemples :

```
*20 + 13
=33
*4A-C
=3E
*FF+4
=03
*3-4
=FF
*
```

Astuces d'utilisation des commandes du moniteur

Ce paragraphe décrit des manières complexes d'utiliser les commandes du moniteur.

Lignes à commandes multiples

Vous pouvez mettre autant de commandes du moniteur sur une simple ligne que vous le désirez, si vous les séparez par des espaces et si le nombre total de caractères dans la ligne reste inférieur à 254. Des commandes adjacentes d'une seule lettre comme L, S, I et N doivent être séparées par des espaces.

Vous pouvez mélanger librement toutes les commandes sauf la commande STORE (:) ou mémorisation de données. Puisque le moniteur prend toutes les valeurs qui suivent un deux-points et les place dans des adresses de mémoire consécutives, la dernière valeur dans une commande STORE doit être suivie d'une lettre de commande avant qu'une autre adresse ne soit rencontrée dans la ligne de commandes. Vous pouvez utiliser la commande N comme lettre requise dans de tels cas ; elle n'a aucun effet habituellement et peut être utilisée n'importe où.

Dans l'exemple suivant, on affiche une zone-mémoire, on la modifie, et on l'affiche à nouveau, en une seule ligne de commandes.

```
*300.307 300:18 69 1 N 300.302

0300- 00 00 00 00 00 00 00 00
0300- 18 69 01
*
```

Si le moniteur rencontre un caractère dans la ligne de commandes qu'il ne reconnaît ni comme un chiffre hexadécimal, ni comme un caractère valide de commande, il exécute toutes les commandes de cette ligne jusqu'à ce caractère, puis s'arrête en faisant un drôle de bruit et ignore le reste de la ligne.

Remplissage de la mémoire

La commande MOVE peut être utilisée pour recopier une séquence de données à travers la mémoire. Pour ce faire, enregistrez d'abord la séquence de données dans les premières adresses de la zone de mémoire :

```
*300:11 22 33

*
```

Rappelez-vous du nombre de données dans la séquence à recopier : ici c'est 3. Utilisez le nombre pour calculer les adresses de la commande MOVE, comme ceci :

{début + nombre} < {début} . {fin-nombre} M

La commande MOVE va d'abord recopier la séquence dans les adresses qui suivent immédiatement la séquence originale, ensuite recopiera cette séquence se suivant elle-même, et ainsi de suite jusqu'à remplir la zone entière.

```
*303<300.32DM
*300.32F
```

```
0300- 11 22 33 11 22 33 11 22
0308- 33 11 22 33 11 22 33 11
0310- 22 33 11 22 33 11 22 33
0318- 11 22 33 11 22 33 11 22
0320- 33 11 22 33 11 22 33 11
0328- 22 33 11 22 33 11 22 33
*
```

Vous pouvez utiliser une astuce similaire avec la commande VERIFY pour vérifier qu'une séquence se répète à travers la mémoire. Ceci est particulièrement utile pour vérifier que toute une zone-mémoire donnée contient la même valeur. Dans cet exemple, on remplit d'abord la zone-mémoire de $300 à $320 avec des zéros et on le vérifie, puis on change une position de mémoire et on le revérifie, pour voir la commande VERIFY détecter la différence :

```
*300:0
*301< 300.31FM
*301< 300.31FV
*304:02
*301< 300.31FV
```

```
0303-00 (02)
0304-02 (00)
*
```

Répétition de commandes

Vous pouvez créer une ligne de commandes qui répète une ou plusieurs commande à l'infini. Vous faites cela en commençant la partie de la ligne de commandes que vous voulez répéter avec une lettre de commande, telle que N, et en terminant par la séquence 34:n, où n est un nombre héxadécimal qui spécifie la position dans la ligne de commande où vous voulez commencer la boucle de répétition ; pour le premier caractère dans la ligne, n = 0. La valeur pour n doit être suivie d'un espace pour que le bouclage s'effectue correctement.

Cette astuce bénéficie du fait que le moniteur utilise un registre d'index pour parcourir la mémoire-tampon du clavier, commençant à l'adresse $200. Chaque fois que le moniteur exécute une commande, il enregistre la valeur de l'index dans l'adresse $34 ; quand cette commande se termine, le moniteur recharge le registre d'index avec la valeur à l'adresse $34. Par la dernière commande qui fait changer le contenu de la position d'adresse $34, vous changez cet index pour que le moniteur récupère le prochain caractère de commande à une position antérieure dans la mémoire-tampon.

La seule façon d'arrêter une boucle de ce type est d'appuyer sur Ctrl - Reset ; c'est ainsi que se termine cet exemple.

```
*N 300 302 34:0

0300- 11
0302- 33
0300- 11
0302- 33
0300- 11
0302- 33
0300- 11
0302- 33
0300- 11
0302- 33
0300- 11
0302- 33
030
*
```

Création de vos propres commandes

La commande USER, Ctrl – Y, force le moniteur à sauter à l'adresse de mémoire $3F8. Vous pouvez y mettre une instruction JMP qui sautera à votre programme en langage-machine. Votre programme peut alors examiner les registres et les pointeurs du moniteur ou la mémoire-tampon d'entrée elle-même. Par exemple, voici un programme qui affiche tout ce qui sera tapé après

Ctrl – Y. Ce programme débute à l'adresse $300 ; la ligne de commande qui commence avec $3F8 permet d'enregistrer à l'adresse $3F8 un saut à l'adresse $300.

```
*300:A4 34 B9 00 02 20 ED FD C8 C9 8D D0 F5 4C 69 FF

*3F8:4C 00 03

*Ctrl – Y THIS IS A TEST

THIS IS A TEST
*
```

Les programmes en langage-machine

La raison principale de la programmation en langage-machine est une exécution plus rapide. Un programme en langage-machine peut s'exécuter beaucoup plus vite que le même programme écrit en langages évolués comme BASIC ou PASCAL, mais la version en langage-machine demande habituellement plus de temps pour être écrite. Il y a d'autres raisons de programmer en langage-machine : si vous voulez que votre programme fasse quelque chose qui ne soit pas prévu dans votre langage évolué, ou si vous voulez seulement avoir le plaisir de réussir à utiliser le langage-machine pour travailler directement sur des bits et des octets.

Si vous n'avez jamais utilisé le langage-machine, vous aurez besoin d'apprendre les instructions du 6502 répertoriées à l'annexe A. Pour devenir un expert de la programmation en langage-machine, vous devrez y passer du temps et étudier un des livres sur la programmation du 6502 donnés dans la Bibliographie.

Vous pourrez obtenir un affichage de votre programme en hexadécimal, le déplacer dans la mémoire, ou le sauvegarder sur cassette et le rappeler à nouveau en mémoire en utilisant les commandes décrites dans les paragraphes précédents. Dans ce paragraphe les commandes du moniteur sont décrites spécialement afin que vous utilisiez pour créer, écrire, et mettre au point vos programmes en langage-machine.

Exécution d'un programme

La commande du moniteur que vous utiliserez pour faire démarrer votre programme en langage-machine est la commande GO. Quand vous taperez une adresse et la lettre G, l'Apple *//e* commencera l'exécution des instructions en langage-machine à partir de l'adresse spécifiée. En ne tapant que G, l'exécution débute à la dernière adresse ouverte.

Le moniteur traite ce programme comme un sous-programme : il doit se terminer par RTS (retour d'un sous-programme), l'instruction qui redonne le contrôle au moniteur.

Le moniteur a des fonctions caactéristiques spéciales qui vous facilitent l'écriture et la recherche d'erreurs des programmes en langage-machine, mais avant de les étudier, voici un petit programme en langage-machine que vous pouvez exécuter en utilisant les commandes simples du moniteur déjà décrites. Dans cet exemple, le programme affiche les lettres A à Z : vous commencez à l'enregistrer à l'adresse $300, vous l'examinez pour vous assurer qu'il a été tapé correctement, puis vous tapez 300G pour commencer son exécution.

```
*300:A9 C1 20 ED FD 18 69 1 C9 DB D0 F6 60

*300:30C

0300- A9 C1 20 ED FD 18 69 01
0308- C9 DB D0 F6 60

*300 G

ABCDEFGHIJKLMNOPQRSTUVWXYZ
*
```

Désassemblage d'un programme

Un programme en langage-machine affiché en hexadécimal n'est pas une chose facile à lire et à comprendre. Pour rendre ce travail un peu plus facile, les programmes en langage-machine sont habituellement écrits en langage d'assemblage et convertis en code ou langage-machine par des programmes appelés assembleurs.

La commande LIST du moniteur affiche un programme en code machine dans sa représentation en langage d'assemblage. Au lieu d'un charabia en hexadécimal non structuré, la commande LIST affiche chaque instruction sur une ligne séparée, avec un nom d'instruction à trois lettres, ou mnémonique, et une opérande en hexadécimal aligné. La commande LIST convertit aussi les adresses relatives utilisées dans les instructions de branchement en adresse absolues.

Puisque les programmes qui traduisent du langage d'assemblage en langage machine s'appellent des **assembleurs**, un programme comme la commande LIST qui traduit du langage-machine en langage d'assemblage s'appelle un désassembleur.

Le mot **mnémonique** a la même racine que mémoire et fait référence à des acronymes courts dont on se souvient plus facilement que les codes d'opération en hexadécimal : par exemple, pour mettre à zéro la retenue (« clear carry ») vous écrivez CLC au lieu de $18.

La commande LIST du moniteur est de la forme :

{adresse} L

La commande LIST débute à une adresse spécifiée et affiche autant de mémoire qu'il est nécessaire pour remplir un écran (20 lignes) d'instructions, comme le montre l'exemple suivant :

```
* 300L

0300-  A9  C1     LDA   $C1
0302-  20  ED FD  JSR   $FDED
0305-  18         CLC
0306-  69  01     ADC   $01
0308-  C9  DB     CMP   $DB
030A-  D0  F6     BNE   $0302
030C-  60         RTS
030D-  00         BRK
030E-  00         BRK
030F-  00         BRK
0310-  00         BRK
0311-  00         BRK
0312-  00         BRK
0313-  00         BRK
0314-  00         BRK
0315-  00         BRK
0316-  00         BRK
0317-  00         BRK
0318-  00         BRK
0319-  00         BRK
*
```

Les sept premières lignes de cet exemple sont la représentation du programme en langage assembleur que vous avez tapé dans l'exemple précédent. Le reste des lignes sont les instructions BRK seulement si cette partie de mémoire contient des zéros : d'autres valeurs seront désassemblés comme d'autres instructions.

Le moniteur sauvegarde l'adresse que vous avez spécifiée dans la commande LIST, mais non en tant que dernière adresse ouverte utilisée par les autres commandes. A la place, le moniteur sauvegarde l'adresse en tant que compteur ordinal ou compteur d'instructions du programme, qu'il utilise pour pointer des adresses à l'intérieur du programme. Chaque fois que le moniteur effectue une commande LIST, il met le compteur ordinal à la valeur de l'adresse suivant immédiatement la dernière adresse affichée sur l'écran, de telle sorte que si vous tapez une autre commande LIST, il affichera un autre plein écran d'instructions, commençant où s'était arrêté l'affichage précédent.

Le mini-assembleur

Sans un assembleur, vous aurez à écrire votre programme en langage-machine, à prendre les valeurs hexadécimales des codes d'opérations et des opérandes, et à les enregistrer en mémoire en utilisant les commandes présentées dans les paragraphes précédents. C'est exactement ce que vous aviez fait en exécutant les exemples précédents.

L'interpréteur BASIC Entier inclut un assembleur appelé mini-assembleur de l'Apple qui vous permet de taper des programmes dans l'Apple //e en utilisant la même formulation en langage d'assemblage que celle que la commande LIST affiche. On l'appelle un mini-assembleur parce qu'il n'inclut pas les étiquettes symboliques, une caractéristique importante de tous les assembleurs complets comme l'Assembleur/Éditeur dans le « DOS Tool Kit » ou boîte à outils du DOS (produit Apple numéro A2D0029).

Avant que vous ne puissiez utiliser le mini-assembleur l'Apple //e doit tout d'abord être en BASIC Entier. Quand vous démarrez l'ordinateur avec le DOS ou même avec BASIC, l'Apple //e charge l'interpréteur BASIC Entier du fichier-programme appelé INTBASIC sur la MEV à banc-commuté.

Pour avoir le BASIC Entier en ligne après avoir démarré l'ordinateur avec le DOS, tapez

```
INT
```

L'Apple //e affiche le caractère de sollicitation de l'interpréteur BASIC Entier (>) et un curseur.

> Si vous n'avez pas mis les sous-programmes pour 80 colonnes en fonction depuis que vous avez démarré avec le DOS, le curseur ressemble maintenant à un rectangle clignotant : c'est précisément un caractère espace affiché en mode clignotant. Ce qui indique que l'ancien moniteur est en fonctionnement (voir le chapitre 3). L'ancien moniteur est chargé en MEV en même temps que le BASIC Entier et le mini-assembleur ; à la prochaine activation des sous-programme pour 80 colonnes, ils copieront la version courante du moniteur de la MEM en MEV. Une fois que cela s'est produit, le moniteur courant est actif même avec le BASIC Entier, et le curseur est soit un damier clignotant soit un rectangle blanc non clignotant.

Après être parvenu dans le moniteur depuis le BASIC Entier, appelez le mini-assembleur en tapant :

```
F666G
```

C'est simplement la commande GO décrite plus haut démarrant le programme enregistré à partir de l'adresse $F666 — le mini-assembleur. Vous pouvez savoir que le mini-assembleur est en ligne à cause d'un point d'exclamation (!) en guise de caractère de sollicitation. Quand le mini-assembleur fonctionne, vous pouvez exécuter toute commande du moniteur en la faisant précéder du symbole ($). A côté de cela, le mini-assembleur a un ensemble d'instructions et une syntaxe qui lui sont propres.

Le mini-assembleur sauvegarde une seule adresse, celle du compteur ordinal. Avant de commencer à taper un programme, vous devez donner au compteur ordinal l'adresse où le mini-assembleur devra mémoriser votre programme. Faites ceci en tapant l'adresse suivie d'un deux-points.

Avec le deux-points, tapez la mnémonique pour la première instruction de votre programme, suivie d'un espace et de l'opérande de l'instruction (les formats des opérandes sont énumérées dans le tableau 5-1). Maintenant appuyez sur ↵. Le mini-assembleur convertit en hexadécimal la ligne que vous venez de taper, l'enregistre en mémoire en commençant à l'adresse du compteur ordinal, et ensuite la désassemble à nouveau et affiche la ligne désassemblée. Il affiche alors un solliciteur de la prochaine ligne à assembler.

Maintenant le mini-assembleur est prêt à accepter la deuxième instruction de votre programme. Pour lui indiquer que vous voulez que la deuxième suive la première, ne taper ni adresse ni deux points : taper seulement un espace et l'opérande et le mnémonique de la prochaine instruction, puis appuyez sur ↵. Le mini-assembleur assemblera cette ligne et attendra la suite.

Si la ligne que vous tapez contient une erreur, le mini-assembleur émet une certaine tonalité et affiche un accent circonflexe (^) sous ou près du caractère erroné dans cette ligne entrée. La plupart des erreurs sont le résultat de fautes typographiques : des mnémoniques mal écrits, des parenthèses manquantes, et ainsi de suite. Le mini-assembleur rejette aussi une ligne entrée si vous oubliez l'espace avant ou après un mnémonique ou si vous incluez un caractère étrange dans une valeur hexadécimale ou une adresse. Si l'adresse de destination d'une instruction de branchement est hors de portée d'un branchement (plus de 127 adresses de distance à l'adresse de l'instruction), le mini-assembleur marque ceci comme une erreur.

```
!300:LDX  02

0300-   A2  02      LDX     $02
! LDA $0,X

0302-   B5  00      LDA     $00,X
! STA $10,X

0304-   95  10      STA     $10,X
! DEX

0306-   CA          DEX
! STA $C030

0307-   8D  30  CD  STA     $C030
! BPL $302

030A-   10  F6      BPL     $0302
! BRK

030C-   00          BRK
```

Il y a deux façons de quitter le mini-assembleur et de revenir dans le moniteur. L'une consiste à taper la commande du moniteur, FF69G, précédée du signe dollar :

```
!$FF69G
*
```

L'autre façon de quitter le mini-assembleur est d'appuyer sur Ctrl - Reset, puis de taper

```
CALL-151
```

Utilisation du moniteur

Votre programme en langage d'assemblage est maintenant en mémoire. Vous pouvez l'afficher avec la commande LIST :

```
*300L

0300-   A2  02      LDX   #$02
0302-   B5  00      LDA   $00,X
0304-   95  10      STA   $10,X
0306-   CA          DEX
0307-   8D  30  CD  STA   $C030
030A-   10  F6      BPL   $0302
030C-   00          BRK
030D-   00          BRK
030E-   00          BRK
030F-   00          BRK
0310-   00          BRK
0311-   00          BRK
0311-   00          BRK
0313-   00          BRK
0314-   00          BRK
0315-   00          BRK
0316-   00          BRK
0317-   00          BRK
0318-   00          BRK
0319-00 BRK
*
```

Formats des instructions en mini-assembleur

Le mini-assembleur de l'Apple reconnaît 56 mnémoniques et 13 modes d'adressage utilisés en programmation en langage-machine du 6502. Ces mnémoniques sont standards, conformément au manuel de programmation Synertek (produit Apple numéro A2L0003), mais les formats d'adressage sont légèrement différents. Le tableau 5-1 montre quels sont les formats d'adressages standards d'Apple pour le langage d'assemblage du 6502.

Une adresse comprend un ou plusieurs chiffres hexadécimaux. Le mini-assembleur interprète les adresses de la même manière que le moniteur les interprète : si une adresse a plus de quatre chiffres hexadécimaux, alors il ne prend que les quatre derniers.

> Dans ce livre, les signes dollar ($) dans les adresses signifient que les adresses sont en notation hexadécimale. Ils sont ignorés par le mini-assembleur et peuvent être omis quand on tape les programmes.

Il n'y a pas de distinction syntaxique entre les modes d'adressage en page zéro et absolue. Si vous donnez une instruction au mini-assembleur qui puisse être utilisée dans les deux modes page zéro et adresse absolue, le mini-assembleur assemble cette instruction en mode absolu si l'opérande de cette instruction est plus grand que $FF, et il l'assemble en mode page-zéro si l'opérande est plus petit que $100.

Le mini-assembleur

Tableau 5-1. Formats des adresses en mini-assembleur

*Note : les instructions d'adresse implicite et Accumulateur n'ont pas d'opérandes

Mode d'adressage	Format	Notes
Accumulateur		*
Implicite		*
Immédiat	$ valeur	
Absolu	$ adresse	
Page zéro	$ adresse	
Indexé Page zéro	$ adresse , X $ adresse , Y	
Indexé Absolu	$ adresse , X $ adresse , Y	
Relatif	$ adresse	
Indexé indirect	($ adresse , X)	
Indirect indexé	($ adresse), Y	
Absolu indirect	($ adresse)	

Les instructions en mode accumulateur et à adressage implicite n'ont pas besoin d'opérandes.

Les instructions de branchement, qui utilisent le mode relatif d'adressage, demande l'adresse de destination du branchement. Le mini-assembleur calcule la distance relative pour la mettre automatiquement dans l'instruction. Si l'adresse de destination est à plus de 128 adresses de distance de l'instruction, le mini-assembleur émet une sonnerie, affiche un accent circonflexe (^) sous l'adresse de destination, et n'assemble pas la ligne.

Si vous donnez au mini-assembleur le mnémonique d'une instruction en guise d'opérande, ou si le mode d'adressage de l'opérande ne peut pas être utilisé avec l'instruction que vous entrez, le mini-assembleur n'acceptera pas la ligne.

Résumé des commandes du moniteur

Voici un résumé des commandes du moniteur, montrant le diagramme de la syntaxe de chacune d'elles. Les commandes du mini-assembleur sont incluses, même si elles ne sont disponibles que si le BASIC Entier est en ligne (voir le paragraphe « Le mini-assembleur »).

Examen des mémoires

{adrs}	Examine la valeur contenue dans une adresse.
{adrs1} . {adrs2}	Affiche les valeurs contenues dans toutes les adresses entre {adrs1} et {adrs2}.
↵	Affiche les valeurs jusqu'à huit adresses suivant la dernière position ouverte.

Modification des contenus de mémoire

{adrs} : {val} {val}...	Enregistre les valeurs dans des positions de mémoire consécutives à partir de adrs .
: {val} {val}...	Enregistre les valeurs en mémoire à partir de la prochaine adresse modifiable.

Déplacement et comparaison

{dest} < {début} . {fin} M	Copie les valeurs de la zone {début} . {fin} dans la zone commençant en dest .
{dest} < {début} . {fin} V	Compare les valeurs de la zone {début} . {fin} avec celles de la zone commençant en dest .

Les commandes sur registres

Ctrl – E	Affiche les positions dans lesquelles sont mémorisées les registres du 6502 et les ouvrent pour les modifier.

Les commandes pour cassettes

{début} . {fin}W	Écrit les valeurs de la zone-mémoire {début} . {fin} sur cassette, précédées d'un en-tête de dix secondes.
{début} . {fin}R	Lit les valeurs sur cassette, les mémorisant à partir de début et jusqu'à {fin}. Affiche « ERR » s'il y a erreur.

Diverses commandes du moniteur

I	Met en mode Inverse
N	Met en mode Normal l'affichage.
Ctrl – B	Réinitialise le langage en-ligne actuellement (habituellement Applesoft).
Ctrl – C	Retourne au langage en-ligne actuellement (habituellement Applesoft).
{val} + {val}	Additionne les deux valeurs et affiche le résultat hexadécimal.
{val} – {val}	Soustrait la seconde valeur de la première et affiche le résultat hexadécimal.

numéro de connecteur Ctrl – P	Dirige la sortie vers le dispositif dont la carte d'interface est dans le connecteur spécifié. Si c'est 0, accepte les entrées par le clavier.
Ctrl – Y	Saute au sous-programme en langage-machine situé à l'adresse $3F8

Exécution et listage des programmes

{adrs}G	Transfère le contrôle au programme en langage-machine débutant à adrs .
{adrs}L	Désassemble et affiche 20 instructions, en commençant à adrs . Les L suivants affichent 20 instructions de plus.

Le mini-assembleur

Le mini-assembleur n'est disponible qu'avec le BASIC Entier en ligne.

F666G	Appelle le mini-assembleur
$ {commande}	Exécute une commande du moniteur depuis le mini-assembleur.
$FF69G	Quitte le mini-assembleur.

Résumé des commandes du moniteur

Programmation pour les cartes périphériques

Programmation pour les cartes périphériques

Les sept connecteurs d'extension présents sur la carte-mère de l'Apple //e servent à installer des cartes de circuits électroniques contenant le hardware et les sous-programmes en mémoire morte nécessaires pour interfacer les dispositifs périphériques à l'Apple //e. Ces connecteurs ne sont pas de simples ports d'E/S ; les cartes de périphériques peuvent avoir accès aux lignes de données, d'adresses, et de contrôle à travers ces connecteurs. Les connecteurs d'extension sont numérotés de 1 à 7, et certains signaux, décrits plus loin, servent à sélectionner un connecteur particulier parmi d'autres.

Les anciens modèles Apple II et Apple II Plus ont un huitième connecteur d'extension : le connecteur 0, qui est normalement utilisé pour mettre une carte-langage ou une carte de MEM ; les fonctions de la carte-langage de l'Apple II sont implémentées sur la carte-mère de l'Apple //e.

Espaces de mémoire pour carte périphérique

Puisque le microprocesseur 6502 de l'Apple //e fait toutes ses E/S à travers des adresses de mémoire, les parties de l'espace de la mémoire de l'Apple //e ont été attribuées pour l'usage exclusif des cartes placées dans le connecteur d'extension. En plus des adresses de mémoire utilisées par les E/S actuelles, il y a des espaces de mémoire disponibles en mémoire programmable (MEV) dans la mémoire principale et en mémoire à lecture-seulement (MEM ou MEM reprogrammable) dans les cartes périphériques elles-mêmes.

Les espaces de mémoire attribués aux cartes périphériques sont décrits ci-dessous. Ces espaces de mémoire sont utilisés pour de petits programmes spécialisés tels que les logiciels de commandes des E/S (« I/O drivers »). Les cartes périphériques qui contiennent leurs propres sous-programmes de commande en mémoire morte sont appelés périphériques intelligents. Ils vous permettent d'ajouter des périphériques à votre Apple //e sans avoir à changer vos programmes, pourvu que vos programmes suivent les pratiques normales d'entrée et de sortie de données.

Espace d'E/S pour carte périphérique

Chaque connecteur d'extension a l'usage exclusif de seize adresses de mémoire pour des données d'entrée et de sortie dans l'espace de mémoire commençant à l'adresse $C090. Le connecteur 1 utilise les adresses $C090 à $C09F, le connecteur 2 les adresses $C0A0 à $C0AF, et ainsi de suite jusqu'à l'adresse $C0FF, comme le montre le tableau 6-1.

Ces adresses de mémoire sont utilisées pour différentes fonctions d'E/S, dépendant de la conception de chaque carte périphérique. Chaque fois que l'Apple *//e* adresse une des seize adresses attribuées à un connecteur particulier, le signal de la broche 41 de ce connecteur, appelé DEVICE SELECT', bascule dans l'état actif (bas). Ce signal peut être utilisé pour valider les circuits logiques de la carte périphérique qui utilisent les lignes des quatre bits d'adresses de plus faible poids pour déterminer quelle est celle qui est appelée parmi les seize adresses.

Tableau 6-1. Adresses de mémoire d'E/S pour carte périphérique

Note : le signal de validation est marqué d'un prime ('), pour indiquer que c'est à son niveau bas que le signal est actif.

Connecteur	Adresses	Validées par
1	$C090-$C09F	DEVICE SELECT'
2	$C0A0-$C0AF	DEVICE SELECT'
3	$C0B0-$C0BF	DEVICE SELECT'
4	$C0C0-$C0CF	DEVICE SELECT'
5	$C0D0-$C0DF	DEVICE SELECT'
6	$C0E0-$C0EF	DEVICE SELECT'
7	$C0F0-$C0FF	DEVICE SELECT'

Espace de MEM pour carte périphérique

Une page de 256 octets d'espace de mémoire est attribuée à chaque carte de périphérique. Cet espace est normalement utilisé par de la mémoire à lecture-seulement (MEM ou MEM reprogrammable) sur la carte contenant des programmes de commande qui contrôlent le fonctionnement du dispositif périphérique connecté à la carte.

La carte de mémoire attribuée à chaque connecteur d'extension débute à l'adresse $Cn00, où n est le numéro du connecteur, comme le montrent le tableau 6-2 et la Figure 6-3. Chaque fois que l'Apple *//e* adresse une des 256 adresses de MEM attribuée à un connecteur particulier, le signal sur la broche 1, appelé I/0 SELECT', bascule dans l'état actif (bas). Ce signal valide les

dispositifs en MEM ou en PMEM sur la carte, et les lignes des huit adresses de plus faible poids déterminent laquelle des 256 adresses de mémoire est adressée.

Tableau 6-2. Adresses de mémoire de MEM pour carte périphérique

Note : Le signal de validation est marqué avec un prime ('), pour indiquer que c'est à son niveau bas que le signal est actif.

Connecteur	Adresses	Validée par
1	$C100-$C1FF	I/O SELECT'
2	$C200-$C2FF	I/O SELECT'
3	$C300-$C3FF	I/O SELECT'
4	$C400-$C4FF	I/O SELECT'
5	$C500-$C5FF	I/O SELECT'
6	$C600-$C6FF	I/O SELECT'
7	$C700-$C7FF	I/O SELECT'

S'il y a une carte de texte en 80 colonnes installée dans le connecteur auxiliaire, certaines des fonctions normalement associées au connecteur 3 sont réalisées par la carte de texte en 80 colonnes et ses sous-programmes de base pour 80 colonnes. Avec une carte de texte en 80 colonnes installée, le signal I/O SELECT' n'est pas disponible pour le connecteur 3, donc les sous-programmes en MEM d'une carte dans le connecteur 3 ne marcheront pas.

Espace de MEM d'extension

En plus des petites zones de MEM attribuées à chaque connecteur d'extension, les cartes périphériques peuvent utiliser l'espace de 2K-octets de mémoire entre $C800 et $CFFF pour des programmes plus longs en MEM ou en PMEM. Cet espace de mémoire est appelé l'espace de MEM d'extension (voir la carte de mémoire de la figure 6-3). En plus d'être plus grand, l'espace de MEM d'extension est toujours aux mêmes adresses indépendamment du connecteur dans lequel est installée la carte, rendant les programmes qui occupent cet espace de mémoire plus facile à écrire. (Voir le paragraphe « Suggestions de programmation des E/S », ci-dessous).

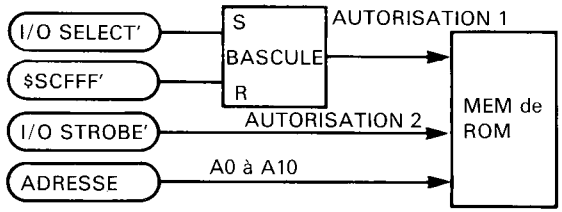
Cet espace de mémoire est disponible pour toutes les cartes périphériques qui en ont besoin. Plusieurs cartes périphériques peuvent avoir leur MEM d'extension, mais l'une seulement d'entre elles peut être active à un moment donné.

Chaque carte périphérique qui utilise la MEM d'extension doit avoir sur elle un circuit pour autoriser la MEM. Ce circuit fait cela en deux temps : en premier, il met à un une bascule quand le signal I/O SELECT', sur la broche 1 du connecteur, devient actif (niveau bas) ; ensuite, il autorise les dispositifs de MEM d'extension quand le signal I/O STROBE', sur la broche 20 du connecteur, devient actif (bas). La figure 6-1 montre un circuit typique d'autorisation de la MEM.

Le signal I/O SELECT' sur un connecteur particulier devient actif chaque fois que le microprocesseur 6502 de l'Apple //e adresse une adresse de l'espace des 256 octets d'adresses de MEM attribué à ce connecteur. Le signal I/O STROBE' sur **tous** les connecteurs d'extension devient actif (bas) quand le 6502 adresse une adresse dans l'espace de MEM d'extension, $C800-$CFFF. Le signal I/O STROBE' est utilisé pour autoriser les dispositifs de MEM d'extension sur une carte périphérique (voir la figure 6-1).

S'il y a une carte de texte en 80 colonnes installée dans le connecteur auxiliaire, certaines des fonctions normalement associées au connecteur 3 sont réalisées par la carte de texte en 80 colonnes et ses sous-programmes de base pour 80 colonnes. Avec une carte de texte en 80 colonnes installée, le signal I/O SELECT' n'est pas disponible pour le connecteur 3, donc les sous-programmes en MEM d'une carte dans le connecteur 3 ne marcheront pas.

Figure 6-1. Circuit d'autorisation de la MEM d'extension



Un programme sur une carte périphérique peut avoir l'usage exclusif de l'espace de MEM d'extension en faisant référence à l'adresse $CFFF dans sa phase d'initialisation. Cette adresse est spéciale : toutes les cartes périphériques qui utilisent de la MEM d'extension doivent reconnaître la référence à $CFFF comme un signal de remise à zéro des bascules d'autorisation de leur MEM et d'inhibition de la MEM d'extension sur cette carte qui est sur le point d'être utilisée, mais la prochaine instruction dans le programme d'initialisation met à un la bascule du circuit d'autorisation de la MEM d'extension sur la carte. Une fois que ceci a été fait, cette carte aura l'usage exclusif de l'espace de mémoire d'extension et son programme peut sauter directement dans la MEM d'extension.

Comme ce fut décrit ci-dessus, le circuit d'inhibition de la MEM d'extension remet à zéro la bascule d'autorisation chaque fois que le 6502 adresse $CFFF.

Programmation pour les cartes périphériques

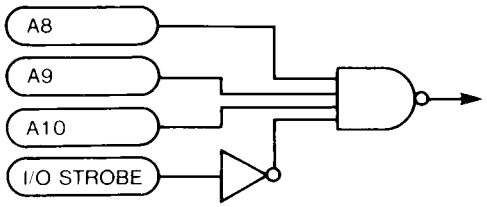


Figure 6-2. Décodage d'adresse provoquant l'inhibition de la MEM

Pour ce faire, la carte périphérique doit détecter la présence de $CFFF sur le bus d'adresses. Vous pouvez utiliser le signal I/O STROBE' comme élément de décodage d'adresse, puisqu'il est actif pour les adresses de $C800 à $CFFF. Si vous avez les moyens de sacrifier un peu d'espace de MEM, vous pouvez encore simplifier encore plus le décodage d'adresse et éliminer des circuits sur la carte. Par exemple, si vous abandonnez les derniers 256 octets de l'espace de MEM d'extension, votre circuit d'inhibition n'a besoin de détecter que la formule $CFxx,, et vous pourrez utiliser le circuit minimal de décodage d'inhibition donné en Figure 6-2.

Espace de MEV pour carte périphérique

Il y a 56 octets de mémoire principale attribués aux cartes périphériques, huit octets par carte, comme le montre le tableau 6-3. Ces 56 adresses sont effectivement dans la mémoire MEV réservée aux affichages de texte et de graphiques en basse-résolution, mais ces adresses particulières ne sont pas affichées sur l'écran et leur contenu n'est pas modifié par le sous-programme de base de sortie COUT1. Les programmes en MEM des cartes périphériques se servent de ces adresses pour mémoriser temporairement des données.

Tableau 6-3. Adresses de mémoire MEV pour carte périphérique

*Note : Les adresses de MEV normalement allouées au connecteur 3 sont reprises par toute carte installée dans le connecteur auxiliaire.

Adresse de base	Numéro de Connecteur						
	1	2	3	4	5	6	7
$0478	$0479	$047A	$047B*	$047C	$047D	$047E	$047F
$0478	$04F9	$04FA	$04FB*	$04FC	$04FD	$04FE	$04FF
$0578	$0579	$057A	$057B*	$057C	$057D	$045E	$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

Espaces de mémoire pour carte périphérique

Un programme sur une carte périphérique peut utiliser les huit adresses de base indiquées sur le tableau pour avoir accès aux huit positions de MEV attribuées à son usage, comme on le montrera dans le prochain paragraphe, « Suggestions de programmation des E/S ».

Suggestions de programmation des E/S

Un programme en MEM sur une carte périphérique devrait fonctionner indépendamment du connecteur que la carte occupe. Si le programme contient un saut à une adresse absolue dans un des espaces de 256 octets, alors la carte ne fonctionnera que si elle est mise dans le connecteur qui utilise cet espace de mémoire. Si vous écrivez ce programme pour une carte périphérique qui sera utilisée par beaucoup de monde, vous devriez éviter d'introduire une telle restriction dans l'usage de cette carte.

