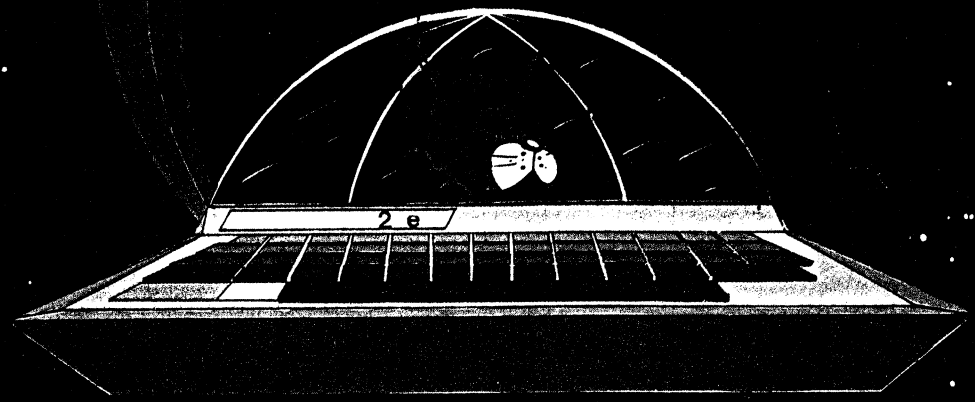
9, rue Arthur Rozier - 75019 - PARIS

SIMPARIC

RVB 80 COL. 64 K



LE CHAT MAUVE

Micro informatique

AVERTISSEMENT :

La société LE CHAT MAUVE micro-informatique se réserve le droit d'apporter des améliorations au produit (logiciel et matériel) décrit dans le présent manuel à tout moment et sans préavis.

Aucune garantie explicite ou implicite n'est donnée sur le logiciel décrit dans ce manuel.

Dessins de M. Michel BOUVET

Sommaire

CHAPITRE V

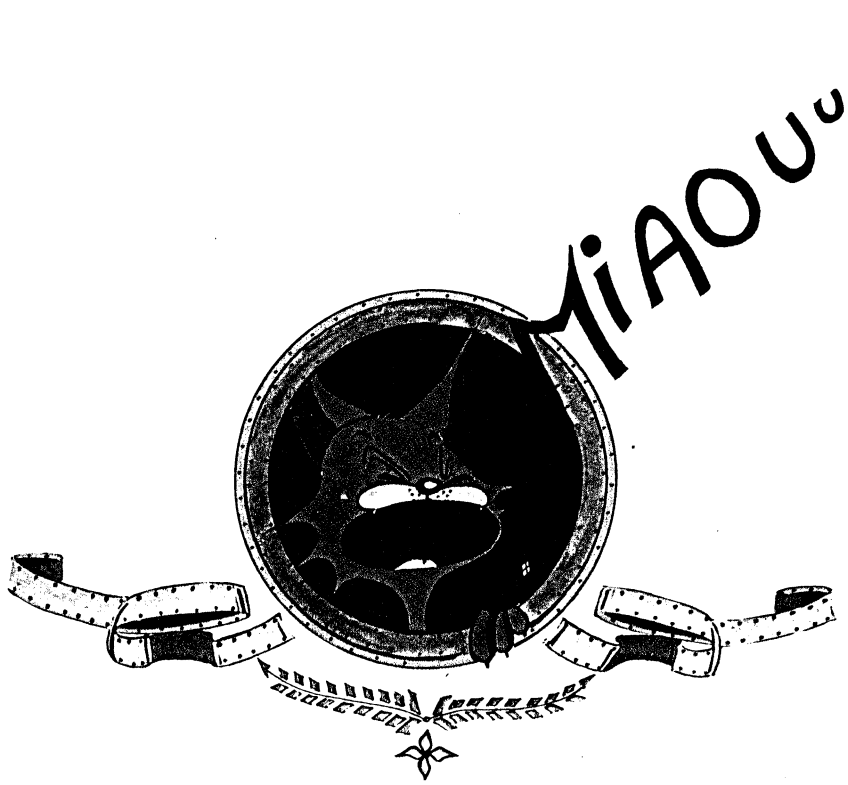
Le logiciel Basic Purplesoft

CHAPITRE VI

Visualisation couleur en Pascal

L'extension mémoire 64 K

Introduction



CHAPITRE I

Introduction

I-1 PRÉSENTATION

Votre nouvelle carte «ève» pour Apple *//e* a de multiples fonctions :

— Visualisation texte sur 80 colonnes (au lieu de 40).
— Extension mémoire (mémoire utilisateur portée de 64K à 128K).
— Visualisation couleur sur votre téléviseur par la prise PÉRITEL.
— Extension des possibilités graphiques de votre Apple *//e*.

Il est possible d'utiliser simultanément la sortie PÉRITEL de votre carte ève sur votre téléviseur et la sortie vidéo standard de l'Apple *//e* sur un moniteur monochrome.

Un moniteur couleur RVB peut être utilisé sous certaines conditions (consulter à ce sujet le chapitre IX).

Tous les programmes conçus pour Apple *//e* fonctionnent directement, ils utilisent éventuellement la visualisation du texte en 80 colonnes, l'extension mémoire 64K èt les possibilités graphiques standard de l'Apple *//e*.

Ce manuel est accompagné de deux disquettes correspondant aux logiciels d'exploitation en Basic Applesoft et en Pascal 1.1 permettant d'accéder facilement aux extensions graphiques spécifiques à la carte ève.

Vous pourrez obtenir quelques informations complétant ce manuel en exécutant le programme Basic INFORMATION présent sur la disquette DOS 3.3 PURPLESOFT.

I-2 COMMENT UTILISER CE MANUEL ?

Nous espérons que ce manuel saura répondre à toutes les questions que vous êtes en droit de vous poser au sujet de votre carte «ève». Néanmoins, nous savons que votre pratique de l'ordinateur individuel et plus généralement de la micro-informatique peut varier du tout au rien en fonction du temps dont vous disposez et de ce que vous attendez d'un tel outil.

Ainsi, nous pouvons distinguer grossièrement trois catégories d'utilisateurs :

— Ceux pour qui l'APPLE *IIe* et la carte « ève » ne sont qu'un moyen efficace d'utiliser au mieux des logiciels professionnels et autres conçus en fonction de ce support : si vous êtes dans ce cas, il vous faut lire au minimum le chapitre « Installation ». Par curiosité vous pouvez, bien entendu, lire les chapitres d'intérêt général; celui sur le graphisme par exemple.

Ce sigle est placé en marge des paragraphes compréhensibles par tous.. Ce qui ne veut pas dire qu'ils sont sans intérêt !!

— Les débutants devront commencer par assimiler une compréhension minimum de leur machine en consultant attentivement le « guide de l'utilisateur » et le manuel Apple correspondant au langage informatique choisi : par exemple le manuel de référence APPLESOFT ou les manuels fournis avec le système Pascal.

Dans un second temps, ils pourront alors utiliser le logiciel Basic « PURPLESOFT » ou bien l'unité Pascal « PURPLEGRAPHICS » gérant les modes textes et graphiques propres à la carte « ève ».

Et bien entendu, nous n'interdisons pas à l'athlète de partager le savoir du brillant docteur d'Oxford..

— Les programmeurs confirmés seront sans doute intéressés par les paragraphes repérés par ce dernier sigle (informations techniques, langage machine..) !

A vous d'apprécier, ces symboles ne sont qu'indicatifs : nous vous souhaitons en tous cas une lecture agréable et de longues heures de passion en compagnie d'ève et de sa pomme !

Installation

Le texte

CHAPITRE III

Le texte

III-1 L'IMPLÉMENTATION DE LA VISUALISATION DU TEXTE SUR L'Apple *//e*

III-1.0 *Introduction*

En visualisation texte, la fonction 80 colonnes permet d'afficher à l'écran 40 caractères supplémentaires par ligne.

L'Apple *//e* standard affiche 24 lignes de 40 caractères pour un plein écran, la fonction 80 colonnes permet de doubler le nombre de caractères par ligne. Il est possible de passer facilement de la visualisation 40 à 80 colonnes et vice-versa.

Note : L'affichage sur un téléviseur standard par la prise PÉRITEL ne permet pas une lecture très satisfaisante des caractères 80 colonnes. Nous vous conseillons dans ce cas l'utilisation d'un moniteur monochrome de bonne qualité.

III-1.1 *Commutation du texte 80 colonnes*

a) Sous Pascal et CP/M

Au chargement du Pascal ou du CP/M, la commutation 80 colonnes est effectuée automatiquement par le système si toutefois votre carte est correctement installée.

En Pascal, il est possible d'utiliser les touches ↑ et ↓ en exécutant au préalable le programme SETUP situé sur la disquette APPLE 3 (se reporter au manuel "APPLE PASCAL OPERATING SYSTEM REFERENCE MANUAL").

Remarque : Pour éviter le passage automatique en 80 colonnes vous pouvez utiliser la « recette » suivante :

— Allumez votre Apple *//e* sans disquette dans les lecteurs.

— Enfoncez simultanément les touches <Ctrl> et <Reset>
puis relâchez, le symbole] de l'Applesoft apparaît alors en bas et à gauche de l'écran.
— Mettez vos disquettes dans les lecteurs.
— Tapez l'instruction suivante
POKE-16373,0:PR #6

Le système présentera un écran texte 40 colonnes comme si... la carte ève n'existait pas.

b) Sous Basic ou DOS 3.3

A l'allumage de votre Apple *//e*, l'affichage est en 40 colonnes; pour obtenir un affichage 80 colonnes, il suffit de taper PR#3 (comme toute commande Basic, PR est obligatoirement en lettres MAJUSCULES).

Cette instruction étant exécutée, il suffit de taper :

Esc 4 pour passer en 40 colonnes.
Esc 8 pour retrouver les 80 colonnes.

Esc désigne ici la touche Esc et non pas la suite de lettres E, s, c. Une telle séquence : touche Esc puis autre(s) caractère(s) est appelée séquence d'évasion.
Il en existe de nombreuses; reportez-vous au paragraphe III-1.2 pour plus de détails.

Il peut vous être agréable d'obtenir 80 colonnes dès l'allumage de votre Apple *//e* sous DOS 3.3; il vous suffit de modifier le programme d'initialisation en y incorporant les deux instructions suivantes :

```
1 D$=CHR$(4)
2 PRINT D$: « PR#3 »
```

III-1.2 *Modes de visualisation sous Basic et DOS 3.3*

Ces modes sont étroitement liés à l'activation de la FONCTION 80 colonnes. Lorsque cette fonction est activée, deux modes de visualisation sont possibles en texte : 24 lignes de 40 caractères ou bien 24 lignes de 80 caractères et dans ces deux cas le curseur est un rectangle blanc. Si la fonction 80 colonnes n'est pas activée, un seul mode de visualisation texte est possible : 24 lignes de 40 caractères. Le curseur est alors un damier clignotant.

Comme nous l'avons déjà vu, l'activation de cette fonction est réalisée par l'instruction Basic PR#3.

La désactivation quant à elle peut s'opérer de deux façons différentes :

— soit par la séquence d'évasion Esc Ctrl-Q
Le curseur en forme de damier clignotant apparaît en bas de l'écran surmonté d'un caractère « backslash » sur la ligne du dessus. Vous êtes de nouveau en 24 lignes de 40 caractères et tout programme BASIC en mémoire au moment de la désactivation est conservé intact.