> Pour fonctionner correctement et indépendamment du connecteur dans lequel la carte est installée, le programme dans l'espace des 256 octets de la carte ne doit faire aucune référence absolue à lui-même. Au lieu d'utiliser des instructions de saut, vous devrez forcer des conditions sur des instructions de branchement, qui utilisent l'adressage relatif.

La première chose qu'un sous-programme de carte périphérique devrait faire est de sauvegarder les contenus des registres du 6502. Une façon de le faire consiste à se servir du sous-programme du Moniteur IOSAVE. Ce sous-programme qui débute à l'adresse $FF4A, mémorise les registres dans les positions de la page zéro d'adresses $45-$49. Le sous-programme allant avec IOSAVE est IOREST qui restaure les registres depuis ces adresses de mémoire. Votre programme devra appeler IOREST, qui débute à l'adresse $FF3F, juste avant de rendre le contrôle au programme qui l'a appelé.

Cette méthode de sauvegarde des registres convient, mais elle n'est pas toujours sûre. Si un second sous-programme appelle IOSAVE, ou si une interruption se produit, le nouveau contenu du registre est sauvegardé à la même position de mémoire, et l'ancien contenu sera détruit. Il est plus sûr, bien que plus lent, de sauvegarder les registres sur la pile et de les restaurer juste avant de rendre le contrôle au programme appelant.

La plupart des E/S d'un seul caractère est réalisée à travers l'accumulateur du 6502. Un caractère à sortir par votre sous-programme sera dans l'accumulateur avec son bit de plus fort poids à un quand votre sous-programme est appelé. De même, si votre sous-programme effectue une entrée de caractère, il doit laisser le caractère dans l'accumulateur avec son bit de plus fort poids mis à un quand il retourne au programme appelant.

Recherche du numéro de connecteur

Les adresses de mémoire utilisées par un programme sur une carte périphérique diffèrent suivant le connecteur d'extension dans lequel la carte est installée. Avant de se référer à l'une de ces adresses, le programme doit déterminer d'une façon ou d'une autre le numéro correct de connecteur. Une des façons de faire est d'exécuter une instruction JSR (« Jump to Subroutine » ou saut à un sous-programme) vers une adresse contenant une instruction RTS (« Return from Subroutine » ou retour depuis un sous-programme), et de réduire le numéro de connecteur de l'adresse sauvegardée sur la pile, comme le montre l'exemple suivant.

```
PHP                 ; save status
SEI                 ; inhibit interrupts
JSR $FF58           ; → a know RTS instruction
TSX                 ; get high byte of the...
LDA $0100, X        ; ... return address from stack
AND *$0F            ; low-order digit is slot no.
PLP                 ; restore status
```

Le numéro de connecteur peut être utilisé au moment d'adresser la mémoire attribuée à la carte périphérique, comme cela est indiqué ci-dessous.

Adressage des E/S

Une fois que votre programme de carte périphérique a récupéré le numéro de connecteur, il peut l'utiliser pour adresser les positions d'E/S attribuées à ce connecteur. Le tableau 6-4 montre comment ces positions sont en relation avec les seize adresses de base débutant en $C080. Remarquez que la différence entre l'adresse de base et la position désirée est de la forme $n0, où n est le numéro de connecteur. En commençant avec le numéro de connecteur dans l'accumulateur, l'exemple suivant calcule cette différence par quatre décalages à gauche, puis la charge dans un registre d'index et utilise l'adresse de base pour spécifier une des seize positions de mémoire d'E/S.

```
ASL                 ; get n into...
ASL                 ;
ASL                 ;
ASL                 ; ... high-order nybble...
TAX                 ; ... of index register.
LDA $C080, X        ; load from first I/O location
```

Vous devez être bien sûr que vous prenez la bonne valeur dans le registre d'index quand vous adressez les positions d'E/S de cette façon. Par exemple, en débutant avec 1 dans l'accumulateur, les instructions dans l'exemple ci-dessus réalisent un LDA avec la position $C090, la première position attribuée au connecteur 1. Si la valeur dans l'accumulateur avait été 0, le LDA aurait adressé la position $C080, d'où le positionnement du commutateur logiciel qui sélectionne le second banc de MEV à l'adresse $D000 et l'autorise en lecture (voir le chapitre 5).

Tableau 6-4. Adresses de base des E/S pour carte périphérique

Adresse de base	Numéro de Connecteur 1	2	3	4	5	6	7
$C080	$C090	$C0A0	$C0B0	$C0C0	$C0D0	$C0E0	$C0F0
$C081	$C091	$C0A1	$C0B1	$C0C1	$C0D1	$C0E1	$C0F1
$C082	$C092	$C0A2	$C0B2	$C0C2	$C0D2	$C0E2	$C0F2
$C083	$C093	$C0A3	$C0B3	$C0C3	$C0D3	$C0E3	$C0F3
$C084	$C094	$C0A4	$C0B4	$C0C4	$C0D4	$C0E4	$C0F4
$C085	$C095	$C0A5	$C0B5	$C0C5	$C0D5	$C0E5	$C0F5
$C086	$C096	$C0A6	$C0B6	$C0C6	$C0D6	$C0E6	$C0F6
$C087	$C097	$C0A7	$C0B7	$C0C7	$C0D7	$C0E7	$C0F7
$C088	$C098	$C0A8	$C0B8	$C0C8	$C0D8	$C0E8	$C0F8
$C08A	$C09A	$C0AA	$C0BA	$C0CA	$C0DA	$C0EA	$C0FA
$C08B	$C09B	$C0AB	$C0BB	$C0CB	$C0DB	$C0EB	$C0FB
$C08C	$C09C	$C0AC	$C0BC	$C0CC	$C0DC	$C0EC	$C0FC
$C08D	$C09D	$C0AD	$C0BD	$C0CD	$C0DD	$C0ED	$C0FD
$C08E	$C09E	$C0AE	$C0BE	$C0CE	$C0DE	$C0EE	$C0FE
$C08F	$C09F	$C0AF	$C0BF	$C0CF	$C0DF	$C0EF	$C0FF

Adressage en MEV

Un programme sur une carte périphérique peut utiliser les huit adresses de base énumérées dans le tableau 6-3 pour avoir accès aux huit positions de MEV qui lui sont attribuées pour son usage. Le programme fait cela en mettant le numéro du connecteur dans le registre d'index Y et en utilisant le mode d'adressage indexé avec l'adresse de base. Les adresses de base peuvent être définies comme constantes car elles sont les mêmes quel que soit le connecteur que la carte périphérique occupe.

Si vous commencez avec un numéro correct de connecteur dans l'accumulateur (en vous servant de l'exemple donné plus haut), l'exemple suivant utilise toutes les huit positions de MEV attribuées à ce connecteur.

```
TAY
LDA     $478, Y
STA     $4F8, Y
LDA     $0578, Y
STA     $0578, Y
LDA     $0678, y
STA     $06F8, Y
LDA     $0778, Y
STA     $07F8, Y
```

Avertissement
Les programmes de carte périphérique ne doivent pas mémoriser des données dans les positions dont les adresses sont les propres adresses de base ; la MEV est utilisée à ces adresses par le Système d'Exploitation des Disquettes. Le DOS enregistre le premier octet de la postion de MEM du connecteur d'expension qui est actuellement actif ($Cn) dans la position $7F8, et le premier octet de la position de MEM du connecteur sur lequel est installée la carte du contrôleur du lecteur de disque d'amorçage, dans la position $5F8.

Changement des liaisons standards d'E/S

Il y a deux paires d'adresses dans l'Apple //e qui servent à contrôler l'entrée et la sortie de caractère. Elles s'appellent liaisons d'E/S (voir le chapitre 3). Avec un Apple //e tournant sans Système d'exploitation de Disquettes, les liaisons d'E/S contiennent normalement les adresses de début des sous-programmes standards d'entrée et de sortie KEYIN et COUT1. Si un système d'exploitation est en ligne, l'une des deux ou les deux liaisons contiendront les adresses des sous-programmes d'entrée et de sortie du DOS.

La liaison aux adresses $36 et $37 (décimal 54 et 55) est appelée CSW, qui veut dire « Character output Switch » ou commutateur de sortie de caractère. Individuellement $36 s'appelle CSWL (CSW « Low » ou poids faible ou basse) et la position $37 s'appelle CSWH (CSW « High » ou haute ou poids forts). Cette liaison contient l'adresse de début du sous-programme que l'Apple //e est en train d'utiliser pour la sortie d'un caractère. Cette adresse est normalement $FDF0, l'adresse du sous-programme COUT1, décrit au chapitre 3.

Quand vous entrez un PR# n depuis BASIC ou un n Ctrl -P depuis le moniteur, l'Apple //e change cette adresse de liaison en la première adresse de l'espace de mémoire MEM attribué au connecteur numéro n. Cette adresse est de la forme $Cn00. Des appels postérieurs de sortie de caractère seront donc transférés au programme de la carte périphérique. Ce programme peut utiliser les séquences d'instructions données plus haut pour rechercher le numéro du connecteur et utiliser les positions de MEV qui lui sont attribuées. Quand il a terminé, le programme peut exécuter une instructions RTS (Retour depuis un sous-programme) pour rendre le contrôle au programme appelant, ou sauter au sous-programme de sortie COUT1 à l'adresse $FDF0 pour afficher le caractère à sortir (qui doit être dans l'accumulateur) sur l'écran, puis laisser COUT1 revenir au programme appelant.

Suggestions de programmation des E/S

Une liaison similaire aux adresses $38 et $39 (décimal 56 et 57) est appelée KSW, qui veut dire « Keyboard input Switch » ou commutateur d'entrée au clavier. Individuellement la position $38 est appelée KSWL (pour KSW « Low » ou basse) et la position $39 est appelée KSWH (KSW « High » pour haute). Cette liaison contient l'adresse de début du sous-programme utilisé actuellement en entrée d'un seul caractère. Cette adresse vaut normalement $FD1B, l'adresse de début du sous-programme standard d'entrée KEYIN (voir au chapitre 3).

Quand vous entrez une commande IN # depuis BASIC ou un n Ctrl -K depuis le moniteur, l'Apple *//e* change cette adresse de liaison en $Cn00, le début de l'espace de mémoire MEM qui est attribué au connecteur n. Des appels postérieurs pour entrer un caractère seront donc transférés au programme de la carte périphérique. Ce programme devra mettre le caractère entré, avec son bit de poids fort à un, dans l'accumulateur et exécuter une instruction RTS pour rendre le contrôle au programme qui a demandé cette entrée.

Quand le Système d'Exploitation de Disquettes (DOS) est en ligne, l'une des deux ou les deux liaisons standards contiennent les adresses des sous-programmes d'entrée et de sortie du Système d'exploitation de Disquettes. Le DOS a des positions internes qui contiennent les adresses des sous-programmes d'entrée et de sortie actuellement actifs.

> Si un programme fonctionnant avec le DOS change les adresses standards de liaison, ou directement ou par les commandes IN # et PR # , le DOS sera déconnecté du système.

Pour éviter de déconnecter le DOS à chaque fois qu'ils commencent des E/S sur un connecteur, les programmes en BASIC qui fonctionnent avec le DOS doivent toujours envoyer une commande IN # ou PR # précédée d'un caractère Ctrl-D écrite à l'intérieur d'une instruction PRINT. Pour les programmes en langage d'assemblage, il existe un appel à un sous-programme du DOS à utiliser quand on change les adresses de liaison. Après avoir changé CSW ou KSW, le programme appelle ce sous-programme à l'adresse $3EA (décimal 1002). Le sous-programme transfère les adresses de liaison vers une position interne du DOS et restaure l'adresse du DOS dans la position de mémoire standard de la liaison. Se reporter au paragraphe sur les registres de liaison d'entrée et de sortie dans le *Manuel du DOS* pour plus de détails.

Utilisation des interruptions

Bien que les programmes fonctionnant sur l'Apple //e n'utilisent pas normalement les interruptions, il est possible de le faire. Pour utiliser les interruptions sur l'Apple //e, votre carte périphérique doit être capable d'envoyer une demande d'interruption (IRQ') au microprocesseur 6502, et vous devez mémoriser l'adresse de votre sous-programme de traitement de l'interruption dans le vecteur d'interruption de l'utilisateur.

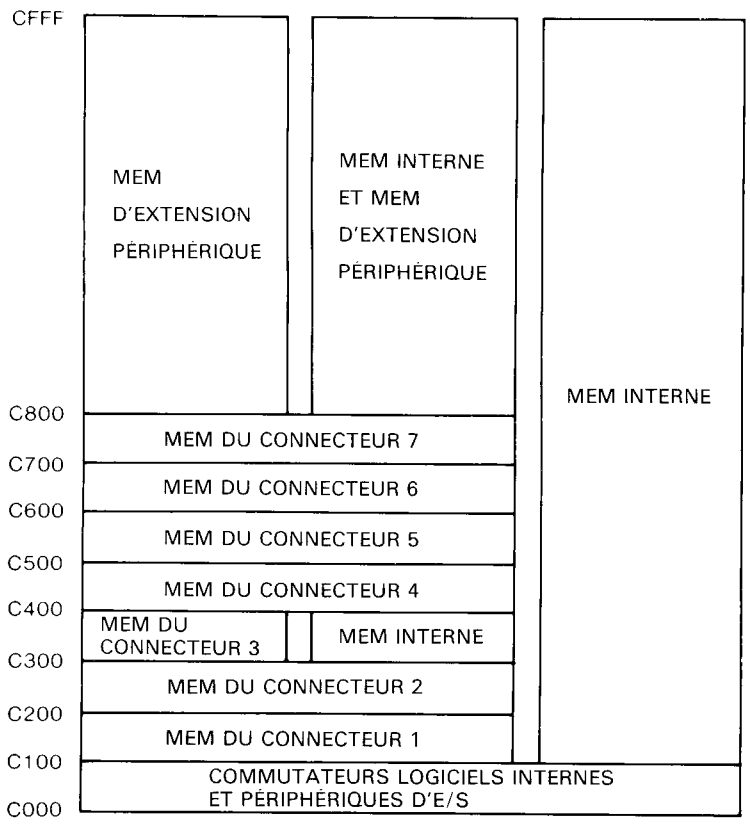
La priorité des interruptions est prise en charge par un arrangement en chaînage-marguerite utilisant deux broches , INT IN et INT OUT, sur chaque connecteur d'extension. Comme cela est décrit au chapitre 7, chaque carte périphérique casse le chaînage quand elle fait une demande d'interruption. Sur les cartes périphériques qui n'utilisent pas d'interruptions, ces broches doivent être connectées ensemble. Le chaînage circulaire donne la priorité à la carte périphérique du connecteur 7 : si cette carte ouvre la connexion entre INT IN et INT OUT, ou s'il n'y a pas de carte dans ce connecteur, les demandes d'interruption de cartes dans les connecteurs 1 à 6 peuvent être prises en compte. De même, le connecteur 6 contrôle l'interruption des connecteurs 1 à 5, et ainsi de suite vers le bas de la ligne.

Quand la ligne IRQ' du microprocesseur 6502 est activée (mise au niveau bas) le 6502 transfère le contrôle à travers le vecteur contenu dans les adresses $FFFE-$FFFF. Ce vecteur est l'adresse du programme de traitement des interruptions du moniteur, lequel détermine si la demande est due à un IRQ' externe ou à une instruction BRK et transfère le contrôle au sous-programme approprié par les vecteurs mémorisés en page 3. Le vecteur de BRK est aux adresses $3F0-$3F1 et le vecteur de IRQ est aux adresses $3FE-$3FF (voir le tableau 4-10). Le moniteur mémorise normalement l'adresse de son sous-programme de réinitialisation dans le vecteur de IRQ ; vous pouvez lui substituer l'adresse de votre sous-programme de traitement d'interruption.

Autres utilisations de l'espace de mémoire d'E/S

La partie de l'espace de mémoire depuis l'adresse $C000 jusqu'à l'adresse $CFFF (décimal 49152 à 53247) est normalement attribuée aux E/S et aux mémoires des programmes sur les cartes périphériques, mais il a deux autres fonctions qui utilisent aussi cet espace : les sous-programmes d'auto-test du système et les sous-programmes pour 80 colonnes. Les commutateurs logiciels qui contrôlent l'attribution de cet espace de mémoire sont décrits ci-dessous.

Figure 6-3. Carte des mémoires d'E/S



Commutation des mémoires d'E/S

Les sous-programmes déjà implantés dans le système se servent de deux commutateurs logiciels pour contrôler l'allocation de l'espace de mémoire d'E/S de $C000 à $CFFF. Les adresses de ces commutateurs logiciels, SLOTXROM et SLOTC3ROM, figurent dans le tableau 6-5.

Comme pour les commutateurs d'affichage décrits au chapitre 2, ces commutateurs logiciels partagent leurs adresses avec les données du clavier et les fonctions d'échantillonnage du clavier. Ces commutateurs ne sont activés que par écriture, et leurs états peuvent être déterminés seulement par lecture, comme le montre le tableau 6-5.

Si SLOTC3ROM est à un, alors la zone de 256 octets de MEM en $C300 est disponible pour une carte périphérique installée dans le connecteur 3, qui est le connecteur normalement utilisé par une interface de terminal. Si une carte est installée dans le connecteur auxiliaire lorsque vous mettez sous-tension ou que vous réinitialisez

Tableau 6-5. Commutateurs des mémoires d'E/S

Nom	Fonction	Adresse Hex	Adresse Décimal	Notes
SLOTC3ROM	MEM périph. en $C300	$C00B	49163	16373 Écrire
	MEM interne en $C300	$C00A	49162	−16374 Écrire
	Lire le comm. SLOTC3ROM	$C017	49175	−16361 Lire
SLOTCXROM	MEM périph. en $CxOO	$C007	49159	−16377 Écrire
	MEM interne en $CxOO	$C006	49158	−16378 Écrire
	Lire le comm. SLOTCXROM	$C015	49173	−16363 Lire

l'Apple //e, le commutateur SLOTC3ROM est remis à zéro. En remettant SLOTC3ROM à zéro, vous inhibez la MEM de la carte périphérique du connecteur 3 et vous autorisez les sous-programmes pour 80 colonnes, comme indiqué sur la figure 6-3. Les sous-programmes pour 80 colonnes sont affectés à l'espace d'adresse du connecteur 3 parce que le connecteur 3 est normalement utilisé avec une interface de terminal, donc les sous-programmes du système fonctionneront avec les programmes qui utiliseront le connecteur 3 de cette façon.

L'installation d'une carte de texte en 80 colonnes dans le connecteur auxiliaire rend impossible le fonctionnement dans le connecteur 3 d'une carte périphérique qui contient des sous-programmes en MEM. Si la carte de texte en 80 colonnes n'est pas installée, une carte périphérique installée dans le connecteur 3 fonctionnera parfaitement.

Le bus et les signaux d'E/S sont toujours disponibles à une carte périphérique installée dans le connecteur 3, même si les sous-programmes et les circuits pour 80 colonnes sont en fonctionnement. Donc il est toujours possible d'utiliser ce connecteur pour tout accessoire d'E/S qui ne contienne **pas** de sous-programme implanté en MEM dedans.

Si SLOTCXROM est actif (niveau haut), l'espace de mémoire d'E/S de $C100 à $C7FF est attribué aux connecteurs d'expension, comme on l'a décrit précédemment. En rendant SLOTCXROM inactif (niveau bas) on inhibe la MEM de carte périphérique et on sélectionne la MEM interne dans tous les espaces de mémoire d'E/S sauf la partie comprise entre $C000 et C0FF (utilisée par les commutateurs logiciels et les données d'E/S) comme indiqué en figure 6-3. En plus des sous-programmes pour 80 colonnes en $C300 et $C800, la MEM interne comprend les sous-programmes qui réalisent l'auto-test des circuits de l'Apple //e.

En mettant au niveau bas SLOTCXROM, on autorise la MEM interne dans tout l'espace de mémoire d'E/S (sauf la zone des commutateurs logiciels), y compris l'espace $C300, qui contient les sous-programmes pour 80 colonnes.

Autres utilisations de l'espace de mémoire d'E/S

Implantation des circuits

Implantation des circuits

La plus grande partie de ce manuel décrit des fonctions - celles de l'Apple //e. Ce chapitre, par contre, décrit des objets : les circuits que l'Apple //e utilise pour réaliser ses fonctions. Si vous concevez un circuit périphérique à connecter à l'Apple //e, ou si vous voulez simplement en savoir plus sur la façon dont l'Apple //e a été conçu, vous devriez étudier ce chapitre.

Les spécifications d'environnement

L'Apple //e est tout-à-fait fiable quand on l'utilise dans les conditions pour lesquelles il a été projeté. Le tableau 7-1 définit les conditions sous lesquelles l'Apple //e a été conçu pour fonctionner correctement.

Tableau 7-1. Résumé des spécifications d'environnement

Température de fonctionnement :	0° à 45° C (30° à 115° F)
Humidité relative :	5 % à 85 %
Tension du secteur :	214 à 264 V AC

Vous devez traiter l'Apple //e avec le même soin qu'avec n'importe quel autre appareil électrique. Vous devez le protéger contre des violences physiques, telles que coups de marteau ou défenestration. Vous devez protéger le clavier mécanique et les connecteurs électriques à l'intérieur de projections de liquides, particulièrement ceux qui contiennent des contaminants dissous, comme le café ou les boissons au cola.

En fonctionnement normal, il y a suffisamment d'air qui passe à l'intérieur par les fentes à l'intérieur pour l'empêcher de surchauffer, bien que quelques composants à l'intérieur de l'Apple //e soient plutôt chauds au toucher. Si vous surchauffez votre Apple //e, en bloquant les fentes d'aération du haut et du bas par exemple, le premier symptôme sera un fonctionnement erratique. Les circuits de mémoire de l'Apple //e sont sensibles à la chaleur : quand ils

deviennent trop chauds, ils changent occasionnellement un bit de données. Le résultat exact dépend de quel genre de programme vous faites exécuter et de quel bit de mémoire est affecté.

Le boîtier d'alimentation

Le boîtier d'alimentation électrique de l'Apple //e fonctionne sur le réseau de courant alternatif domestique normal et fournit suffisamment de puissance électrique en basse tension pour les circuits électroniques implémentés dans le système, plus un ensemble complémentaire de cartes périphériques, y compris les cartes de contrôleur de disquettes et les interfaces de communication. Les spécifications du boîtier d'alimentation sont données à le tableau 7-2.

Le cordon d'alimentation de l'Apple //e doit être branché dans une prise 220-240 volts à trois trous. Vous devez connectez l'Apple //e à une prise de terre ou à une bonne masse à la terre. De plus, la tension sur la ligne doit être comprise dans les limites données dans le tableau 7-2. Si vous tentez de faire fonctionner l'Apple //e sous une tension de plus de 264 volts, vous allez abimer votre boîtier d'alimentation.

Tableau 7-2. Spécifications d'environnement du boîtier d'alimentation.

* Fonctionnement par intermittence : l'Apple //e peut fonctionner en toute sécurité jusqu'à 20 minutes à la plus haute charge si cela est suivi d'au-moins 10 minutes à charge normale.

Tension du secteur :	214 V à 264 V AC
Consommation maximum :	60 W continuellement 80 W par intermittence*
Tensions fournies :	+ 5 V ± 3 % + 11.8 V ± 6 % − 5.2 V ± 10 % − 12 V ± 10 %
Courants maxima fournis :	+ 5 V : 2.5 A + 12 V : 1.5 A continuellement 2.5 A par intermittence* − 5 V : 250 mA − 12 V : 250 mA
Température maximum à l'intérieur :	55° C (130° F)

L'Apple //e utilise une alimentation à commutation conçue sur mesure. Elle est petite et légère, et dégage moins de chaleur que tous les autres types d'alimentation.

L'alimentation de l'Apple //e opère en convertissant la tension alternative (AC) du secteur en tension continue (DC) et en utilisant cette tension continue pour alimenter un oscillateur à fréquence variable. L'oscillateur commande un petit transformateur ayant de nombreux enroulements séparés pour produire les différents voltages requis. Un circuit compare la tension de + 5 Volts avec une tension de référence et fournit en rétroaction un signal d'erreur

au circuit oscillateur. Le circuit oscillateur utilise ce signal d'erreur pour contrôler la fréquence des oscillations et maintenir les tensions de sortie dans leurs limites normales.

Le boîtier d'alimentation comprend des circuits pour se protéger lui-même et protéger les autres composants électroniques de l'Apple //e en mettant à zéro les quatre tensions d'alimentation chaque fois qu'il détecte un de ces cas de mauvais fonctionnement :

- Une des tensions d'alimentation est en court-circuit à la terre ;
- Le câble d'alimentation est débranché ;
- Une des tensions d'alimentation est en dehors de ses valeurs limites normales.

Chaque fois qu'une de ces pannes se produit, le circuit de protection arrête l'oscillateur, et toutes les tensions tombent à zéro. Après environ une demi-seconde, l'oscillateur redémarre. Si le mauvais fonctionnement se produit toujours, le circuit de protection arrête à nouveau l'oscillateur. L'alimentation continuera à démarrer et à s'arrêter de cette façon jusqu'à ce que le mauvais fonctionnement soit corrigé ou que l'alimentation soit déconnectée.

Avertissement
Si vous pensez que le boîtier d'alimentation est en panne, n'essayez pas de le réparer vous-même. Le boîtier d'alimentation est dans une enceinte scellée car quelques-uns des circuits sont connectés directement sur le secteur. Un équipement spécial est nécessaire pour réparer le boîtier d'alimentation, donc voyez votre revendeur Apple pour la réparation.

Le connecteur d'alimentation

Le câble du boîtier d'alimentation est connecté à la carte-mère par un connecteur à six broches avec une prise spéciale. Les broches du connecteur sont identifiées dans le tableau 7-3 et la figure 7-14d.

Tableau 7-3. Spécifications des signaux du connecteur d'alimentation.

Numéro de broche	Nom	Description
1,2	Ground	Masse (terre) électrique commune
3	+ 5 V	+ 5 V du boîtier d'alimentation
4	+ 12 V	+ 12 V du boîtier d'alimentation
5	– 12 V	– 12 V du boîtier d'alimentation
6	– 5 V	– 5 V du boîtier d'alimentation

Le microprocesseur 6502

L'Apple //e utilise un microprocesseur 6502A comme unité centrale (UC). Le 6502A dans l'Apple //e tourne à une fréquence d'horloge de 1.018 MHz et effectue jusqu'à 500 000 opérations par seconde sur 8 bits. Il ne faudrait pas se servir de la fréquence d'horloge comme critère de comparaison entre différents microprocesseurs. Le 6502 a un cycle d'instruction plus simple que celui de la plupart des autres microprocesseurs et il utilise l'exécution d'instructions en cascade (« instruction pipelining ») pour un traitement plus rapide. La vitesse du 6502, à une fréquence d'horloge de 1 MHz, équivaut pour les autres types de microprocesseurs à une fréquence d'horloge allant jusqu'à 2,5 MHz.

Le 6502 a un bus d'adresses à seize bits, lui conférant un espace d'adresses de 64 K (2 à la puissance seize ou 65536) octets. L'Apple //e se sert de techniques spéciales pour adresser un total de plus de 64 K : voir le paragraphe « Mémoire à banc-commuté » et « La mémoire auxilaire et ses sous-programmes » du chapitre 4 et le paragraphe « Commutation des mémoires d'E/S » du chapitre 6.

Tableau 7-4. Spécifications du microprocesseur 6502

Type :	6502A
Registres :	Accumulateur (A) Registres d'index (X, Y) Pointeur de pile (S) État du processeur (P)
Taille du registre :	Huit bits
Bus de données :	Huit bits de large
Bus d'adresses :	Seize bits de large
Interruptions :	IRQ (masquable) NMI (non masquable) BRK (programmable)
Tension de fonctionnement :	+ 5 V (± 5 %)
Dissipation de puissance :	500 mW (typique)

Le timing du 6502

Le fonctionnement de l'Apple //e est contrôlé par un ensemble de signaux de timing, appelés quelque fois signaux d'horloge. En électronique, le mot *horloge* est utilisé pour identifier des signaux qui contrôlent le cadencement des opérations du circuit. L'Apple //e ne contient pas le type d'horloge qui vous donne l'heure, bien que son timing interne soit suffisamment précis pour qu'un programme tournant sur l'Apple //e puisse simuler une telle horloge.

La fréquence de l'oscillateur que génère le signal de timing-maître est de 14.25045 MHz. Les circuits de l'Apple //e utilisent ce signal d'horloge, appelé 14M, pour produire tous les autres signaux de timing. Ces signaux de synchronisation effectuent deux tâches majeures : contrôler les fonctions de calcul et générer l'affichage vidéo. Les signaux de timing directement impliqués dans l'opération du 6502 sont décrits dans ce paragraphe. D'autres signaux de synchronisation sont décrits dans les paragraphes « Adressage de MEV », « Modes d'affichage vidéo », et « Les connecteurs d'expension ».

Les principaux signaux de timing du 6502 sont énumérés dans le tableau 7-5, et leurs correlations sont schématisées dans le chronogramme de la figure 7-1. Les signaux d'horloge du 6502 sont ϕ 1 et ϕ 0, des signaux complémentaires à une fréquence de 1.01799 MHz. Si vous avez besoin de plus d'informations sur le 6502 lui-même, reportez-vous au *Manuel hardware* Synertek (Produit Apple numéro A2L0002). Le signal de l'Apple //e appelé 0 est équivalent au signal appelé 2 dans le *Manuel hardware* (il n'est pas identique : il est un tout petit peu en avance).

Tableau 7-5. Description des signaux de timing du 6502

Nom du signal	Description
14M	Oscillateur maître, 14.25045 MHz ; aussi l'horloge des points en 80 colonnes.
7M	Signal intermédiaire de timing et horloge des points en 80 colonnes.
Q3	Signal intermédiaire de timing, 2.03578 MHz avec un cycle asymétrique d'utilisation.
ϕ0	Phase 0 de l'horloge du 6502, 1.017989 MHz ; Complément de ϕ1.
ϕ1	Phase 1 de l'horloge du 6502, 1.017989 MHz ; Complément de ϕ0.

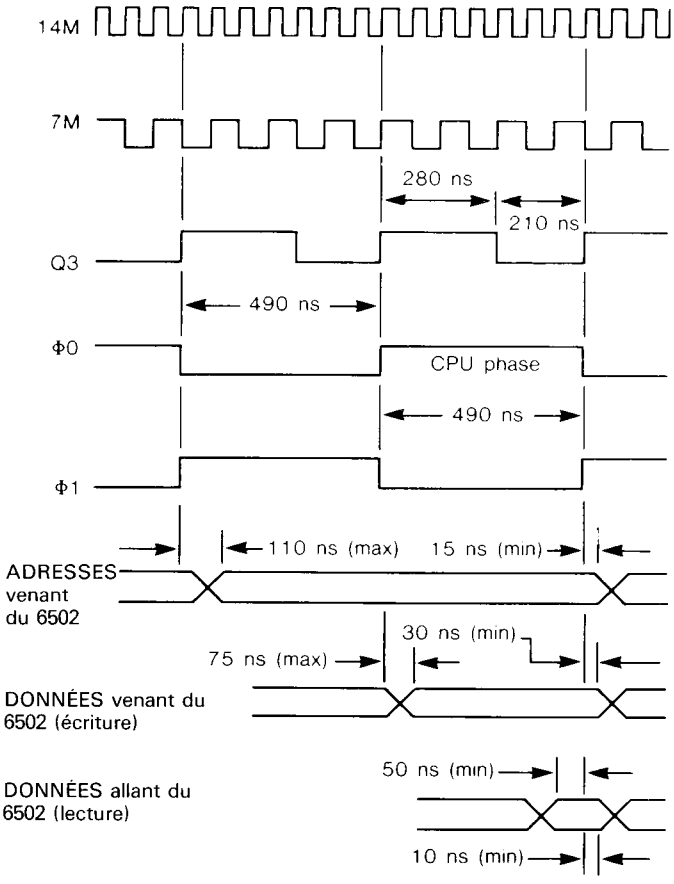


Figure 7-1. Signaux de timing du 6502

Note : Les temps donnés sont ceux d'établissement et de maintien du 6502A. Le timing sur les connecteurs périphériques comprend aussi 25 ns approximativement de temps de propagation dans la mémoire-tampon.

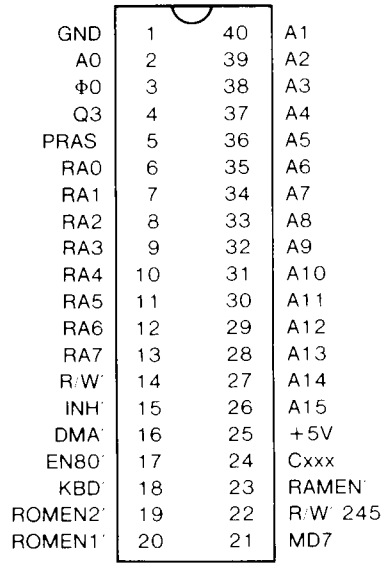
Les opérations du 6502 sont reliées aux signaux d'horloge d'une façon simple : adresse pendant $\phi 1$, donnée durant $\phi 0$. Le 6502 met une adresse sur le bus d'adresses pendant $\phi 1$. Cette adresse est valide pas plus longtemps que 110 nanosecondes après que $\phi 1$ soit au niveau haut et reste valide pendant toute la phase $\phi 0$. Le 6502 lit ou écrit une donnée pendant $\phi 0$. Si le 6502 est en train d'écrire, le signal de lecture/écriture est au niveau bas pendant $\phi 0$ et le 6502 met la donnée sur le bus de données. La donnée est valide pas plus longtemps que 75 nanosecondes après que $\phi 0$ passe au niveau haut. Si le 6502 est en train de lire, le signal de lecture/écriture reste haut. La donnée sur le bus de données doit être valide pas plus longtemps que 50 nanosecondes avant la fin de $\phi 0$.

Implantation des circuits

Les circuits intégrés fabriqués sur mesure

La plupart des circuits qui contrôlent l'adressage de la mémoire et des E/S dans l'Apple //e est dans trois circuits intégrés faits sur mesure appelés l'unité de gestion de la mémoire (MMU), l'unité d'entrée/sortie (IOU), et le dispositif PAL ou « Programmed Array Logic » ou réseau logique programmé. Les commutateurs logiciels utilisés pour contrôler les différentes E/S et les modes d'adressage de l'Apple //e sont des signaux adressables à l'intérieur de la MMU et de l'IOU. Les fonctions de ces dispositifs ne sont pas aussi indépendantes que leur nom le suggère ; travaillant ensemble elles génèrent tous les signaux d'adressage. Par exemple, la MMU génère les signaux d'adresses pour l'UC, tandis que l'IOU génère des signaux similaires d'adresses pour l'affichage vidéo.

Figure 7-2. Le brochage de la MMU



L'unité de gestion de la mémoire (MMU)

Les commutateurs logiciels sont implémentés dans les circuits internes de la MMU et sont décrits dans les chapitres suivants :

Affichage de la Page 2 (PAGE2) : chapitre 2
Mode haute-résolution (HIRES) : chapitre 2
Mémorisation sur la carte 80 colonnes (80STORE) : chapitre 2
Sélection du banc 2 : chapitre 4
Autorisation de la MEV à banc-commuté : chapitre 4
Lecture en mémoire auxiliaire (RAMRD) : chapitre 4
Écriture en mémoire auxiliaire (RAMWRT) : chapitre 4
Pile et page zéro auxiliaire (ALTZP) : chapitre 4
MEM du connecteur n° 3 (SLOTC3ROM) : chapitre 6
MEM périphériques dans l'espace d'E/S (SLOTCXROM) : chapitre 6

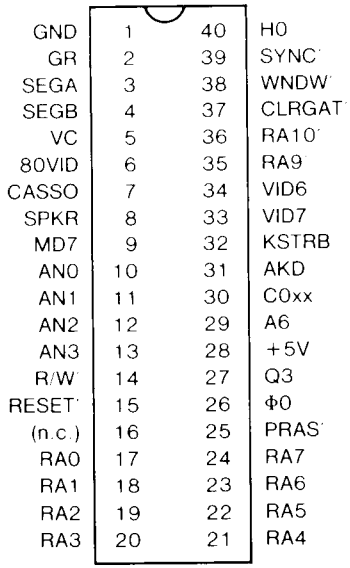
les 64 K de MEV dynamiques utilisés dans l'Apple //e se servent d'une adresse multiplexée, comme cela sera décrit dans le paragraphe « Timing de la MEV dynamique ». La MMU génère cette adresse multiplexée pour la lecture et l'écriture en mémoire par le 6502.

Tableau 7-6. Description des signaux de la MMU

Numéro de broche	Nom	Description
1	GND	Commun d'alimentation et des signaux
2	A0	Entrée d'adresse venant du 6502
40-26	A1-A15	Entrée d'adresse venant du 6502
3	O 0	Phase 0 de l'horloge
4	Q3	Signal de timing
5	PRAS'	Échantillonnage de l'adresse de rangée de mémoire
6-13	RA0-RA7	Sortie d'adresse multiplexée
14	R/W'	Signal de contrôle de lecture-écriture du 6502
15	INH'	Inhibe la mémoire principale
16	DMA'	Contrôle le bus de données pour les transferts en DMA (accès direct en mémoire)
17	EN80'	Autorise la MEV auxiliaire
18	KBD'	Autorise les bits 0-6 de données du clavier
19	ROMEN2'	Autorise les sous-programmes en MEM 2
20	ROMEN1'	Autorise les sous-programme en MEM 1
21	MD7	État des indicateurs de la MMU
22	RW'245	Contrôle le circuit 74LS245 amplificateur-tampon des données du bus
23	RAMEN'	Autorise la mémoire principale
24	CXXX	Autorise la mémoire des cartes-périphériques
25	+5v	Alimentation

Implantation des circuits

Figure 7-3. Le brochage de l'IOU



L'unité d'Entrée/Sortie

Les commutateurs logiciels suivants sont implémentés dans les circuits de l'IOU, tous sont décrits au chapitre 2 :

Affichage de la Page 2 (PAGE2)
Mode haute-résolution (HIRES)
Mode texte (TEXT)
Mode mixte (MIXED)
Affichage en 80 colonnes (80COL)
Sélection d'un ensemble de caractères (ALTCHARSET)
Toute-touche-enfoncée
Annonciateurs
Effacement vertical (VBL)

Les 64K de MEV dynamiques utilisées par l'Apple //e requièrent une adresse multiplexée, comme cela est décrit au paragraphe « Timing des MEV dynamiques ». L'IOU génère cette adresse multiplexée pour les transferts de données nécessaires au rafraîchissement d'affichage et de mémoire pendant la phase 1. La façon dont cette adresse est générée est expliquée plus loin dans le paragraphe « Génération de l'affichage vidéo ».

Table 7-7. Description des signaux de l'IOU

Note : La broche 16 n'est pas connectée.

Numéro de broche	Nom	Description
1	GND	Commun d'alimentation et des signaux
2	GR	Autorisation du mode graphique
3,4	SEGA, SEGB	Bits du compteur vertical d'affichage
5	VC	Bit du compteur vertical d'affichage
6	80VID'	Autorisation du mode vidéo 80 colonnes
7	CASSO	Signal de sortie sur cassette
8	SPKR	Signal de sortie sur haut-parleur
9	MD7	Indicateurs internes au bus de données
10-13	AN0-AN3	Sorties annonciateurs
14	R/W'	Signal de contrôle de lecture-écriture du 6502
15	RESET'	Sortie de Mise sous-tension et Reset
17-24	RA0-RA7	Adresse multiplexée de MEV (phase 0)
25	PRAS'	Échantillonneur d'adresse-rangée (p. 0)
26	$\phi 0$	Phase 0 de l'horloge maître
27	Q3	Signal de timing intermédiaire
28	+5V	Alimentation
29	A6	Bit 6 d'adresse du 6502
30	C0XX'	Autorisation d'adresse d'E/S
31	AKD	Signal de toute-touche enfoncée
32	KSTRB	Signal d'échantillonnage du clavier
33, 34	VID7, VID6	Bits de contrôle d'affichage vidéo
35, 36	RA9',RA10'	Bits de contrôle d'affichage vidéo
37	CLRGAT'	Porte de rafale de couleur (autorise)
38	WNDW'	Signal d'effacement d'affichage
39	SYNC'	Signal de synchronisation de l'affichage
40	H0	Signal de synchronisation horizontale de l'affichage

Implantation des circuits

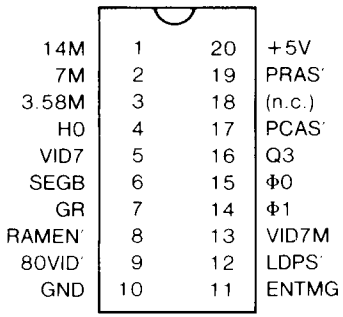
Le circuit PAL

Un dispositif de réseau logique programmable, type PAL 16R8, génère plusieurs signaux de timing et de contrôle dans l'Apple //e. Ces signaux sont énumérés dans la table 7-8.

Tableau 7-8. Description des signaux PAL

N° de broche	Nom	Description
1	14M	Signal maître de timing à 14.25045 Mhz
2	7M	Signal de timing à 7.125225 MHz
3	3.58M	Signal de timing à 3.562612 MHz
4	H0	Signal de timing de vidéo horizontal
5	VID7	Bit 7 de données vidéo
6	SEGB	Signal de timing vidéo
7	GR	Autorisation du mode graphique vidéo
8	RAMEN'	Autorisation de MEV (autorise CAS)
9	80VID'	Autorisation du mode vidéo 80 colonnes
10	GND	Commun d'alimentation et des signaux
11	ENTMG	Autorise le timing-maître
12	LDPS'	Autorise le chargement du registre à décalage de l'affichage vidéo
13	VID7M	Horloge des points vidéo, 7 ou 14 MHz
14	ϕ0	Horloge du système-phase 1
15	ϕ0	Horloge du système-phase 0
16	Q3	Signal intermédiaire d'échantillonnage et de timing
17	PCAS'	Échantillonneur d'adresse-colonne de MEV
18	N.C.	(Non connecté)
19	PRAS'	Échantillonneur d'adresse-rangée de MEV
20	+5V	Alimentation

Figure 7-4. Le Brochage du PAL



L'adressage des mémoires

Le microprocesseur 6502 peut adresser 65 536 positions de mémoire. L'Apple //e utilise cet espace d'adresses en entier et, en plus, quelques zones en mémoire sont utilisées pour plus d'une fonction. Les paragraphes suivants décrivent les dispositifs de mémoire utilisés dans l'Apple //e et la façon de les adresser. Les entrées et sorties se servent aussi de portions de l'espace d'adresses de mémoire ; pour information se reporter au paragraphe « Espaces de mémoire pour cartes périphériques » dans le chapitre 6.

Adressage de la mémoire morte (MEM)

Dans l'Apple //e, les programmes suivants sont enregistrés définitivement dans deux MEM (mémoire à lecture seulement) de type 2364 de 8K par 8 bits :

- L'interpréteur et l'éditeur Applesoft
- Le programme moniteur
- Les sous-programmes d'affichage en 80 colonnes
- Les sous-programmes d'auto-test

Ces deux MEM sont autorisées par deux signaux appelés ROMEN1 et ROMEN2. La MEM autorisée par ROMEN1, quelquefois appelée MEM de diagnostic, occupe l'espace d'adresses de mémoire de $C100 à $DFFF. L'espace d'adresses de $C300 à $C3FF et de $C800 à $CFFF contient les sous-programmes d'affichage en 80 colonnes. Ces espaces d'adresses sont normalement affectés à la MEM d'une carte périphérique installée dans le connecteur 3 ; pour comprendre comment les sous-programmes pour 80 colonnes prennent le dessus sur la carte périphérique, voir le paragraphe « Autres utilisations de l'espace de mémoire d'E/S » du chapitre 6.

Deux autres portions de la MEM de Diagnostic, d'adresses $C100 à $C2FF et $C400 à $C7FF, contiennent les sous-programmes d'auto-test du système. Ces espaces d'adresses sont normalement affectés aux cartes périphériques ; quand les programmes d'auto-test sont en marche, les cartes périphériques sont inhibées.

Le reste de la MEM de diagnostic, d'adresses $D000 à $DFFF, contient une partie de l'interpréteur Applesoft.

La MEM autorisée par ROMEN2, appelée quelque fois la MEM du moniteur, occupe l'espace d'adresses de mémoire de $E000 à $FFFF. Cette MEM contient le reste de l'interpréteur Applesoft, dans l'espace d'adresses de $E000 à $EFFF, et les sous-programmes du moniteur, de $F000 à $FFFF.

Figure 7-5. Le brochage de la MEM 2364

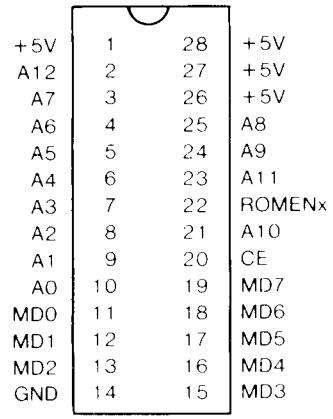


Figure 7-6. Le brochage de la MEM 2316

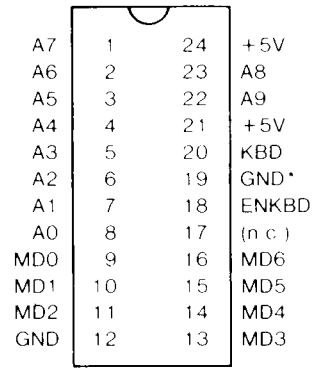
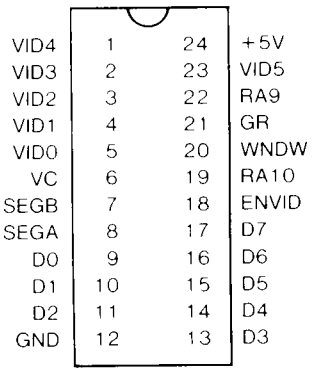


Figure 7-7. Le brochage de la MEM 2333



Les autres MEM de l'Apple //e sont une MEM de type 2316 utilisée par le décodeur de caractères du clavier et une MEM de type 2333 utilisée pour les ensembles de caractères de l'affichage vidéo. Cette MEM 2333 est plutôt grande parce qu'elle comprend une partie des fonctions logiques traitées exhaustivement par application bit à bit (« bit mapping ») pour les modes graphiques. De cette façon, l'affichage vidéo des graphiques peut passer par les mêmes circuits que pour le texte sans qu'il y ait besoin de circuits logiques additionnels.

Adressage de la mémoire vive (MEV)

La mémoire MEV (programmable) dans l'Apple //e est utilisée à la fois pour l'enregistrement des programmes et des données et pour l'affichage vidéo. Les zones en MEV qui sont utilisées pour l'affichage sont accessibles à la fois au microprocesseur 6502 et aux circuits d'affichage vidéo. Dans certains ordinateurs, cette dualité d'accès conduit à des conflits d'adressage (vol de cycle) qui peuvent provoquer des absences temporaires dans l'affichage vidéo. Ce problème ne se rencontre pas dans l'Apple //e, grâce à la façon dont le microprocesseur et les circuits vidéo se partagent la mémoire.

Les circuits de mémoire dans l'Apple //e profitent de l'horloge du système à deux phases décrite dans le paragraphe « Timing du système » pour entrelacer les accès en mémoire du microprocesseur et les accès en mémoire des circuits d'affichage de telle sorte qu'ils ne pourront pas interférer l'un avec l'autre. Le microprocesseur n'écrit ou ne lit en MEV que durant la phase 0, et les circuits d'affichage ne lisent leurs données que durant la phase 1.

Rafraîchissement des MEV dynamiques

L'image sur un écran vidéo n'est pas fixe ; elle s'affaiblit rapidement et doit être rafraîchie périodiquement. Pour rafraîchir l'écran vidéo, l'Apple //e lit les données dans la page d'affichage active et les envoie sur l'écran. Pour empêcher un clignotement visible sur l'écran, et pour se conformer à l'usage standard en diffusion vidéo, l'Apple //e rafraîchit l'écran cinquante fois par seconde.

Les dispositifs des MEV dynamiques utilisés dans l'Apple //e demandent aussi une sorte de rafraîchissement, parce que les données sont mémorisées sous forme de charges électriques qui diminuent avec le temps et doivent être réapprovisionnées souvent. L'Apple //e est conçu de telle sorte que le rafraîchissement de l'écran rafraîchit aussi les MEV dynamiques. Les prochains paragraphes expliquent comme cela est réalisé.

Le travail de rafraîchissement des composants des MEV dynamiques est minimisé par la structure des composants eux-mêmes. Les cellules individuelles de données dans chaque composant de MEV sont arrangées en réseau rectangulaire de rangées et de colonnes. Quand le composant est adressé, la partie de l'adresse qui spécifie

une rangée est présentée en premier, suivie par l'adresse de la colonne. La division d'informations en partie qui se succèdent chacune à son tour est appelée *multiplexation.* Puisque seulement la moitié de l'adresse est nécessaire à chaque instant, la multiplexation de l'adresse réduit le nombre de broches nécessaires pour connecter les MEV.

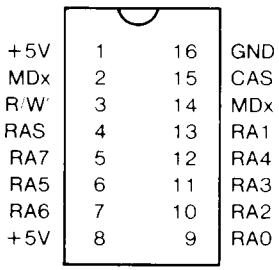
Des MEV de 64K de différents fabricants ont des réseaux de cellules de 128 rangées par 512 colonnes ou bien de 256 rangées par 256 colonnes. Seule la partie rangée de l'adresse est utilisée dans le rafraîchissement des MEV.

Maintenant voyons comment l'écran est rafraîchit. Comme cela est décrit plus loin dans le paragraphe « Les compteurs vidéo », les circuits d'affichage génèrent une séquence de 8 192 adresses de mémoire en mode haute-résolution ; en modes texte et basse-résolution, cette séquence est la page d'adresses d'affichage de 1 024 positions répétée huit fois. L'adresse d'affichage boucle dans cette séquence 50 fois par seconde, ou une fois chaque 20 millisecondes. La façon dont les lignes d'adresses de plus faible poids sont affectées aux MEV consiste à ce que l'adresse-rangée boucle à travers les 256 valeurs possibles une fois par demi-seconde (voir le tableau 7-9). Ceci est plus que suffisant pour satisfaire les besoins de rafraîchissement des MEV dynamiques.

Tableau 7-9. Multiplexation de l'adresse de MEV

Adresse Multiplexée	Adresse rangée	Adresse colonne
RA0	A0	A9
RA1	A1	A6
RA2	A2	A10
RA3	A3	A11
RA4	A5	A13
RA6	A7	A14
RA7	A8	A15

Figure 7-8. Le brochage de la MEV 64K



Timing des MEV dynamiques

L'horloge du microprocesseur de l'Apple *//e* marche à une vitesse modérée d'environ 1.018 MHz, mais l'entrelacement des cycles de l'UC et de l'affichage signifie que l'on accède à la MEV à une fréquence de 2 MHz, ou que le temps de cycle est juste au-dessous de 500 nanosecondes. Une donnée pour l'UC est échantillonnée par le front descendant de $\phi 0$, et une donnée pour l'écran est échantillonnée par le front descendant de $\phi 0$, comme le montre la figure 7-9.

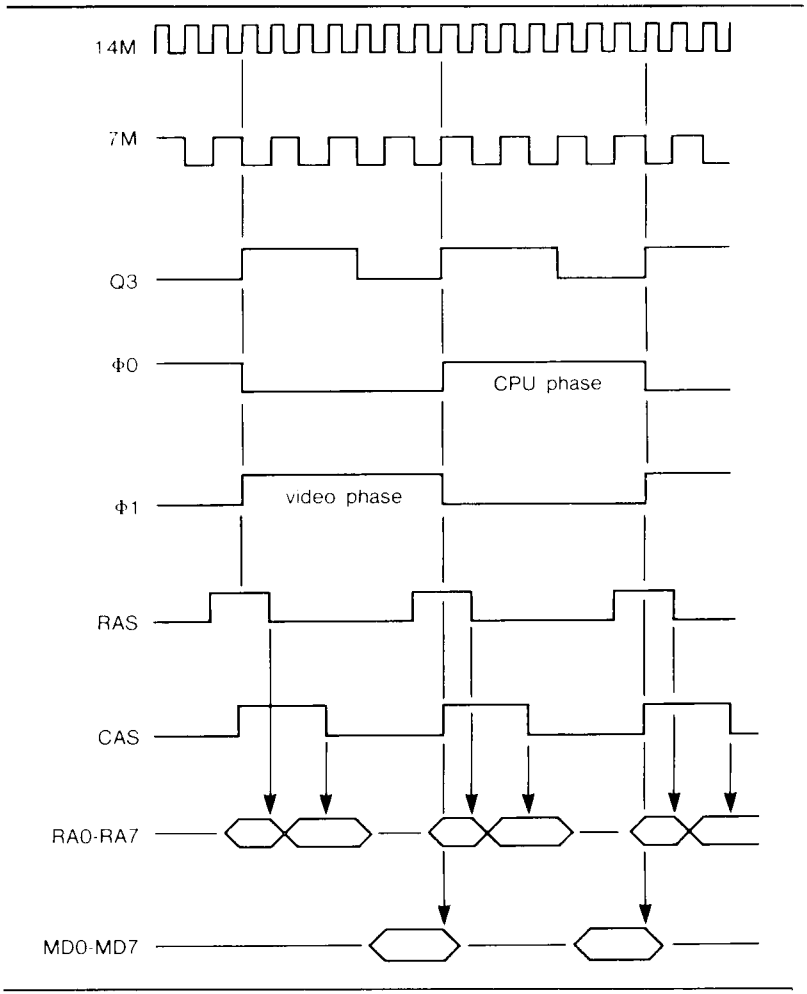
Le timing de la MEV paraît compliqué parce que l'adresse de MEV est multiplexée, comme cela fut décrit dans le paragraphe précédent. La MMU prend soin de la multiplexation de l'adresse pour le cycle de l'UC, puis l'IOU effectue la même opération pour le cycle de l'affichage. L'adresse multiplexée est envoyée aux CI de MEV sur les lignes marquées RA0-RA7. Avec d'autres signaux de timing, le PAL génère deux signaux qui contrôlent l'adressage de MEV : l'échantillonneur d'adresse-rangée (RAS) et l'échantillonneur d'adresse-colonne (CAS).

Tableau 7-10. Signaux de timing des MEV dynamiques

Nom du signal	Description
$\phi 0$	Phase 0 de l'horloge (phase de l'UC)
$\phi 1$	Phase 1 de l'horloge (phase de l'affichage)
RAS	Échantillonneur d'adresse-rangée
CAS	Échantillonneur d'adresse-colonne
Q3	Échantillonneur alternatif d'adresse-colonne
RA0-RA7	Bus d'adresses multiplexées
MD0-MD7	Bus de données internes

L'adressage des mémoires

Figure 7-9. Signaux de timing des MEV



L'affichage sur écran vidéo

L'Apple //e produit un signal vidéo qui crée un affichage sur un moniteur vidéo standard ou, si vous ajouter un modulateur HF, sur un téléviseur noir-et-blanc ou couleurs. Le signal vidéo est un signal composite constitué des données qui sont affichées et, en plus, des signaux de synchronisations horizontale et verticale dont le moniteur vidéo se sert pour arranger les lignes de données affichées sur l'écran.

Les Apple //e fabriqués pour la vente aux U.S.A. génèrent un signal vidéo qui est compatible avec les normes imposées par le NTSC (« National Television Standards Committee » ou Commission Nationale des Normes de Télévision). Les Apple //e internationaux fabriqués pour la vente dans les autres pays contiennent des circuits additionnels pour générer un signal vidéo qui soit compatible avec la norme P.A.L. (« Phase Alternating Lines » ou Lignes en Phase Alternée). Ce manuel ne décrit que la version P.A.L. des circuits de vidéo.

Implantation des circuits

La portion à afficher du signal vidéo est un voltage variable dans le temps généré à partir d'un flux de bits de données, ou le un correspond à un voltage qui génère un point brillant, et le zéro à un point sombre. Le flux de bits à afficher est généré en rafales qui correspondent aux lignes horizontales de points sur l'écran vidéo. Le signal appelé WNDW' est à son niveau bas pendant ces rafales.

Pendant les intervalles de temps entre les rafales de données, rien n'est affiché sur l'écran. Pendant ces intervalles, appelés *intervalles de suppression* (« blanking intervals »), l'affichage est suspendu et le signal WNDW' est à son niveau haut. Les signaux de synchronisation, abrégés sync, sont produits en rendant le signal appelé SYNC' bas pendant des portions des intervalles de suppression. Les impulsions de sync sont à un voltage équivalent au niveau vidéo plus-noir-que-noir et ne se voient pas sur l'écran.

Les compteurs vidéo

Les signaux de synchronisation et d'adresses qui contrôlent la génération de l'affichage vidéo sont tous dérivés d'une chaîne de compteurs à l'intérieur de l'IOU. Seulement quelques uns de ces signaux sont disponibles à l'extérieur de l'IOU, mais ils sont tous importants pour comprendre le fonctionnement du processus de génération de l'affichage, en particulier l'adressage des mémoires d'affichage décrit au prochain paragraphe.

Le compteur horizontal est constitué de sept étages : H0, H1, H2, H3, H4, H5 et HPE'. L'entrée du compteur horizontal est un signal à 1 MHz qui contrôle la lecture des données à afficher. Le cycle complet du compteur horizontal comprend 65 états. Les six bits H0 à H5 comptent normalement de 0 à 63, puis repartent à zéro. Lorsque ceci se produit, HPE' force un autre pas de comptage avec H0 à H5 maintenus à zéro, donc étendant le nombre total de pas de comptage à 65.

L'IOU utilise les quarante valeurs de comptage horizontal de 25 à 64 en générant la partie de plus faible poids de l'adresse des données d'affichage, comme cela est décrit dans le paragraphe « Projection des adresses d'affichage » plus bas. L'IOU utilise les valeurs de comptage de 0 à 24 pour générer la suppression horizontale, l'impulsion de sync horizontale, et le passage de rafale de couleurs.

Quand le comptage horizontal atteint 65, il signale la fin d'une ligne en déclenchant le comptage vertical. Le compteur vertical a neuf étages : VA, VB, VC, V0, V1, V2, V3, V4 et V5. Quand le comptage vertical atteint 262, l'IOU le réinitialise et il reprend son comptage de zéro. Seules les 192 lignes de balayage sont effectivement affichées ; l'IOU utilise les valeurs de comptage 192 à 261 pour générer la suppression verticale et l'impulsion de sync.

Rien n'est affiché pendant l'intervalle de suppression vertical. (Le comptage des lignes verticales est de 312 au lieu du standard 312.5 parce que, au contraire d'un téléviseur normal, l'écran vidéo de l'Apple //e n'est pas entrelacé.)

Les affichages animés ont quelquefois un clignotement erratique dû au changement des données à afficher au même instant où elles sont affichées. Vous pouvez éviter cela sur l'Apple //e en lisant le signal de suppression verticale (VBL) à l'adresse $C019 et en ne changeant les données à afficher qu'au moment où VBL se trouve à son niveau bas (valeur inférieure à 128).

Adressage des mémoires d'affichage

Comme cela est décrit dans le chapitre 2 au paragraphe « Adressage direct des pages d'affichage », les octets de données ne sont pas enregistrés en mémoire dans le même ordre que celui dans lequel ils apparaissent sur l'écran. Vous pouvez avoir une idée de la façon dont les données à afficher sont mémorisées en utilisant le programme moniteur pour mettre l'écran en mode graphique, puis en enregistrant des données en commençant au début de la page d'affichage à l'adresse héxadécimale $400 et en regardant les effets sur l'écran. Si vous faites cela, vous devez utiliser le mode graphique à la place du texte pour éviter des confusions : l'écran de texte est aussi utilisé pour les entrées et sorties du moniteur.

Si vous voulez que votre programme affiche des données en les enregistrant directement dans les mémoires d'affichage, vous devez tout d'abord transformer les coordonnées d'écran en adresses appropriées de mémoire, comme cela est montré au chapitre 2. Les descriptions qui suivent vous aideront à comprendre comment cette transformation d'adresses est faite et pourquoi elle est nécessaire. Elles n'élimineront pas (hélas !) cette nécessité.

La transformation d'adresses qui replie trois rangées de quarante octets affichés en 128 positions de mémoire continues est la même pour tous les modes d'affichage, aussi est-elle est décrite en premier. Les différences entre les différents modes d'affichage sont expliquées dans le paragraphe « Modes d'affichage vidéo », ci-après.

Projection des adresses d'affichage

Considérons l'affichage le plus simple de l'Apple //e, le mode texte en 40 colonnes. Pour adresser quarante colonnes il faut six bits, et pour adresser vingt-quatre rangées il faut encore cinq bits, soit un total de onze bits d'adresses. L'adressage de l'affichage de cette façon impliquerait 2 048 (deux à la puissance onze) octets de mémoire pour n'afficher que 960 caractères. Le mode 80 colonnes nécessiterait 4 096 octets pour afficher 1 920 caractères. Les portions restantes de mémoire qui ne seraient pas affichées pourraient être utilisées pour mémoriser d'autres données, mais pas facilement, parce qu'elles ne seraient pas contigües.

Implantation des circuits

Au lieu d'utiliser les comptages horizontal et vertical pour adresser directement la mémoire, les circuits internes de l'IOU les transforment en nouveaux signaux d'adresses décrits plus loin. L'adresse d'affichage transformée doit respecter les critères suivants :

- Transformez les 960 octets de texte en 40 colonnes en seulement 1024 octets.
- Balayez les adresses-rangées pour rafraîchir les MEV dynamiques.
- Continuer à rafraîchir les MEV pendant la suppression vidéo.

Les besoins de rafraîchissement de la MEV sont discutés plus haut, dans le paragraphe « Rafraîchissement des MEV dynamiques ».

La transformation ne fait intervenir que les comptages verticaux H3, H4, et H5, et les comptages horizontaux V3 et V4. Les bits de comptage vertical VA, VB et VC adressent les lignes constituant les caractères, et ne sont pas impliqués dans la transformation d'adresses. Les bits de comptage de plus faible poids restant, H0, H1, H2, V0, V1 et V2 sont utilisés directement, et ne sont pas impliqués dans la transformation.

l'IOU effectue une addition qui réduit les cinq bits de poids forts à quatre nouveaux signaux appelés S0, S1, S2 et S3 où S veut dire somme. La figure 7-10 est un diagramme montrant l'addition sous forme binaire, avec V3 pris comme retenue entrante et H5 pris sous sa forme complémentée H5'. Une valeur constante à un est prise comme bit de plus faible poids du premier opérande. Le bit de retenue généré par la somme n'est pas pris en compte.

			V3	retenue entrante
H5'	V3	H4	H3	deuxième opérande
V4	H5'	V4	1	premier opérande
S3	S2	S1	S0	Somme

Figure 7-10. Transformation d'adresse d'affichage

Si cette transformation vous paraît terriblement obscure, essayez-la avec des valeurs effectives. Par exemple, pour le coin le plus à gauche et en haut de l'écran, le comptage vertical vaut 0 et le comptage horizontal 24 : H0, H1, H2 et H5 sont à zéros et H3 et H4 sont à un. La valeur de la somme est zéro, donc la position de mémoire du premier caractère de l'écran est la première position de la page d'affichage, comme vous vous y attendiez.

Les bits horizontaux H0, H1, et H2 et les bits de la somme S0, S1, et S2 constituent l'adresse horizontale transformée (A0 à A6 dans le tableau 7-11). Quand le comptage horizontal augmente de 24 à 63, la valeur de la somme (S3 S2 S1 S0) augmente de zéro à quatre et l'adresse transformée va de 0 à 39, par rapport au début de la page d'affichage.

Les trois bits de plus faible poids du compteur vertical sont V0, V1, et V2. Ces bits contrôlent les bits d'adresses A7, A8, et A9 comme indiqué dans le tableau 7-11, de telle sorte que les rangées 0 à 7 commencent exactement tous les 128 octets. Quand le compteur vertical de rangées atteint 8, V0, V1 et V2 repassent à nouveau par zéro, et V3 passe à un. Si vous faites l'addition de la Figure 7-10 avec H égal 24 (comptage horizontal de la première colonne affichée) et avec V égal à 8, la somme vaut 5 et l'adresse horizontale est 40 ; le premier caractère de la rangée 8 est mémorisé dans la position de mémoire à 40 octets du début de la page d'affichage.

Figure 7-11 Mémoire d'affichage de texte en 40 colonnes. Les positions de mémoire marquées d'un astérisque (*) sont réservées à l'usage des sous-programmes d'E/S périphériques : se reporter au paragraphe « Espace de MEV pour carte périphérique », au Chapitre 6.

	← 128 octets →			
	← 40 octets →	← 40 octets →	← 40 octets →	← 8 octets →
$400	rangée 0	rangée 8	rangée 16	*
$480	rangée 1	rangée 9	rangée 17	*
$500	rangée 2	rangée 10	rangée 18	*
$580	rangée 3	rangée 11	rangée 19	*
$600	rangée 4	rangée 12	rangée 20	*
$680	rangée 5	rangée 13	rangée 21	*
$700	rangée 6	rangée 14	rangée 22	*
$780	rangée 7	rangée 15	rangée 23	*

La Figure 7-11 montre comment les groupes de trois rangées de quarante caractères sont mémorisés en blocs de 128 octets contigus commençant aux adresses limites de 128 octets. Ce diagramme est une autre façon de décrire la projection d'affichage décrite en Figure 2-5. Remarquez que les trois rangées dans chaque bloc de 120 octets ne sont pas adjacentes sur l'écran.

Tableau 7-11 Adressage de la mémoire d'affichage

* Pour ces bits d'adresse, voir le texte et la Table 7-12.

Bit d'adresse de mémoire	Bit d'adresse d'écran
A0	H0
A1	H1
A2	H2
A3	S0
A4	S1
A5	S2
A6	S3
A7	V0
A8	V1
A9	V2
A10	*
A11	*
A12	*
A13	*
A14	*
A15	GND

Le tableau 7-11 montre comment les signaux des compteurs vidéo sont affectés aux lignes d'adresses. H0, H1, et H2 sont les bits de comptage horizontal, et V0, V1, et V2 sont les bits de comptage vertical. S0, S1, S2, et S3 sont les bits d'adresses repliées décrits ci-dessus. Les bits d'adresses marqués d'un astérisque (*) sont différents suivant les modes : voir le tableau 7-12 et les trois prochains paragraphes.

Tableau 7-12 Bits d'adresse de mémoire pour les modes d'affichage

Bits d'adresse	Modes d'affichage : Texte et Basse-Résolution	Haute-Résolution
A10	80VID + PG2'	VA
A11	80VID'.PG2	VB
A12	0	VC
A13	0	80VID + PG2'
A14	0	80VID'.PG2

Modes d'affichage vidéo

Les différents modes d'affichage utilisent tous le système de projection en mémoire décrit au paragraphe précédent, mais ils utilisent des zones de mémoire de différentes tailles à des adresses différentes. Les trois prochains paragraphes décrivent les systèmes d'adressage et les méthodes de génération des signaux vidéo effectifs pour les différents modes d'affichage.

Affichage de texte

Les pages de texte et de graphique basse-résolution débutent aux adresses de mémoire $400 et $800. Le tableau 7-12 montre comment les signaux de mode d'affichage contrôlent les bits d'adresses pour produire ces adresses. Les bits d'adresses A10 et A11 sont contrôlés par PG2 et 80VID, qui sont positionnés par les commutateurs logiciels de pages d'affichage et de texte en 80 colonnes. Les bits d'adresses A12, A13, et A14 sont mis à zéro. Remarquez que si 80VID est actif, il inhibe PG2 : il n'y a qu'une seule page d'affichage en mode 80 colonnes.

Les six bits de plus faible poids de chaque octet de données parviennent directement au générateur de caractères, par le bus de données vidéo VID0-VID5. Les deux bits de plus fort poids sont modifiés par IOU pour sélectionner un des ensembles de caractères primaire ou alternatif et sont envoyés au générateur de caractères sur les lignes RA9 et RA10.

Les données pour chaque rangée de caractères sont lues huit fois, une fois pour chacune des huit lignes de points constituant la rangée de caractères. Les bits de données sont envoyés au générateur de caractères en parallèle avec VA, VB, et VC, les bits de plus faible poids du compteur vertical. Pour chaque caractère à afficher, le générateur de caractères sort une des huit combinaisons préenregistrées de bits, sélectionnée par le nombre de 3 bits constitué de VA, VB, et VC.

Les combinaisons de bits fournies par le générateur de caractères sont chargées dans un registre à décalage 74166 à entrées parallèles et sortie série et elles sortent en un flux de bits en série qui va vers le circuit de sortie vidéo. Le registre à décalage est contrôlé par les signaux appelés LDPS' (pour « load parallel-to-serial shifter » ou chargement parallèle à série du décaleur) et VID7M (pour Vidéo 7MHz). Dans le mode 40 colonnes, LDPS' échantillonne la sortie du générateur de caractère dans le registre à décalage toutes les microsecondes, et VID7M décale les bits vers la sortie 7MHz.

L'adressage pour l'affichage en 80 colonnes est le même que pour l'affichage en 40 colonnes : les 40 colonnes de mémoire d'affichage disponibles sur la carte de texte en 80 colonnes sont adressées en parallèle avec les 40 colonnes de la mémoire principale. Les données provenant de ces deux mémoires atteignent le bus de données vidéo (lignes VID0-VID7) par des registres-tampons 74LS374 à trois états. Ces registres-tampons sont chargés simultanément, mais leurs sorties sont envoyées au générateur de caractères alternativement par ϕ 0 et ϕ 1. Dans le mode 80 colonnes, LDPS' charge les données venant du générateur de caractère dans le registre à décalage deux fois par microseconde, une fois pendant ϕ 0 et une fois pendant ϕ 1, et VID7M marche à 14MHz pour décaler les bits de données en sortie deux fois plus vite.