— soit par la commande Ctrl-Reset
Vous retrouvez le curseur à damier mais vos lignes de texte sont devenues incompréhensibles. Un programme Basic en mémoire peut alors être irrémédiablement endommagé. Il convient donc de le sauver avant d'utiliser cette commande.
Si toutefois vous ne l'avez pas fait, ne répandez pas inutilement vos larmes et tapez LIST à tout hasard; vous avez malgré tout de grandes chances de le récupérer.

a) Aspect commun à tous les modes texte

Toutes les commandes Basic fonctionnent également dans tous les modes à l'exception de celles affectant la tabulation horizontale, l'aspect du curseur, celui des caractères et bien entendu le passage 40/80 colonnes et vice-versa.

b) Visualisation fonction non activée

Vous disposez de tous les caractères minuscules et majuscules du clavier en 24 lignes de 40 caractères.
Lorsque la touche ALPHA LOCK est enfoncée, vous obtenez l'ancienne visualisation Apple II et Apple II Plus et de ce fait une totale compatibilité avec les logiciels écrits pour ces machines. Les modes spéciaux obtenus par les commandes Basic FLASH (caractère clignotant) et INVERSE (caractère noir sur fond blanc) fonctionnent parfaitement dans ce mode mais n'affectent pas les caractères minuscules.
Vous avez également accès à toutes les séquences d'évasion hormis celles ayant trait à la fonction 80 colonnes (Voir Tableau 1).

c) Visualisation fonction activée

c.1 La tabulation horizontale en 80 colonnes

Dans tous les cas, l'instruction BASIC HTAB peut être suivie d'un nombre entre 1 et 255; si ce nombre est inférieur ou égal à 40, pas de problème : le PRINT suivant imprimera le mot à cette tabulation. Par contre, si la valeur suivant HTAB est supérieure à 40, l'expression s'affichera alors sur les lignes suivantes.
Ceci pose un problème en 80 colonnes : pour y remédier, vous pouvez remplacer HTAB N par POKE 36, N si N est supérieur à 40. Votre expression s'affichera alors où vous le souhaitez.

Essayez donc le programme suivant :

```
10 HTAB 55 : PRINT « TABULATION H=55 »
20 POKE 36,55 : PRINT « TABULATION EXACTE »
```

Tapez RUN et voyez par vous-même ce qu'il advient !

De même la tabulation horizontale obtenue par des virgules entre les expressions à imprimer ne fonctionne pas correctement en 80 colonnes.

Tapez Esc 4 puis exécutez ce programme :

```
10 PRINT «POMME», «CHAT», «CHAPEAU»
```

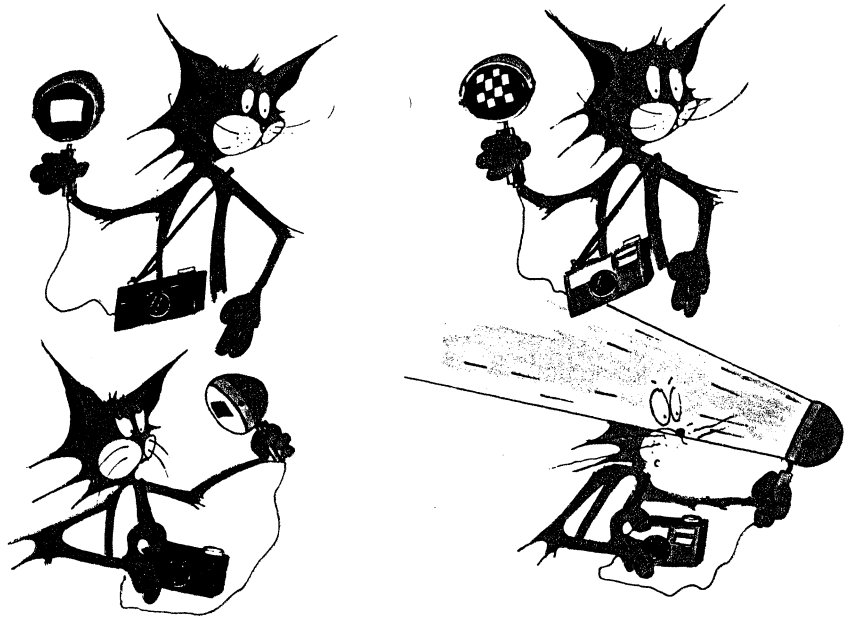
Tapez RUN puis Esc 8 puis de nouveau RUN
Tenez compte du résultat pour vos prochains programmes...

c.2 Les instructions FLASH, INVERSE et NORMAL

Lorsque la fonction 80 colonnes est activée, la commande INVERSE affecte aussi les caractères minuscules tandis que la commande FLASH n'a aucun effet.

Toutefois, méfiez-vous, il ne faut pas être en mode FLASH lorsque vous tapez PR#3 pour activer la fonction 80 colonnes; vous travaillerez alors en noir sur fond blanc et toute une partie de votre texte après LIST ou un RUN serait incompréhensible.

Pour annuler les commandes FLASH et INVERSE, tapez NORMAL.



c.3. Le mode majuscule restreint

Ce mode vous interdit les lettres minuscules à l'exception de celles contenues dans des guillemets si bien entendu la touche ALPHA LOCK n'est pas enfoncée. Dans le cas contraire, les lettres entre guillemets seront aussi des majuscules. On entre dans ce mode en tapant la séquence Esc R; on en sort par Esc T.

Essayez : — Tapez Esc R
— Relevez la touche ALPHA LOCK si elle était enfoncée
— Tapez print «j'ai compris»

Avez vous compris ?

— Tapez Esc T et recommencez : cette fois, le mot print reste en minuscules et le SYNTAX ERROR? bien connu apparaît.

Voici un mode bien pratique !

c.4 Comportement du curseur en mode d'évasion

Dès que vous frappez la touche Esc, le curseur change d'aspect : c'est toujours un rectangle blanc mais à présent il contient un signe +. Il restera ainsi jusqu'à ce que vous ayez terminé votre séquence d'évasion.

Lorsque vous tapez Esc puis les touches ↓ ↑ →et ← le curseur se déplace dans le sens des flèches sur l'écran. En particulier, les caractères sur lesquels il passe ne sont ni recopiés ni effacés de la mémoire.

Pour terminer votre séquence et revenir au mode normal, frappez simplement la barre d'espacement; le curseur redeviendra uniforme.

Pour plus d'information sur les séquences d'évasion, voir le tableau III.1.

c.5 Restriction en affichage 80 colonnes

Lorsque votre écran affiche 80 caractères par ligne, NE TAPEZ PAS PR#6 pour relancer le système. Repassez auparavant en 40 colonnes par Esc 4; tout se passera normalement et votre écran restera compréhensible.

De même, si vous voulez sortir du système Pascal pour revenir au DOS 3.3 : une procédure courante est d'insérer une disquette sous DOS et de taper H)alt au niveau le plus haut de commande. Changez d'habitudes, insérez votre disquette sous DOS et tapez Ctrl-Reset pour les raisons indiquées ci-dessus.

c.6 Passage 80/40 colonnes et vice-versa par programme

A cet effet, deux caractères de contrôle sont indispensables :

— le caractère Ctrl-R (Code ASCII décimal 18) passe en affichage 80 colonnes;
— le caractère Ctrl-Q (Code ASCII décimal 17) vous fait revenir en 40 colonnes.

Essayez donc ce petit programme :

```
10 PRINT «JE TREMBLE DE TOUS MES MEMBRES»
20 PRINT CHR$(17):PRINT CHR$(18)
30 GOTO 20
```

Tapez RUN puis Ctrl-C lorsque ce tremblement aura quitté l'écran.

Pour plus d'information sur les caractères de contrôle, voir le tableau III.2.

III-2 TEXTE COULEUR EN BASIC SANS LOGICIEL SPÉCIFIQUE

En Basic, si vous ne désirez pas charger le logiciel Basic PURPLESOFT et malgré tout utiliser les possibilités de votre carte « ève » en mode texte nous vous donnons ci-dessous le moyen de le faire. Notez tout d'abord que ce logiciel n'occupe qu'environ 1K de la mémoire de l'Apple *//e* et qu'il est donc tout à fait discutable de refuser la souplesse d'utilisation qu'il vous procure.

Vous pouvez remplacer la visualisation monochrome blanche des textes 40 et 80 colonnes par une visualisation monochrome verte en tapant l'instruction :

POKE-16197,0

Pour revenir à la visualisation blanche tapez :

POKE-16198,0

Vous pouvez, en texte 40 colonnes, choisir parmi 16 la couleur de chaque caractère et la couleur du fond. Placer les couleurs souhaitées dans deux variables, par exemple C (caractères) et F (fond) par leur numéro d'ordre :

0 noir	8 marron
1 magenta	9 orange
2 bleu foncé	10 gris 2
3 violet	11 rose
4 vert foncé	12 vert
5 gris 1	13 jaune
6 bleu	14 bleu turquoise
7 bleu clair	15 blanc

Pour passer en mode texte couleur il suffira d'exécuter :

POKE -16199,16*F+C

A partir de cet instant tous les caractères envoyés à l'écran apparaîtront en couleur C sur couleur F.

ATTENTION : Tous les caractères déjà sur l'écran à cet instant apparaîtront dans des couleurs indéterminées. Au premier accès à ce mode texte couleur, exécuter l'instruction HOME, tout l'écran se couvrira d'un fond de couleur F.

A chaque fois que vous changer les valeurs de C ou F, exécuter à nouveau l'instruction POKE -16199,16*F+C

Remarque : Tout « scrolling » de l'écran provoquera l'affichage de tous les caractères dans les couleurs C et F.

Pour quitter le mode texte couleur exécuter :

POKE -16200,16*F+C

Séquences d'évasion

Code	Fonction	Notes
ESC à	: Efface la fenêtre et remonte le curseur en 0,0	
ESC A	: Déplace le curseur d'un caractère vers la droite	
ESC B	: Déplace le curseur d'un caractère vers la gauche	
ESC C	: Descend le curseur d'une ligne	
ESC D	: Remonte le curseur d'une ligne	
ESC E	: Efface la fin de la ligne	
ESC F	: Efface le bas de la fenêtre	
ESC I	: Remonte le curseur d'une ligne	1
ESC ↑	: Remonte le curseur d'une ligne	1
ESC J	: Déplace le curseur d'un caractère vers la gauche	1
ESC ←	: Déplace le curseur d'un caractère vers la gauche	1
ESC K	: Déplace le curseur d'un caractère vers la droite	1
ESC →	: Déplace le curseur d'un caractère vers la droite	1
ESC M	: Descend le curseur d'une ligne	1
ESC ↓	: Descend le curseur d'une ligne	1
ESC R	: Entre dans le mode majuscule restreint	2
ESC T	: Quitte le mode majuscule restreint	2
ESC 4	: Commutation en 40 colonnes (écran effacé, curseur remonté)	2
ESC 8	: Commutation en 80 colonnes (écran effacé, curseur remonté)	2
ESC CONTROL-Q	: Désactivation de la fonction 80 colonnes	2

Notes :

1 : L'écran reste en mode d'évasion sans avoir à refrapper ESC
2 : Uniquement lorsque la fonction 80 colonnes est activée.

Tableau III-1

Caractères de contrôle

Caractère	Code ASCII	Fonction	Notes
CTRL-G	7	L'Apple *//e* produit un « bip » sonore	
CTRL-H	8	Déplace le curseur d'un espace vers la gauche	
CTRL-J	10	Descend le curseur d'une ligne	
CTRL-K	11	Efface le bas de la fenêtre	1
CTRL-L	12	Remonte le curseur et efface la fenêtre	1
CTRL-M	13	Curseur au début de la ligne suivante	
CTRL-N	14	Caractères normaux (blanc sur noir)	1,2
CTRL-O	15	Caractères inversés (noir sur blanc)	1,2
CTRL-Q	17	Commutation en écran 40 colonnes	1,2
CTRL-R	18	Commutation en écran 80 colonnes	1,2
CTRL-S	19	Arrêt de l'envoi des caractères à l'écran	1,3
CTRL-U	21	Désactivation de la fonction 80 colonnes	1,2
CTRL-V	22	L'écran descend d'une ligne	1
CTRL-W	23	L'écran remonte d'une ligne	1
CTRL-Y	25	Remonte le curseur en 0,0 (sans effacer)	1
CTRL-Z	26	Efface la ligne du curseur	1
CTRL-ç	28	Déplace le curseur d'un rang vers la droite	1
CTRL-§	29	Efface la fin de la ligne du curseur	1

Notes :

1 : La fonction 80 colonnes doit être activée
2 : Ne fonctionne que d'un programme
3 : Ne fonctionne que du clavier

Tableau III-2

Le graphisme

CHAPITRE IV

Le graphisme

IV-1 LES DIFFÉRENTS ÉCRANS GRAPHIQUES

Un dessin sur l'écran de votre téléviseur ou de votre moniteur n'est en fait qu'une juxtaposition de petites surfaces élémentaires : les cases d'une grille... Et votre dessin sera d'autant plus fin que le maillage de la grille est resserré.

Pour repérer une de ces cases de cette grille, on donnera sa position horizontale et sa position verticale; l'origine des positions devra également être connue (fig IV_1).

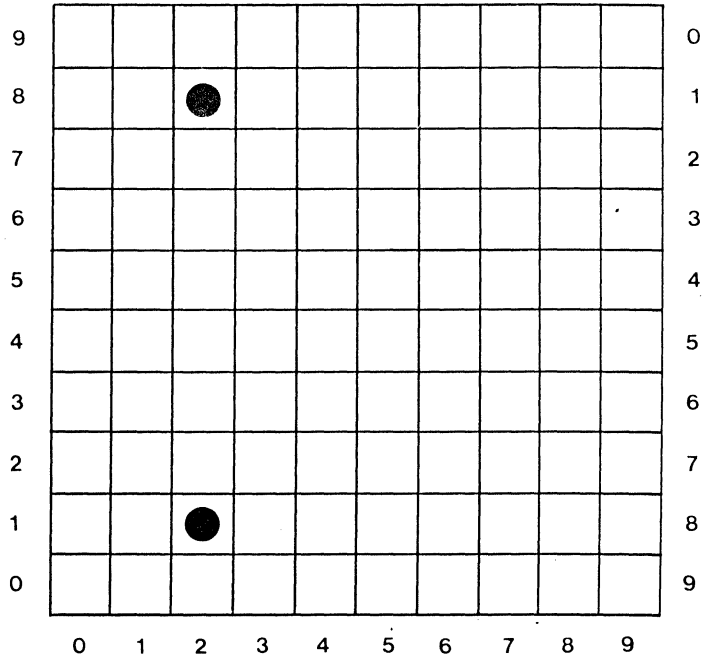


Figure IV-1

Case rouge en (2,8) avec une origine (0,0) en bas et à gauche de la grille.
Case verte en (2,8) avec une origine (0,0) en haut et à gauche de la grille.

On peut caractériser une grille par le nombre des positions horizontales et verticales qu'elle possède.
Plus ces nombres sont élevés, plus les dimensions d'une case sont réduites, plus la finesse du dessin est grande : en effet si le maillage de la grille varie, le moniteur ou le poste de télévision garde des dimensions fixes; vous ne l'avez jamais vu gonfler ni rétrécir !!

Ainsi nous pourrons attribuer à un écran graphique une définition horizontale : nombre de positions ou de cases le long d'une même horizontale (également nombre de colonnes).
De la même manière nous parlerons de définition verticale : nombre de positions ou de cases le long d'une même verticale (également nombre de lignes).

Un très grand nombre de lignes et de colonnes est bien entendu le meilleur garant de la précision d'un graphisme; ce n'est cependant pas une règle absolue : en effet il existe un nombre de cases au-delà duquel le gain de précision est illusoire car les appareils de visualisation sont limités dans leur faculté de reproduire un objet très petit. Telle est la différence fondamentale entre moniteur haute-résolution et un téléviseur classique. Le téléviseur arrive en limite de résolution pour la grille la plus resserrée que peuvent générer l'Apple *//e* et la carte «ève» : c'est en particulier le cas pour une définition horizontale de 560 points. Un moniteur haute résolution peut aller plus loin... si toutefois il mérite son appellation !

Mais revenons à ce qui en définitive vous intéresse le plus : de quels écrans graphiques disposez-vous donc sur Apple *//e* ?

IV-1.1 *Ecran standard Apple //e*

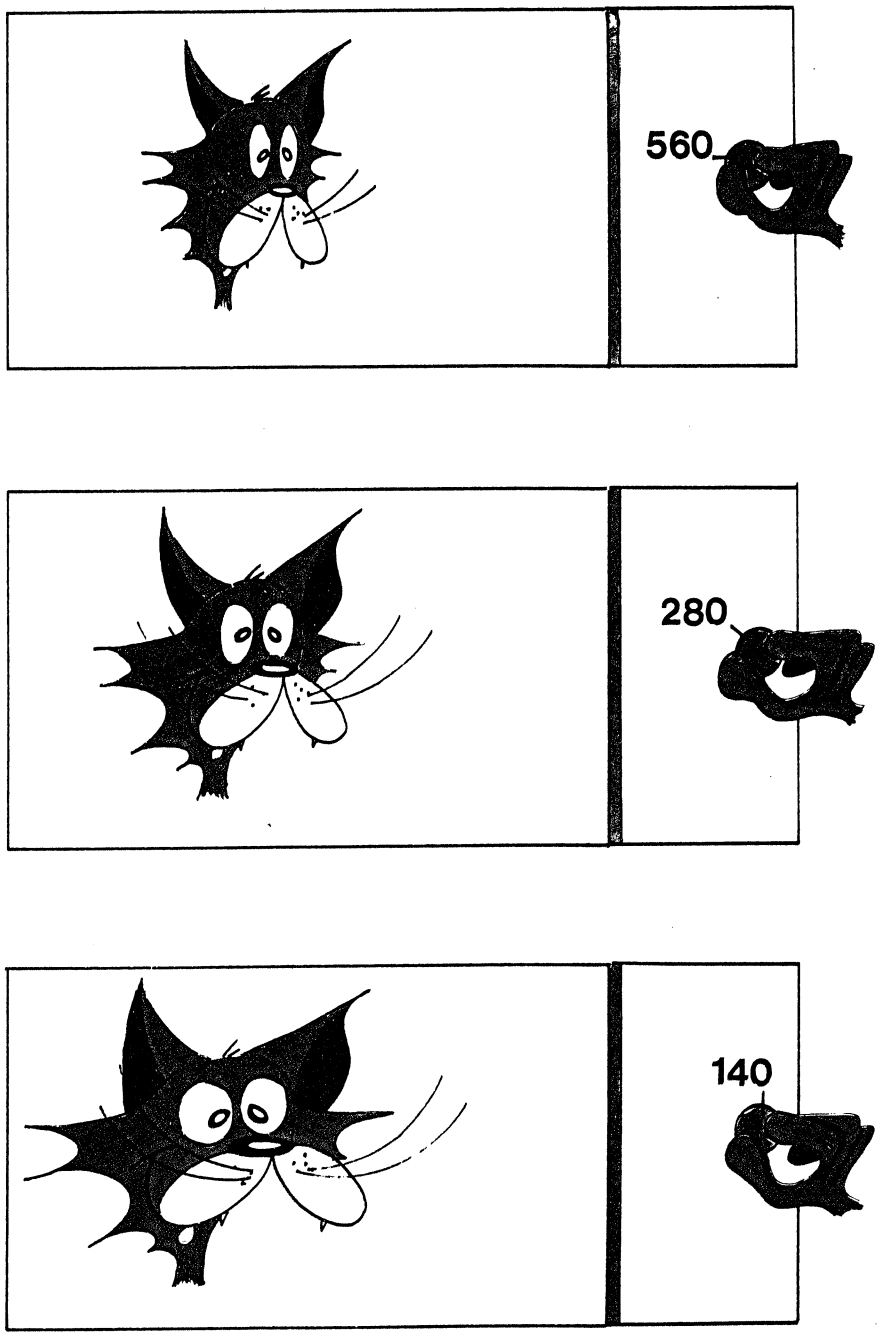
C'est une grille de 280 colonnes sur 192 lignes dont l'origine est située en haut et à gauche en Basic (point(0,0)) et en bas et à gauche en Pascal.

IV-1.2 *Ecrans* «ève»

Il en existe trois qui ont tous leur origine en bas et à gauche de l'écran quel que soit le langage employé :

— Grille 280x192
— Grille 140x192
— Grille 560x192

Notons au passage que toutes ces grilles ont une définition verticale de 192 lignes tandis que la définition horizontale varie du simple au quadruple : ainsi une même case est quatre fois plus large dans une grille 140x192 que dans une grille 560x192.



IV-2 PROBLÈME DE COLORIAGE D'UNE CASE

D'un point de vue informatique colorier une case revient à coder sa couleur et sa position. Dans le cas des 16 couleurs usuelles Apple, nous avons besoin de 4 bits pour en coder la totalité, la position de la case étant fournie par un calcul d'adresse.

Le système Apple exige 8K de mémoire pour une image en 2 couleurs coloriée case par case, 16K pour 4 couleurs et il exigerait 32K en 16 couleurs pour une grille de 280x192.

Etant donné que l'écran standard Apple *//e* vous offre déjà 6 couleurs pour 8K de mémoire, cela signifie donc que les couleurs ne sont pas disponibles case par case et qu'il y a "une astuce" !

Nous vous présentons ci-dessous les trois coloriages utilisés par l'Apple *//e* et ève.

IV-2.1 *Coloriage standard APPLE //e*

L'Apple utilise pour son graphisme une zone mémoire commençant à l'adresse 8192 et finissant en 16383 appelée page haute-résolution n° 1.

Chaque série de 7 points sur une même ligne de l'écran correspond à un octet de cette zone mémoire. Un point est allumé lorsque le bit correspondant est à 1. Une ligne est donc formée de 40 séries de 7 points.

Ainsi les 40 premiers octets de la page HR n° 1 (8192-16383) contiennent les informations de la ligne située en haut de l'écran.