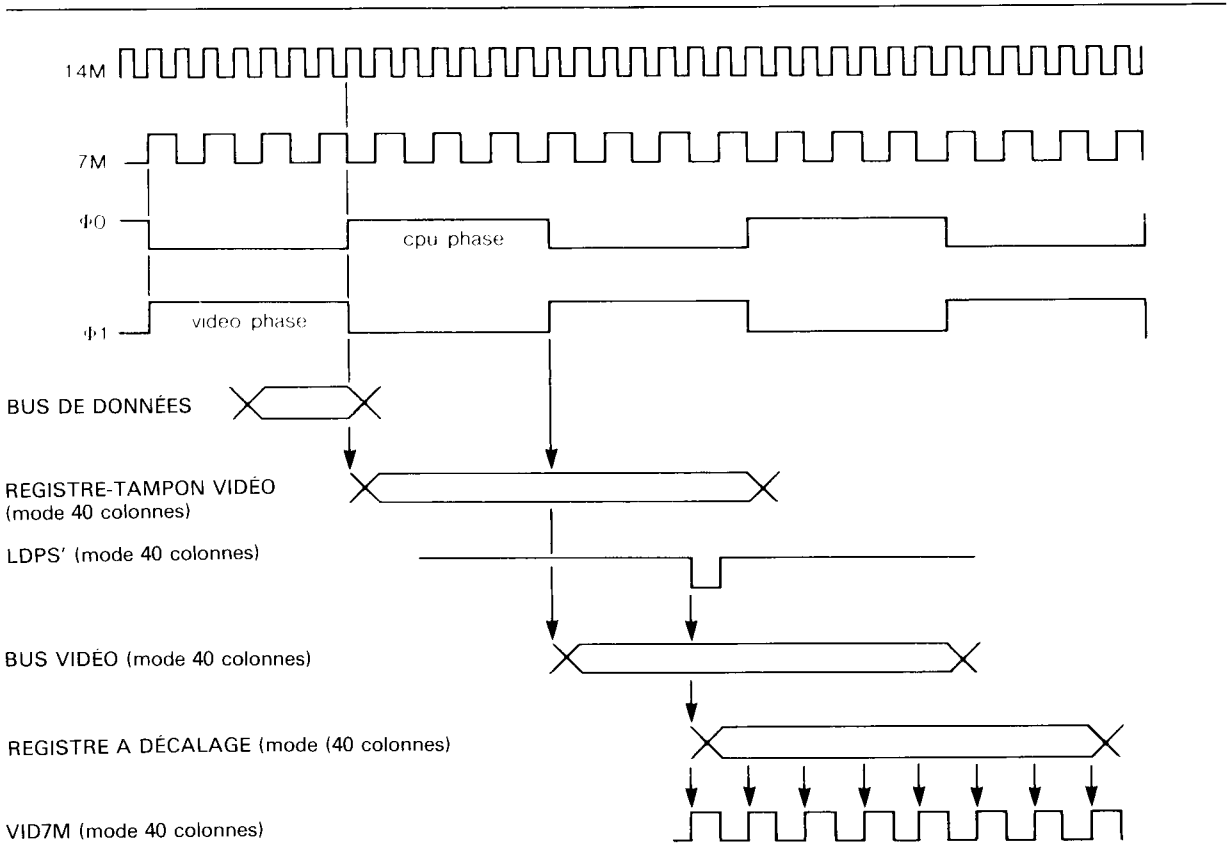


Figure 7-12 Signaux du timing vidéo

L'affichage sur écran vidéo

Affichage en basse-résolution

Dans les modes graphiques, VA et VB ne sont pas utilisés par le générateur de caractères, donc l'IOU utilise les lignes SEGA et SEGB pour transmettre H0 et HIRES', comme indiqué dans le tableau 7-13.

Tableau 7-13. Signaux de contrôle du générateur de caractères

Mode d'affichage	SEGA	SEGB	SEGC
Text	VA	VB	VC
Graphics	H0	HIRES'	VC

L'affichage en graphique basse-résolution utilise VC pour diviser les huit lignes d'écran correspondant à une rangée de caractères en deux groupes de quatre lignes chacun. Chaque rangée d'octets de données est adressée huit fois, comme en mode texte, mais chaque octet est interprété comme deux demi-octets. Chaque demi-octet sélectionne une des seize couleurs. Pendant les quatre lignes d'affichage plus hautes, VC est à son niveau bas et le demi-octet de plus faible poids détermine la couleur. Pendant les quatre lignes d'affichage plus basses, VC est à son niveau haut et le demi-octet de poids fort détermine la couleur.

Les combinaisons de bits qui produisent les couleurs de basse-résolution sont extraites du générateur de caractères en MEM, de la même façon que les combinaisons de bits des caractères sont produites en mode texte. Le registre à décalage parallèle-à-série 74166 convertit les combinaisons de bits en un flux de bits en série pour les circuits de vidéo.

Le signal vidéo généré par l'Apple //e contient une porteuse de couleur en modulation de phase de 4.43MHz utilisable par un moniteur couleur ou un téléviseur couleur P.A.L. Des combinaisons différentes de bits produisent des couleurs différentes en changeant la phase de la porteuse de couleur de 4.43 MHz. Le registre à décalage marche à 14 MHz et décale vers la sortie 14 bits à chaque cycle de l'horloge de données de 1 MHz. Pour générer un flot de quatorze bits de chaque combinaison de huit bits lues dans la MEM, la sortie du registre à décalage est réinjectée à l'entrée série du registre pour répéter les mêmes huit bits ; les deux derniers bits du deuxième tour sont ignorés.

Implantation des circuits

Chaque combinaison est disponible en sortie pendant le même laps de temps que pour un caractère : 0.98 microsecondes. Le générateur de caractères met en sortie une des deux combinaisons différentes pour chaque demi-octet, suivant l'état de H0, le bit de plus faible poids du compteur horizontal.

Affichage en haute-résolution

Les pages graphiques en haute-résolution débutent aux adresses de mémoire $2000 et $4000 (en décimal 8192 et 16384). Ces adresses de page sont sélectionnées par les bits d'adresses A13 et A14. Dans le mode haute-résolution, ces bits d'adresses sont contrôlés par PG2 et 80VID, les signaux contrôlés par les commutateurs logiciels de page d'affichage (PAGE2) et de texte en 80 colonnes (80COL). Comme pour le mode texte, 80VID inhibe l'adressage de la seconde page parce qu'il n'y a qu'une seule page de texte en 80 colonnes en mode mixte.

En mode graphique haute-résolution, les données affichées sont toujours enregistrées en blocs comme ceux montrés à la Figure 7-11, mais il y a huit de ces blocs. Comme l'indique les tableaux 7-11 et 7-12, les compteurs verticaux VA, VB, et VC sont utilisés pour les bits d'adresses A10, A11, et A12, qui adressent huit blocs de 1024 octets chacun. Rappelez-vous que sur l'écran VA, VB, et VC comptent des lignes horizontales adjacentes en groupes de huit. Ce système d'adressage projette chacune de ses huit lignes en un bloc différent de 1024 octets. Cela pourrait aider de penser à une sorte de multiplexeur à huit voies : c'est comme si huit affichages de textes étaient combinés pour produire un seul affichage haute-résolution, où chaque affichage de texte fournirait une ligne de points à son tour, au lieu d'une rangée de caractères.

Les combinaisons de bits en haute-résolution sont produites par le générateur de caractères en MEM. Dans ce mode, les combinaisons de bits reproduisent simplement les huits bits de données à afficher. Les six bits de plus faible poids atteignent le générateur de caractères par le bus de données vidéo VID0-VID5. L'IOU envoie les deux autres bits de données à la MEM par RA9 et RA10.

Les couleurs en haute-résolution décrites au chapitre 2 sont produites par le signal vidéo que les combinaisons de bits génèrent. Les combinaisons de bits en haute-résolution sont toujours décalées à 7 MHz vers la sortie. Toute partie du signal vidéo qui produit un seul point blanc entre deux points noirs, ou vice-versa, est effectivement une courte rafale de 3.58 MHz et est donc affichée en couleur. En d'autres termes, une combinaison de bits constituée de uns et de zéros alternés sera affichée comme une ligne en couleur. Les sous-programmes de graphiques en haute-résolution produisent les combinaisons appropriées de bits en masquant les bits de données avec des uns et des zéros alternés.

Pour produire des couleurs différentes, les combinaisons de bits doivent provoquer des décalages de phase du signal intermédiaire de couleur à 3.58 MHz. Si des uns et des zéros alternés produisent une certaine couleur, disons du vert, alors en inversant la combinaison en zéros et en uns, on produira la couleur complémentaire, le mauve. Les combinaisons de bits produites par les circuits sont les mêmes pour des octets adjacents ; la compensation de couleur est réalisée par le logiciel de haute-résolution, qui utilise des masques de couleur différents pour les données à afficher dans les colonnes paires et impaires.

Pour produire d'autres couleurs, les combinaisons de bits doivent avoir d'autres relations temporelles avec le signal de couleur à 3.58 MHz. En mode haute-résolution, l'Apple //e produit deux couleurs de plus en retardant la sortie du registre à décalage par un demi-point (70 ns), suivant la valeur du bit de poids le plus fort de l'octet de données à afficher. (Le bit de plus fort poids n'est pas affiché effectivement comme un point, parce qu'à 7 MHz, on ne peut décaler que sept des huit bits.)

Pendant que chaque octet de donnée est envoyé du générateur de caractères au registre à décalage, le bit de donnée D7 de plus fort poids est aussi envoyé au dispositif PAL. Si D7 est à zéro, le dispositif PAL transmet les signaux de timing du registre à décalage LDPS' et VID7M normalement. Si D7 est à un, le dispositif PAL retarde LDPS' et VID7M de 70 nanosecondes, le temps correspondant à la moitié d'un point. La combinaison de bits qui avant produisait du vert donne maintenant de l'orange ; la combinaison pour le violet produit maintenant du bleu.

Une remarque sur le timing : Pour le texte en 80 colonnes, le registre à décalage est stimulé par l'horloge à deux fois la vitesse normale. Quand le texte en 80 colonnes est affiché avec des graphiques en mode mixte, le dispositif PAL contrôle les signaux LDPS' et VID7M de timing du registre à décalage de telle sorte que la partie graphique de l'écran fonctionne correctement même si la fenêtre d'écran est en mode 80 colonnes.

Implantation des circuits

Les signaux de sortie vidéo

Le flux de données vidéo, en série NTSC à 3.58 MHz généré par les circuits d'affichage décrits plus haut, est converti en données parallèles sur quatre bits et mises en mémoire dans l'UA15. Les sorties de l'UA15 sont appliquées à travers les résistances R47-R52 montées en circuit linéaire de sommation pour le circuit UA13, un double modulateur balancé. Une porteuse de couleur à 4.43 MHz issue de l'oscillateur à quartz Q7 est appliquée au double modulateur balancé. La sortie de ce modulateur est un signal à 4.43 MHz PAL de couleur à modulation de phase. Le flux de données en série (les points de l'écran), qui contrôle la luminance, est ajouté au signal de chrominance (de couleur) à l'entrée de l'amplificateur vidéo Q1-Q2. Le signal GR sert à insérer un filtre de déroutement (C111, L7, Q3) pour le signal à 3.58 MHz pour empêcher des lignes d'interférences pendant les affichages graphiques. Le commutateur BW/COLOR sur la carte-mère peut être utilisé pour mettre le filtre de déroutement (position BW) pour empêcher des lignes d'interférences pendant les opérations de texte. Les composantes U et V de couleur sont balancées par les potentiomètres R63 et R64.

Le signal vidéo résultant est un signal vidéo composite compatible P.A.L. qui peut être affiché sur un moniteur vidéo standard. Les caractéristiques de ce signal sont données dans le tableau 7-14. Ce signal est disponible à deux endroits dans l'Apple //e :

- à la prise phone à l'arrière de l'Apple //e. L'extérieur de la prise est connecté à la terre et l'extrémité est connectée à la sortie vidéo à travers un réseau de résistances qui l'atténue à environ 1 volt et ajuste son impédance à 75 ohms.
- au connecteur vidéo interne sur la carte-mère de l'Apple //e près de la prise RCA, en J13 sur la figure 7-14c. Il est constitué de quatre broches de type Molex, de 0.25 pouces de haut, et centrées à 0.10 pouce. Ce connecteur transporte le signal vidéo, la terre, et les deux alimentations, comme l'indique le tableau 7-14.

Tableau 7-14 Les signaux du connecteur vidéo interne

Numéro de broche	Nom	Description
1	GROUND	Terre commune du système
2	VIDEO	Signal vidéo composite P.A.L. Le niveau blanc est d'environ 2.0 volts, le niveau noir est d'environ 0.75 volts, le niveau de sync est de 0.0 volts. La sortie **n'est pas** protégée contre les court-circuits.
3	− 5V	Alimentation de −5 volts
4	+ 12V	Alimentation de + 12 volts

L'adressage des mémoires

Les circuits d'E/S de base

L'utilisation des fonctions d'E/S de base de l'Apple //e est décrite dans le chapitre 2. Ce paragraphe décrit l'implémentation du hardware de toutes ces fonctions sauf l'affichage vidéo décrit dans les paragraphes précédents. L'IOU (l'Unité d'Entrée/Sortie) génère directement les signaux de sortie pour le haut-parleur, l'interface cassette et les annonciateurs. Les autres fonctions d'E/S sont prises en charge par des CI plus petits, comme on le décrira plus loin.

Les adresses des fonctions d'E/S de base sont décrites au chapitre 2 et énumérés dans le tableau 2-2, le tableau 2-12, et le tableau 2-13. Toutes les fonctions d'E/S de base sauf les affichages utilisent des positions de mémoire entre $C000 et $C070 (en décimal 49152 et 49264). Le décodage des adresses d'E/S est réalisé par trois CI : un 74LS138, un 74LS154, et un 74LS251.

Le 74LS138 décode les lignes d'adresses A8, A9, A10, et A11 pour sélectionner les pages d'adresses aux limites des blocs de 256 octets en commençant à l'adresses $C000 (en décimal 49152). Quand il détecte des adresses entre $C000 et $C0FF, il autorise l'IOU et le 74LS154. Le 74LS154 à son tour décode les lignes d'adresses A4, A5, A6, et A7 pour sélectionner des zones d'adresses de 16 octets entre $C000 et $C0FF. Les adresses entre $C060 et $C06F autorisent le 74LS251 qui multiplexe les boutons-poussoirs et les manettes de jeu ; les adresses entre $C070 et $C07F remettent à zéro le quadruple temporisateur NE558 qui fait l'interface aux manettes de jeu, comme cela est décrit dans le paragraphe « Signaux d'E/S de jeu ».

Le clavier

Le clavier de l'Apple //e est une matrice de touches connectées à un décodeur de clavier de type AY-3600 par un câble en ruban et un connecteur à 26 broches. Le AY-3600 balaye sans arrêt la matrice de touches pour détecter si une touche est enfoncée. La fréquence de balayage est donnée par un circuit résistance-capacité externe constitué de C70 et R32. Le temps de rebondissement est aussi fixé extérieurement par C71.

Les sorties de l'AY-3600 comprennent cinq bits du code de la touche plus des lignes séparées pour les touches Ctrl , ⇧ , une-touche-enfoncée, et l'échantillonneur du clavier. Les lignes d'échantillonneur du clavier et d'une-touche-enfoncée sont connectées à l'IOU, qui les adressent comme des commutateurs logiciels. Les lignes du code de la touche, en parallèle avec celles de Ctrl et ⇧ , sont des entrées d'une MEM 2316 séparée. La MEM les traduit en codes de caractères qui sont autorisés sur le bus de données par les signaux appelés KBD' et ENKBD'. Sur l'Apple //e International, ces codes de caractères seront, soit pour le jeu de

caractères local (français) ou soit pour le jeu de caractères Nord-Américain, suivant la position de l'inverseur situé sous le côté droit du clavier. Le signal KBD' est autorisé par la MMU chaque fois qu'un programme lit à l'adresse $C000, comme cela est décrit au chapitre 2.

Tableau 7-15 Les signaux du connecteur du clavier

Numéro de broche	Nom	Description
1, 2, 4, 6, 8, 10, 23, 25, 12, 22	Y0-Y9	Connexions de la matrice des touches dans la direction Y
3	+ 5	Alimentation de + 5 Volts
5, 7, 9, 15	n.c.	non connectées
1	LCNTL'	Ligne de la touche Ctrl
13	GND	Terre commune du système
14, 16, 20, 21, 19, 26, 17	X0-X7	Connexions de la matrice des touches dans la direction X
24	LSHFT'	Ligne de la touche ⇧

Le branchement d'un clavier numérique

Il y a un plus petit connecteur branché en parallèle sur le connecteur du clavier. Vous pouvez brancher un bloc-clavier de dix touches numériques à l'Apple //e sur ce connecteur.

Tableau 7-16 Les signaux du connecteur du clavier numérique

Numéro de broche	Nom	Description
1, 2, 5, 3, 4, 6	Y0-Y5	Connexions de la matrice de touches dans la direction Y
7	n.c.	non connectée
9, 11, 10, 8	X4-X7	Connexions de la matrice de touches dans la direction X

L'E/S sur cassette

Les deux prises miniatures de type jack à l'arrière de l'Apple //e sont utilisées pour brancher un lecteur/enregistreur de cassettes audio pour sauvegarder des programmes. Le signal de sortie allant à l'enregistreur provient d'une broche de l'IOU par un circuit de résistances R6 et R9, qui atténue le signal à un niveau approprié à l'entrée microphone de l'enregistreur. L'entrée venant du lecteur est amplifiée et conditionnée par un amplificateur opérationnel de type 741 et envoyée à une des entrées du multiplexeur d'entrées 74LS251.

Les spécifications du signal pour l'E/S sur cassette sont :

Entrée : 1 volt (nominal) de la sortie Earphone (écouteur) ou Monitor du magnétophone. L'impédance d'entrée est de 12 K ohms.

Sortie : 25 millivolts pour l'entrée Microphone du magnétophone. L'impédance de sortie est de 100 ohms.

Le haut-parleur

Le haut-parleur installé dans l'Apple //e est contrôlé par un seul bit de sortie venant de l'IOU (Unité d'Entrée/Sortie). Le signal de l'IOU est couplé en AC à Q5, un amplificateur Darlington à transistor MPSA13. Le connecteur du haut-parleur est un connecteur Molex KK100, J18 dans la Figure 7-14b, avec deux broches carrées de 0.25 pouce de haut et centrées à 0.10 pouces.

Une diode émettrice de lumière est branchée en parallèle sur les broches du haut-parleur de telle sorte que, lorsque le haut-parleur n'est pas connecté, la diode scintille chaque fois que le signal du haut-parleur est à un. La diode est utilisée comme un indicateur diagnostic pendant l'assemblage et les tests de l'Apple //e.

Tableau 7-17 Les signaux du connecteur du haut-parleur

Numéro de broche	Nom	Description
1	SPKR	Signal du haut-parleur. Cette ligne délivrera environ 0.5 watts dans un haut-parleur de 5 ohms.
2	+ 5	Alimentation à + 5V. Notez que le haut-parleur n'est pas connecté à la terre du système.

Les signaux d'E/S de jeu

Plusieurs signaux d'E/S, qui sont individuellement contrôlés par des commutateurs logiciels, sont collectivement référencés comme signaux de jeu. Bien qu'ils soient normalement utilisés pour les manettes de jeu, ces signaux peuvent être utilisés pour d'autres applications simples d'E/S. Il y a cinq signaux de sortie : les quatre annonciateurs, numérotés de A0 à A3, et une sortie d'échantillonnage. Il y a trois entrées à un bit, appelées entrées logiques et numérotées SW0 à SW2, et quatre entrées analogiques, appelées manettes ou leviers de jeu et numérotées PDL0 à PDL3.

Les sorties annonciateurs sont commandées directement par l'IOU (Unité d'Entrée/Sortie). Ces sorties peuvent commander ou supporter une charge TTL (Transistor-Transistor Logic) chacune : pour des charges plus lourdes, vous devez utiliser un transistor ou un amplificateur-tampon TTL sur ces sorties. Ces signaux ne sont accessibles que sur le connecteur interne à 16 broches (voir le tableau 7-18).

La sortie d'échantillonnage est une impulsion transmise chaque fois qu'un programme lit ou écrit dans l'adresse $C040. La broche d'échantillonnage est connectée à une sortie du décodeur d'adresses 74LS154. Ce signal TTL est normalement au niveau haut mais il passe au niveau bas pendant ϕ 0 du cycle de l'instruction qui adresse la position $C040. Ce signal n'est disponible que sur le connecteur interne à 16 broches (voir le tableau 7-18).

Les entrées de jeu sont multiplexées ainsi que le signal d'entrée de cassette par le multiplexeur à 8 entrées 74LS251 autorisé par le signal C06X' venant du décodeur d'adresses d'E/S 74LS154. Suivant les bits de poids faible de l'adresse, l'entrée de jeu appropriée est connectée au bit 7 du bus de données.

Les entrées logiques sont des entrées standards TTL Shottky à faible consommation. Pour les utiliser, connectez chacune d'elles à une résistance de 220 ohm connectée à l'alimentation de + 5 volts et à travers des boutons-poussoirs à pôle simple et à contact momentané, à la terre.

Les entrées reliées aux manettes sont connectées aux entrées de temporisation d'un quadruple temporisateur analogique NE558 de type 555. En adressant $C07X, on envoie un signal venant du 74LS154 qui remet à zéro les quatre temporisateurs et fait monter à un (haut) leur sortie. Une résistance variable de moins de 150K ohms, connectée entre une de ces entrées et l'alimentation de + 5v contrôle le temps de charge de l'une des quatre capacités de 0.022 microfarad. Quand la tension aux bornes de la capacité dépasse un certain seuil, la sortie du NE558 repasse à zéro (bas).

Les programmes peuvent déterminer la valeur de la résistance variable en remettant à zéro les temporisateurs et en comptant le temps que met la sortie du temporisateur sélectionné pour passer de haut en bas. Le comptage résultant est proportionnel à la résistance.

Les signaux d'E/S de jeu sont tous disponibles sur un support DIP à 16 broches marqué GAME I/O sur la carte-mère à l'intérieur. L'état des boutons-poussoirs et des manettes sont aussi disponibles sur un connecteur miniature à l'arrière de l'Apple //e ; voir J8 et J15 dans la Figure 7-14d.

Tableau 7-18 Les Signaux du connecteur d'E/S de jeu

Numéro de broche du connecteur interne	Numéro de broche du connecteur arrière	Nom du Signal	Description
1	2	+ 5V	Alimentation + 5V. Le courant total tiré de cette broche ne doit pas dépasser 100 mA.
2, 3, 4	7, 1, 6	PB0-PB2	Entrées logiques. Elles sont des entrées standards 74LS.
5	—	STROBE'	Sortie d'échantillonnage. Cette ligne devient basse pendant ϕ0 d'une instruction de lecture ou d'écriture à l'adresse $C040.
6, 10, 7, 11	5, 8, 4, 9	PDL0-PDL3	Entrées de manettes. Chacune devra être reliée à une résistance variable de 150K ohms connectée au + 5V.
8	3	GND	Terre du système
15, 14, 13, 12	—	AN0-AN3	Annonciateurs. Ils sont des sorties standards 74LS TTL et doivent être amplifiés pour commander autre chose que des entrées TTL.
9, 16	—	n.c.	Rien n'est branché sur ces broches.

Extension de l'Apple IIe

Le circuit imprimé de la carte-mère contient huit connecteurs vides de cartes. Ces connecteurs rendent possible l'ajout de fonctions à l'Apple //e en y installant des cartes périphériques contenant des circuits additionnels. Le chapitre 6 décrit les normes de programmation des cartes périphériques de l'Apple //e. Ce paragraphe décrit les circuits qui les supportent, y compris tous les signaux disponibles sur les connecteurs d'extension.

Les connecteurs d'extension

Les sept connecteurs alignés en travers de la partie arrière de la carte-mère de l'Apple //e sont les connecteurs d'extension, appelés aussi connecteurs périphériques ou simplement connecteurs, numérotés de 1 à 7. Ce sont des connecteurs pour des cartes PC à 50 broches avec des broches séparées de 0.10 pouce. Une carte PC, installée dans un de ces connecteurs, a accès à tous les signaux nécessaires pour réaliser des entrées et des sorties et pour exécuter des programmes en MEV ou en MEM sur la carte. Ces signaux sont décrits brièvement dans les tableaux 7-19a, 7-19b et 7-19c. Les paragraphes suivants décrivent les signaux en général et mentionnent quelques points qui sont souvent survolés. Pour plus de détails, se reporter au diagramme schématique des Figures 7-14a, 7-14b, 7-14C, et 7-14d.

Le bus d'adresses périphériques

Le bus d'adresses du 6502 est amplifié par deux registres-tampons à trois états à huit bits 74LS244. Ces tampons, ainsi qu'un tampon dans la ligne R/W' du 6502, sont validés par un signal dérivé de la chaine DMA' implémentée sur les connecteurs d'extension. En mettant au niveau bas la ligne périphérique DMA', on inhibe les registres-tampons d'adresses et de R/W' pour que les circuits de DMA périphérique puissent prendre le contrôle du bus d'adresses. Les signaux R/W' et d'adresses DMA fournies par une carte périphérique doivent être stables sur toute la durée de ϕ 0 du cycle d'instruction, comme le montre la Figure 7-13.

Un autre signal qui peut être utilisé pour inhiber le fonctionnement normal de l'Apple //e est INH'. En mettant INH' au niveau bas, on inhibe toutes les mémoires de l'Apple //e sauf la partie de l'espace d'E/S entre $C000 et $CFFF. Une carte périphérique, qui utilise soit INH' soit DMA', doit observer un timing approprié ; pour inhiber proprement la MEV et le MEM, le signal d'inhibition doit être stable sur toute la durée ϕ du cycle d'instruction (se référer au chronogramme de la Figure 7-13).

Les dispositifs périphériques doivent utiliser I/O SELECT' et DEVICE SELECT' comme validation. La plupart des CI périphériques ont

besoin que les signaux de validation soient présents pendant un certain laps de temps avant que les données ne soient échantillonnées dans ou hors du dispositif. Rappelez-vous que I/O SELECT' et DEVICE SELECT' ne sont sûrs que pendant que ϕ 0 est à son niveau haut.

Le bus de données périphériques

L'Apple //e a deux versions de bus de données du 6502 : un bus interne MD0-MD7, connecté directement au 6502 ; un bus externe, D0-D7, amplifié en puissance par un registre-tampon de bus octal bidirectionnel 74LS245. Le 6502 est fabriqué en circuit MOS, donc il ne peut supporter que des charges d'une capacité d'environ 130 pF. Si des cartes périphériques sont installées dans tous les sept connecteurs, la charge totale sur le bus de données pourra atteindre 500 pF, donc c'est le 74LS245 qui va amplifier la puissance du bus de données pour les cartes périphériques. Le même argument s'applique si vous utilisez des circuits MOS sur des cartes périphériques : ils n'ont pas assez de puissance pour le bus complètement chargé, donc vous devrez ajouter des amplificateurs-tampons.

Les règles de charge et de couplage

Les tableaux 7-19a, 7-19b et 7-19c montrent les contraintes de couplage et les limites de charge de chaque broche sur les connecteurs d'extension. Le bus d'adresses, le bus de données et la ligne R/W' devront être amplifiés en puissance par des registres tampons à trois états. Rappelez-vous qu'il y a une capacité distribuée considérable sur ces bus et que vous devrez prévoir une tolérance de charge totale correspondante à six cartes périphériques additionnelles. Les dispositifs MOS tels que les PIA et les ACIA ne peuvent faire commuter des charges d'une si lourde capacité. En connectant de tels dispositifs directement sur le bus, cela conduira à des erreurs possibles de niveau et de timing.

Les chaînages du DMA et des interruptions

Les demandes d'interruption (IRQ' et NMI') et le signal d'accès direct en mémoire (DMA') sont disponibles sur tous les sept connecteurs d'extension. Une carte périphérique demande une interruption ou un transfert DMA en mettant la ligne de sortie appropriée au niveau bas (actif). Si deux cartes périphériques demandent une interruption ou un transfert DMA au même instant, elles vont se disputer les bus d'adresses et de données. Pour empêcher cela, deux paires de broches sur chaque connecteur sont cablées en chaîne-marguerite prioritaire circulaire. Les broches du chaînage-marguerite pour les interruptions sont INT IN et INT OUT, et les broches pour le DMA sont DMA IN et DMA OUT, comme l'indique la Figure 7-14d en J1-J7.

Chaque chaine-marguerite travaille comme ceci : la sortie de chaque connecteur va à l'entrée du prochain connecteur de numéro plus élevé. Pour que ces signaux soient utilisables par les cartes dans les connecteurs de numéros plus bas, tous les connecteurs aux numéros plus élevés doivent contenir des cartes et toutes ces cartes doivent relier DMA IN à DMA OUT et INT IN à INT OUT. Chaque fois qu'une carte périphérique utilise la broche DMA', elle ne doit le faire que si sa ligne DMA IN est active, et elle doit inhiber sa ligne DMA OUT pendant qu'elle utilise DMA'. Les lignes INT IN et INT OUT doivent être utilisées de la même manière : valider les circuits d'interruption de la carte avec INT IN, et inhiber INT OUT quand IRQ' ou NMI' sont utilisés.

Figure 7-13 Le timing des signaux des périphériques

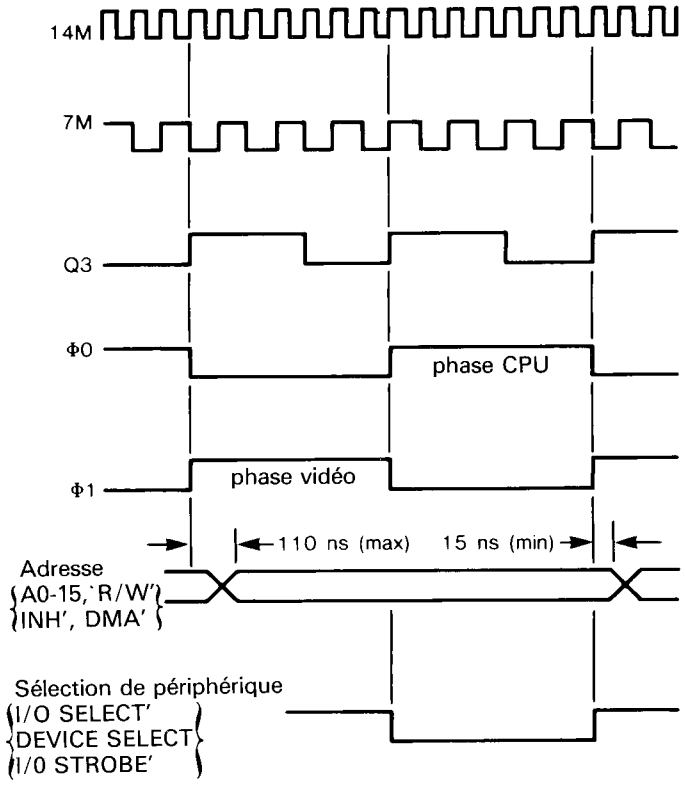


Tableau 7-19a Les signaux de connecteur d'extension

* Les limites de charge sont pour chaque carte

Broche	Nom	Description
1	I/O SELECT	Normalement haut ; passe en bas pendant ϕ Ø quand le 6502 adresse les positions $CnXX, où n est le numéro de connecteur. Cette ligne peut supporter 10 charges LS TTL.*
2-17	A0-A15	Bus d'adresses à trois-états. L'adresse devient valide pendant ϕ 1 et reste valide durant ϕ 0. Chaque ligne d'adresse peut supporter 5 charges LS TTL.*
18	R5W'	Ligne de lecture/écriture à trois-états. Valide au même moment que le bus d'adresses ; au niveau haut pendant un cycle de lecture, bas durant un cycle d'écriture. Elle peut supporter 2 charges LS TTL.*
19	SYNC'	Sync composite horizontale et verticale sur le connecteur d'expansion 7, **seulement**. Cette ligne peut supporter 2 charges LS TTL.*
20	I/O STROBE'	Normalement haut ; passe au niveau bas pendant ϕ0 quand le 6502 adresse une position entre $C800 et $CFFF. Cette ligne peut supporter 4 charges LS TTL.
21	RDY	Entrée pour le 6502. En mettant cette ligne au niveau bas pendant ϕ 1, on arrête le 6502 et le bus d'adresses contient l'adresse de la position recherchée à ce moment. Cette ligne a une résistance de rappel de 3300 ohms connectée à + 5V.
22	DMA'	Entrée pour les registres-tampons du bus d'adresse. En mettant cette ligne au niveau bas pendant ϕ 1, cela déconnecte le 6502 du bus d'adresses. Cette ligne a une résistance de rappel de 3300 ohms au + 5V.
23	INT OUT	Sortie de la chaîne-marguerite prioritaire d'interruption. Habituellement connectée à la broche 28 (INT IN). (Note : sur le connecteur 7 **seulement**, cette broche peut être connectée au signal GR du mode graphique. (Voir les détails dans le texte.)
24	DMA OUT	Sortie de la chaîne-marguerite prioritaire de DMA. Normalement connectée à la broche 22 (DMA IN).
25	+5V	Alimentation + 5 volts. Un courant total de 500 mA est à la disposition des cartes périphériques.
26	GND	Terre commune du système.

Tableau 7-19b Les signaux de connecteur d'extension (suite)

* Les limites de charge sont pour chaque carte.

Broche	Nom	Description
27	DMA IN	Entrée de la chaîne-marguerite prioritaire de DMA. Habituellement connectée à la broche 24 (DMA OUT).
28	INT IN	Entrée de la chaîne-marguerite prioritaire d'interruption. Habituellement connectée à la broche 23 (INT OUT).
29	NMI'	Interruption non-masquable du 6502. En mettant cette ligne au niveau bas cela déclenche un cycle d'interruption avec un sous-programme de traitement de l'interruption débutant à l'adresse $03FB. Cette ligne a une résistance de rappel de 3300 ohms connectée à + 5V.
30	IRQ'	Demande d'interruption au 6502. En mettant cette ligne au niveau bas cela déclenche un cycle d'interruption à condition que l'indicateur (I) d'inhibition d'interruption soit à zéro. Utilise le sous-programme de traitement de l'interruption à l'adresse $03FE. Cette ligne a une résistance de rappel de 3300 ohms connectée à + 5V.
31	RES'	En mettant cette ligne au niveau bas cela initialise une procédure de « reset », comme c'est décrit au chapitre 4.
32	INH'	En mettant cette ligne au niveau bas pendant O 0, cela inhibe la mémoire principale sur la carte-mère. Cette ligne a une résistance de rappel de 3300 ohms au + 5V.
33	− 12V	Alimentation de − 12 volts. Un total de 200 mA est disponible pour toutes les cartes périphériques.
34	− 5V	Alimentation de − 5 volts. Un total de 200 mA est disponible pour toutes les cartes périphériques.
35	3.58M	Signal de référence couleur à 3.58 MHz sur le connecteur 7 **seulement**. Cette ligne peut supporter 2 charges LS TTL.*
36	7M	Horloge du système à 7 MHz. Cette ligne peut supporter 2 charges LS TTL.*
37	Q3	Horloge asymmétrique du système à 2 MHz. Cette ligne peut supporter 2 charges LS TTL.*
38	ϕ1	Phase 1 de l'horloge du 6502. Cette ligne peut supporter 2 charges LS TTL.*

Tableau 7-19c Les signaux de connecteur d'extension, (suite)

* Les limites de charge sont pour chaque carte.