Le bit de poids fort de chaque octet (bit n° 7) n'est pas visualisé mais il conditionne en partie la couleur des 7 points liés à l'octet dont il fait partie.

Ainsi, si le bit 7 est à 0, tout point isolé correspondant à cet octet et situé en colonne paire apparaît violet, en colonne impaire apparaît vert.

Si le bit 7 est à 1, les points isolés en colonne paire apparaissent bleus, ceux en colonne impaire apparaissent rouges.

Si deux points ou plus sont côte à côte sur une même ligne, ils apparaissent blancs même s'ils ne dépendent pas du même octet.

Pour mieux comprendre, tapez en Basic APPLESOFT le programme suivant :

```
10 HGR
20 HCOLOR=3
30 HPLOT 21,0 TO 122,0 TO 122,150 TO 21,150 TO 21,0
70 END
```

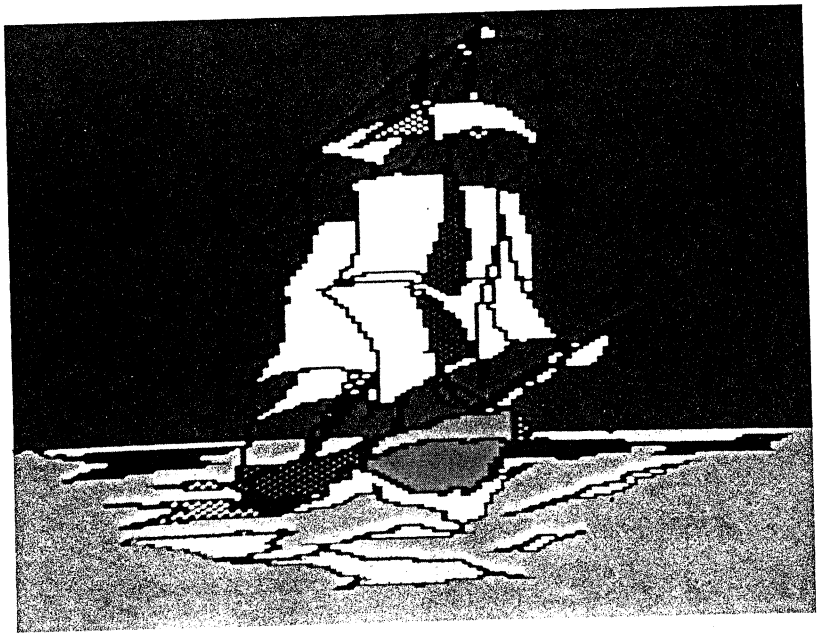
puis RUN

La ligne 20 met la couleur au blanc pour les instructions de dessin qui suivent; le résultat n'est pas tout à fait celui escompté; à vous de l'expliquer en sachant que dans ce cas (HCOLOR=3), le bit 7 de chaque octet est à 0.

Tapez à présent HCOLOR=7 (autre valeur de HCOLOR pour le blanc) puis RUN 30.

Autre résultat ! à votre avis pourquoi ?

Si vous désirez de plus amples précisions sur ce « mystérieux comportement », reportez-vous au manuel de référence de l'Apple *//e*.



mode HRAPPLE II

IV-2.2 *Coloriage par série de sept points* « foreground-background »

Comme dans le coloriage standard, les points sont regroupés 7 par 7 et chaque groupe de 7 points correspond à un octet (bits 0 à 6) de la page haute résolution n° 1.

Tout point allumé (bit à 1) sera de la couleur de « premier plan » (foreground), tout point éteint (bit à 0) sera de la couleur d'« arrière-plan » (background).

La définition de ces deux couleurs exige un octet de mémoire supplémentaire pour chaque série de sept points : il est fourni par la mémoire auxiliaire de votre carte ève.

Ainsi à une même adresse, l'octet en mémoire principale déterminera si les sept points qui lui correspondent sont allumés ou non ; en mémoire auxiliaire le demi-octet de poids fort donnera la couleur d'arrière-plan (16 possibilités), le demi-octet de poids faible la couleur de premier plan (également 16 possibilités).

Voici un exemple qui facilitera votre compréhension :

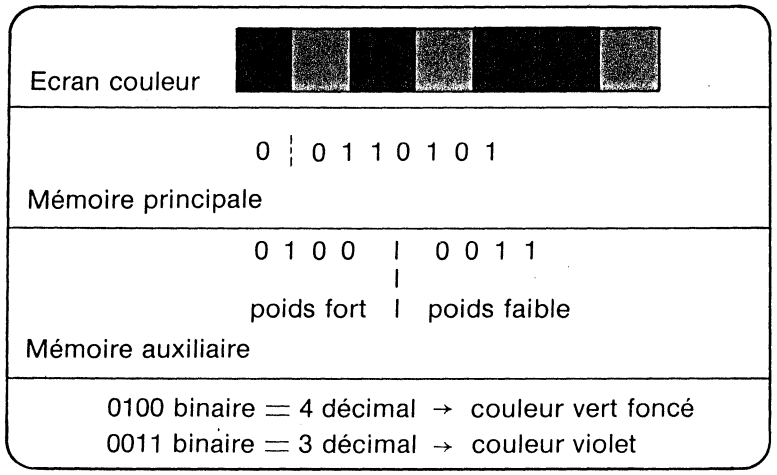


Figure IV-2

Nous précisons que pour chaque octet de la mémoire principale le point le plus à gauche correspond au bit de poids faible de cet octet.

IV-2.3 *Coloriage point par point*

Avec 16 couleurs disponibles, ce type de coloriage nécessite un demi-octet par point : il est donc bien plus gourmand en mémoire ; aussi les modes graphiques l'utilisant devront diminuer soit le nombre de couleurs disponibles, soit la définition horizontale afin que l'occupation d'une page graphique reste compatible avec les possibilités de votre Apple *II*e associé à la carte ève.

Il n'existe dans ce cas aucune limitation à l'existence d'une couleur où que ce soit dans la grille et la programmation s'en trouve d'autant plus facilitée.

IV-2.4 *Avantages et inconvénients des différents types de coloriage*

De toute évidence, le coloriage point par point est appelé à devenir le standard en la matière. Sa facilité d'emploi, son absence de surprises (!) le détermine en ce sens. Toutefois sa gourmandise en mémoire ne permettait pas jusqu'à présent de l'utiliser sur tous les types de machine.

Le coloriage standard Apple *II*e a pour lui l'indiscutable avantage de la compatibilité avec l'ancien Apple II et l'Apple II Plus et donc avec tous les programmes graphiques écrits pour ces machines.
Il exige une gestion plus rigoureuse de l'espace mémoire et rend la programmation plus délicate. Testez-le en détail grâce au programme du paragraphe IV-2.1 en faisant progresser HCOLOR de 1 à 7. Ce que vous verrez vous en apprendra plus que toute information écrite.

Enfin, le recouvrement couleur « foreground-background » est un moyen terme entre le point par point et le standard Apple. Bien employé, il permet de créer une image « fouillée » en 16 couleurs avec une résolution identique au mode standard.

IV-3 MODES GRAPHIQUES

Un mode graphique est défini par son nombre de colonnes et son nombre de lignes ainsi que par le type de coloriage qu'il emploie.

L'occupation mémoire d'une page graphique varie en fonction du mode choisi ; les modes « ève » occupent 16K-octets tandis que le mode standard Apple *II*e se contente de 8K-octets.

IV-3.1 *Mode standard Apple IIe*

Cette appellation recouvre en fait différents mode qui se distinguent par leur type de coloriage et leur interprétation par la carte ève :

—Mode HRAPPLEII: écran 280x192 et coloriage standard Apple *II*e.

—Mode HRSPEC1: même écran, coloriage standard Apple avec cette différence : les points isolés de couleur sur un fond blanc apparaissent noirs.

—Mode HRSPEC2: identique à HRSPEC1 mais en plus les points isolés sur fond noir apparaissent blancs.

—Mode HRDASH: les lignes horizontales de couleur violette, verte, rouge ou bleue apparaissent pointillées.

—Mode HRBW: écran 280x192 en noir et blanc.

HRSPEC1 et HRSPEC2 servent à obtenir une meilleure lisibilité des caractères graphiques ; HRSPEC1 affecte très peu le reste de l'image tandis que HRSPEC2 la blanchit.

Tous ces modes sont des dérivés du mode standard Apple *II*e et occupent au total 8K-octets de mémoire ; nous les appelons aussi « modes 8K ».

IV-3.2 *Modes « ève »*

—COL140: écran 140x192; coloriage en 16 couleurs point par point.

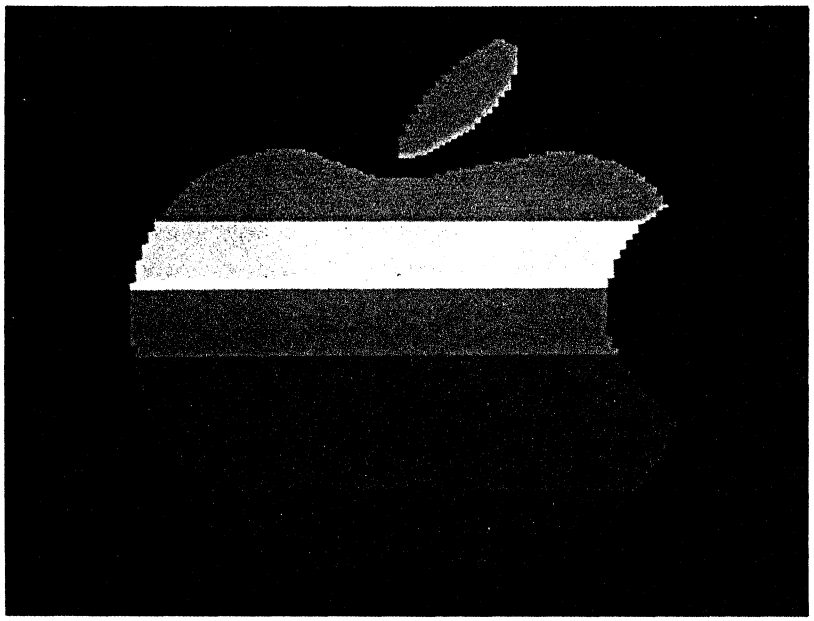
—COL280A: écran 280x192; coloriage point par point en 4 couleurs (noir, orange, vert, blanc).

—COL280B: écran 280x192; coloriage point par point en 4 couleurs (noir, bleu-clair, rose, jaune).

—CP280: écran 280x192; coloriage « foreground-background » en 16 couleurs par séries de 7 points horizontaux.

—BW560: écran 560x192; coloriage noir et blanc.

L'avantage de ces différents modes est d'offrir soit plus de couleurs, soit une meilleure résolution. L'inconvénient est qu'ils occupent deux fois plus de mémoire : 8K-octets en mémoire principale et 8K-octets en mémoire auxiliaire : ce sont des « modes 16K ».