Broche	Nom	Description
39	μ PSYNC	Le 6502 signale une recherche d'opérande en mettant cette ligne au niveau haut pendant le premier cycle de lecture de chaque instruction.
40	ϕ0	Phase ϕ 0 de l'horloge du 6502. Cette ligne peut supporter 2 charges LS TTL.*
41	DEVICE SELECT	Normalement haut ; passe au niveau bas pendant ϕ 0 quand le 6502 adresse les positions $C0nX, où n est le numéro du connecteur plus 8. Cette ligne peut supporter 10 charges LS TTL.*
42-49	D0-D7	Bus de données bi-directionnel amplifié par registre-tampon. Les données deviennent valides pendant que ϕ 0 est au niveau haut et restent valides jusqu'à ce que ϕ 0 passe au niveau bas. Chaque ligne peut supporter une charge LS TTL.*
50	+ 12V	Alimentation de + 12V. Un total de 250 mA est disponible pour toutes les cartes périphériques.

Les signaux vidéo sur le connecteur 7

Les signaux vidéo ne sont disponibles que sur le connecteur auxiliaire et ne le sont pas sur les connecteurs d'extension numérotés, sauf le connecteur 7. Les signaux vidéo accessibles sur le connecteur 7 sont SYNC', le signal sync horizontal et vertical composite, sur la broche 19, et 3.58 M, le signal de référence couleur, sur la broche 35.

Le signal qui autorise les modes graphiques, appelé GR, n'est pas disponible normalement sur les connecteurs d'extension numérotés. Vous pouvez le rendre accessible sur la broche 23 du connecteur 7 en fermant le circuit à la position X7 sur la carte-mère. N'oubliez pas de mettre hors tension avant de changer quoique ce soit à l'intérieur de l'Apple *//e*. N'oubliez pas aussi que de tels changements sont à votre propre risque et peuvent annuler la garantie.

Le connecteur auxiliaire

Le grand connecteur sur la carte-mère de l'Apple *//e* est le connecteur auxiliaire. C'est un connecteur pour carte PC à 60 broches espacées de 0.10 pouce. Une carte PC installée dans ce connecteur a accès à tous les signaux utilisés pour produire de l'affichage vidéo. Ces signaux sont décrits brièvement dans les

Tableaux 7-20a, 7-20b et 7-20c. Pour plus de détails, se reporter au diagramme schématique des Figures 7-14a, 7-14b, 7-14c et 7-14d.

Plusieurs des signaux internes qui ne sont pas accessibles sur les connecteurs d'extension le sont sur le connecteur auxiliaire. En utilisant les deux sortes de connecteurs, le personnel de fabrication et de maintenance peut avoir accès à la plupart des signaux nécessaires aux problèmes de diagnostic dans l'Apple *//e*.

Les signaux d'affichage en 80 colonnes

La mémoire additionnelle nécessaire à l'affichage de texte en 80 colonnes est sur la carte de texte en 80 colonnes, ainsi que des registres-tampons qui transfèrent les données au bus de données vidéo, comme c'est décrit plus haut dans le paragraphe «Affichages de texte ». Les signaux qui contrôlent les données de texte en 80 colonnes comprennent les signaux d'horloge $\phi 0$ et $\phi 1$, les signaux d'adresse multiplexée de MEV RA0-RA7, les signaux d'échantillonnage d'adresse MEV PRAS' et PCAS', et les signaux de validation de la mémoire MEV auxiliaire, EN80' et R/W80. Le signal d'autorisation EN80' est contrôlé par le commutateur logiciel 80STORE décrit au chapitre 4. La donnée est envoyée à la mémoire auxiliaire par le bus de données interne MD0-MD7 ; la donnée est transférée vers le générateur vidéo par le bus de données vidéo VID0-VID7.

Tableau 7-20a Les signaux du connecteur auxiliaire

Broche	Nom	Description
1	3.58M	Signal de référence couleur vidéo de 3.58 MHz. Cette ligne peut supporter deux charges LS TTL.
2	VID7M	Signal d'horloge de sortie des points vidéo du registre à décalage parallèle-à-série 74166. Cette ligne peut supporter deux charges LS TTL.
3	SYNC'	Signal sync horizontale et verticale vidéo. Cette ligne peut supporter deux charges LS TTL.
4	PRAS'	Échantillonnage de la rangée d'adresse multiplexée de MEV. Cette ligne peut supporter deux charges LS TTL.
5	VC	Troisième bit de plus faible poids du compteur vertical. Cette ligne peut supporter deux charges LS TTL.
6	C07X'	Signal de remise à zéro pour les manettes de jeu. Cette ligne peut supporter deux charges LS TTL.
7	WNDW'	Fenêtre vidéo non supprimée. Cette ligne peut supporter deux charges LS TTL.
8	SEGA	Premier bit de plus faible poids du compteur vertical. Cette ligne peut supporter deux charges LS TTL.
51, 10, 49, 48, 13, 14, 46, 9	RA0-RA7	Bus d'adresse multiplexée de MEV. Cette ligne peut supporter deux charges LS TTL.
11, 12	ROMEN1, ROMEN2	Signaux d'autorisation des MEM de la carte-mère.
15	R/W'	Signal de lecture/écriture venant du 6502. Cette ligne peut supporter deux charges LS TTL.
44, 43, 40, 39, 21, 20, 17, 16	MD0-MD7	Bus de données interne (non-amplifié). Cette ligne peut supporter deux charges LS TTL.

Tableau 7-20b Les signaux du connecteur auxiliaire (suite)

Broche	Nom	Description
45, 42, 41, 38, 22, 19, 18, 15	VID0-VID7	Bus de données vidéo. Ce bus à trois états transportent les données vidéo vers le générateur de caractères.
23	ϕ0	Phase 0 de l'horloge du 6502. Cette ligne peut supporter deux charges LS TTL.
24	CLRGAT'	Signal de passage de la rafale de couleur. Cette ligne peut supporter deux charges LS TTL.
25	80VID'	Autorise le timing d'affichage en 80 colonnes. Cette ligne peut supporter deux charges LS TTL.
26	EN80'	Autorise la MEV auxiliaire. Cette ligne peut supporter deux charges LS TTL.
27	ALTVID'	Sortie vidéo alternative pour l'amplificateur sommateur.
28	SEROUT'	Sortie vidéo en série venant du registre à décalage parallèle-à-série 74166.
29	ENVID'	Normalement bas ; en mettant cette ligne au niveau haut, cela inhibe le générateur de caractères de telle sorte que les points vidéo sortant du registre à décalage sont tous hauts (blancs), et le signal vidéo alternatif peut être envoyé par ALTVID'. Cette ligne a une résistance de rabaissement de 1000 ohms à la terre.
30	+5	Alimentation de + 5V.
31	GND	Terre commune du système
32	14M	Signal d'horloge maître de 14.3 MHz. Cette ligne peut supporter deux charges LS TTL.
33	PCAS'	Échantillonnage de l'adresse-colonne multiplexée. Cette ligne peut supporter deux charges LS TTL.
34	LDPS'	Échantillonnage du registre à décalage parallèle-à-série vidéo. Ce signal passe au niveau bas pour charger le contenu du bus de données vidéo dans le registre à décalage. Cette ligne peut supporter deux charges LS TTL.

Tableau 7-20c Les signaux du connecteur auxiliaire (suite)

Broche	Nom	Description
35	R/W80	Signal de lecture/écriture dans la MEV de la carte de texte en 80 colonnes. Cette ligne peut supporter deux charges LS TTL.
36	ϕ1	Phase 1 de l'horloge du 6502. Cette ligne peut supporter deux charges LS TTL.
37	CASEN'	Autorisation de l'adresse-colonne. Ce signal est inhibé (maintenu au niveau haut) pendant les accès dans la mémoire de la carte auxiliaire. Cette ligne peut supporter deux charges LS TTL.
47	HO	Bit de plus faible poids du compteur horizontal. Cette ligne peut supporter deux charges LS TTL.
50	AN3	Sortie de l'annonciateur numéro 3. Cette ligne peut supporter deux charges LS TTL.
52	R/W'	Signal de lecture/écriture du 6502. Cette ligne peut supporter deux charges LS TTL.
53	Q3	Horloge asymmétrique de 2 MHz. Cette ligne peut supporter deux charges LS TTL.
54	SEGB	Deuxième bit de plus faible poids du compteur vertical. Cette ligne peut supporter deux charges LS TTL.
55	ENFIRM	Normalement haut ; en mettant cette ligne au niveau bas on inhibe les MEM1 et MEM2 de la carte-mère. Cette ligne a une résistance de rappel de 3300 ohm au + 5V.
56, 57	RA9', RA10'	Signaux de contrôle du générateur de caractères venant de l'IOU. Cette ligne peut supporter deux charges LS TTL.
58	GR	Signal d'autorisation du mode graphique. Cette ligne peut supporter deux charges LS TTL.
59	7M	Signal de timing à 7 MHz. Cette ligne peut supporter deux charges LS TTL.
60	ENTMG'	Normalement bas ; en mettant cette ligne au niveau haut on inhibe le timing maître venant du PAL. Cette ligne a une résistance de rabaissement de 1000 ohms à la terre.

Figure 7-14a Diagramme schématique, partie 1

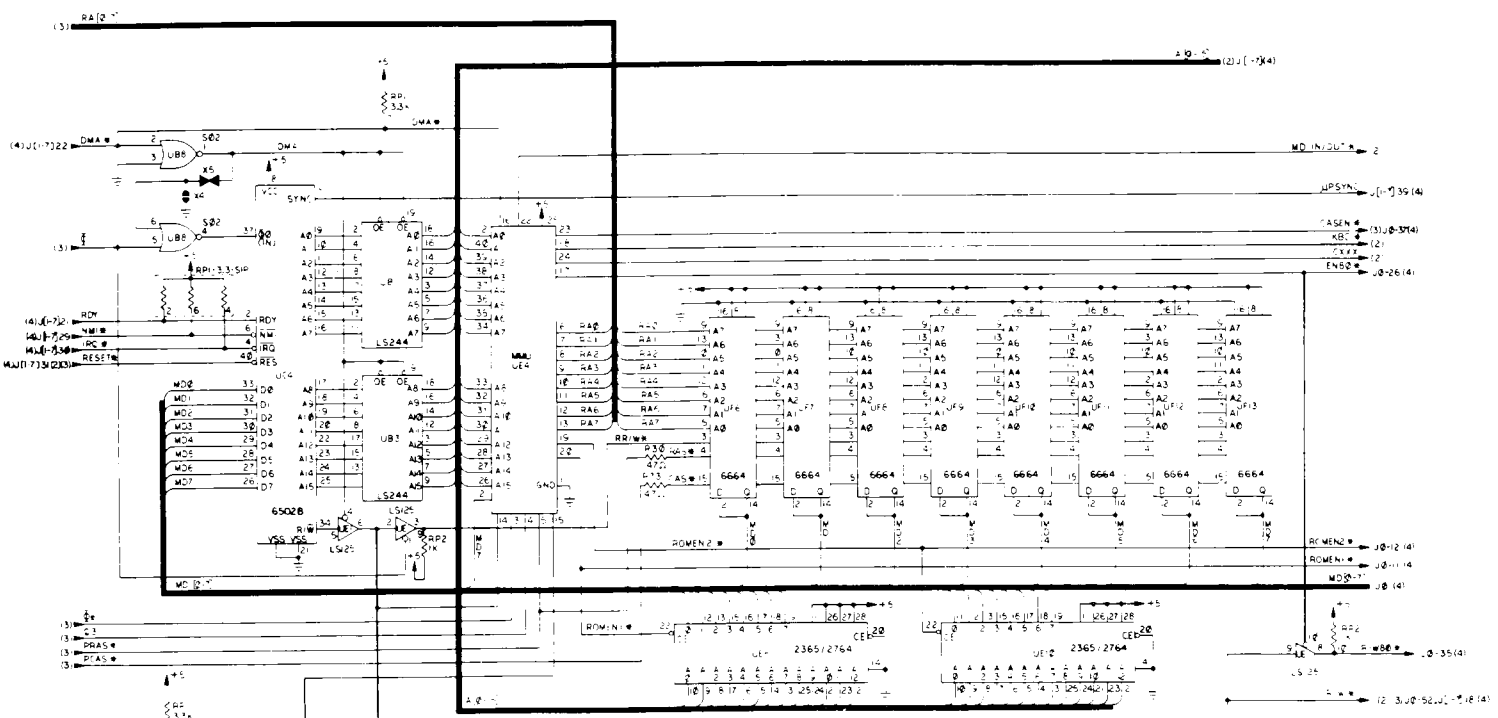
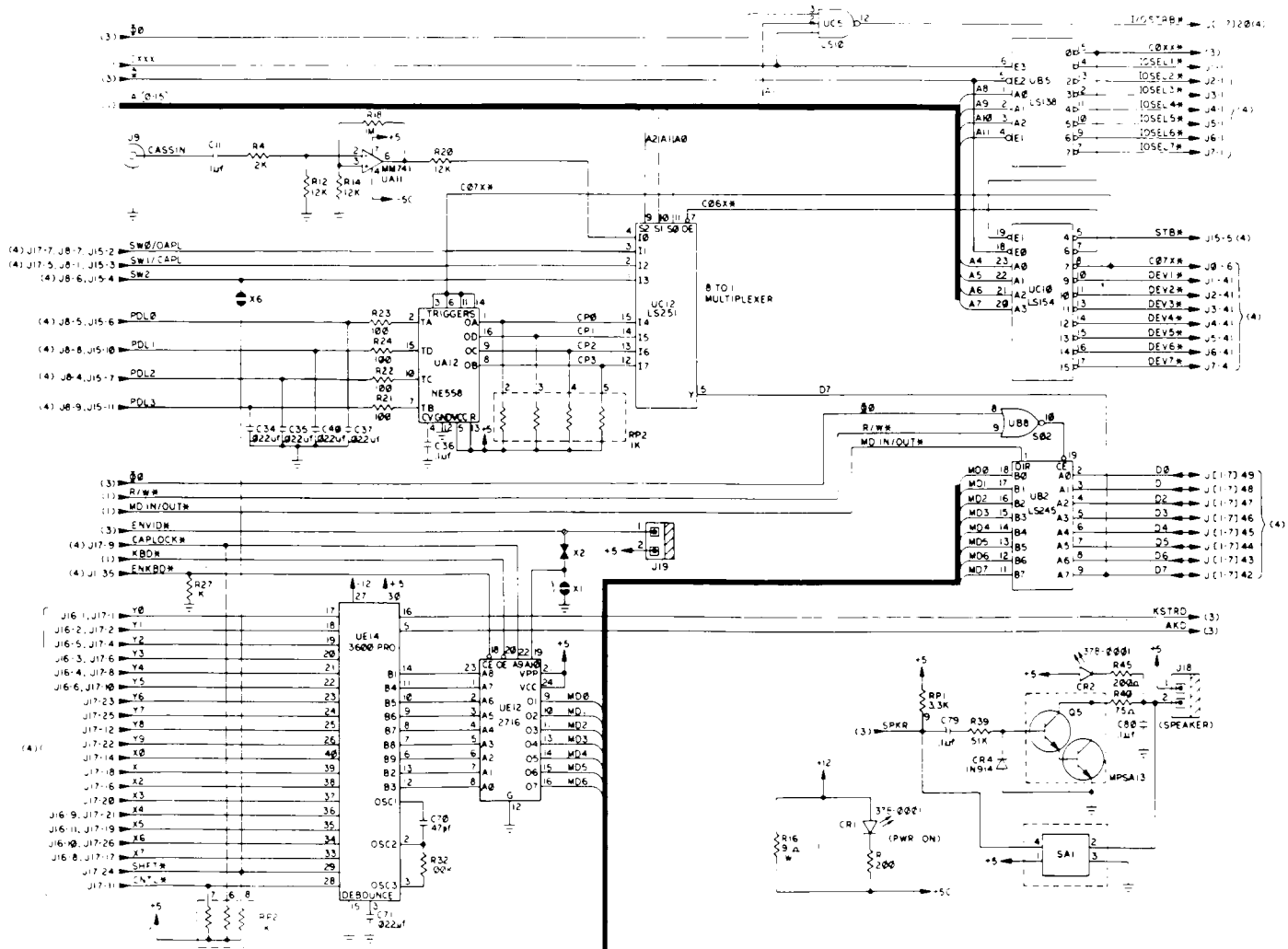


Figure 7-14b Diagramme schématique, partie 2



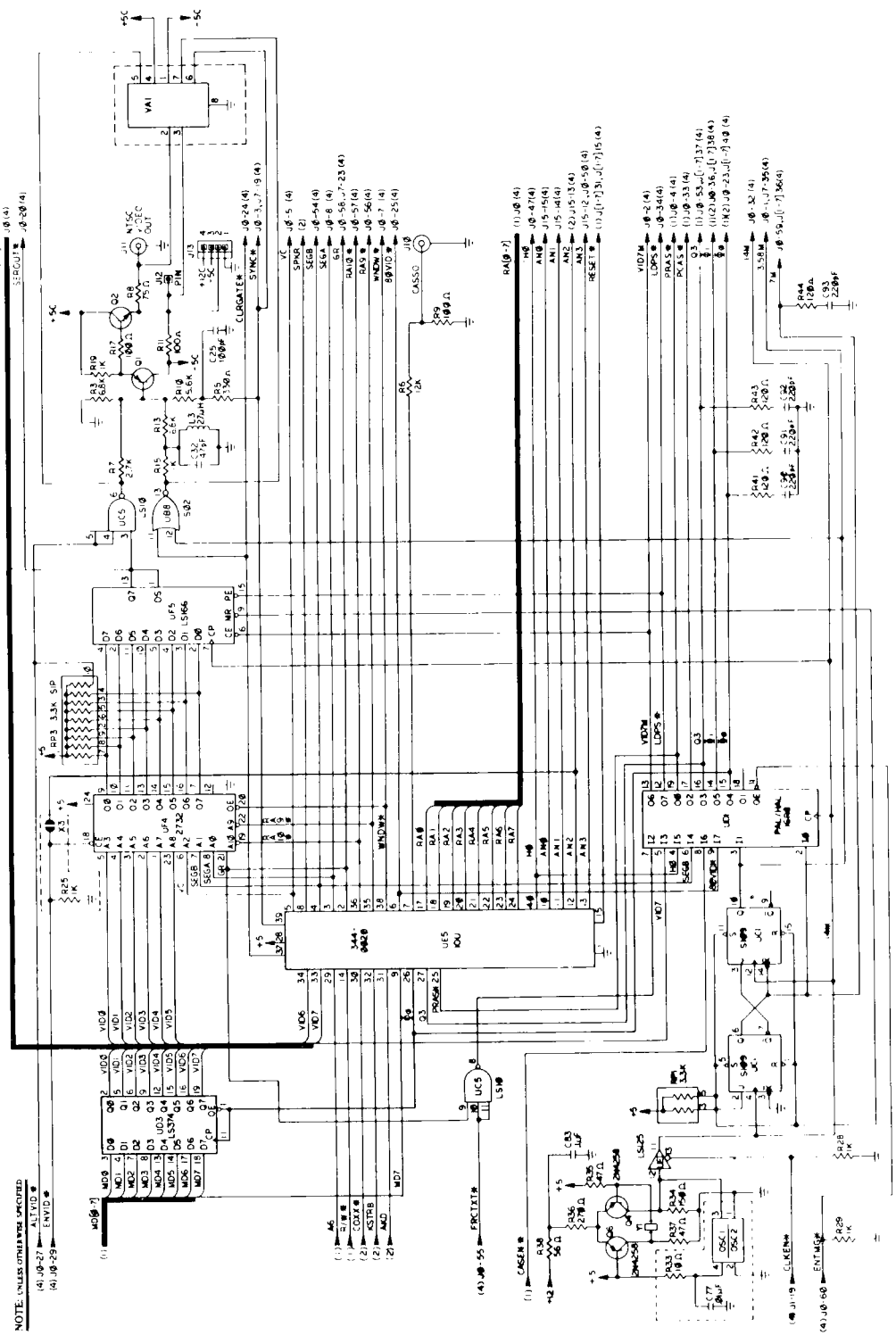
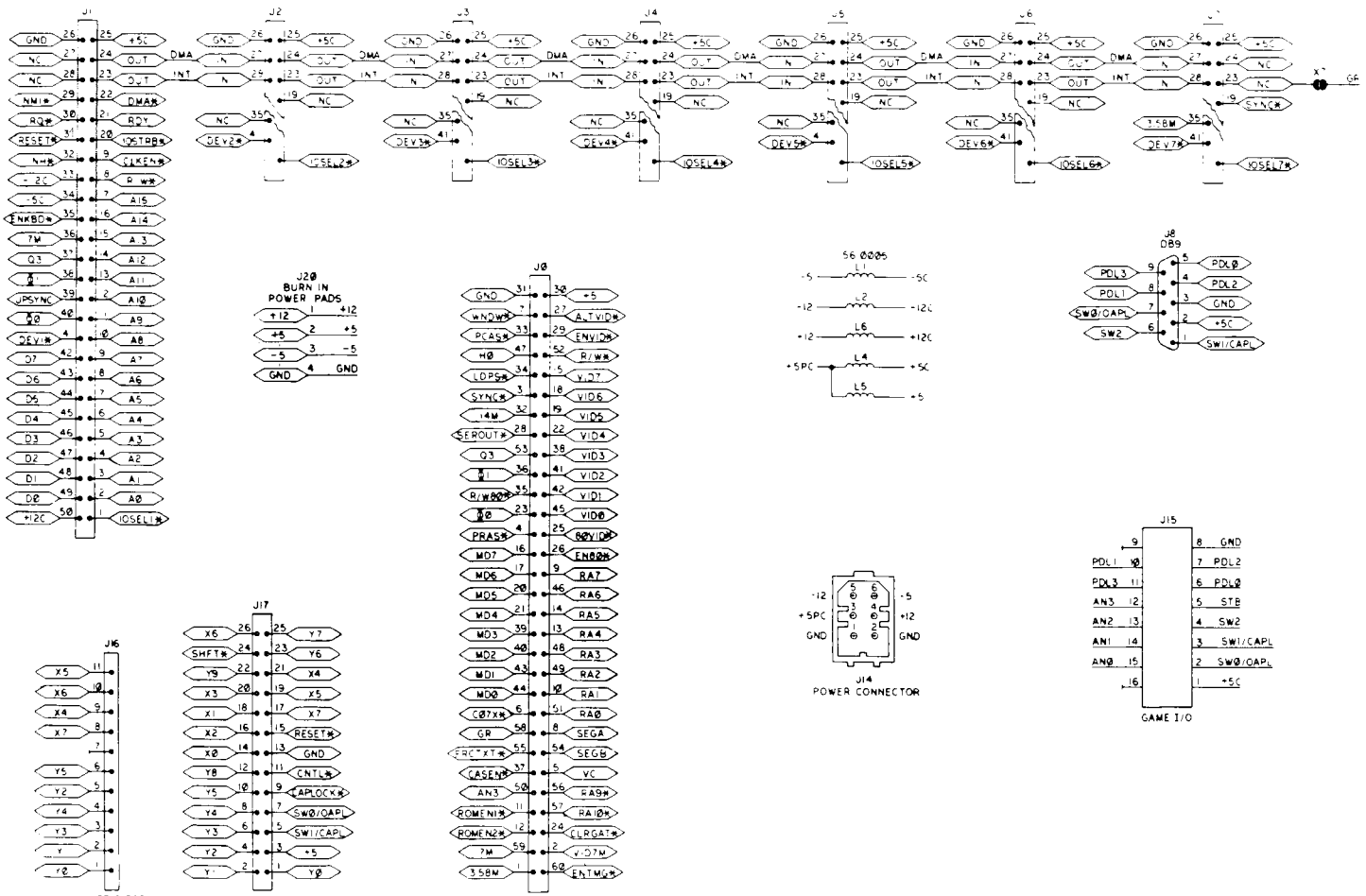


Figure 7-14c Diagramme schématique, partie 3

Figure 7-14d Diagramme schématique, partie 4



Annexes

Le jeu des instructions du 6502

La notation suivante s'applique à ce résumé :

A	Accumulateur
X,Y	Registre d'index
M	Mémoire
C	Retenue complémentée
P	Registre d'état du processeur
S	Pointeur de pile
	Changement
	Pas de changement
	Addition
	AND Logique
	Soustraction
	OR exclusif logique
	Transfert depuis la pile
	Transfert sur la pile
	Transfert vers
	Transfert vers
	VOR logique
PC	Compteur ordinal
PCH	Poids fort du PC
PCL	Poids faible du PC
OPER	Opérande
	Mode d'adressage immédiat

FIGURE 1. ASL-DECALAGE D'UN BIT VERS LA GAUCHE

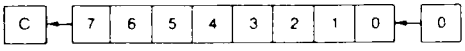


FIGURE 2. ROL-ROTATION D'UN BIT VERS LA GAUCHE (MEMOIRE OU ACCUMULATEUR)

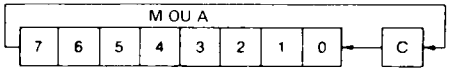
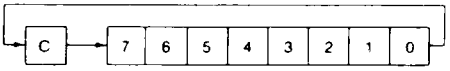


FIGURE 3. ROR-ROTATION D'UN BIT SUR LA DROITE (MÉMOIRE OU ACCUMULATEUR)



NOTE 1 : BIT — TEST DES BITS

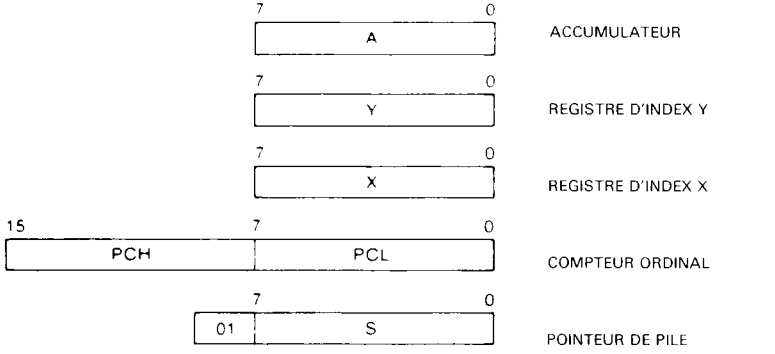
Les bits 6 et 7 sont transférés dans le registre d'état. Si le résultat de A M vaut zéro alors Z = 1, sinon Z = 0.

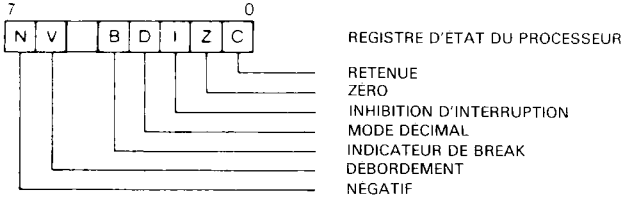
Instructions du microprocesseur 6502

ADC	Addition de la mémoire à l'accumulateur plus la retenue
AND	« ET logique » bit à bit entre accumulateur et mémoire
ASL	Décalage à gauche de l'accumulateur ou d'une mémoire
BCC	Branchement si pas de retenue
BCS	Branchement si retenue à un
BEQ	Branchement si résultat = 0
BIT	Test de bits de la mémoire avec l'accumulateur
BMI	Branchement si résultat négatif
BNE	Branchement si résultat non nul
BPL	Branchement si résultat positif
BRK	Arrêt par interruption programmée
BVC	Branchement si non débordement
BVS	Branchement si débordement
CLC	Annuler la retenue
CLD	Annuler le mode décimal
CLI	Mettre à 1 le bit d'inhibition des interruptions
CLV	Annuler l'indicateur de débordement
CMP	Comparer la mémoire avec l'accumulateur
CPX	Comparer la mémoire avec le registre d'index X
CPY	Comparer la mémoire avec le registre d'index Y
DEC	Diminuer de 1 la mémoire
DEX	Diminuer de 1 le registre X
DEY	Diminuer de 1 le registre Y
EOR	« OU exclusif » entre mémoire et accumulateur
INC	Augmenter de 1 la mémoire
INX	Augmenter de 1 l'index X
INY	Augmenter de 1 l'index Y
JMP	Sauter à une nouvelle adresse
JSR	Sauter à une nouvelle adresse en sauvegardant l'adresse de retour
LDA	Charger l'accumulateur avec la mémoire
LDX	Charger le registre d'index X avec la mémoire
LDY	Charger le registre d'index Y avec la mémoire
LSR	Décaler à droite mémoire ou accumulateur
NOP	Pas d'opération
ORA	« OU » entre mémoire et accumulateur
PHA	Empiler l'accumulateur
PHP	Empiler le registre d'état P
PLA	Dépiler vers l'accumulateur
PLP	Dépiler vers le registre d'état
ROL	Décaler à gauche avec rotation (la mémoire ou l'accumulateur)
ROR	Décaler à droite avec rotation (la mémoire ou l'accumulateur)
RTI	Retour d'interruption
RTS	Retour de sous-programme
SBC	Soustraire la mémoire de l'accumulateur avec la retenue
SEC	Mettre la retenue à un
SED	Mettre en mode décimal
SEI	Mettre à 1 le bit d'inhibition des interruptions
STA	Ranger l'accumulateur dans la mémoire
STX	Ranger l'index X en mémoire
STY	Ranger l'index Y en mémoire
TAX	Transférer A dans X
TAY	Transférer A dans Y
TSX	Transférer S dans X
TXA	Transférer X dans A
TXS	Transférer X dans S
TYA	Transférer Y dans A

Le jeu des instructions du 6502

Registres de programmation





Code des instructions

Name Description	Operation	Addressing Mode	Assembly Language Form	HEX OP Code	No Bytes	"P" Status Reg N Z C I D V
ADC						
Add memory to accumulator with carry	A+M+C → A C	Immediate	ADC #Oper	69	2	√√√ · · √
		Zero Page	ADC Oper	65	2	
		Zero Page,X	ADC Oper,X	75	2	
		Absolute	ADC Oper	6D	3	
		Absolute,X	ADC Oper,X	7D	3	
		Absolute,Y	ADC Oper,Y	79	3	
		(Indirect,X)	ADC (Oper,X)	61	2	
		(Indirect),Y	ADC (Oper),Y	71	2	
AND						
"AND" memory with accumulator	A∧M → A	Immediate	AND #Oper	29	2	√√· · · ·
		Zero Page	AND Oper	25	2	
		Zero Page,X	AND Oper,X	35	2	
		Absolute	AND Oper	2D	3	
		Absolute,X	AND Oper,X	3D	3	
		Absolute,Y	AND Oper,Y	39	3	
		(Indirect,X)	AND (Oper,X)	21	2	
		(Indirect),Y	AND (Oper),Y	31	2	
ASL						
Shift left one bit (Memory or Accumulator)	(See Figure 1)	Accumulator	ASL A	0A	1	√√√· · ·
		Zero Page	ASL Oper	06	2	
		Zero Page,X	ASL Oper,X	16	2	
		Absolute	ASL Oper	0E	3	
		Absolute,X	ASL Oper,X	1E	3	
BCC						
Branch on carry clear	Branch on C=0	Relative	BCC Oper	90	2	· · · · · ·
BCS						
Branch on carry set	Branch on C=1	Relative	BCS Oper	B0	2	· · · · · ·
BEQ						
Branch on result zero	Branch on Z=1	Relative	BEQ Oper	F0	2	· · · · · ·
BIT						
Test bits in memory with accumulator	A∧M, M_7 → N, M_6 → V	Zero Page	BIT[1] Oper	24	2	M_7√· · M_6
		Absolute	BIT[1] Oper	2C	3	
BMI						
Branch on result minus	Branch on N=1	Relative	BMI Oper	30	2	· · · · · ·
BNE						
Branch on result not zero	Branch on Z=0	Relative	BNE Oper	D0	2	· · · · · ·
BPL						
Branch on result plus	Branch on N=0	Relative	BPL oper	10	2	· · · · · ·
BRK						
Force Break	Forced Interrupt PC+2 ↓ P ↓	Implied	BRK[2]	00	1	· · · 1 · ·
BVC						
Branch on overflow clear	Branch on V=0	Relative	BVC Oper	50	2	· · · · · ·

Note 1: Bits 6 and 7 are transferred to the status register. If the result of A AND M is zero, then Z = 1; otherwise Z = 0.

Note 2: A BRK command cannot be masked by setting I.

Name Description	Operation	Addressing Mode	Assembly Language Form	HEX OP Code	No Bytes	"P" Status Reg N Z C I D V
BVS Branch on overflow set	Branch on V=1	Relative	BVS Oper	70	2	
CLC Clear carry flag	0 → C	Implied	CLC	18	1	0
CLD Clear decimal mode	0 → D	Implied	CLD	D8	1	0
CLI	0 → I	Implied	CLI	58	1	0
CLV Clear overflow flag	0 → V	Implied	CLV	B8	1	0

CMP Compare memory and accumulator	A — M	Immediate
		Zero Page
		Zero Page, X
		Absolute
		Absolute,X
		Absolute,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
				C5
				D5
				CD
				DD
				D9
				C1
				D1
					2
					2
					2
					3
					3
					3
					2
					2
						√√√

CPX Compare memory and index X	X — M	Immediate Zero Page Absolute	CPX #Oper CPX Oper CPX Oper	E0 E4 EC	2 2 3	√√√
CPY Compare memory and index Y	Y — M	Immediate Zero Page Absolute	CPY #Oper CPY Oper CPY Oper	C0 C4 CC	2 2 3	√√√
DEC Decrement memory by one	M — 1 → M	Zero Page Zero Page,X Absolute Absolute,X	DEC Oper DEC Oper,X DEC Oper DEC Oper,X	C6 D6 CE DE	2 2 3 3	√√ - - - -
DEX Decrement index X by one	X — 1 → X	Implied	DEX	CA	1	√√ - - - -
DEY Decrement index Y by one	Y — 1 → Y	Implied	DEY	88	1	√√ - - - -

Name Description	Operation	Addressing Mode	Assembly Language Form	HEX OP Code	No. Bytes	"P" Status Reg N Z C I D V
EOR						
"Exclusive-Or" memory with accumulator	A V M → A	Immediate	EOR #Oper	49	2	√√ - - -
		Zero Page	EOR Oper	45	2	
		Zero Page X	EOR Oper,X	55	2	
		Absolute	EOR Oper	4D	3	
		Absolute X	EOR Oper X	5D	3	
		Absolute Y	EOR Oper Y	59	3	
		(Indirect X)	EOR (Oper,X)	41	2	
		(Indirect) Y	EOR (Oper) Y	51	2	
INC						
Increment memory by one	M + 1 → M	Zero Page	INC Oper	E6	2	√√- - - -
		Zero Page,X	INC Oper,X	F6	2	
		Absolute	INC Oper	EE	3	
		Absolute,X	INC Oper,X	FE	3	
INX						
Increment index X by one	X + 1 → X	Implied	INX	E8	1	√√- - - -
INY						
Increment index Y by one	Y + 1 → Y	Implied	INY	C8	1	√√- - - -
JMP						
Jump to new location	(PC+1) → PCL	Absolute	JMP Oper	4C	3	- - - - - -
	(PC+2) → PCH	Indirect	JMP (Oper)	6C	3	
JSR						
Jump to new location saving return address	PC+2 ↓	Absolute	JSR Oper	20	3	- - - - - -
	(PC+1) → PCL					
	(PC+2) → PCH					
LDA						
Load accumulator with memory	M → A	Immediate	LDA #Oper	A9	2	√√- - - -
		Zero Page	LDA Oper	A5	2	
		Zero Page,X	LDA Oper,X	B5	2	
		Absolute	LDA Oper	AD	3	
		Absolute,X	LDA Oper,X	BD	3	
		Absolute,Y	LDA Oper,Y	B9	3	
		(Indirect,X)	LDA (Oper,X)	A1	2	
		(Indirect),Y	LDA (Oper),Y	B1	2	
LDX						
Load index X with memory	M → X	Immediate	LDX #Oper	A2	2	√√- - - -
		Zero Page	LDX Oper	A6	2	
		Zero Page,Y	LDX Oper,Y	B6	2	
		Absolute	LDX Oper	AE	3	
		Absolute,Y	LDX Oper,Y	BE	3	
LDY						
Load index Y with memory	M → Y	Immediate	LDY #Oper	A0	2	√√- - - -
		Zero Page	LDY Oper	A4	2	
		Zero Page,X	LDY Oper,X	B4	2	
		Absolute	LDY Oper	AC	3	
		Absolute,X	LDY Oper,X	BC	3	

Name Description	Operation	Addressing Mode	Assembly Language Form	HEX OP Code	No Bytes	"P" Status Reg. N Z C I D V
LSR						
Shift right one bit (memory or accumulator)	(See Figure 1)	Accumulator	LSR A	4A	1	0√√---
		Zero Page	LSR Oper	46	2	
		Zero Page,X	LSR Oper,X	56	2	
		Absolute	LSR Oper	4E	3	
		Absolute X	LSR Oper,X	5E	3	
NOP						
No operation	No Operation	Implied	NOP	EA	1	------
ORA						
"OR" memory with accumulator	A V M → A	Immediate	ORA #Oper	09	2	√√----
		Zero Page	ORA Oper	05	2	
		Zero Page,X	ORA Oper,X	15	2	
		Absolute	ORA Oper	0D	3	
		Absolute,X	ORA Oper,X	1D	3	
		Absolute,Y	ORA Oper,Y	19	3	
		(Indirect,X)	ORA (Oper,X)	01	2	
		(Indirect),Y	ORA (Oper),Y	11	2	
PHA						
Push accumulator on stack	A ↓	Implied	PHA	48	1	------
PHP						
Push processor status on stack	P ↓	Implied	PHP	08	1	------
PLA						
Pull accumulator from stack	A ↑	Implied	PLA	68	1	√√----
PLP						
Pull processor status from stack	P ↑	Implied	PLP	28	1	From Stack
ROL						
Rotate one bit left (memory or accumulator)	(See Figure 2)	Accumulator	ROL A	2A	1	√√√---
		Zero Page	ROL Oper	26	2	
		Zero Page,X	ROL Oper,X	36	2	
		Absolute	ROL Oper	2E	3	
		Absolute,X	ROL Oper,X	3E	3	
ROR						
Rotate one bit right (memory or accumulator)	(See Figure 3)	Accumulator	ROR A	6A	1	√√√---
		Zero Page	ROR Oper	66	2	
		Zero Page,X	ROR Oper,X	76	2	
		Absolute	ROR Oper	6E	3	
		Absolute,X	ROR Oper,X	7E	3	

Name Description	Operation	Addressing Mode	Assembly Language Form	HEX OP Code	No. Bytes	"P" Status Reg N Z C I D V
RTI Return from interrupt	P↓ PC↓	Implied	RTI	40	1	From Stack
RTS Return from subroutine	PC↓, PC+1 → PC	Implied	RTS	60	1	- - - - - -
SBC Subtract memory from accumulator with borrow	A - M - $\overline{C}$ → A	Immediate Zero Page Zero Page,X Absolute Absolute,X Absolute,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 E5 F5 ED FD F9 E1 F1	2 2 2 3 3 3 2 2	√√√- - - √
SEC Set carry flag	1 → C	Implied	SEC	38	1	- - 1 - - -
SED Set decimal mode	1 → D	Implied	SED	F8	1	- - - - 1 -
SEI Set interrupt disable status	1 → I	Implied	SEI	78	1	- - - 1 - -
STA Store accumulator in memory	A → M	Zero Page Zero Page,X Absolute Absolute,X Absolute,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 95 8D 9D 99 81 91	2 2 3 3 3 2 2	- - - - - -
STX Store index X in memory	X → M	Zero Page Zero Page,Y Absolute	STX Oper STX Oper,Y STX Oper	86 96 8E	2 2 3	- - - - - -
STY Store index Y in memory	Y → M	Zero Page Zero Page,X Absolute	STY Oper STY Oper,X STY Oper	84 94 8C	2 2 3	- - - - - -
TAX Transfer accumulator to index X	A → X	Implied	TAX	AA	1	√√- - - -
TAY Transfer accumulator to index Y	A → Y	Implied	TAY	A8	1	√√- - - -
TSX Transfer stack pointer to index X	S → X	Implied	TSX	BA	1	√√- - - -

Name Description	Operation	Addressing Mode	Assembly Language Form	HEX OP Code	No. Bytes	"P" Status Reg. N Z C I D V
TXA Transfer index X to accumulator	X → A	Implied	TXA	8A	1	√ √ - - - -
TXS Transfer index X to stack pointer	X → S	Implied	TXS	9A	1	- - - - - -
TYA Transfer index Y to accumulator	Y → A	Implied	TYA	98	1	√ √ - - - -

Codes hexadécimaux des opérations

00 — BRK
01 — ORA — (Indirect, X)
02 —
03 —
04 —
05 — ORA — Zero Page
06 — ASL — Zero Page
07 —
08 — PHP
09 — ORA — Immediate
0A — ASL — Accumulator
0B —
0C —
0D — ORA — Absolute
0E — ASL — Absolute
0F —
10 — BPL
11 — ORA — (Indirect), Y
12 —
13 —
14 —
15 — ORA — Zero Page, X
16 — ASL — Zero Page, X
17 —
18 — CLC
19 — ORA — Absolute, Y
1A —
1B —
1C —
1D — ORA — Absolute, X
1E — ASL — Absolute, X
1F —
20 — JSR
21 — AND — (Indirect, X)
22 —
23 —
24 — BIT — Zero Page
25 — AND — Zero Page
26 — ROL — Zero Page
27 —
28 — PLP
29 — AND — Immediate
2A — ROL — Accumulator
2B —
2C — BIT — Absolute
2D — AND — Absolute
2E — ROL — Absolute
2F —
30 — BMI
31 — AND — (Indirect), Y
32 —
33 —
34 —
35 — AND — Zero Page, X
36 — ROL — Zero Page, X
37 —
38 — SEC
39 — AND — Absolute, Y
3A —
3B —
3C —
3D — AND — Absolute, X
3E — ROL — Absolute, X
3F —
40 — RTI
41 — EOR — (Indirect, X)
42 —
43 —
44 —
45 — EOR — Zero Page
46 — LSR — Zero Page
47 —
48 — PHA
49 — EOR — Immediate
4A — LSR — Accumulator
4B —
4C — JMP — Absolute
4D — EOR — Absolute
4E — LSR — Absolute
4F —
50 — BVC
51 — EOR (Indirect), Y
52 —
53 —
54 —
55 — EOR — Zero Page, X
56 — LSR — Zero Page, X
57 —
58 — CLI
59 — EOR — Absolute, Y
5A —
5B —
5C —
5D — EOR — Absolute, X
5E — LSR — Absolute, X
5F —
60 — RTS
61 — ADC — (Indirect, X)
62 —
63 —
64 —
65 — ADC — Zero Page
66 — ROR — Zero Page
67 —
68 — PLA
69 — ADC — Immediate
6A — ROR — Accumulator
6B —
6C — JMP — Indirect
6D — ADC — Absolute
6E — ROR — Absolute
6F —
70 — BVS
71 — ADC — (Indirect), Y
72 —
73 —
74 —
75 — ADC — Zero Page, X
76 — ROR — Zero Page, X
77 —
78 — SEI
79 — ADC — Absolute, Y
7A —
7B —
7C —
7D — ADC — Absolute, X
7E — ROR — Absolute, X
7F —
80 —
81 — STA — (Indirect, X)
82 —
83 —
84 — STY — Zero Page
85 — STA — Zero Page
86 — STX — Zero Page
87 —
88 — DEY
89 —
8A — TXA
8B —
8C — STY — Absolute

8D — STA — Absolute
8E — STX — Absolute
8F —
90 — BCC
91 — STA — Indirect, Y
92 —
93 —
94 — STY — Zero Page, X
95 — STA — Zero Page, X
96 — STX — Zero Page, Y
97 —
98 — TYA
99 — STA — Absolute, Y
9A — TXS
9B —
9C —
9D — STA — Absolute, X
9E —
9F —
A0 — LDY — Immediate
A1 — LDA — Indirect, X
A2 — LDX — Immediate
A3 —
A4 — LDY — Zero Page
A5 — LDA — Zero Page
A6 — LDX — Zero Page
A7 —
A8 — TAY
A9 — LDA — Immediate
AA — TAX
AB —
AC — LDY — Absolute
AD — Absolute
AE — LDX — Absolute
AF —
B0 — BCS
B1 — LDA — Indirect, Y
B2 —
B3 —
B4 — LDY — Zero Page, X
B5 — LDA — Zero Page, X
B6 — LDX — Zero Page, Y
B7 —
B8 — CLV
B9 — LDA — Absolute, Y
BA — TSX
BB —
BC — LDY — Absolute, X
BD — LDA — Absolute, X
BE — LDX — Absolute, Y
BF —
C0 — CPY — Immediate
C1 — CMP — Indirect, X
C2 —
C3 —
C4 — CPY — Zero Page
C5 — CMP — Zero Page
C6 — DEC — Zero Page
C7 —
C8 — INY
C9 — CMP — Immediate
CA — DEX
CB —
CC — CPY — Absolute
CD — CMP — Absolute
CE — DEC — Absolute
CF —
D0 — BNE
D1 — CMP — Indirect, Y
D2 —
D3 —
D4 —
D5 — CMP — Zero Page, X
D6 — DEC — Zero Page, X
D7 —
D8 — CLD
D9 — CMP — Absolute, Y
DA —
DB —
DC —
DD — CMP — Absolute, X
DE — DEC — Absolute, X
DF —
E0 — CPX — Immediate
E1 — SBC — Indirect, X
E2 —
E3 —
E4 — CPX — Zero Page
E5 — SBC — Zero Page
E6 — INC — Zero Page
E7 —
E8 — INX
E9 — SBC — Immediate
EA — NOP
EB —
EC — CPX — Absolute
ED — SBC — Absolute
EE — INC — Absolute
EF —
F0 — BEQ
F1 — SBC — Indirect, Y
F2 —
F3 —
F4 —
F5 — SBC — Zero Page, X
F6 — INC — Zero Page, X
F7 —
F8 — SED
F9 — SBC — Absolute, Y
FA —
FB —
FC —
FD — SBC — Absolute, X
FE — INC — Absolute, X
FF —

Codes hexadécimaux des opérations

Tableaux

Cette annexe contient les copies des tableaux dont vous aurez souvent besoin, comme par exemple les codes ASCII et les adresses des commutateurs logiciels. Ces tableaux ont leur numéro original pour que vous puissiez vous reporter aux paragraphes qui y font référence dans le texte.

Tableau 2-2 Adresses de mémoire du clavier

Adresse Hex	Décimale	Description
$C000	49152 — 16384	Données du clavier et échantillonnage
$C010	49168 — 16368	Indicateur qu'une touche est enfoncée et commutateur de remise à zéro de l'échantillonneur du clavier

Tableau 2-3. Les touches et les codes ASCII, jeu de caractères français

Touche	Normal	Ctrl	Shift	Ctrl et SHIFT
Del	7F	7F	7F	7F
←	08	08	08	08
→|	09	09	09	09
↓	0A	0A	0A	0A
↑	0B	0B	0B	0B
↵	0D	0D	0D	0D
→	15	15	15	15
Esc	1B	1B	1B	1B
Espace	20	20	20	20
! 8	21	21	38	38
" 3	22	22	33	33
$ *	24	24	2A	2A
& 1	26	26	31	31
' 4	27	27	34	34
(5	28	28	35	35
) °	29	29	5B	1B
, ?	2C	2C	3F	3F
- _	2D	2D	5F	1F
: /	3A	3A	2F	2F
; .	3B	3B	2E	2E
< >	3C	3C	3E	3E
= +	3D	3D	2B	2B
à 0	40	00	30	00
ç 9	5C	1C	39	39
§ 6	5D	1D	36	1D
^ ¨	5E	1E	7E	1E
` £	60	60	23	23
a A	61	01	41	01
b B	62	02	42	02
c C	63	03	43	03
d D	64	04	44	04
e E	65	05	45	05
f F	66	06	46	06
g G	67	07	47	07
h H	68	08	48	08
i I	69	09	49	09
j J	6A	0A	4A	0A
k K	6B	0B	4B	0B
l L	6C	0C	4C	0C
m M	6D	0D	4D	0D
n N	6E	0E	4E	0E
o O	6F	0F	4F	0F
p P	70	10	50	10
q Q	71	11	51	11
r R	72	12	52	12

Touche	Normal	Ctrl	Shift	Crtl et SHIFT
s S	73	13	53	13
t T	74	14	54	14
u U	75	15	55	15
v V	76	16	56	16
w W	77	17	57	17
x X	78	18	58	18
y Y	79	19	59	19
z Z	7A	1A	5A	1A
è 2	7B	7B	32	32
ù %	7C	7C	25	25
è 7	7D	7D	37	37

Tableau 2-4. Les touches et les codes ASCII, jeu de caractères nord-américains

Key	Normal	Control	Shift	Both
DELETE	7F	7F	7F	7F
LEFT-ARROW	08	08	08	08
TAB	09	09	09	09
DOWN-ARROW	0A	0A	0A	0A
UP-ARROW	0B	0B	0B	0B
RETURN	0D	0D	0D	0D
RIGHT-ARROW	15	15	15	15
ESC	1B	1B	1B	1B
space	20	20	20	20
' "	27	27	22	22
, <	2C	2C	3C	3C
- _	2D	2D	5F	1F
. >	2E	2E	3E	3E
/ ?	2F	2F	3F	3F
0)	30	30	29	29
1 !	31	31	21	21
2 @	32	00	40	00
3 #	33	33	23	23
4 $	34	34	24	24
5 %	35	35	25	25
6 ^	36	1E	5E	1E
7 &	37	37	26	26
8 *	38	38	2A	2A
9 (	39	39	28	28
; :	3B	3B	3A	3A
= +	3D	3D	2B	2B

Tableau 2-4. Les touches et les codes ASCII, jeu de caractères nord-américains

Key	Normal	Control	Shift	Both
[{	5B	1B	7B	1B
|	5C	1C	7C	1C
]}	5D	1D	7D	1D
`~	60	60	7E	7E
A	61	01	41	01
B	62	02	42	02
C	63	03	43	03
D	64	04	44	04
E	65	05	45	05
F	66	06	46	06
G	67	07	47	07
H	68	08	48	08
I	69	09	49	09
J	6A	0A	4A	0A
K	6B	0B	4B	0B
L	6C	0C	4C	0C
M	6D	0D	4D	0D
N	6E	0E	4E	0E
O	6F	0F	4F	0F
P	70	10	50	10
Q	71	11	51	11
R	72	12	52	12
S	73	13	53	13
T	74	14	54	14
U	75	15	55	15
V	76	16	56	16
W	77	17	57	17
X	78	18	58	18
Y	79	19	59	19
Z	7A	1A	5A	1A

Tableau 2-5 L'ensemble des caractères ASCII

Dec	Hex	Char	Dec	Hex	Char	Dec	Hex	Char	Dec	Hex	Char
0	00	NUL	32	20	SP	64	40	@	96	60	`
1	01	SOH	33	21	!	65	41	A	97	61	a
2	02	STX	34	22	"	66	42	B	98	62	b
3	03	ETX	35	23	#	67	43	C	99	63	c
4	04	EOT	36	24	$	68	44	D	100	64	d
5	05	ENQ	37	25	%	69	45	E	101	65	e
6	06	ACK	38	26	&	70	46	F	102	66	f
7	07	BEL	39	27	'	71	47	G	103	67	g
8	08	BS	40	28	(	72	48	H	104	68	h
9	09	HT	41	29	)	73	49	I	105	69	i
10	0A	LF	42	2A	*	74	4A	J	106	6A	j
11	0B	VT	43	2B	+	75	4B	K	107	6B	k
12	0C	FF	44	2C	,	76	4C	L	108	6C	l
13	0D	OR	45	2D	-	77	4D	M	109	6D	m
14	0E	SO	46	2E	.	78	4E	N	110	6E	n
15	0F	SI	47	2F	/	79	4F	O	111	6F	o
16	10	DLE	48	30	0	80	50	P	112	70	p
17	11	DC1	49	31	1	81	51	Q	113	71	q
18	12	DC2	50	32	2	82	52	R	114	72	r
19	13	DC3	51	33	3	83	53	S	115	73	s
20	14	DC4	52	34	4	84	54	T	116	74	t
21	15	NAK	53	35	5	85	55	U	117	75	u
22	16	SYN	54	36	6	86	56	V	118	76	v
23	17	ETB	55	37	7	87	57	W	119	77	w
24	18	CAN	56	38	8	88	58	X	120	78	x
25	19	EM	57	39	9	89	59	Y	121	79	y
26	1A	SUB	58	3A	:	90	5A	Z	122	7A	z
27	1B	ESC	59	3B	;	91	5B	[	123	7B	{
28	1C	FS	60	3C	<	92	5C	\	124	7C	|
29	1D	GS	61	3D	=	93	5D	]	125	7D	}
30	1E	RS	62	3E	>	94	5E	^	126	7E	~
31	1F	US	63	3F	?	95	5F	_	127	7F	DEL

Tableau 2-7. Les ensembles de caractères affichables

Pour identifier des caractères particuliers et des valeurs, se reporter au tableau 2-5

Inv. : Inverse, Cli. : Clignotant
Nor. : Normal

Valeurs Hexa.	Ensemble Primaire : Type de caractères	Mode	Ensemble Alternatif : Type de caractères	Mode
$00-$1F	Lettres majuscules	Inv.	Lettres minuscules	Inv.
$20-$3F	Caractères spéciaux	Inv.	Caractères spéciaux	Inv.
$40-$5F	Lettres majuscules	Cli.	Lettres majuscules	Inv.
$60-$7F	Caractères spéciaux	Cli.	Lettres minuscules	Inv.
$80-$9F	Lettres majuscules	Nor.	Lettres majuscules	Nor.
$A0-$BF	Caractères spéciaux	Nor.	Caractères spciaux	Nor.
$C0-$DF	Lettres majuscules	Nor.	Lettres majuscules	Nor.
$E0-$FF	Lettres minuscules	Nor.	Lettres minuscules	Nor.

Tableau 2-8. Couleurs des graphiques en basse-résolution

Les couleurs peuvent varier, suivant les contrôles du moniteur ou du téléviseur

Valeur du demi-octet Dec	Hex	Couleur	Valeur du demi-octet Dec	Hec	Couleur
0	$0	Noir	8	$8	Marron
1	$1	Magenta	8	$9	Orange
1	$2	Bleu foncé	10	$A	Gris 2
3	$3	Violet	11	$B	Rose
4	$4	Vert foncé	12	$C	Vert
5	$5	Gris 1	13	$D	Jaune
6	$6	Bleu	14	$E	Bleu turquoise
7	$7	Bleu clair	15	$F	Blanc

Tableau 2-9. Couleurs des graphiques haute-résolution

Les couleurs peuvent varier avec le réglage du moniteur ou du téléviseur

Bits 0-6	Bit 7 à zéro	Bit 7 à un
Colonnes adjacentes à zéro	Noir 1	Noir 2
Colonnes paires à un	Violet	Bleu
Colonnes impaires à un	Vert	Orange
Colonnes adjacentes à un	Blanc 1	Blanc 2

Tableau 2-10. Adresses des pages d'affichage vidéo

Mode d'affichage	Page	Adresse la plus basse		Adresse la plus haute	
Texte en 40 colonnes, Graphiques en basse-résolution	1	$400	1024	$7FF	2047
	2	$800	2048	$BFF	3071
Texte en 80 colonnes	1*	$400	1024	$7FF	2047
Graphiques en haute-résolution	1	$2000	8192	$3FFF	16383
	2	$4000	16384	$5FFF	24575

* Note : Le mode 80 colonnes utilise les adresses des 1 024 octets de la Page Texte 1 dans les deux mémoires principale et auxiliaire. Le commutateur PAGE2 est utilisé pour sélectionner l'une ou l'autre pour enregistrer des données (voir le paragraphe « Commutations des modes d'affichage »).

Tableau 2-11. Commutateurs logiciels d'affichage

(1) Ce mode n'est opérationnel que si le commutateur de mode graphique est à UN.
(2) Ce commutateur a une fonction différente si la page auxiliaire de texte d'une carte de texte en 80 colonnes a été autorisée en écriture. Se reporter au prochain paragraphe « Adressage direct des pages d'affichage ».
(3) Ce commutateur change la fonction du commutateur PAGE2 pour permettre l'adressage de la mémoire auxiliaire de texte sur une carte de texte en 80 colonnes. Le prochain paragraphe décrit comment le faire.
(4) En lisant à cette adresse, on obtient l'état du signal VBL de suppression verticale. La fonction de VBL est décrite au chapitre 7 dans le paragraphe « Signaux de sortie vidéo ».

Nom	Fonction	Adresse Hex	Adresse Décimale		Notes
ALTCHARSET	Ens. Alternatif à un	$C00F	49167	−16369	Ecrire
	Ens. Alternatif à zéro	$C00E	49166	−16370	Ecrire
	Lire le comm. ALTCHARSET	$C010	49182	−16354	Lire
TEXT	Mode Texte à un	$C051	49233	−16303	
	Mode Texte à zéro (GR)	$C050	49232	−16304	
	Lire le comm. TEXT	$C01A	49178	−16358	Lire
MIXED	Mode Mixte à un	$C053	49235	−16301	1
	Mode Mixte à zéro	$C052	49234	−16302	1
	Lire le comm. MIXED	$C01B	49179	−16357	Lire
PAGE2	Page 2 à un	$C055	49237	−16299	2
	Page 2 à zéro (Page 1)	$C054	49236	−16300	2
	Lire le comm. PAGE2	$C01C	49180	−16356	Lire
HIRES	Mode haute-res à un	$C057	49239	−16297	1
	Mode haute-res à zéro	$C056	49238	−16298	1
	Lire le comm. HIRES	$C01D	49181	−16355	Lire
80COL	Affi. 80 colonnes à un	$C00D	49165	−16371	Ecrire
	Aff. 80 colonnes à zéro	$C00C	49164	−16372	Ecrire
	Lire le comm. 80COL	$C01F	49183	−16353	Lire
80STORE	Enrgt en mémoire aux.	$C001	49153	−16383	Ecrire, 3
	Enrgt en mém. centrale	$C000	49152	−16384	Ecrire, 3
	Lire le comm. 80STORE	$C018	49176	−16360	Lire
VBL	Lire la suppression verticale	$C019	49177	−16359	Lire, 4

Table 2-12. Adresses des mémoires des annonciateurs

*les numéros de broches sont ceux du connecteur de CI à 16 broches sur la carte mère

Annonciateur			Adresse		
N°	Broche*	Etat	Décimale		Hex
0	15	zéro	49240	16296	$C058
		un	49241	16295	$C059
1	14	zéro	49242	16294	$C05A
		un	49243	16293	$C05B
2	13	zéro	49244	16292	$C05C
		un	49245	16291	$C05D
3	12	zéro	49246	16290	$C05E
		un	49247	16289	$C05F

Tableau 2-13. Adresses des mémoires d'E/S secondaires

Pour l'identification du connecteur et des numéros de broches, se reporter aux Tableaux 7-17 et 7-18

Fonction	Adresse Décimale		Hex	Notes
Haut-parleur	49200	−16336	$C030	Lire
Sortie de cassette	49184	−16352	$C020	Lire
Entrée de cassette	49248	−16288	$C060	Lire
Annonciateur 0 à Un	49241	−16295	$C059	
Annonciateur 0 à Zéro	49240	−16296	$C058	
Annonciateur 1 à Un	49243	−16293	$C05B	
Annonciateur 1 à Zéro	49242	−16294	$C05A	
Annonciateur 2 à Un	49245	−16291	$C05D	
Annonciateur 2 à Zéro	49244	−16292	$C05C	
Annonciateur 3 à Un	49247	−16289	$C05F	
Annonciateur 3 à Zéro	49246	−16290	$C05E	
Sortie d'échantillonnage	49216	−16320	$C040	Lire
Commutateur d'entrée 0 (touche ⓐ)	49249	−16287	$C061	Lire
Commutateur d'entrée 1 (touche)	49250	−16286	$C062	Lire
Commutateur d'entrée 2	49251	−16285	$C063	Lire
Remise à zéro sur les entrées analogiques	49264	−16272	$C070	
Entrée analogique 0	49252	−16284	$C064	Lire
Entrée analogique 1	49253	−16283	$C065	Lire
Entrée analogique 2	49254	−16282	$C066	Lire
Entrée analogique 3	49255	−16281	$C067	Lire

Tableau 3-3b. Les caractères de contrôle avec COUT1, suite.

(1) N'est disponible que si les sous-programmes pour 80 colonnes sont en fonction.
(2) gotoXY n'est pas implémenté en BASIC : voir le *Manuel de référence du système d'exploitation en Pascal sur Apple IIe*.

Caractère de contrôle	Nom ASCII	Nom Apple *IIe*	Action prise par COUT1	Notes
Ctrl-S	(DS3)	stop-list	Arrête d'envoyer des caractères sur l'écran, jusqu'à ce qu'une touche soit enfoncées.	
Ctrl-U	(NAK)	quit	Met les sous-programmes pour 80 colonnes hors-fonction, remonte le curseur en haut et à gauche et efface l'écran.	1
Ctrl-V	(SYN)	scroll	Fait descendre l'affichage d'une ligne, en laissant le curseur dans sa position courante.	1
Ctrl-W	(ETB)	scroll-up	Fait défiler l'affichage d'une ligne vers le haut en laissant le curseur dans sa position courante.	1
Ctrl-Y	(EM)	home	Met le curseur dans le coin en haut et à gauche de la fenêtre (mais n'efface pas).	1
Ctrl-Z	(SUB)	clear line	Efface la ligne sur laquelle se trouve le curseur.	1
Ctrl-\	(FS)	fwd. space	Déplace le curseur d'une position vers la droite ; du bord droit de la fenêtre, le déplace à l'extrême gauche de la ligne du dessous.	1
Ctrl-]	(GS)	clear EOL	Efface la ligne depuis la position du curseur jusqu'au bord droit de la fenêtre	1
Ctrl-^	(RS)	gotoXY	En utilisant les deux caractères moins 32, comme valeurs de X et Y sur un octet, déplace le curseur en CH = X, CV = Y.	1,2

Tableau 3-4. Les adresses de mémoire de la fenêtre de texte.

Paramètre de la fenêtre	Adresse		Valeurs minimales		Valeurs normales : 40 col.		Valeurs normales : 80 col.		Valeurs maximales : 40 col.		Valeurs maximales : 80 col.	
	Dec	Hex	Dec	Hex	Dec	Hex	Dec	Hex	Dec	Hex	Dec	Hex
Bord gauche	32	$20	0	$0	0	$0	0	$0	39	$27	79	$4F
Largeur	33	$21	0	$0	40	$28	80	$50	40	$28	80	$50
Haut	34	$22	0	$0	0	$0	0	$0	23	$17	23	$17
Bas	35	$23	1	$1	24	$18	24	$18	24	$18	24	$18

Tableau 3-6. Les codes d'évasion

(1) Touche de contrôle du curseur dans l'ancien style : voir le texte
(2) Touche de contrôle du curseur : voir le texte
(3) Ce code ne fonctionne que si les sous-programmes pour 80 colonnes sont en fonction.

Code d'évasion	Fonction	Notes
Esc @	Efface la fenêtre et remonte le curseur dans le coin en haut à gauche	
Esc A	Remonte le curseur d'une ligne	1
Esc B	Déplace le curseur d'une position à droite	1
Esc C	Déplace le curseur d'une position à gauche	1
Esc D	Descend le curseur d'une ligne	1
Esc E	Efface la fin de la ligne	
Esc F	Efface le bas de la fenêtre	
Esc I Esc ↑	Remonte le curseur d'une ligne et met en mode d'évasion	2
Esc J Esc ←	Déplace le curseur d'une position à gauche et met en moded'évasion	2
Esc K Esc →	Déplace le curseur d'une position à droite et met en moded'évasion	2
Esc M Esc ↓	Descend le curseur d'une ligne et met en mode d'évasion	2
Esc R	Entre dans le mode majuscule restreint	3
Esc T	Sort du mode majuscule restreint	3
Esc 4	Commute en mode à 40 colonnes, met le curseur dans le coin en haut à gauche,et efface l'écran	3
Esc 8	Commute en mode à 80 colonnes, met le curseur dans le coin en haut à gauche,et efface l'écran	3
Esc Ctrl – Q	Met les sous-programmes pour 80 colonnes hors-fonction	3

Tableau 4-5. Commutateurs de sélection de banc

(1) Ce commutateur autorise l'écriture en MEV et la lecture en MEM.
(2) Deux lectures successives à ce commutateur autorise l'écriture et la lecture en MEV.

Adresse de commutateur	Écriture en MEV	Lecture en MEV	Lecture en MEM	Banc de 4K de MEV : Premier	Second	Notes
$C080		•			•	
$C081	•		•		•	1
$C082			•		•	
$C083	•	•			•	2
$C084		•			•	
$C085	•		•		•	1
$C086			•		•	
$C087	•	•			•	2
$C088		•		•		
$C089	•		•	•		1
$C08A			•	•		
$C08B	•	•		•		2
$C08C		•		•		
$C08D	•		•	•		1
$C08E			•	•		
$C08F	•	•		•		2

Tableau 4-6. Commutateurs de sélection de la mémoire auxiliaire

(1) Si 80STORE est à un, le commutateur PAGE2 sélectionne la mémoire d'affichage, soit auxiliaire soit principale.
(2) Si 80STORE est à un, le commutateur HIRES vous autorise à utiliser le commutateur PAGE2 pour commuter la zone Page 1 en haute résolution entre la mémoire principale et la mémoire auxiliaire.

Nom	Fonction	Adresse Hex	Adresse Décimale	Notes
RAMRD	Lire en mémoire auxiliaire	$C003	49155 —16381	Écrire
	Lire en mémoire principale	$C002	49154 —16382	Écrire
	Lire l'état de RAMRD	$C013	49171 —16365	Lire
RAMWRT	Écrire en mémoire auxiliaire	$C005	49157 —16379	Écrire
	Écrire en mémoire principale	$C004	49156 —16380	Écrire
	Lire l'état de RAMWRT	$C014	49172 —16354	Lire
80STORE	A un : accès page d'affich.	$C001	49153 —16383	Écrire
	A zéro : utilisez RAMRD/WRT	$C000	49152 —16384	Écrire
	Lire l'état de 80STORE	$C018	49176 —16360	Lire
PAGE2	Page 2 à un (mémoire aux.)	$C055	49237 —16299	1
	Page 2 à zéro (mémoire princ.)	$C054	49236 —16300	1
	Lire l'état de PAGE2	$C01C	49180 —16356	Lire
HIRES	A un : accès pages hte-res.	$C057	49239 —16297	2
	A zéro : utilisez RAMRD/WRT	$C056	49238 —16298	2
	Lire l'état de HIRES	$C01D	49181 —16355	Lire
ALTZP	Pile et page zéro aux.	$C009	49161 —16373	Écrire
	Pile et page zéro princ.	$C008	49160 —16374	Écrire
	Lire l'état de ALTZP	$C016	49174 —16352	Lire

Tableau 4-10. Vecteurs de la page 3

Adresse du vecteur		Fonction du vecteur
Dec	**Hex**	
1008 1009	$3F0 $3F1	Adresse du sous-programme qui gère une demande de BRK (arrêt en langage assembleur) (normalement $59, $FA)
1010 1011	$3F2 $3F3	Vecteur de réinitialisation (Reset) (voir le texte)
1012	$3F4	Octet de mise-sous-tension (« power-up ») (voir le texte)
1013 1014 1015	$3F5 $3F6 $3F7	Instruction de saut au sous-programme qui prend en charge la commande « & » de l'Applesoft (normalement $4C, $58, $FF).
1016 1017 1018	$3F8 $3F9 $3FA	Instruction de saut au sous-programme qui prend en charge la commande (Ctrl - Y) donnée par l'utilisateur.
1019 1020 1021	$3FB $3FC $3FD	Instruction de saut au sous-programme qui gère les interruptions non masquables
1022 1023	$3FE $3FF	Vecteur d'interruption (adresse du sous-programme qui exploite les demandes d'interruption) programme qui exploite les demandes d'interruption)

Tableau 6-1. Adresses de mémoire d'E/S pour carte périphérique

Note : le signal de validation est marqué d'un prime (′), pour indiquer que c'est à son niveau bas que le signal est actif.

Connecteur	Adresses	Validées par
1	$C090-$C09F	DEVICE SELECT′
2	$C0A0-$C0AF	DEVICE SELECT′
3	$C0B0-$C0BF	DEVICE SELECT′
4	$C0C0-$C0CF	DEVICE SELECT′
5	$C0D0-$C0DF	DEVICE SELECT′
6	$C0E0-$C0EF	DEVICE SELECT′
7	$C0F0-$C0FF	DEVICE SELECT′

Tableau 6-2. Adresses de mémoire de MEM pour carte périphérique

Note : Le signal de validation est marqué avec un prime (′), pour indiquer que c'est à son niveau bas que le signal est actif.