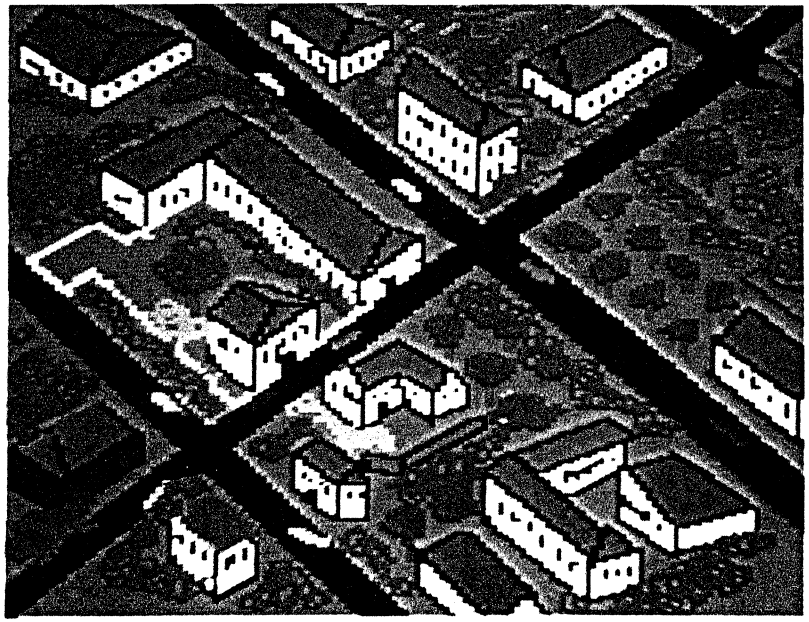
mode COL140

mode COL140

mode COL140

mode BW560

mode COL140



mode COL140

IV-4 LE CURSEUR GRAPHIQUE

Tout comme en texte, la page graphique dispose de son curseur; la différence est qu'il n'apparaît pas sur l'écran, mais sa position est présente en mémoire. Ce curseur peut être déplacé en laissant trace de son passage ou non de façon absolue ou relative : vous pouvez en fait le considérer comme la pointe d'un crayon se déplaçant sur une feuille de papier (votre écran). On associera au crayon une « couleur de crayon » et au papier une « couleur de papier ».

Plusieurs instructions permettent de gérer le déplacement de ce curseur.

IV-5 LES POINTS ET LES LIGNES

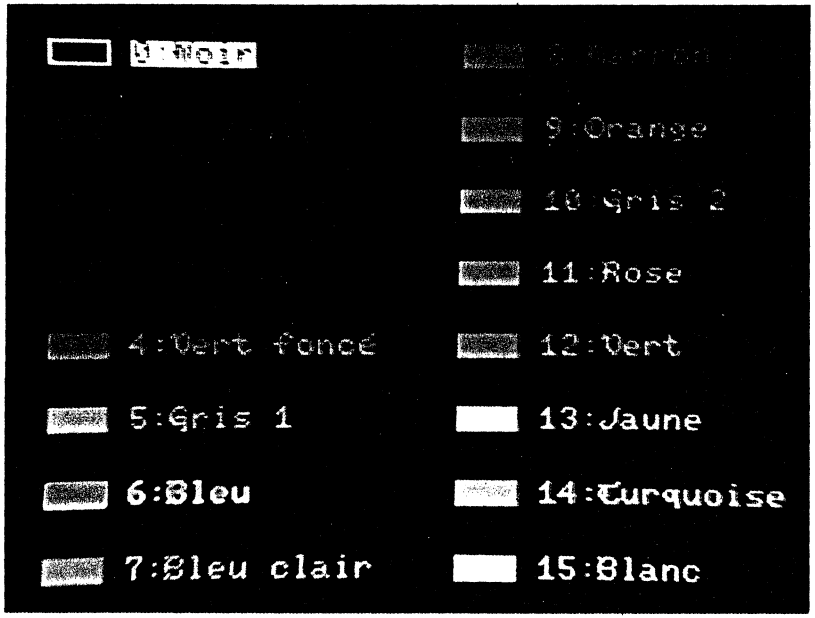
On peut tracer un point soit à l'endroit où se trouve le curseur graphique, soit à un endroit repéré par ses deux coordonnées d'écran X et Y.

Tracer une ligne peut être équivalent à tracer un certain nombre de points, mais il est possible de le faire directement.

La connaissance du point de départ et du point d'arrivée de la ligne suffit alors à la définir. On peut envisager synthétiquement ce tracé sous deux angles différents :
— tracé d'un point (X1, Y1) à un autre point (X2, Y2).
— tracé de la position du curseur à un point d'arrivée de coordonnées inconnues en déplaçant le curseur de DX colonnes et de DY lignes.

IV-6 LES COULEURS

Nous disposons au total de 16 couleurs différentes



mode CP280

et d'une « couleur transparente ».

A tout moment, lors des opérations de traçage et de remplissage, deux couleurs sont sélectionnées parmi les dix-sept spécifiées :
— la couleur de traçage (encre)
— la couleur de remplissage (papier)
Par défaut, la couleur de traçage est au blanc, et celle de remplissage au noir.

La façon la plus simple de dessiner est de tracer un point à l'écran dans la couleur de traçage, sans se préoccuper ni de la couleur, ni de la localisation du point antérieurement affiché. Ceci à l'intérêt de fournir une plus grande vitesse d'exécution, mais pose le problème de la protection de certaines parties de l'écran (protection locale) et de certaines couleurs (protection des couleurs). A ce dernier problème, deux techniques simples apportent une solution partielle :

IV-6.1 *Protection des couleurs*

Une table de commande des couleurs peut être éventuellement utilisée dans toutes les opérations de traçage et de remplissage. La couleur du point tracé dépend alors de la couleur du point préexistante et de celle qui vient se superposer dessus. Par défaut, la table de commande des couleurs n'est pas utilisée : le point tracé « sort » alors de la couleur de traçage prédeterminée (ou de remplissage, selon l'opération effectuée).

Supposons que vous dessiniez des arbres en employant les couleurs d'encre vert, vert-clair et marron; vous allez constituer au bout d'un temps plus ou moins long une forêt touffue.
Vous avez commencé votre dessin avec un fond bleu-clair et vous voulez à présent rajouter un château blanc derrière la forêt et devant le ciel.
Il suffira de positionner ainsi la table des couleurs :
BLANC sur VERT donne VERT
BLANC sur VERT-CLAIR donne VERT-CLAIR
BLANC sur MARRON donne MARRON
BLANC sur BLEU-CLAIR donne BLANC

A l'initialisation, la table des couleurs renvoie toujours un résultat tel que : couleur d'encre sur couleur d'écran donne couleur d'encre.
De même, la table des couleurs n'est pas toujours activée : à vous de le faire en sachant qu'un tracé est plus lent lorsqu'il tient compte de cette table. Par défaut la table de commande des couleurs est inactive.

IV-6.2 *Opérations logiques bit à bit*

L'option de transfert est un outil un peu plus compliqué et qui peut donner des résultats étonnants pour qui l'emploie sans la maîtriser.

Cette option permet de faire une opération logique bit à bit entre la couleur d'encre et celle de l'écran.

Les différentes opérations logiques sont :
— « ou inclusif » : encre ou écran
— « ou exclusif » : encre ou bien écran (opération très utile car elle permet de tracer une première fois sur un fond complexe et de réobtenir le fond en traçant une deuxième fois)
— « et » : encre et écran
— non encre (négatif)
— non encre ou écran
— non encre ou bien écran
— non encre et écran

Par défaut le transfert affiche la couleur encre quelle que soit la couleur écran.
Il est à noter que lorsque la table de commande des couleurs est active, encre désigne ici la couleur retournée par celle-ci.

La couleur affichée est déterminée ainsi :
COULEUR D'ENCRE → TABLE DES COULEURS → OPTION DE TRANSFERT → COULEUR AFFICHÉE

IV-7 LA FENÊTRE

De même qu'il peut être utile de protéger une couleur, il est parfois indispensable de soustraire toute une zone de l'écran à des tracés éventuels.

A cet effet vous disposez d'une fenêtre, elle peut être imaginée comme un rectangle fictif à l'intérieur duquel ont lieu toutes les opérations de traçage et de remplissage. Par exemple, si on trace une ligne en travers de l'écran, seule la portion de la ligne située à l'intérieur de la fenêtre sera visualisée; le reste de l'écran ne sera pas affecté par le tracé : il se trouve donc ainsi protégé.

N.B. : — La position courante du curseur n'est pas limité à l'intérieur fenêtre; en fait, l'abscisse ou l'ordonnée du curseur peuvent même prendre des valeurs négatives.

Par défaut, ce rectangle est l'écran graphique tout entier dans lequel vous dessinez. Bien entendu, vous pouvez modifier les dimensions de la fenêtre. Tout tracé à l'extérieur de cette fenêtre n'apparaît pas sur l'écran.

IV-8 LES FORMES

On peut afficher à l'écran à la position du curseur de petits dessins en deux couleurs. Ces petits dessins doivent être créés en mémoire sous la forme d'un tableau rectangulaire de bits. Les 0 seront affichés dans la couleur de remplissage et les 1 dans la couleur de traçage (éventuellement modifiées par la table de commande des couleurs et l'option de transfert). La hauteur et/ou la largeur de ces dessins sur l'écran peut être multipliée par 1, 2, 3...

IV-9 LES CARACTÈRES GRAPHIQUES

Il est possible d'afficher sur l'écran graphique des caractères et des chaînes de caractères de couleur à la position du curseur. Huit ensembles de caractères sont disponibles en permanence, il est même possible de remplacer un ou plusieurs d'entre eux en cours de programme. La hauteur et/ou la largeur de ces caractères sur l'écran peut être multipliée par 1, 2, 3...

Le logiciel Basic Purplesoft

CHAPITRE V

Le logiciel Basic Purplesoft

V-1 QUE VOUS OFFRE CE LOGICIEL ?

Pour une occupation minime de la mémoire principale (1K), PURPLESOFT gère la totalité des modes de visualisation, texte et graphique, spécifiques à la carte ève.

En plus de ces modes, PURPLESOFT vous permet de travailler conjointement avec les anciennes instructions texte et graphique Applesoft moyennant quelques précautions.

V-2 MISE EN ŒUVRE DE PURPLESOFT

Ce logiciel utilise la fonction & (Ampersand) du Basic Applesoft qui permet d'étendre l'ensemble des instructions de ce langage. Toutes les nouvelles instructions commenceront par ce symbole &.

Toutes les routines nécessaires à ces nouvelles instructions seront chargées en mémoire par le programme HELLO présent sur la disquette PURPLESOFT.

Ceci sera fait automatiquement si vous allumez votre Apple *//e* ou si vous effectuez un <Ctrl-pomme ouverte-Reset> avec PURPLESOFT comme disquette d'amorçage.

Note : Il est conseillé de dupliquer cette disquette avant toute utilisation et de garder l'original en lieu sûr. (Consulter à ce propos le manuel du DOS).

IMPORTANT : Pour bien comprendre les explications qui suivent, il est nécessaire de lire au préalable les chapitres III et IV.

V-3 COMMUTATION DES DIFFÉRENTS MODES TEXTE

Cette commutation s'effectue par l'instruction &TEXT n où n est un paramètre variant de 0 à 6.
Les caractères seront imprimés par l'ordre PRINT habituel du Basic Applesoft.

&TEXT 1 Ecran de 24 lignes de 40 caractères noir et blanc; curseur damier. L'écran est mis au noir et le curseur en haut à gauche.

&TEXT 2 Identique à &TEXT 1. Curseur bloc.