Connecteur	Adresses	Validée par
1	$C100-$C1FF	I/O SELECT′
2	$C200-$C2FF	I/O SELECT′
3	$C300-$C3FF	I/O SELECT′
4	$C400-$C4FF	I/O SELECT′
5	$C500-$C5FF	I/O SELECT′
6	$C600-$C6FF	I/O SELECT′
7	$C700-$C7FF	I/O SELECT′

Tableau 6-3. Adresses des mémoires MEV pour carte périphérique

*Note : Les adresses de MEV normalement allouées au connecteur 3 sont reprises par toute carte installée dans le connecteur auxiliaire.

Adresse de base	Numéro de Connecteur						
	1	2	3	4	5	6	7
$0478	$0479	$047A	$047B*	$047C	$047D	$047E	$047F
$0478	$04F9	$04FA	$04FB*	$04FC	$04FD	$04FE	$04FF
$0578	$0579	$057A	$057B*	$057C	$057D	$045E	$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

Tableau 6-4. Adresses de base des E/S pour carte périphérique

Adresse de base	Numéro de Connecteur 1	2	3	4	5	6	7
$C080	$C090	$C0A0	$C0B0	$C0C0	$C0D0	$C0E0	$C0F0
$C081	$C091	$C0A1	$C0B1	$C0C1	$C0D1	$C0E1	$C0F1
$C082	$C092	$C0A2	$C0B2	$C0C2	$C0D2	$C0E2	$C0F2
$C083	$C093	$C0A3	$C0B3	$C0C3	$C0D3	$C0E3	$C0F3
$C084	$C094	$C0A4	$C0B4	$C0C4	$C0D4	$C0E4	$C0F4
$C085	$C095	$C0A5	$C0B5	$C0C5	$C0D5	$C0E5	$C0F5
$C086	$C096	$C0A6	$C0B6	$C0C6	$C0D6	$C0E6	$C0F6
$C087	$C097	$C0A7	$C0B7	$C0C7	$C0D7	$C0E7	$C0F7
$C088	$C098	$C0A8	$C0B8	$C0C8	$C0D8	$C0E8	$C0F8
$C08A	$C09A	$C0AA	$C0BA	$C0CA	$C0DA	$C0EA	$C0FA
$C08B	$C09B	$C0AB	$C0BB	$C0CB	$C0DB	$C0EB	$C0FB
$C08C	$C09C	$C0AC	$C0BC	$C0CC	$C0DC	$C0EC	$C0FC
$C08D	$C09D	$C0AD	$C0BD	$C0CD	$C0DD	$C0ED	$C0FD
$C08E	$C09E	$C0AE	$C0BE	$C0CE	$C0DE	$C0EE	$C0FE
$C08F	$C09F	$C0AF	$C0BF	$C0CF	$C0DF	$C0EF	$C0FF

Tableau 6-5. Commutateurs des mémoires d'E/S

Nom	Fonction	Adresse Hex	Décimal		Notes
SLOTC3ROM	MEM périph. en $C300	$C00B	49163	−16373	Écrire
	MEM interne en $C300	$C00A	49162	−16374	Écrire
	Lire le comm. SLOTC3ROM	$C017	49175	−16361	Lire
SLOTCXROM	MEM périph. en $CxOO	$C007	49159	−16377	Écrire
	MEM interne en $CxOO	$C006	49158	−16378	Écrire
	Lire le comm. SLOTCXROM	$C015	49173	−16363	Lire

Répertoire des sous-programmes de base

Voici une liste de sous-programmes utiles faisant partie du Moniteur de l'Apple //e. Pour se servir de ces programmes à partir de programmes en langage-machine, il faut enregistrer les données dans les adresses de mémoire indiquées ou dans les registres du 6502 suivant les conditions requises par le sous-programme avant d'exécuter un JSR à l'adresse de début du sous-programme. Après que le sous-programme est réalisé sa fonction, le retour s'effectue avec des changements sur les registres du 6502 comme indiqué ci-dessous.

Avertissement
Pour assurer la comptabilité entre l'Apple II Plus et l'Apple IIe, ne sautez pas au milieu d'un sous-programme du Moniteur. Les adresses de début ou points sont les mêmes pour tous les modèles Apple II, mais le code réel est différent.

BELL sortir un caractère de signal sonore $FF3A

BELL écrit un caractère de signal sonore (Ctrl - G) dans le dispositif de sortie courant. Il laisse $87 dans l'accumulateur.

BELL 1 Envoyer une tonalité sonore vers le haut-parleur $FBDD

BELL 1 génère une tonalité de 1 kHz par le haut-parleur de l'Apple //e pendant 0,1 seconde. Il détruit le contenu des registres A et X.

CLREOL Effacer jusqu'à la fin de la ligne $FC9C

CLREOL efface une ligne de texte depuis la position du curseur jusqu'au bord droit de la fenêtre. CLREOL détruit le contenu des registres A et Y.

CLEOLZ Effacer jusqu'à la fin de la ligne $FC9E

CLEOLZ efface une ligne de texte jusqu'au bord droit de la fenêtre en commençant à la position donnée par l'adresse de base BASL indéxée par le contenu du registre Y. CLEOLZ détruit le contenu des registres A et Y.

CLREOP Effacer jusqu'à la fin de la fenêtre $FC42

CLREOP efface la fenêtre de texte depuis la position du curseur jusqu'en bas de la fenêtre. CLREOP détruit le contenu des registres.

CLRSCR Effacer l'écran en basse-résolution $F832

CLRSCR efface l'affichage graphique en basse-résolution avec du noir. Si vous appeler CLRSCR, alors que l'affichage est en mode texte, il remplit l'écran de caractères à ou @ en mode inverse. CLRSCR détruit le contenu des registres A et Y.

CLRTOP Effacer l'écran en basse-résolution $F836

CLRTOP est pareil à CLRSCR (ci-dessus), sauf qu'il n'efface que les 40 lignes les plus hautes de l'affichage en basse-résolution.

COUT Sortir un caractère $FDED

COUT appelle le sous-programme courant de sortie de caractère. Le caractère à sortir devra se trouver dans l'accumulateur. COUT appelle le sous-programme dont l'adresse est mémorisée en CSW (adresses $36 et $37), qui est la sortie habituelle et standard de caractère COUT1.

COUT1 Sortir sur écran $FDF0

COUT1 affiche le caractère présent dans l'accumulateur sur l'écran de l'Apple //e à la position courante du curseur et avance le curseur, lequel étant lié à une opération de sortie. Il place le caractère en se servant de l'état de l'adresse du mode Normal/Inverse. Il prend en compte les codes des caractères Return, linefeed, backspace, et bell. Il laisse tous les registres intacts à son retour.

CROUT Produire un Return $FD8E

CROUT envoie un caractère Return au dispositif courant de sortie.

CROUT1 Return avec effacement d'écran $FD8B

CROUT1 efface l'écran depuis la position courante du curseur jusqu'au bas de la fenêtre d'écran, puis appelle CROUT.

GETLN Entrer une ligne de commande avec le solliciteur $FD6A

GETLN est le sous-programme standard d'entrée pour des lignes entières de caractères, comme cela est décrit au Chapitre 3. Votre programme appellera GETLN après avoir placé le solliciteur à l'adresse $33 ; GETLN met la ligne de commande dans la zone-tampon d'entrée (commençant à l'adresse $200) et le registre X contient la longueur de la ligne d'entrée au retour du sous-programme.

GETLNZ Entrer une ligne de commande $FD67

GETLNZ est un autre point d'entrée pour GETLN qui enverra un caractère Return à la sortie standard, puis continuera dans GETLN.

GETLN1 Entrer une ligne de commande sans solliciteur $FD6F

GETLN1 est un autre point d'entrée pour GETLN qui n'affichera pas de solliciteur avant d'accepter la ligne de commande. Si, pourtant, l'utilisateur annule la ligne de commande, soit à cause de trop nombreux backspaces, soit avec - X, alors GETLN1 utilisera le contenu de l'adresse $33 comme solliciteur quand il entrera une nouvelle ligne.

HLINE Tracer une ligne horizontale de blocs $F819

HLINE trace une ligne horizontale de blocs sur l'affichage graphique basse-résolution de la couleur fixée par SETCOL. Appelez HLINE avec la coordonnée verticale de la ligne dans l'accumulateur, la coordonnée horizontale la plus à gauche dans le registre Y et la coordonnée horizontale la plus à droite dans l'adresse $2C. HLINE détruit A et Y et laisse X intact à son retour.

HOME Mettre le curseur en haut à gauche et effacer $FC58

HOME efface l'affichage et met le curseur dans le coin supérieur gauche de l'écran.

IOREST Restaurer les registres $FF3F

IOREST charge les registres internes du 6502 avec les contenus des adresses de mémoire $45 à $49.

IOSAVE Sauvegarder les registres $FF4A

IOSAVE mémorise les contenus des registres internes du 6502 dans les adresses $45 à $49 dans l'ordre suivant A, X, Y, P, S. Les contenus de A et X sont modifiés et le mode décimal est mis à zéro.

KEYIN Lire le clavier $FD1B

KEYIN est le sous-programme d'entrée par le clavier. Il lit le clavier de l'Apple //e, attend qu'une touche soit enfoncée, et rend aléatoire le nombre mémorisé en $4E - $4F, servant d'amorce au générateur de nombres pseudo-aléatoires. Quand une touche est enfoncée, KEYIN supprime le curseur clignotant de l'écran et renvoie le code de la touche dans l'accumulateur. KEYIN est décrit au chapitre 3.

MOVE Déplacer un bloc de mémoires $FE2C

MOVE recopie les contenus de mémoire d'une zone d'adresses dans une autre. Ce sous-programme est le même que la commande MOVE dans le moniteur, sauf qu'il prend ses arguments dans une paire d'adresses en mémoire, l'octet de poids faible en premier. L'adresse de destination doit se trouver dans A4 ($42 - $43), l'adresse de début de la zone originale dans A1 ($3C - $3D), et l'adresse de fin de la zone originale dans A2 ($3E - $3F) quand votre programme appelera MOVE.

NEXTCOL Augmenter la couleur de 3 $F85F

NEXTCOL ajoute 3 à la couleur courante (fixée par SETCOL) utilisée en graphique basse-résolution.

PLOT Tracer un bloc sur l'écran en basse-résolution $F800

PLOT met un seul bloc de la couleur fixée par SETCOL sur l'écran d'affichage en basse-résolution. La position verticale du bloc est transmise par l'accumulateur, sa position verticale est dans le registre Y. PLOT détruit l'accumulateur et laisse intacts X et Y en revenant.

PRBLNK Imprimer trois espaces $F948

PRBLNK sort trois espaces vers le dispositif standard de sortie. Au retour, l'accumulateur contient habituellement $A0, le registre X contient 0.

PRBL2 Imprimer plusieurs espaces $F94A

PRBL2 sort de 1 à 256 espaces vers le dispositif standard de sortie. En entrant, le registre X devra contenir le nombre d'espaces à sortir. Si X = $00, alors PRBL2 sortira 256 espaces.

PRBYTE Imprimer un octet hexadécimal $FDDA

PRBYTE sort le contenu de l'accumulateur en hexadécimal vers le dispositif courant de sortie. Le contenu de l'accumulateur est détruit.

PREAD Lire une manette de jeu $FB1E

PREAD renvoie un nombre qui représente la position de la manette. Vous transmettez le numéro de la manette par le registre X. Si ce nombre n'est pas valable (différent de 0, 1, 2, ou 3), d'étranges choses peuvent se produire. PREAD revient avec un nombre dans le registre Y, nombre compris entre $00 et $FF. L'accumulateur est détruit.

PRERR Imprime ERR $FF2D

PRERR envoie le mot ERR, suivi d'un caractère bell, au dispositif de sortie standard. Au retour, l'accumulateur est détruit.

PRHEX imprime un chiffre hexadécimal $FDE3

PRHEX imprime le demi-octet de plus faible poids de l'accumulateur, l'accumulateur est détruit.

PRNTAX Imprime A et X en hexadécimal $F941

PRNTAX imprime les contenus des registres A et X sous forme de deux chiffres hexadécimaux. L'accumulateur contient le premier octet à sortir, le registre X le second. Au retour, le contenu de l'accumulateur est détruit.

RDCHAR Entrer un seul caractère ou le code Esc $FD35

RDCHAR est un autre sous-programme d'entrée de caractère. Il place un curseur clignotant sur l'écran à la position du curseur et saute au sous-programme dont l'adresse est enregistrée en KSW (adresses $38 et $39), habituellement le sous-programme d'entrée standard KEYIN, qui revient avec un caractère dans l'accumulateur.

RDKEY Entrer un seul caractère $FD0C

RDKEY est le sous-programme d'entrée de caractère. Il place un curseur clignotant à la position courante et saute à la routine dont l'adresse est dans KSW ($38 et $39), le plus souvent c'est la routine standard d'entrée KEYIN qui retourne avec le caractère dans l'accumulateur.

READ Lire un enregistrement venant d'une cassette $FEFD

READ lit une série de tonalités au port d'entrée de la cassette, les convertit en octets de données, et enregistre les données dans une zone spécifiée d'adresses de mémoire. Avant d'appeler READ, l'adresse du premier octet doit se trouver dans A1 ($3C—$3D) et l'adresse du dernier octet dans A2 ($3E—$3F).

READ fait tourner en OR exclusif sur les octets de données dans CHKSUM ($2E). Dès que la dernière adresse est remplie, READ lit un octet de plus et le compare avec CHKSUM. S'ils sont égaux, READ envoie une tonalité sonore et revient ; sinon, il envoie « ERR » par COUT, envoie la tonalité sonore et revient.

SCRN Lire l'écran graphique basse-résolution $F871

SCRN renvoie la valeur de la couleur d'un seul bloc de l'affichage graphique en basse-résolution. Appelez-le, avec dans l'accumulateur, la position verticale du bloc et, dans le registre Y, la position horizontale. Appelez-le de la même manière que PLOT (ci-dessus). La couleur du bloc sera renvoyée dans l'accumulateur. Aucun autre registre n'est changé.

SETCOL Donner une couleur aux graphiques en basse-résolution $F864

SETCOL donne la valeur transmise par l'accumulateur à la couleur utilisée pour le tracé de graphiques en basse-résolution. Les couleurs et leur valeur sont énumérées dans le tableau 2-8.

SETINV Mettre en mode inverse $FE80

SETINV met en mode inverse l'affichage. COUT1 affichera alors tous les caractères de sortie en points noirs sur un fond blanc. Le registre Y contient $3F, tous les autres sont inchangés.

SETNORM Mettre en mode normal $FE84

SETNORM met l'affichage en mode normal. COUT1 affichera alors tous les caractères de sortie en points blancs sur un fond noir. Au retour, Y contient $FF et les autres sont inchangés.

VERIFY Comparer deux blocs de mémoire $FE36

VERIFY compare les contenus d'une zone de mémoire à ceux d'une autre. Ce sous-programme est le même que la commande VERIFY dans le moniteur, sauf qu'il prend ses arguments d'une paire d'adresses de mémoire, l'octet de plus faible poids en premier. L'adresse de destination doit se trouver dans A4 ($42-$43), l'adresse de début de la zone source dans A1 ($3C-$3D), et l'adresse de fin de la source dans A2 ($3E-$3F) quand votre programme appelle VERIFY.

VLINE Tracer une ligne verticale de blocs $F828

VLINE trace une ligne verticale de blocs de la couleur fixée par SETCOL dans l'affichage basse-résolution. Vous devrez appeler VLINE en ayant mis la position horizontale de la ligne dans le registre Y, la coordonnée verticale la plus haute dans l'accumulteur, et la coordonnée verticale la plus basse dans l'adresse $2D. VLINE reviendra avec l'accumulateur détruit.

WAIT Temporisation $FCA8

La routine WAIT temporise pendant un certain laps de temps et renvoie ensuite au programme d'appel. Le délai d'attente est donné par la valeur de l'accumulateur. Avec A comme contenu de l'accumulateur, le délai de la temporisation est 1/2(26+27A+5A 2) microsecondes. Wait place 0 dans l'accumulateur et ne modifie pas les registres X et Y.

WRITE Écrire sur cassette $FECD

WRITE convertit les données dans une zone-mémoire. Avant d'appeler WRITE, l'adresse de la première donnée doit se trouver dans A1 ($3C-$3D) et l'adresse du dernier octet dans A2 ($3E-$3F). Le sous-progrmme écrit une tonalité continue de dix secondes comme en-tête, puis écrit les données suivies d'un octet de somme de contrôle.

Différences entre l'Apple IIe et l'Apple II Plus

L'Apple //e est le dernier modèle Apple II, et il contient de nombreuses améliorations par rapport aux anciens modèles. Une liste des améliorations et autres différences est donnée ici approximativement dans l'ordre dans lequel vous allez les rencontrer : en premier les différences évidentes, après les détails techniques. Chaque terme dans la liste contient des références aux chapitres de ce manuel où le terme est expliqué.

Un clavier complet

L'Apple //e a un clavier complet de 62 touches majuscules et minuscules. Le clavier contient deux touches entièrement opérationnelles ⇧ et ⇩ (blocage de majuscules). Il comprend aussi quatre touches directionnelles pour déplacer le curseur. Le Chapitre 2 contient une description du clavier. Les touches de déplacement du curseur sont décrites au Chapitre 3.

Des touches POMME

Le clavier de l'Apple //e a deux touches marquées du logo Apple. Ces touches, appelées ◌ et ●, s'utilisent avec la touche **Reset** pour sélectionner des fonctions spéciales d'initialisation. Elles sont connectées aux bouton-poussoirs des manettes de jeu, donc elles peuvent être utilisées pour des fonctions spéciales dans des programmes.

Un affichage en minuscule

L'Apple //e peut afficher l'ensemble complet des caractères ASCII, majuscules et minuscules. Pour rester compatible avec les anciens Apple II, l'ensemble standard de caractères affichables contient les majuscules clignotantes à la place des minuscules inversées ; vous pouvez aussi changer d'ensemble de caractères affichables : l'ensemble alternatif contient les minuscules et les majuscules inversées mais n'a pas de mode clignotant. Le Chapitre 2 comprend une description des ensembles de caractères affichables. Le Chapitre 3 vous dit comment changer de mode d'affichage.

Un format d'affichage optionnel en 80 colonnes

En ajoutant une carte de texte en 80 colonnes, l'Apple //e peut afficher 80 colonnes par ligne de texte. L'affichage en 80 colonnes est entièrement compatible avec les deux modes graphiques — vous pouvez même utiliser le mode mixte. (Si vous préférez, vous pouvez mettre à la place une carte 80 colonnes ancienne dans un des connecteurs d'extension). Le Chapitre 2 contient une description de l'affichage en 80 colonnes.

Des touches de contrôle et d'évasion supplémentaires

Les caractéristiques d'affichage mentionnées ci-dessus (et beaucoup d'autres non mentionnées) sont commandables depuis le clavier par des séquences d'évasion et depuis un programme par des caractères de contrôle. Le Chapitre 3 contient des descriptions de ces codes d'évasion et de ces caractères de contrôle.

Une carte-langage dans la configuration de base

Les 16K octets de MEV que vous ajoutez à l'Apple II Plus en installant la Carte-Langage sont déjà implantés dans l'Apple //e, lui donnant une taille standard de 64K octets de mémoire. Dans l'Apple //e, ce bloc de 16K octets est appelé la mémoire à banc-commuté. Elle est décrite dans le Chapitre 4.

Une mémoire auxiliaire optionnelle

En installant une carte de texte en 80 colonnes étendue, vous pouvez ajouter une autre MEV de 64K à l'Apple //e. Le Chapitre 4 vous montre comment utiliser la mémoire additionnelle. (Note sur la compatibilité : la carte de texte en 80 colonnes fournit aussi l'option d'affichage en 80 colonnes).

Un connecteur auxiliaire

En plus des connecteurs d'expansion normaux il y a un connecteur spécial qui est utilisé soit pour une carte de texte en 80 colonnes soit pour une carte de texte en 80 colonnes étendues. Ce connecteur est localisé dans le Chapitre 1 et décrit au Chapitre 7.

Des connecteurs sur le panneau arrière

L'Apple //e a un panneau arrière métallique avec des emplacements pour plusieurs connecteurs de type D. Chaque carte périphérique

que vous rajoutez est livrée avec un connecteur que vous installerez sur le panneau arrière. Le Chapitre 1 contient une description du panneau arrière ; pour plus de détails, reportez-vous aux explications concernant l'installation fournies avec les cartes périphériques.

Des commutateurs logiciels additionnels - lisibles, aussi

Les fonctions supplémentaires d'affichage et de mémoire de l'Apple //e sont commandables et contrôlables par des commutateurs logiciels comme ceux de l'Apple II Plus. Sur l'Apple //e, les programmes peuvent aussi lire l'état de ces commutateurs logiciels. Le Chapitre 2 décrit les commutateurs logiciels qui contrôlent les fonctions d'affichage, et le Chapitre 4 décrit les commutateurs logiciels qui contrôlent les fonctions de mémoire.

Un auto-test dans la configuration de base

L'Apple //e a des sous-programmes supplémentaires implantés en MEM dont un sous-programme d'auto-test. Cet auto-diagnostic a comme but premier les essais pendant la fabrication, mais vous pouvez le faire fonctionner pour être sur que l'Apple //e travaille correctement. L'auto-test est décrit dans le Chapitre 4.

Réinitialisation forcée

Certains programmes de l'Apple II Plus prennent le contrôle de la fonction d'initialisation dite « Reset » pour empêcher l'utilisateur d'arrêter l'ordinateur et de copier le programme. L'Apple //e a une réinitialisation forcée qui réécrit dans le programme en mémoire pour le modifier. En utilisant la réinitialisation forcée, vous pouvez redémarrer l'Apple //e sans couper le courant puis réallumer ce qui provoque des à-coups inutiles sur les circuits.

Une prise en compte des interruptions

Même si la plupart des programmes d'applications n'utilisent pas d'interruptions, l'Apple //e permet d'avoir des programmes déclenchés par interruption. Par exemple, les sous-programmes pour 80 colonnes autorise périodiquement des interruptions pendant qu'il efface l'écran (normalement une opération longue durant laquelle les interruptions sont verrouillées). Les interruptions sont expliquées au Chapitre 6.

Un signal de synchronisation pour les graphiques animés

Des programmes avec des graphiques animés peuvent se mettre au pas sur l'affichage et éviter des clignotements des formes affichées sur l'écran. Le Chapitre 7 contient une description de la génération de signaux vidéo et de la synchronisation verticale.

Un octet identifiant l'Apple IIe

Un programme peut savoir s'il est exécuté sur un Apple //e ou sur un ancien modèle Apple II en lisant l'octet à l'adresse $FBB3 dans le programme moniteur du système. Dans le moniteur de l'Apple //e, cet octet vaut $06 ; dans le moniteur Autostart (le moniteur standard sur l'Apple II Plus), sa valeur est $EA. (Remarque : si vous démarrez avec le DOS et que vous passez sur le BASIC Entier, le moniteur Autostart est en fonction et la valeur à l'adresse $FBB3 sera $EA, même sur l'Apple //e.) Evidemment, il y a de nombreuses autres adresses qui ont des valeurs différentes suivant les différentes versions du moniteur ; l'adresse $FBB3 a été choisie parce qu'elle aura la valeur $06 même dans les versions révisées futures du moniteur de l'Apple //e.

Implantation des circuits

L'implantation des circuits de l'Apple //e est radicalement différente de celle de l'Apple II et de l'Apple II Plus. Trois des plus importantes différences sont

- Les circuits intégrés fabriqués sur mesure, l'IOU et la MMU.
- Les circuits de vidéo, qui se servent de MEM pour générer à la fois du texte et du graphique.
- Le bus de données périphériques, qui est entièrement amplifié par des registres-tampons.

Toutes ces caractéristiques sont décrites dans le Chapitre 7.

Glossaire

6502 : Le microprocesseur utilisé par l'ordinateur Apple //e.

accumulateur : Le registre du microprocesseur 6502 dans lequel la plupart des calculs s'effectuent.

acronyme : Un mot formé des lettres initiales d'un nom ou d'une phrase, comme « laser », qui vient de « Light Amplification by Stimulated Emission of Radiation » ou amplification de lumière par émission stimulée de radiation.

adresse : Un nombre utilisé pour identifier quelque chose, comme par exemple une position (un emplacement) dans la mémoire de l'ordinateur.

adresse de retour : Le point dans un programme où le contrôle repasse au programme appelant après l'achèvement d'un sous-programme.

adressage indéxé : Une méthode pour spécifier des adresses de mémoire en programmation en langage-machine.

afficher : Représenter visuellement l'information.

affichage : Information visualisée sur un écran de téléviseur ou de moniteur vidéo.

adresse de base : En mode d'adressage indexé, c'est la composante fixe d'une adresse.

alimentation électrique : Voir **boitier d'alimentation.**

amorcer : Mettre en fonctionnement un ordinateur en chargeant un programme en mémoire venant d'un support externe de mémoire comme par exemple une disquette. L'amorçage est souvent réalisé en chargeant d'abord un petit programme dont le but est de lire sur disquette un plus grand programme et de l'écrire en mémoire vive. En américain, c'est le verbe « to boot », qui vient de « bootstrap » ou chausse-pied. On dit que le programme se met lui-même dedans avec ses propres chausse-pieds.

amorçage : Voir **amorcer.** Le terme américain équivalent est « bootstrap ».

analogique : Représentée sous forme d'une quantité physique, comme par exemple une tension, une fréquence, une longueur, ou une position, qui peut varier doucement et continuellement dans un domaine de valeurs. Par exemple une montre conventionnelle est un dispositif analogique qui représente l'heure du jour en fonction des angles des aiguilles de la montre. Comparez avec **digital.**

AND : L'opérateur logique ET qui produit un résultat vrai si les deux opérandes sont vrais, un résultat faux si l'un ou l'autre ou les deux opérandes sont faux ; comparez avec OR, OR exclusif, NOT.

Apple *//e* : Un ordinateur personnel de la famille APPLE II, fabriqué et vendu par Apple Computer Inc.

Applesoft : Une version étendue du langage de programmation BASIC sur l'Apple //e. Un interprêteur qui permet de créer et d'exécuter des programmes en Applesoft est implanté en MEM dans le système Apple //e.

ASCII : « American Standard Code for Information Interchange » ou code américain standard pour l'échange d'informations ; un code dans lequel les nombres de 0 à 127 correspondent à des caractères, utilisés pour représenter du texte à l'intérieur d'un ordinateur et pour transmettre des textes entre ordinateurs ou entre un ordinateur et un dispositif périphérique.

assembleur : Un traducteur de langage qui convertit un programme écrit en langage d'assemblage en un programme équivalent en langage-machine.

bande passante : Une mesure du domaine de fréquences qu'un dispositif peut manipuler. Dans le cas d'un moniteur vidéo, une plus grande bande passante l'autorise à afficher plus d'informations ; pour afficher 80 colonnes de caractères, un moniteur doit avoir une bande passante minimum de 12 MHz.

binaire : La représentation des nombres en fonction des puissances de deux, en utilisant les deux chiffres 0 et 1. Utilisée couramment dans les ordinateurs, puisque les valeurs 0 et 1 peuvent être facilement représentées physiquement de nombreuses façons, comme par exemple la présence ou l'absence de courant, d'une tension positive ou négative, d'un point blanc ou noir sur l'écran.

bit : Un chiffre binaire (0 et 1) ; la plus petite unité d'information possible, représentant un choix simple à deux états, comme par exemple oui ou non, sous-tension ou hors-tension, positif ou négatif, quelque chose ou rien (contraction de « binary digit »).

boitier à deux lignes de broches : Un circuit intégré placé dans un boitier rectangulaire étroit avec une rangée de broches alignées de chaque côté ; ressemblant pourquoi pas à un millepatte en armure ! Appelé DIP en américain.

boitier d'alimentation : Le composant hardware d'un ordinateur qui utilise le courant électrique d'une prise du réseau électrique et le convertit sous une forme adaptée pour d'autres composants hardware.

bootstrap : Voir **amorçage.**

bus : Un groupe de fils électriques qui transmettent des informations corrélées, comme les bits d'une adresse, d'un endroit de l'ordinateur jusqu'à un autre.

bus périphérique : Le bus utilisé pour transmettre des informations entre l'ordinateur Apple //e et les dispositifs périphériques branchés dans les connecteurs d'expansion de l'ordinateur.

CAD : Voir **convertisseur analogique-digital.**

caractère : Une lettre, un chiffre, un symbole de ponctuation, ou tout autre symbole écrit utilisé en impression ou en affichage d'informations sous une forme lisible par une personne.

caractère de Contrôle : Un caractère qui contrôle ou modifie la façon dont une information est transmise ou affichée. Les caractères de contrôle ont des codes ASCII compris entre 0 et 31 et sont tapés au clavier de l'Apple //e en maintenant enfoncée la touche Ctrl tout en appuyant sur un autre caractère. Par exemple le caractère Ctrl-M (de code ASCII 13) signifie « revenir au début de la ligne» et il équivaut à la touche ↵.

caractère de sollicitation : Un caractère affiché sur l'écran pour solliciter une action de l'utilisateur. Identifie souvent le programme ou la composante du système qui est en train de faire la sollicitation ; par exemple, le caractère de sollicitation est utilisé par l'interprêteur BASIC Applesoft, > par le BASIC Entier, et * par le programme moniteur du système. Appelé aussi *caractère solliciteur.*

caractère espace : Un caractère dans un texte dont la représentation imprimée ou affichée est un espace vide, tapé du clavier en appuyant sur la barre d'espacement.

carte : Voir **carte périphérique.**

carte contrôleur : Une carte périphérique qui relie un dispositif comme une imprimante ou un lecteur/enregistreur de disquette à l'Apple //e et contrôle le fonctionnement du dispositif périphérique.

carte « ève » : carte fabriquée en France par *Le Chat Mauve micro-informatique* ayant de multiples fonctions : visualisation de texte en 80 colonnes, extension mémoire de 64 Koctets, visualisation couleur sur un téléviseur muni d'une prise Péritel ou sur un moniteur RVB et extension des possibilités graphique de l'Apple //e.

carte de texte en 80 colonnes : Une carte périphérique fabriquée et vendue par Apple Computer qui s'installe dans le connecteur auxiliaire de l'Apple //e et convertit l'affichage de texte sur écran de l'Apple //e de la largeur 40 colonnes à la largeur 80 colonnes.

carte de texte en 80 colonnes Etendue : Une carte périphérique fabriquée et vendue par Apple Computer qui se branche dans le connecteur auxiliaire de l'Apple //e et convertit la largeur d'affichage de texte sur écran de 40 colonnes à 80 colonnes et qui augmente la capacité de mémoire de 64 octets.

carte périphérique : Une carte sous forme de circuit imprimé que l'on peut enlever et qui s'installe dans un des connecteurs d'extension de l'Apple //e et augmente ou modifie les capacités de l'ordinateur en connectant un dispositif périphérique ou en réalisant une fonction périphérique ou subsidiaire.

CDA : Voir **convertisseur digital-analogique**

chaine de caractères : Un élément d'information fait d'une séquence de caractères de texte ou alphanumériques.

chiffre : (1) Un des caractères de 0 à 9, utilisé pour exprimer des nombres en numération décimale. (2) Un des caractères utilisés pour exprimer des nombres dans une autre base de numération, comme 0 et 1 en binaire ou 0 à 9 et A à F en héxadécimal. Se dit « digit » en américain.

CI : Voir **circuit intégré.**

circuit imprimé : Un élément de hardware d'un ordinateur ou d'un dispositif électronique, composé d'une plaquette plate, rectangulaire de matériau rigide, habituellement en fibre de verre, sur laquelle des circuits intégrés et d'autres composants électroniques sont connectés.

circuit intégré : Un composant électronique fait de plusieurs éléments de circuits fabriqués sur une seule pièce (c'est la puce !) d'un matériau semi-conducteur comme le silicium ; voir **pastille.**

clavier : L'ensemble des touches faisant partie de l'ordinateur Apple //e, ressemblant à un clavier de machine à écrire, pour taper des informations pour l'ordinateur.

charger : Transférer des informations depuis un support de mémorisation externe (tel qu'un disque) jusqu'à la mémoire principale pour l'utiliser ; par exemple, pour transférer un programme en mémoire et l'exécuter.

code de caractère : Un nombre utilisé pour représenter un caractère devant être traité par un système informatique.

code : (1) Un nombre ou un symbole utilisé pour représenter une information sous une forme condensée et facile à traiter. (2) Les déclarations ou les instructions constituant un programme.

code-objet : Voir **programme-objet.**

code-opération : La partie d'une instruction en langage-machine qui spécifie l'opération à effectuer ; souvent appelé *op code*.

commande : Un message communiqué par l'utilisateur à l'ordinateur (en général tapé au clavier) lui ordonnant d'exécuter immédiatement une action.

commutateur logiciel : Un moyen d'intervenir sur une fonction caractéristique de l'Apple //e à partir d'un programme ; spécifiquement, une position de mémoire qui produit un certain effet chaque fois que son adresse est référencée soit pour y lire soit pour y écrire.

compilateur : Un traducteur de langage qui convertit un programme écrit dans un langage de programmation évolué ou de haut-niveau en un programme équivalent dans un langage peu évolué (comme le langage-machine) pour son exécution ultérieure. Comparez avec **interprèteur**.

composant : Une pièce ou une partie ; en particulier une partie d'un ordinateur.

composite : Se dit d'un signal vidéo dans lequel sont inclus l'information à afficher et les signaux de synchronisation (et d'autres) nécessaires à l'affichage de cette information.

connecteur : Un dispositif physique comme par exemple une prise, un support ou une broche utilisé pour relier un composant hardware d'un système à un autre.

connecteur auxiliaire : Le connecteur spécial à l'intérieur de l'Apple //e utilisé par la carte de texte en 80 colonnes ou la carte de texte en 80 colonnes étendue ou la carte « ève ».

connecteur d'expansion : Un connecteur à l'intérieur de l'ordinateur Apple //e dans lequel une carte périphérique peut être installée ; appelé quelque fois *connecteur périphérique*.

connecteur d'E/S de jeu : Un connecteur spécial à 16 broches à l'intérieur de l'Apple //e, conçu à l'origine pour brancher des manettes de jeu à l'ordinateur, mais aussi utilisable pour connecter d'autres dispositifs périphériques. Comparez avec le connecteur pour manettes de jeu.

connecteur périphérique : Voir **connecteur d'extension.**

connecteur pour manettes de jeu : Un connecteur à 9 broches sur le panneau arrière de l'Apple //e, utilisé pour brancher des manettes de jeu à l'ordinateur. Comparez avec le **connecteur d'E/S de jeu.**

convertisseur analogique-digital : Un dispositif qui convertit des quantités analogiques en informations digitales. Par exemple, le contrôleur de manette de jeu de l'Apple //e convertit la position de la manette de jeu (une quantité analogique) en un nombre discret (une quantité digitale ou numérique) qui change brusquement même si on fait tourner la manette doucement.

convertisseur digital-analogique : Un dispositif qui convertit des quantités sous forme digitale ou numérique en analogique.

curseur : Une marque ou un symbole affiché sur l'écran qui indique où la prochaine action de l'utilisateur prendra effet ou bien où le prochain caractère tapé au clavier apparaitra sur l'écran.