&TEXT 3 Ecran de 24 lignes de 40 caractères couleur; curseur damier. L'écran est rempli de la couleur fixée par &BACK=. Curseur en haut et à gauche. Les caractères seront imprimés dans la couleur fixée par &COLOR= sur un fond fixé par &BACK=.

&TEXT 4 Texte couleur (identique à &TEXT 3). Curseur bloc.

&TEXT 5 Ecran de 24 lignes de 80 caractères noir et blanc, l'écran est mis au noir et le curseur en haut à gauche.

&TEXT 6 Ecran de 24 lignes de 80 caractères noir et vert, l'écran est mis au noir et le curseur en haut à gauche.

&TEXT 0 Retour au dernier mode texte utilisé sans effacement de l'écran et sans positionnement du curseur. Cette instruction est conçue pour revenir d'un mode graphique vers un mode texte sans effacer l'écran texte; elle NE DOIT PAS être utilisée après un <Ctrl-Reset> , il faut alors repréciser le mode texte dans lequel on désire travailler.

IMPORTANT : Pour le bon fonctionnement de la visualisation, &TEXT 0 doit être utilisé à la place de l'instruction Applesoft : TEXT.

Le programme DEMO TEXTE sur la disquette PURPLESOFT vous donne un exemple d'utilisation de cette instruction. Exécutez le par RUN DEMO TEXTE puis listez-le par LIST.

V-4 COMMUTATION DES DIFFÉRENTS MODES GRAPHIQUES

Cette commutation s'effectue par l'instruction &GR n où n est un paramètre variant de 0 à 10.

&GR 1 Mode HRAPPLEII (mode standard Apple //e)

&GR 2 Mode HRSPEC1

&GR 3 Mode HRSPEC2

&GR 4 Mode HRDASH

&GR 5 Mode HRBW

Tous les modes ci-dessus sont des modes 8K, l'écran n'est pas effacé. Seules les instructions Applesoft de la haute résolution y opèrent : HPLOT, HCOLOR, DRAW, XDRAW, ..etc.. L'instruction Applesoft HGR permet alors d'effacer l'écran au noir.

IMPORTANT : L'instruction HGR du Basic Applesoft DOIT être impérativement précédée d'une instruction du type &GR n (n entre 1 et 5) correspondant à l'un des modes précédents. Par exemple effectuer un &GR1 pour obtenir la haute résolution Apple //e puis éventuellement un HGR pour effacer l'écran.

&GR 6 Mode COL140

&GR 7 Mode COL280A

&GR 8 Mode COL280B

&GR 9 Mode CP280

&GR 10 Mode BW560

Ces derniers modes sont des modes 16K, les instructions PURPLESOFT leur sont destinées : &PLOT, &COLOR=, &BACK=, &PRINT ..etc.. La fenêtre est fixée à tout l'écran graphique et remplie de la couleur fixée au préalable par &BACK=. La table des couleurs est déconnectée (mais pas modifiée) et l'option de transfert mise à zéro.

&GR 0 Retour d'un mode texte vers le dernier mode graphique utilisé.

Remarque : contrairement à &TEXT 0, l'instruction &GR 0 peut être utilisée après un <Ctrl-Reset> .

Note : Exécutez et listez DEMO GR16K pour un exemple commenté des modes graphiques étendus (modes 16K).

IMPORTANT : Interférences entre modes texte et graphique.

Lorsqu'un mode texte est visualisé, il est toujours possible de continuer à travailler sur l'écran graphique (non visualisé) avec les instructions &PLOT, &COLOR=, &CLEAR ,..etc.. L'inverse n'est PAS TOUJOURS possible, si l'un des modes graphiques COL140, COL280A, COL280B ou BW560 est visualisé et si le mode texte avait été choisi par &TEXT 2 ou par &TEXT 4 (texte 40 colonnes avec curseur bloc), l'instruction Applesoft PRINT ne doit pas être utilisée jusqu'au retour à une visualisation texte.

De même, il n'est pas toujours possible de placer 4 lignes de l'écran texte à la place du bas de l'écran graphique : les modes graphiques COL140,

COL280A, COL280B ou BW560 ne peuvent être « mixés » qu'avec un mode texte 80 colonnes (choisi au préalable par &TEXT 5 ou &TEXT 6). Cette dernière possibilité décrite dans la documentation Apple n'est pas toujours utile dans la mesure ou l'affichage de caractères est possible sur l'écran graphique (mode 16K) avec PURPLESOFT.

V-5 LES POINTS ET LES LIGNES

Les points et les lignes sont tracés par l'instruction &PLOT.

Pour ceux d'entre vous qui sont déjà familiarisés avec Applesoft, la syntaxe de l'instruction &PLOT est similaire à celle de l'instruction HPLOT.

&PLOT X,Y — trace un point de coordonnées X,Y dans la couleur définie auparavant par &COLOR.

&PLOT TO X,Y — trace une ligne de la position du dernier point tracé par une instruction &PLOT, à la position X,Y

&PLOT X1,Y1 TO X2,Y2 TO X3,Y3 TO .... TO XN,YN
trace une ligne brisée de point de départ X1,Y1 passant par X2,Y2 .. X3,Y3 .. et finissant en XN,YN.

Dans tous les cas, la couleur du tracé est définie par &COLOR=

IMPORTANT : Dans tous les modes graphiques étendus (16K) l'origine des coordonnées est située en bas et à gauche de l'écran, le point en haut à gauche ayant pour coordonnées 0,191.(ceci est différent dans le cas de l'écran graphique Haute Résolution standard en Basic).

Note : Exécutez et listez DEMO LIGNES pour un exemple pratique d'utilisation.

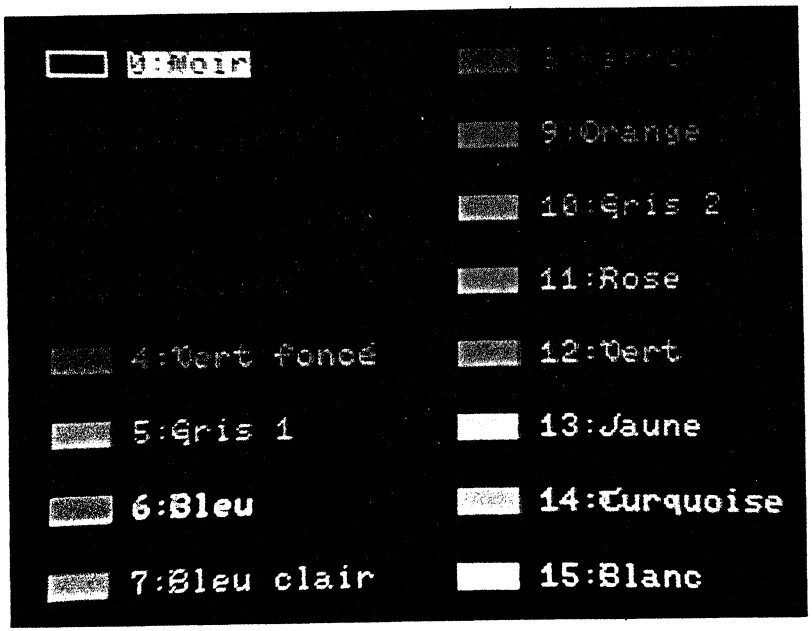
V-6 LES COULEURS ET LEUR CONTROLE

V-6.1 *Les instructions &COLOR= et &BACK=*

&COLOR=n avec n compris entre 0 et 16 fixe la couleur de traçage (encre).

&BACK=n avec n compris entre 0 et 16 fixe la couleur de remplissage (papier).

Consultez l'image qui suit pour la correspondance entre numéro et couleur.



mode CP280

Le rôle de la valeur 16 (couleur transparente) est un peu particulier :
— En graphisme si on effectue un &BACK=16 les formes et les caractères sont tracés sur le fond existant.
— En texte couleur une instruction &BACK=16 ou &COLOR=16 a pour effet de bloquer la répartition des couleurs sur l'écran mais n'interdit pas l'impression de caractères, par l'instruction Applesoft PRINT, qui s'inscrivent alors dans les couleurs et sur les fonds préexistants.

ATTENTION : Les 16 couleurs ne sont pas disponibles en totalité dans tous les modes graphiques.
COL280A, COL280B n'offrent que 4 couleurs : noir, orange, vert et blanc pour COL280A. Noir, bleu clair, rose et jaune pour COL280B. Quoiqu'il en soit la couleur 0 donne toujours le noir et un tracé dans la couleur 16 n'a aucun effet quel que soit le mode.
En BW560, toute couleur différente de 0 donne le blanc.

V-6.2 *L'instruction &SCRN(X,Y,C)*

L'instruction &SCRN(X,Y,C) retourne dans la variable C le numéro de la couleur du point de coordonnés X,Y.

Si le point de coordonnées X,Y n'est pas dans l'écran défini par le mode graphique dans lequel vous vous trouvez, &SCRN(X,Y,C) renvoie 16 dans la variable C.

V-6.3 *La protection des couleurs*

Vous disposez à cet effet de la table de commande des couleurs. Vous l'activerez par POKE-26074,1 et la désactiverez par POKE-26074,0.

Pour modifier la table de commande des couleurs de façon à ce que tout point tracé dans la couleur C sur un point déjà existant de la couleur E apparaisse de la couleur R, effectuez l'instruction :

```
POKE -26368+C+16*E,R
```

Pour réinitialiser la table des couleurs à ses valeurs par défaut, exécutez la séquence d'instructions suivante :

```
10 FOR C=0 TO 15
20 FOR E=0 TO 15
30 POKE -26368+C+16*E,C
40 NEXT E,C
```

Pour mieux comprendre le fonctionnement de la table de commande des couleurs exécuter puis examiner le programme DEMO TABLE.

V-6.4 *Opérations logiques*

Pour modifier l'option de transfert et la fixer à la valeur T comprise entre 0 et 7 exécuter l'instruction :

```
POKE-26112,T:CALL-25984
```

Options de transfert :

0 : encre
1 : encre ou écran
2 : encre ou bien écran
3 : encre et écran
4 : non encre
5 : non encre ou écran
6 : non encre ou bien écran
7 : non encre et écran

Pour plus de précisions, consulter le paragraphe IV-6.2.

V-7 LA FENÊTRE

Le positionnement de la fenêtre s'effectue par l'instruction

```
&WINDOW G,D,B,H
```

Vous fixez ainsi les quatre côtés de la fenêtre : G,D,B,H en représentant respectivement les limites gauche, droite, basse et haute.

Le remplissage de cette fenêtre d'une couleur C s'effectue par l'instruction suivante :

```
&BACK=C:&CLEAR
```

Ceci est également très utile pour tracer des rectangles pleins.

Le programme DEMO FENETRE vous donne un exemple d'utilisation dans ce sens.

Remarque : En mode CP280 (« foreground-background »), la limite gauche est automatiquement arrondie au multiple de 7 inférieur et la limite droite au multiple de 7 supérieur moins un. C'est une conséquence des limitations de ce mode.

V-8 LE CURSEUR

Un curseur est utilisé pour un positionnement dans les ordres &PRINT (impression de caractères graphiques) et &DRAW (traçage de tableaux bicolores).

Le positionnement de ce curseur sur l'écran graphique s'effectue par l'instruction &POS X,Y où X est l'abcisse de cette position et Y son ordonnée.

L'instruction &POS peut très bien conduire le curseur en dehors de la fenêtre courante, mais seuls apparaîtront les tracés situés à l'intérieur de cette fenêtre.

V-9 LES CARACTÈRES GRAPHIQUES

L'instruction &PRINT permet d'afficher à l'écran graphique des caractères et des chaînes de caractères, le premier caractère étant affiché au-dessus et à droite de la position du curseur définie au préalable par une instruction &POS.
Le curseur est ensuite repositionné automatiquement à la fin de cette chaîne de caractères.

ATTENTION : Aucune gestion d'écran n'est effectuée, ni passage à la ligne, ni « scrolling ». Il est donc nécessaire de repositionner le curseur pour changer d'ordonnée.

Les caractères sont affichés dans la couleur d'encre sur la couleur de remplissage, dans les mêmes conditions que les points ou les lignes (fenêtre, table de commande des couleurs...).

La syntaxe de l'instruction &PRINT est identique à celle de l'instruction Applesoft PRINT. Il est conseillé de la faire suivre d'un point virgule afin d'éviter l'inscription du caractère graphique correspondant au <Return> .

La taille des caractères peut être multipliée en hauteur et en largeur grâce à l'instruction &SCALE=L,H où L et H sont compris entre 1 et 128 et représentent respectivement les coefficients multiplicateurs de la largeur et de la hauteur.

Huit ensembles de caractères sont disponibles. L'instruction &CHR$ n avec n compris entre 0 et 7 permet de choisir.

0 : SYSTEM.CHARSET (Caractères américains)
1 : FRENCH.FONT (Caractères français)
2 : BYTE.FONT
3 : ITALIC.FONT
4 : GOTHIC.FONT
5 : ASCII.FONT
6 : LARGE.FONT
7 : GREEK.FONT

SYSTEM.CHARSET est choisi à l'initialisation.

V-9 LE TRACÉ DES FORMES

L'instruction &DRAW n permet de tracer la forme chargée en mémoire à l'adresse n.
Le coin en bas et à gauche de la forme sera la position courante du curseur graphique (le positionner éventuellement par &POS).

Une forme a la structure suivante :
1er octet : Hauteur de la forme
2e octet : Largeur de la forme
Les octets suivants contiennent la forme proprement dite représentée en mémoire par une suite de 1 et de 0.

Les bits à 1 représentent les points qui seront tracés dans la couleur d'encre et les bits à 0 les points qui seront tracés dans la couleur de remplissage.

Pour vous expliquer simplement comment coder votre forme en mémoire, nous allons recourir à un exemple :

Supposons que vous désiriez représenter la forme suivante :

```
  xxxxxx
  xx  xx
 xxxx  xxxx
xxxxxxxxxxxxxxx
 x        x
```

Ce dessin est inscrit dans un rectangle de hauteur H=5 (hexa 05) et de largeur L=15 (hexa 0F).

Vous allez à présent « lire » votre forme de gauche à droite et de bas en haut à partir du coin inférieur gauche du rectangle en remplaçant les blancs par des 0 et les x par des 1.

```
000011111100000
000011001100000
001111001111000
111111111111111
001000000001000
```

Cette opération va vous donner la suite binaire :

```
00100000000100011111111111111110011110011110000000110011000000000111111000000
```

Réorganisons cette suite par séries de huit bits en rajoutant éventuellement des 0 à la fin :

```
00100000
00010001
11111111
11111100
11110011
11000000
01100110
00000000
11111100
00000000
```

Chaque série de 8 bits est alors transformé en un octet en étant relu de droite à gauche :

binaire 00000100	hexa 04	décimal 4
binaire 10001000	hexa 88	décimal 136
binaire 11111111	hexa FF	décimal 255
binaire 00111111	hexa 3F	décimal 63
binaire 11001111	hexa CF	décimal 207
binaire 00000011	hexa 03	décimal 3
binaire 01100110	hexa 66	décimal 102
binaire 00000000	hexa 00	décimal 0
binaire 00111111	hexa 3F	décimal 63
binaire 00000000	hexa 00	décimal 0

Exécutez le petit programme suivant :

```
10 DATA 5,15,4,136,255,63,207,3,102,0,63,0
20 FOR I=0 TO 11:READ A:POKE768+I,A:NEXT I
30 &COLOR=15:&BACK=0:&GR7:&POS130,95:&DRAW768
40 END
```

Les instructions 10 et 20 placent cette forme aux adresses 768 à 779. L'instruction 30 trace cette forme dans le mode graphique COL280A.

ATTENTION : Choisissez avec soin l'adresse n car vous pourriez sans le vouloir écraser des données et tout votre programme pourrait être détruit.

Il est possible d'agrandir une forme en hauteur et/ou en largeur en exécutant l'instruction &SCALE=ML,MH où ML est le facteur entier multiplicateur de la largeur de la forme et MH un entier multiplicateur de la hauteur de la forme.

La séquence d'instructions : &SCALE=2,1
&DRAW n

dessinera une forme deux fois plus large que l'original. Multiplier les dimensions d'une forme diminuera malheureusement la précision de son tracé.

V-10 SAUVEGARDE ET CHARGEMENT DES IMAGES

Deux petits programmes GRLOAD et GRSAVE présents sur la disquette PURPLESOFT permettent de charger et de sauvegarder des images sur disque. Ils utilisent les routines BLOAD et BSAVE du DOS et sont de ce fait assez lents.
Vous pouvez inclure les sous-programmes utilisés, dans vos programmes ou vous en inspirez pour concevoir des routines plus adaptées et/ou plus performantes.

V-11 INFORMATIONS DIVERSES

V-11.1 *Utilisation de compilateurs Applesoft*

Certains compilateurs n'implémentent pas l'ampersand (&) et il sera donc nécessaire pour les utiliser de transcrire les instructions PURPLESOFT par des POKE et des CALL. Le chapitre « Eve et le langage machine » vous donnera toutes les informations nécessaires à cette traduction.

V-11.2 *Messages d'erreurs*

Trois messages d'erreur peuvent être retournés par PURPLESOFT :
— SYNTAX ERROR signale une erreur de syntaxe.
— ILLEGAL QUANTITY ERROR signale une erreur quantitative dans les paramètres.
— BAD SUBSCRIPT ERROR signale que le mode graphique basse résolution est sélectionné et qu'il est nécessaire d'effectuer tout d'abord un &GR n pour utiliser les autres instructions graphiques de PURPLESOFT. En particulier, il faut savoir qu'après un <Ctrl-Reset> l'Apple *IIe* sélectionne automatiquement le mode graphique basse résolution.

V-11.3 *Modification de &GR*

Le fonctionnement de l'ordre &GR peut être modifié par l'utilisateur pour éviter l'effacement d'écran et/ou la commutation en graphisme dans les modes 16K.

Modification de OPTCLEAN
L'instruction POKE-26066,0 permet d'éviter le nettoyage de l'écran provoqué par un &GR n (n compris entre 6 et 10).
L'instruction POKE-26066,1 permet de revenir au fonctionnement normal de &GR au niveau de l'effacement.

Modification de SWITXTGR
L'instruction POKE-26064,0 permet d'éviter la commutation de l'affichage de texte en graphisme provoquée par &GR n (n compris entre 1 et 10).
L'instruction POKE-26064,1 permet de revenir au fonctionnement normal de &GR au niveau de la commutation d'affichage.

V-11.4 *Routine CPCHANGE*

En mode CP280, l'instruction CALL-25996 a pour effet de remplacer la couleur de tous les points « allumés » (bits à 1) par la couleur de traçage et celle des points « éteints » (bits à 0) par celle de remplissage. (Cette routine n'a d'effet que dans la fenêtre graphique et utilise éventuellement la table de commande des couleurs et l'option de transfert).

V-11.5 *Routine TRANSCHAR*

L'instruction CALL-25932 permet de remplacer le charset courant sélectionné par l'instruction &CHR$n par un autre préalablement chargé à l'adresse ADDR.

— L'adresse -26112 devra contenir la partie basse de l'adresse ADDR et l'adresse -26111 la partie haute.

```
10 D$=CHR$ (4)
20 PRINT D$;"BLOAD SPANISH.FONT,A16384"
30 &CHR$7
40 POKE-26112,0:POKE-26111,64:CALL-25932
50 END
```

Ce petit programme charge en premier lieu l'ensemble des caractères espagnols en mémoire à l'adresse 16384=64x256+0, initialise le contenu des adresses -26112 et -26111 puis remplace l'ensemble des caractères numéro 7 par les caractères espagnols.

V-11.6 *MAXFILES et INT*

Une partie du logiciel PURPLESOFT est logée entre le DOS et ses « buffers » et peut donc être détruite par les deux instructions MAXFILES et INT. Ne pas utiliser ces instructions si vous désirez conserver PURPLESOFT en mémoire.

V-11.7 *Utilisation de la page graphique n° 2*

Toutes les instructions de PURPLESOFT débranchent le commutateur PAGE2. Si vous désirez utiliser la page graphique n° 2 en exécutant HGR2, notez bien que toute utilisation d'une instruction PURPLESOFT replacera la visualisation en page n° 1. Comme pour l'instruction HGR n'oubliez pas de sélectionner au préalable la Haute Résolution standard par &GR n avec n compris entre 1 et 5.

L'utilisation de la page graphique n° 2 est de toute façon rendue impossible si vous utilisez l'écran texte 80 colonnes.

Visualisation couleur en Pascal

L'extension mémoire 64 K

CHAPITRE VII

L'extension mémoire 64 K

VII-1 POURQUOI UNE MÉMOIRE AUXILIAIRE ?

L'Apple *//e*, dans sa version standard, contient déjà sur sa carte-mère 64 K-octets de mémoire utilisateur.
Le microprocesseur 6502 qui l'équipe ne peut effectivement adresser que cette quantité de mémoire; aussi peut-on se demander à quoi servent les 64 K-octets supplémentaires enfichés sur votre carte.

Pour le comprendre, imaginez une grande avenue où chaque maison sur le trottoir de droite ferait face à une maison sur le trottoir de gauche et où toutes deux porteraient le même numéro.
Cette avenue est très longue : les numéros vont de 0 à 65535 et en fait chaque maison représente un octet (64 K = 64x1024 = 65536 « maisons » ou octets).

Lorsque vous marchez sur le trottoir de gauche, vous êtes dans la mémoire principale : sur la plaque de l'Apple *//e*; sur le trottoir de droite vous vous trouvez dans la mémoire auxiliaire : celle de votre carte ève.

A présent, la question se pose de savoir comment passer d'un trottoir à l'autre : ceci est réalisé par l'intermédiaire de commutateurs logiques (les « switches » pour les initiés) que l'on peut actionner facilement par programme. Ils permettent de traverser l'avenue à des adresses précises.

Le fait d'écrire (en Basic par POKE adresse, valeur) ou de lire (par PEEK (adresse)) à certaines adresses mémoire indiquera à l'Apple *//e* ce qu'il doit porter à l'écran et où il doit lire/écrire les données qui lui sont nécessaires : mémoire principale et/ou auxiliaire.