CRT : Voir **Tube à rayon cathodique.**

debug : Verbe américain dérivé de « bug » ; localiser et corriger une erreur ou la cause d'un problème ou d'un mauvais fonctionnement dans un système informatique. S'utilise typiquement pour des problèmes de logiciels ; comparez à détection de panne.

décimale : La formulation habituelle pour représenter les nombres dans la vie courante : les nombres sont exprimés en fonction des puissances de dix, avec les chiffres de 0 à 9.

défaut (par) : Une valeur, une action ou un positionnement qui est pris automatiquement par le système informatique si aucune autre information explicite n'a été donnée. Par exemple, si une commande d'exécution d'un programme sur disquette ne précise pas quel lecteur utiliser, le Système d'Exploitation de Disquette (DOS) va automatiquement choisir le dernier lecteur utilisé.

défenestration : Action de jeter quelque chose par la fenêtre. Ce n'est pas une opération à faire subir à l'Apple //e.

défiler (faire) : Changer le contenu de tout ou partie de l'écran en décalant l'information hors de l'écran (le plus souvent vers le haut) pour laisser de la place à de nouvelles informations apparaissant à l'autre bord (le plus souvent le bas), en produisant un effet similaire à celui d'un rouleau de papier défilant derrière une fenêtre fixe. Voir **fenêtre.**

démarrage à chaud : La procédure de réinitialisation de l'Apple //e bien qu'il soit toujours sous-tension, sans recharger le système d'exploitation en mémoire principale et souvent sans perdre le programme ou les informations encore en mémoire principale. Comparez au **démarrage à froid.**

démarrage à froid : La procédure de démarrage de l'Apple //e quand on le met d'abord sous-tension (ou comme si l'on venait de le mettre sous-tension). Il se produit le chargement du système d'exploitation en mémoire principale, puis le chargement d'un programme et son exécution. Comparez avec le **démarrage à chaud.**

demi-octet : Une unité d'information faite de quatre bits ; peut contenir toute valeur de 0 à 15. Appelé en américain « nibble » ou « nybble ».

démoduler : Retrouver l'information transmise par un signal modulé ; par exemple, un récepteur de radio conventionnel démodule un signal entrant de radio-diffusion pour le convertir en sons émis par un haut-parleur.

désassembleur : Un traducteur de langage qui convertit un programme en langage-machine en un programme équivalent en langage d'assemblage, plus facilement compréhensible par une personne qui programme. C'est l'inverse d'un assembleur.

désempiler : Enlever l'élément situé en haut d'une pile. Appelé « pop » en américain.

détection de panne : Localisation et correction de la cause d'un problème ou du mauvais fonctionnement d'un système informatique.
S'applique typiquement aux pannes de hardware.

digital : Représenté sous forme discrète (dicontinue), comme les chiffres numériques. Par exemple les montres à quartz actuelles affichent le temps sous forme numérique (comme 2 : 57) au lieu d'utiliser les positions des deux aiguilles d'un cadran. Comparez avec **analogique.**

DIP : « Dual in-line package » ; voir **boitier à deux lignes de broches.**

dispositif : (1) Un appareil physique effectuant une tâche particulière ou réalisant un objectif particulier. (2) En particulier, un composant hardware d'un système informatique.

dispositif d'E/S : Un dispositif d'entrée/sortie ; un dispositif qui transfère des informations dans ou hors d'un ordinateur. Voir **Entrée, Sortie, dispositif périphérique.**

dispositif périphérique : Un dispositif, comme un moniteur vidéo, un lecteur de disque, une imprimante ou un modem, utilisé en conjonction avec un ordinateur. Souvent (mais pas nécessairement) séparé physiquement de l'ordinateur et connecté à lui par des fils, des câbles, ou toute autre forme d'interface, typiquement au moyen d'une carte périphérique.

disque : Un support de mémorisation d'information fait d'une surface plate, circulaire magnétique sur laquelle l'information peut être enregistrée sous la forme de petites zones magnétisées, de la même façon que les sons sont enregistrés sur une bande magnétique.

disque dur : Un disque fait de matériau dur non flexible. Comparez à **floppy (disque).**

disquette : Un terme utilisé pour les petits disques souples (13 cm de diamètre) qui vont dans le lecteur de disque Disk II de l'Apple //e.

donnée : Information ; en particulier une information utilisée ou traitée par un programme.

DOS : « Disk Operating System » ; voir **Système d'Exploitation de Disque.**

échantillonnage : Un événement, comme le changement d'un signal, qui déclenche une certaine action.

échantillonneur : Un signal dont le changement est utilisé pour déclencher une certaine action.

écrire : Transférer de l'information depuis un ordinateur vers une destination externe à l'ordinateur (comme un lecteur de disque, une imprimante, ou un modem) ou bien depuis le processeur de l'ordinateur vers une destination externe au processor (comme la mémoire principale).

éditer : Changer ou modifier ; par exemple, insérer, enlever, remplacer ou déplacer des textes dans un document.

éditeur : Un programme qui permet à l'utilisateur de créer et d'éditer des informations d'une forme particulière ; par exemple, un *éditeur de textes* ou un *éditeur de graphiques*.

effective (adresse) : En programmation en langage-machine, l'adresse de la position de mémoire sur laquelle opère actuellement une instruction particulière, cette adresse ayant été obtenue par adressage indexé ou une autre méthode d'adressage.

empiler : Ajouter un élément sur le sommet de la pile. Appelé « push » en américain.

entier : Un nombre sans partie fractionnaire ; représenté dans l'ordinateur en virgule fixe. Comparez avec **nombre réel.**

entrée : Information transférée dans l'ordinateur depuis une source extérieure, comme par exemple le clavier, le lecteur de disque, ou un modem.

exécuter : Effectuer ou réaliser une action spécifique ou une séquence d'actions, comme celles décrites dans un programme.

E/S : Entrée/Sortie ; le transfert d'informations dans et hors d'un ordinateur. Voir **Entrée, Sortie.**

fenêtre : (1) La portion d'une collection d'information (comme un document, un dessin, ou un plan de travail) qui est rendue visible sur l'écran à travers un masque ; comparez à un **masque d'écran**. (2) Un masque d'écran. (3) Un panneau rectangulaire et plat, habituellement fait de verre, utilisée dans de nombreuses structures archaïques en tant qu'interface homme-environnement.

fenêtre de texte : Une zone de l'écran de l'Apple *//e* à l'intérieur de laquelle le texte est affiché et défile éventuellement.

firmware : Les composants d'un système informatique consistant en programmes enregistrés de manière permanente en mémoire à lecture seulement. De tels programmes (par exemple, l'interpréteur Applesoft et le programme moniteur de l'Apple //e) sont installés dans l'ordinateur à la fabrication ; ils peuvent être exécutés à tout moment mais ne peuvent être ni modifiés ni effacés de la mémoire principale. Appelés *sous-programme de base* du système dans ce manuel. Comparez avec **hardware, logiciel.**

floppy (disque) : Un floppy disque est un disque fait en plastique souple. Comparez avec **disque dur.**

frappe de touche: Le fait d'appuyer sur une seule touche ou une combinaison de touches (par exemple Ctrl-C) sur le clavier de l'Apple //e.

graphique : (1) Information présentée sous la forme de figures ou d'images. (2) L'affichage de dessins et d'images sur un écran d'ordinateur. Comparez avec **texte.**

graphique en haute-résolution : L'affichage de graphique sur l'écran de l'Apple //e en une matrice de points à six couleurs, de 280 colonnes de large et de 192 points de haut.

graphique en basse-résolution : L'affichage de graphique sur l'écran de l'Apple //e sous forme d'une matrice de blocs à seize couleurs, de 40 colonnes de large et 48 lignes de haut.

hardware : Les composants d'un ordinateur faits avec des dispositifs physiques (électroniques ou mécaniques). Comparez avec **software, firmware.**

Hertz : L'unité de mesure de la fréquence de vibrations ou d'oscillations, appelée aussi cycles par seconde ; son nom vient du physicien Heinrich Hertz et s'abrège en Hz. Le courant fournit par une prise du réseau alternatif a une fréquence de 50 Hz ; ce qui veut dire qu'il change de signe 50 fois par seconde. Le microprocesseur 6502 de l'Apple //e fonctionne à la fréquence de 1 million d'hertz, ou 1 mégahertz (MHz).

hexadécimale : La représentation des nombres en fonction des puissances de seize, en utilisant les seize chiffres de 0 à 9 et de A à F. Les nombres hexadécimaux sont plus faciles à lire et à comprendre que les nombres binaires, mais peuvent être facilement et directement convertis en numération binaire : à chaque chiffre hexadécimal correspond une séquence de quatre chiffres binaires ou bits.

Hz : Voir **Hertz**

information : Des faits, des concepts, des instructions représentés sous une forme organisée.

imprimante : Un dispositif périphérique qui écrit des informations sur du papier sous une forme facilement lisible par les humains ou par des singes alphabétisés.

index : (1) Un nombre servant à identifier un membre d'une liste ou d'une table par sa position séquentielle. (2) Une liste ou une table dont les cases sont identifiées par leur position séquentielles. (3) En programmation en langage-machine, la composante variable d'une adresse indexée, contenue dans un registre d'index et ajoutée à l'adresse de base pour obtenir l'adresse effective.

instruction : Une unité d'un programme en langage-machine ou en langage d'assemblage correspondant à une action unique que le processeur de l'ordinateur doit effectuer.

interface : Les dispositifs, règles et conventions par lesquels un composant d'un système communique avec un autre.

interface parallèle : Une interface dans lequel plusieurs bits d'information (typiquement huit bits, ou un octet) sont transmis simultanément sur différents fils ou canaux. Comparez avec **interface série.**

interface série : Une interface dans laquelle l'information est transmise séquentiellement, un bit à chaque fois, sur un seul fil ou un seul canal. Comparez avec **interface parallèle.**

interface-utilisateur : Les règles et les conventions par lesquelles un système informatique communique avec la personne qui le fait fonctionner.

interprèteur : Un traducteur de langage qui lit un programme écrit dans un langage de programmation particulier et immédiatement réalise les actions décrites dans ce programme. Comparez avec **compilateur.**

interruption : Une suspension temporaire dans l'exécution d'un programme par un ordinateur de façon qu'il puisse effectuer une autre tâche, souvent en réponse à un signal provenant d'un dispositif périphérique ou de toute autre source extérieure à l'ordinateur.

K : Deux à la puissance dix ou 1024 (vient de la racine grecque *Kilo* signifiant un millier) ; par exemple 64K est égal à 64 fois 1024, soit 65536.

kilo-octets : Une unité d'information consistant en 1K (1024) octets, ou 8K (8192) bits.

KSW : Le nom symbolique de la position de mémoire dans l'Apple //e ou la liaison standard d'entrée est enregistrée ; veut dire « keyboard switch » ou commutateur de clavier. Voir **liaison d'E/S.**

langage machine : La formulation dans laquelle les instructions sont enregistrées en mémoire pour être exécutées directement par le processeur d'ordinateur. Chaque modèle de processeur d'ordinateur (comme le microprocesseur du 6502 utilisé par l'Apple //e) à son propre langage-machine.

liaison d'E/S : Une adresse fixe qui contient l'adresse d'un sous-programme d'entrée/sortie dans le programme moniteur de l'Apple //e.

langage : Voir **langage de programmation.**

langage d'assemblage : Un langage de programmation peu évolué ou de bas-niveau dans lequel les instructions individuelles en langage-machine sont écrites sous forme symbolique pour être plus facilement compréhensibles par un programmeur humain que le langage-machine lui-même.

langage évolué : Un langage de programmation qui est relativement facile à comprendre par les humains. Une simple instruction dans un langage évolué correspond à plusieurs instructions en langage-machine. Un langage évolué est dit aussi de haut-niveau.

langage de bas-niveau : Un langage de programmation qui est relativement proche de la forme qu'un processeur d'ordinateur peut exécuter directement. Les langages de bas-niveau disponibles sur l'Apple //e sont le langage-machine du 6502 et le langage d'assemblage du 6502.

langage de programmation : Un ensemble de règles et de conventions à suivre pour écrire un programme.

lecteur de disque : Un dispositif périphérique qui écrit et lit des informations sur la surface d'un disque magnétique.

lecteur de disque Disk II : Modèle de lecteur de disque fabriqué et vendu par Apple Computer à utiliser avec l'ordinateur Apple //e ; il se sert de disquettes souples (« floppy ») de 13 cm.

lire : Transférer de l'information dans la mémoire de l'ordinateur depuis une source externe à l'ordinateur (comme un lecteur de disque ou un modem) ou bien dans le processeur de l'ordinateur en provenance d'une source extérieure au processeur (comme le clavier ou la mémoire principale).

LS : Voir Shottkey à faible consommation.

logiciel : Les composants d'un système informatique faits de programmes qui déterminent ou contrôle le comportement de l'ordinateur. Correspond à « software ». Comparez à **hardware, firmware.**

manettes de jeu : Un dispositif périphérique qui peut être branché au connecteur pour manettes de jeu et comprend un potentiomètre rotatif et un bouton-poussoir par manette ; typiquement utilisées dans les programmes de jeux d'adresses mais peuvent être utilisées dans des applications plus sérieuses.

masque d'écran : Tout ou partie de l'écran, utilisée par un programme d'application pour afficher une portion de l'information (comme un document, un dessin ou un plan de travail) sur laquelle le programme fonctionne. Comparez à **fenêtre.**

MEM : Voir **mémoire morte.**

mémoire : Un composant hardware d'un système informatique qui peut stocker des informations en vue d'une lecture ou d'une récupération ultérieure ; voir **mémoire principale, mémoire à accès direct ou vive, mémoire morte ou à lecture seulement, mémoire de lecture/écriture, mémoire à écriture seulement.**

mémoire à accès direct : Mémoire dont le contenu de chacune des positions individuelles peut être référencé dans un ordre arbitraire ou au hasard. Ce terme est souvent utilisé incorrectement pour ne parler que des mémoires vives, mais strictement parlant, les mémoires à lecture seulement et les mémoires vives sont à accès direct. Voir **mémoire à lecture seulement, mémoire morte, mémoire vive.**

mémoire à écriture seulement : Une forme de mémoire d'ordinateur dans laquelle l'information peut être stockée mais jamais récupérée, développée avec un contrat du gouvernement américain en 1975 par le Professeur Homberg T. Farnsfarfle. Le prototype original de Farnsfarfle, approximativement de 3 cm de côté, a déjà été utilisé pour enregistrer plus de 100 mille milliards de surplus d'information fédérale. Les critiques de Farnsfarfle ont dénoncé son projet comme une faveur de six millions de dollars, mais son défenseur démontre que ce surplus d'information aurait coûté plus de 250 milliards de dollars à être stocké sur un support conventionnel. Comparez aux **mémoire vive, mémoire morte, mémoire à accès direct.**

mémoire principale : La partie de mémoire d'un système informatique qui est intégrée dans l'ordinateur lui-même et dont les contenus sont directement accessibles par le professeur.

mémoire morte : Mémoire morte : Mémoire dont le contenu peut être lu mais pas écrit ; utilisée pour enregistrer le « firmware » c'est-à-dire les sous-programmes de base et les interprêteurs du système. L'information n'est écrite qu'une fois sur la mémoire morte, à la fabrication ; elle restera là en permanence, même si l'ordinateur est mis hors-tension, et ne peut jamais être effacée ou modifiée. Abrégée MEV. Comparez à **mémoire vive, mémoire à accès direct, mémoire à écriture seulement.**

mémoire-tampon : Une zone en mémoire de l'ordinateur réservée à un usage spécifique, tel que le maintien d'informations graphiques qui devront être affichées sur l'écran ou de caractères devant être lus à partir d'un dispositif périphérique d'entrée. Elle est souvent utilisée comme zone de transit pour transférer des informations entre des dispositifs fonctionnant à des vitesses différentes, comme par exemple le processeur d'un ordinateur et une imprimante ou un contrôleur de disquettes. L'information pourra être enregistrée dans une mémoire-tampon par un dispositif et ensuite lue par un autre à une vitesse différente.

mémoire vive : Mémoire à accès direct dans laquelle on peut lire et écrire des informations à des positions arbitraires référencées par leur adresse. Les informations contenues dans une mémoire vive s'effacent quand l'ordinateur est mis hors-tension, et sont définitivement perdues à moins qu'elles aient été sauvegardées sur un support de mémorisation permanente, telle qu'un disque. Voir **RAM**. Comparez à **mémoire morte, mémoire à écriture seulement.**

message d'erreur : Un message affiché sur l'écran ou imprimé pour notifier l'utilisateur d'une erreur ou d'un problème dans l'exécution d'un programme.

MEV : Voir **mémoire vive.**

MHz : Mégahertz ; un million de hertz. Voir **hertz.**

micro-ordinateur : Un ordinateur comme l'Apple //e, dont le processeur est un microprocesseur.

microprocesseur : Un processeur d'ordinateur contenu dans un seul circuit intégré, comme le microprocesseur 6502 utilisé dans l'Apple //e.

microseconde : Un millionième de seconde ; abrégé μs.

milliseconde : Un millième de seconde ; abrégé ms.

mode : L'état dans lequel se trouve un ordinateur ou un système et qui va déterminer son comportement ultérieur.

mode d'évasion : Un état dans lequel va se trouver l'Apple //e dès qu'on appuie sur la touche Esc, état dans lequel certaines touches sur le clavier prennent une certaine signification permettant de positionner le curseur et contrôler l'affichage de texte sur l'écran.

modem : Modulateur/Démodulateur ; Un dispositif périphérique qui autorise l'ordinateur à transmettre et à recevoir des informations par une ligne téléphonique.

modulateur de radio-fréquence : Un dispositif qui convertit les signaux vidéo produits par un ordinateur en une forme qui puisse être acceptée par un récepteur de télévision.

moduler : Modifier ou altérer un signal pour qu'il transmette de l'information ; par exemple, la radio-diffusion conventionnelle transmet des sons par modulation de l'amplitude (modulation d'amplitude ou AM) ou de la fréquence (modulation de fréquence ou FM) d'un signal porteur dite porteuse.

moniteur : Voir **moniteur vidéo.**

mot : Un groupe de bits d'une taille constante qui est traité comme une unité ; le nombre de bits dans un mot est une caractéristique de chaque ordinateur particulier.

nanoseconde : Un milliardième de seconde ; abrégé ns.

nibble : Voir **demi-octet.**

nombre réel : Un nombre qui peut avoir une partie fractionnaire ; représenté è l'intérieur de l'ordinateur en virgule flottante. Comparez à **entier.**

NOT : L'opérateur logique unaire PAS qui produit un résultat vrai si son opérande est faux, un résultat faux si son opérande est vrai ; comparez avec AND, OR, OR exclusif.

NTSC : (1) « National Television Standards Committe » ou commission nationale des normes de télévision ; la commission qui définit le format standard utilisé pour transmettre les signaux vidéo de diffusion aux États-Unis. (2) Le format standard du signal vidéo défini par la NTSC. Comparez au **P.A.L.**

octet : Une unité d'information composée d'un nombre fixe de bits ; sur l'Apple //e, un octet contient 8 bits et peut mémoriser toute valeur comprise entre 0 et 255.

octet de plus faible poids : La moitié la moins significative d'une adresse de mémoire ou d'une autre quantité sur deux-octets. Dans le microprocesseur de l'Apple //e, l'octet de plus faible poids d'une adresse est habituellement enregistré en premier et l'octet de plus fort poids en second.

opérande : Une valeur sur laquelle s'applique un opérateur.

opérateur : Un symbole ou une séquence de caractères, tels que + ou AND, spécifiant une opération à effectuer sur une ou plusieurs valeurs (les opérandes) pour produire un résultat.

opérateur binaire : Un opérateur qui fait la combinaison de deux opérandes pour produire un résultat ; par exemple, OR est un opérateur binaire logique. Comparez avec un **opérateur unaire.**

opérateur logique : Un opérateur comme le AND, qui combine des valeurs logiques pour produire un résultat logique.

opérateur unaire : Un opérateur qui ne s'applique qu'à un seul opérande. Par exemple, le signe (−) dans un nombre négatif comme −6 est un opérateur arithmétique unaire. Comparez à l'**opérateur binaire.**

OR : L'opérateur logique OU qui produit un résultat vrai si l'un ou l'autre ou les deux opérandes sont vrais, un résultat faux si les deux opérandes sont faux ; comparez avec OR exclusif, AND, NOT.

ordinateur : Un dispositif électronique capable d'effectuer des calculs prédéfinis (programmés) à haute vitesse et avec une grande précision.

OR exclusif : Un opérateur logique qui produit un résultat vrai si un des opérandes est vrai et l'autre faux, et un résultat faux si ses opérandes sont tous les deux ou vrais ou faux ; comparez avec OR, AND, NOT.

page : (1) Un écran plein d'informations sur un affichage vidéo, fait de 24 lignes de 40 ou 80 colonnes chacune sur l'Apple //e. (2) Une zone de mémoire principale contenant du texte ou de l'information graphique à afficher sur l'écran. (3) Un segment de mémoire principale de 256 octets de long en commençant à une adresse qui est un multiple pair de 256.

page zéro : La première page (256 octets) de la mémoire de l'Apple //e. Puisqu'un octet de plus fort poids de toute adresse dans cette page vaut zéro, seul l'octet de plus faible poids est nécessaire pour spécifier une adresse de la page zéro ; ceci rend les positions de mémoire plus efficaces à adresser, en espace et en temps, que des positions dans tout autre page de mémoire.

P.A.L. : « Phase Alternating Lines » ou lignes en phase alternée ; système normalisé en Europe de codage des signaux de télévision en couleur. Les Apple //e vendus en Europe contiennent des circuits additionnels permettant la génération d'un signal vidéo aux normes P.A.L. et la comptabilité avec les téléviseurs de ce type. (En France le codage des signaux de télévision couleur est d'un type différent appelé procédé SECAM).

panneau arrière : La face arrière de l'ordinateur Apple //e, qui comprend l'interrupteur d'alimentation, le connecteur d'alimentation, et des connecteurs pour un dispositif d'affichage vidéo, un magnétophone à cassette, et d'autres dispositifs périphériques.

pastille de silicium : Le petit substrat de matériau semi-conducteur (comme le silicium) sur lequel est fabriqué un circuit intégré. En américain, c'est le terme « chip » à qui on attribue aussi le sens de circuit intégré ; voir **circuit intégré.**

périphérique : Sur ou hors des limites de l'ordinateur lui-même, soit dans un sens physique (comme un dispositif périphérique) soit dans un sens logique (comme une carte périphérique).

phase : (1) Une étape dans un processus périodique ; un point dans un cycle ; par exemple le microprocesseur 6502 utilise un cycle d'horloge constitué de deux phases appelées O 0 et O 1. (2) La relation entre deux signaux ou processus périodiques ; par exemple, en vidéo NTSC couleur, la couleur d'un point sur l'écran est déterminée par le déphasage instantané entre le signal vidéo et le signal de référence de couleur.

pile : Une liste où les éléments sont entrés ou enlevés à une seule extrémité (le haut de la pile), faisant qu'ils sont enlevés dans l'ordre LIFO (« last-in-first-out » ou dernier entré premier sorti). Appelée « stack » en américain.

pipelining : Une caractéristique de fonctionnement d'un microprocesseur qui lui permet de commencer à rechercher la prochaine instruction avant d'avoir terminé l'exécution de l'instruction courante. Toutes autres choses égales par ailleurs, les processeurs qui ont cette caractéristique fonctionnent plus vite que ceux qui ne l'ont pas.

planter (se) : On dit qu'un programme se plante quand il s'arrête sans qu'on s'y attende, et que des informations se détruisent dans ce processus.
Équivant à « crash ».

pointeur : Un élément d'information qui est l'adresse d'un autre élément.

pop : Voir **désempiler.**

port : Le point de connection, habituellement un connecteur physique, entre un ordinateur et un dispositif périphérique, un autre ordinateur ou un réseau.

porteuse : Un signal de radio-diffusion qui est modulé pour transmettre de l'information.

position : Voir **position de mémoire.**

position de mémoire : Une unité de mémoire principale qui est repérée par une adresse et peut contenir un seul élément d'information d'une taille fixe ; dans l'Apple //e, une position de mémoire contient un octet, ou 8 bits, d'information.

processeur : Le composant hardware d'un ordinateur qui effectue réellement les calculs en exécutant directement les instructions formulées en langage-machine et enregistrées en mémoire principale.

programme : Un ensemble d'instructions décrivant des actions que doit effectuer un ordinateur pour accomplir une certaine tâche, conformément aux règles et aux conventions d'un langage de programmation particulier.

programme Moniteur : Un programme du système faisant partie de l'Apple //e implanté en MEV, utilisé pour inspecter ou changer directement les contenus de la mémoire principale et pour faire fonctionner l'ordinateur au niveau du langage-machine.

programme-objet : La forme traduite d'un programme produite par un traducteur de langage comme un complicateur ou un assembleur ; appelé aussi *code-objet.* Comparez avec **programme-source.**

programmer : Écrire un programme ; voir **programme.**

programme-source : La forme originale d'un programme donnée à un traducteur de langage comme un compilateur ou un assembleur pour la convertir en une autre forme ; appelé quelquefois *code source.* Comparez avec **programme-objet.**

push : Voir **empiler.**

RAM : Voir **mémoire à accès direct et mémoire vive.**

réenroulement : La fonction qui permet qu'un texte arrivant en fin de ligne puisse continuer à s'afficher ou s'imprimer automatiquement au début de la ligne suivante. Appelé « wraparound » en américain.

registre : Une position dans le processeur d'ordinateur où un élément d'information, par exemple un octet, est mémorisé et modifié sous le contrôle d'un programme. Les registres du microprocesseur 6502 sont l'accumulateur (A), deux registres d'index (X et Y), le pointeur de pile (S), le registre d'état du processeur (P) et le compteur ordinal ou compteur d'instructions (PC). Le registre PC a deux octets (seize bits) ; les autres ont un octet ou 8 bits chacun.

registre d'index : Un registre du processeur de l'ordinateur qui contient un index à utiliser dans l'adressage indexé. Le microprocesseur de l'Apple //e dispose de deux registres d'index, appelés le registre X et le registre Y.

registre X : Un des registres d'index du microprocesseur 6502.

registre Y : Un des registres d'index du microprocesseur 6502.

récepteur de télévision : Un dispositif de visualisation capable de recevoir des signaux de vidéo-diffusion (comme les chaînes de télévision) au moyen d'une antenne. Il peut être utilisé associé à un modulateur en radio-fréquence comme écran d'affichage pour l'ordinateur Apple //e. Comparez au **moniteur vidéo.**

réseau : Une collection d'ordinateur contrôlés individuellement et interconnectés, associée au matériel et au logiciel utilisés pour les connecter.

résident en mémoire : (1) Enregistré en permanence en mémoire principale, comme par exemple les sous-programmes de base du système ou « firmware ». (2) Maintenu en permanence en mémoire centrale même s'il n'est pas utilisé, comme par exemple le Système d'Exploitation de Disque (DOS).

retenue : Bit d'état dans le microprocesseur 6502, utilisé dans l'addition et la soustraction pour mémoriser le bit de plus fort poids dit bit de retenue.

routine : Une partie d'un programme qui accomplit une certaine tâche subordonnée au travail global d'un programme. Voir **sous-programme.**

RF (modulateur) : Voir **modulateur en radio-fréquence.** Équivaut en France à HF ou VHF.

ROM : Voir **mémoire à lecture seulement.**

RUN : (1) commande permettant l'exécution d'un programme. (2) Commande de chargement d'un programme en mémoire principale depuis un support de mémorisation externe, comme un disque, et exécution de ce programme.

sauvegarder : Transférer des informations depuis la mémoire principale vers un support périphérique de mémoire pour un usage ultérieur.

séquence d'évasion : Une séquence de touches tapées qui commence par la touche (Esc), utilisée pour positionner le curseur et contrôler l'affichage de texte sur l'écran.

solliciter : Rappeler ou signaler à l'utilisateur qu'une certaine action est attendue, typiquement par affichage d'un symbole distinctif, un message de rappel, ou un menu proposant des choix sur l'écran.

sortie : (1) Information transférée depuis un ordinateur jusqu'à une destination externe, comme un écran de visualisation, un lecteur de disque, une imprimante, ou un modem. (2) Le fait ou le processus de transfert d'une telle information.

Shottkey à faible consommation : Un type de circuit intégré TTL ayant une faible consommation de puissance et plus rapide qu'un circuit TTL conventionnel. En abrégé LS.

silicium : Un élément chimique semi-conducteur, non-métallique à partir duquel sont fabriqués les circuits intégrés. A ne pas confondre avec la *silice* —c'est-à-dire du dioxyde de silicium, comme le quartz, l'opale ou le sable— ou le *silicone*, un composant du groupe de composants organiques contenant du silicium.

sous-programme : Une partie d'un programme qui peut être exécutée sur demande de n'importe quel point d'un programme, et qui après sa réalisation retourne le contrôle au point où avait eu lieu la demande.

système : Une collection coordonnée de parties reliées et interagissant entre elles organisée pour réaliser une certaine fonction ou atteindre un certain objectif.

Système d'Exploitation de Disque (DOS) : Un logiciel optionnel du système Apple //e qui permet à l'ordinateur de contrôler et de communiquer avec les un ou plusieurs lecteurs de Disque Disk II.

système informatique : Un ordinateur et ses circuits (« hardware »), ses sous-programmes en mémoire morte (« firmware ») et ses logiciels (« software ») associés.

téléviseur : Voir **récepteur de télévision.**

temps de maintien : Dans les circuits d'un ordinateur, le laps de temps pendant lequel un signal valide doit se maintenir avant qu'un autre signal qui dépend du premier puisse retomber. Comparez avec **temps d'établissement.**

temps d'établissement : Le laps de temps pendant lequel un signal doit être valide avant qu'un événement se produise. Comparez avec le **temps de maintien.**

terminal : Un dispositif fait d'un clavier de type machine à écrire et d'un écran d'affichage, utilisé pour la communication entre un système informatique et un utilisateur. Les ordinateurs personnels comme l'Apple //e ont typiquement tout ou partie d'un terminal intégré en eux.

texte : (1) Information présentée sous la forme de caractères lisibles par les humains. (2) L'affichage de caractères sur l'écran de l'Apple //e. Comparez à **Graphique.**

traducteur de langage : Un programme du système qui lit un programme écrit dans un langage de programmation particulier et soit l'exécute immédiatement soit le convertit dans un autre langage (comme le langage-machine) pour une exécution ultérieure. Voir **interpréteur, compilateur, assembleur.**

trame : Le dessin des lignes constituant l'image sur un écran vidéo. L'image est produite en contrôlant la brillance des points successifs sur chaque ligne de la trame.

Transistor-Transistor-Logic : (1) Une famille de circuits intégrés utilisés dans les ordinateurs et dans des dispositifs associés. (2) Un standard d'interconnection de tels circuits qui définit les tensions utilisés pour représenter le zéro et le un logiques.

TTL : Voir **Transistor-Transistor-Logic.**

tube cathodique : Un dispositif électronique, comme par exemple le tube-couleur d'un téléviseur, qui produit des images sur un écran recouvert de phosphores qui émettent de la lumière quand ils reçoivent un faisceau focalisé d'électrons.

UC : Unité centrale ; voir **processeur.**

unité Centrale : Voir **processeur.**

utilisateur : La personne faisant fonctionner ou contrôlant un système informatique.

vecteur : (1) L'adresse de début d'un segment de programme, quand on l'utilise comme un point commun pour transférer le contrôle depuis d'autres programmes. (2) Une position de mémoire utilisée pour mémoriser un vecteur, ou bien l'adresse d'une telle position.

vidéo : (1) Un moyen de transmission d'information sous la forme d'images devant être affichées sur l'écran d'un tube à rayons cathodiques. (2) Information organisée ou transmise sous forme vidéo.

vidéo inverse : L'affichage d'un texte sur l'écran de l'ordinateur sous forme de points noirs sur un fond noir (ou une autre couleur simple de phosphore), au lieu des points blancs habituels sur fond noir.

virgule fixe : Une méthode de représentation des nombres dans l'ordinateur dans laquelle la virgule décimale (plus exactement la virgule binaire) est considérée comme étant toujours à la même position dans le nombre. Typiquement, la virgule est supposée se trouver à l'extrême droite du nombre, de telle sorte que le nombre est interprété comme un entier. Comparez avec **virgule flottante.**

virgule flottante : Une méthode de représentation des nombres dans l'ordinateur dans laquelle la virgule est autorisée à « flotter » dans différentes positions à l'intérieur du nombre. Quelques uns des bits dans le nombre lui-même sont utilisés pour repérer la position de la virgule. Comparez avec **virgule fixe.**

Bibliographie

Apple Computer, Inc. : *Apple IIe Applesoft Manuel de Référence ;* Apple Seedrin. ; 1983, Numéro de produit Apple A2L2004F

— *Apple IIe Applesoft Travaux Pratiques ;* Apple Seedrin, 1983. Numéro de produit Apple A2L2003F

— *Apple II BASIC Programming Manual ;* Apple Computer Inc., 1978, Cupertino, CA. Numéro de produit Apple A2L0005

— *Apple II Monitors Peeled ;* Apple Computer Inc. ; 1978, Cupertino, California. Numéro de produit Apple D2L0013

— *Apple IIe Guide de l'Utilisateur ;* Apple Seedrin, 1983, France. Numéro de produit Apple A2L2001F

Leventhal, Lance : 6502 *Assembly Language Programming,* Osborne/McGraw-Hill, 1979, Berkeley, CA

Synertek, Incorporated : *Hardware Manual ;* Synertek Incorporated, 1976, Santa Clara, CA

Watson, Allen, III : "More colors for your Apple", *Byte,* Vol. 4, No. 6, Juin 1979. Byte publications, Inc, Peterborough, NH

Guide de l'Apple II par Benoit de Merly. Editions Edi Micro, 10, rue Henri-Pape, 75013 Paris

Le Basic par la Pratique sur Apple II par Jean-Pierre Lamoitier. Editions SYBEX, 4, place Félix-Eboué, 75583 Paris

Clefs pour l'Apple II et la Pratique de l'Apple *II* par Nicole BREAUD POULIQUEN. Editions PSI.

Assembly Lines Apple *II. Manuel de Programmation du 6502* par Robert Wagner.
Editions IS.

Index