L'utilisation de ces commutateurs logiques est réservée aux programmeurs AVERTIS et aux logiciels développés pour l'Apple *//e* par des professionnels. Une mauvaise utilisation peut rendre votre ordinateur fou !!
Vous n'auriez plus alors qu'à relancer le système en perdant tout espoir de retrouver vos données et votre programme...

La mémoire auxiliaire permet donc de doubler la mémoire de votre ordinateur. La plupart des programmes conçus pour Apple //e tirent avantage de cette mémoire additionnelle.

La visualisation du texte en 80 colonnes utilise 1K de cette mémoire auxiliaire, les modes graphiques étendus (double haute résolution) en utilisent 8K. Les logiciels graphiques fournis par Le Chat Mauve en utilise une partie (consulter à ce sujet le chapitre VIII, paragraphe 1.1.).

VII-2 FONCTIONNEMENT DES VISUALISATIONS ÉTENDUES

Toute « image » obtenue sur votre écran correspond en fait à l'interprétation par l'Apple //e et la carte « ève » d'une zone mémoire bien déterminée. Ces zones sont repérées par leurs adresses de début et leurs adresses de fin. Il existe deux zones différentes pour le texte 40 colonnes : page 1 et page 2; il en est de même pour la haute résolution standard. Le texte 80 colonnes et la haute résolution étendue utilisent à la fois, la mémoire principale et la mémoire auxiliaire; une seule zone mémoire est disponible dans ce cas.

Lire le tableau de ces zones. (Tableau VII-1)

Ainsi, en 80 colonnes, les caractères en colonnes impaires sont inscrits en mémoire principale et ceux des colonnes paires en mémoire auxiliaire.

VII-2.1 *Commutation des différents modes de visualisation*

Ces modes sont sélectionnés en écrivant ou en lisant à certaines adresses servant de point d'entrée aux commutateurs logiques. La plupart de ces commutateurs ont trois points d'entrée : l'un pour les connecter, l'autre pour les débrancher et le troisième pour lire leur état.

Consulter le tableau VII-2 sur lequel voici quelques remarques :
— MIXED et HIRES n'ont d'effet sur la visualisation que lorsque TEXT est débranché.
— Le branchement de 80STORE modifie la fonction de PAGE2. Lorsque 80STORE est branché, PAGE2 n'a plus d'effet sur la visualisation texte mais permet alors au processeur d'adresser, soit la zone écran texte en mémoire principale (page 1), soit la zone écran texte en mémoire auxiliaire (page 1X). En raison de cette utilisation particulière de PAGE2, il n'y a qu'une zone écran texte en 80 colonnes.
— Pour actionner certains commutateurs il est nécessaire d'effectuer une écriture aux adresses indiquées et non pas une lecture (ceci est indiqué dans le tableau).

En temps normal, ces commutateurs sont gérés par les instructions et commandes permettant de sélectionner un des modes de visualisation (par exemple en Basic: TEXT,GR,HGR, PR#3...).

Voici un petit programme Basic qui vous permettra de mieux comprendre : à cet effet, sachez que lorsqu'un commutateur logique est branché, la valeur lue est supérieure à 128 et inférieure à cette valeur dans le cas contraire.

```
10 FOR I=16360 TO -16353 : SW(I+16360)=PEEK(I):NEXT
20 PRINT «80STORE =»;SW(0)
30 PRINT «TEXT = »;SW(2)
40 PRINT «MIXED = »;SW(3)
50 PRINT «PAGE2 = »;SW(4)
60 PRINT «HIRES = »;SW(5)
70 PRINT «80COL = »;SW(7)
```

Exécutez ce programme à l'allumage de l'APPLE //e puis tapez PR#3 et recommencez; essayez encore après un Esc 4 puis après un HGR ...etc.

VII-2.2 *Utilisation directe des commutateurs d'affichage*

Pour actionner les commutateurs logiques 80STORE et PAGE2, il est nécessaire de passer par un programme. En effet, dès que l'APPLE //e affiche du texte à l'écran en mode 80 colonnes, il utilise ces commutateurs logiques. Votre action ne dure ainsi que le temps de votre programme.

a) Pour le texte 80 colonnes

La page 1 du texte en mémoire principale a son « reflet » en mémoire auxiliaire aux mêmes adresses, appelé page 1X texte.

Pour stocker directement des données en page 1X, il faut brancher le 80STORE en écrivant à l'adresse 49153 (INTEGER BASIC : -16383, HEXA: $C001) puis actionner PAGE2 en écrivant ou en lisant à l'adresse 49237 (INTEGER BASIC: -16299, HEXA: $C055).

Pour un petit essai, activer la fonction 80 colonnes par PR#3 et exécutez le programme suivant :

```
10 HOME
20 FOR I=1 TO 10:READ A$:POKE 1462+I,ASC(A$)+128:
   GOSUB 1000:NEXT
30 POKE 49237,0 : REM branchement de PAGE2
40 FOR I=0 TO 11:READ A$:POKE 1462+I,ASC(A$)+128:
   GOSUB 1000:NEXT
50 DATA E,O,I,A,I,N,E," ",A,E,1
60 DATA M,M,R,S,T,O," ",N.,P,G," ",X
70 END
1000 FOR T=1 TO 200:NEXT:RETURN
```

Les premiers caracteres qui apparaissent en colonnes impaires sont inscrits en mémoire principale, les suivants en mémoire auxiliaire aux mêmes adresses.

b) La haute résolution étendue

En mode graphique 560 points noir et blanc (double haute résolution), la répartition mémoire est similaire à celle employée pour la haute résolution standard. Dans chaque octet le bit de poids fort est ignoré, un octet correspond donc à 7 points affichés.

On visualise 40 paires de 14 points par ligne, les 7 premiers de chaque paire étant en mémoire auxiliaire (page 1X graphique), les 7 autres en mémoire principale (même adresse). Ainsi les points 0 à 6, 14 à 20.... 545 à 552 d'une même ligne sont implantés en page 1X graphique. Les points 7 à 13,21 à 27.... 553 à 559 sont eux en page 1 graphique (mémoire principale).

Pour stocker des données directement dans la page 1X graphique, branchez 80STORE et HIRES puis PAGE2.

Note : le branchement des deux commutateurs 80STORE et HIRES est nécessaire pour modifier le rôle de PAGE2 au niveau du graphisme haute résolution. Le commutateur d'affichage se transforme alors en un commutateur mémoire principale-auxiliaire sur les zones mémoire $2000-$3FFF et $400-$7FF.

Pour visualiser un mode graphique double haute résolution il faut que 80COL et HIRES soit branchés, que TEXT soit débranché, et de plus que le commutateur AN3 soit débranché (AN3 est branché à l'allumage de l'Apple *//*e).

Commutateur AN3 :	« on » :	CO5F (hexa)	-16289 (décimal)
(« annunciator 3 »)	« off » :	CO5E (hexa)	-16290 (décimal)

En ce qui concerne la visualisation couleur, d'autres commutations peuvent être nécessaires pour choisir parmi les différents modes graphiques étendues de la carte ève (consulter à ce sujet le chapitre IX, paragraphe 1).

Note : le logiciel fourni avec la carte prend en charge toutes les commutations d'affichage.

Les pages graphiques 1 et 1X sont utilisées par tous les modes de visualisation étendue de la carte ève.

VII-3 ACCÈS UTILISATEUR A L'EXTENSION MÉMOIRE

Une série de commutateurs permettent un accès sélectif à la mémoire auxiliaire.

Le tableau VII-3 présente l'ensemble de ces commutateurs. Les figures VII-1 à VII-4 expliquent le fonctionnement de ces commutateurs. Chaque figure présente l'action d'un commutateur marqué d'un X. La mémoire active est celle présentée au microprocesseur, indépendamment de l'état de ce commutateur. Les parties « commutées » sont celles soumises à l'action de ce commutateur qui en présente au microprocesseur, soit la mémoire principale, soit la mémoire auxiliaire.

Les commutateurs RAMRD et RAMWRT permettent de valider indépendamment les fonctions d'écriture et de lecture dans les deux espaces mémoire $0200-$BFFF. A ces adresses, il est donc possible de lire en mémoire principale et écrire en mémoire auxiliaire (ou vice-versa).

Les commutateurs d'accès au pages d'écran ont priorité sur les commutateurs RAMRD et RAMWRT. Lorsque 80STORE est branché, les deux espaces mémoires $400-$7FF sont sélectionnés indépendamment de l'état de RAMRD et RAMWRT par l'action de PAGE2. Si de plus HIRES est branché, il en est de même pour les deux espaces mémoires $2000-$3FFF.

La mémoire morte contenant l'interpréteur Applesoft est en parallèle avec la « bank switched memory » (zone mémoire $D000-$FFFF). Son activation est indépendante de l'état des commutateurs du tableau VII-3.

Pour plus de précisions, se procurer l'« Apple *//*e Reference Manual ». Ce manuel est indispensable pour utiliser toutes les possibilités de l'Apple *//*e au niveau de la gestion mémoire.

VII-4 LES SOUS-PROGRAMMES DE LA MÉMOIRE AUXILIAIRE

Deux sous-programmes implantés en ROM (mémoire morte) sur votre Apple *//*e facilitent l'utilisation de la mémoire auxiliaire.

Installés dans l'espace mémoire $C300-$C3FF réservé au slot 3 leur accès reste indépendant de l'état des commutateurs de mémoire décrits dans ce chapitre.

VII-4.1 *La routine AUXMOVE*

Cette routine située en $C311 permet de transférer des blocs de données de la mémoire principale vers la mémoire auxiliaire et vice-versa, mais seulement dans l'espace mémoire $200-$BFFF.

ATTENTION : Lorsque 80STORE est branché certains espaces mémoires apparaissent à cette routine comme étant à la fois en mémoire principale et en mémoire auxiliaire. En effet cette routine utilise les commutateurs RAMRD et RAMWRT sur lesquels 80STORE prend priorité. Revoir éventuellement à ce sujet les explications du paragraphe VII-2 et les figures VII-1 et VII-2.

Cette particularité peut être bien utile, vous pourrez consulter à ce propos, au paragraphe VII-4, le programme TRANSFOTO de sauvegarde des images « 16K » (haute résolution étendue) en mémoire auxiliaire.

Avant tout appel de cette routine (en assembleur JSR $C311) il faudra :

— positionner le bit de retenue (carry) du microprocesseur en fonction du sens de transfert choisi, à 1 (par SEC en assembleur) pour un transfert de la mémoire principale vers la mémoire auxiliaire, à 0 (par CLC en assembleur) pour l'autre sens ;

— charger l'adresse de départ des données à transférer en $3C (partie basse de cette adresse) et $3D (partie haute). Le pointeur $3C/$3D est appelé A1L/A1H ;

— charger l'adresse du dernier octet à transférer en $3E (partie basse de cette adresse) et $3F (partie haute). Le pointeur $3E/$3F est appelé A2L/A2H ;

— charger l'adresse de destination du bloc de données en $42 (partie basse de cette adresse) et $43 (partie haute). Le pointeur $42/$43 est appelé A4L/A4H ;

Remarque importante : En Pascal, les pointeurs A1, A2 et A4 en page zéro sont utilisés par le système. Pour pouvoir utiliser AUXMOVE, il est donc indispensable de les sauvegarder pour les restaurer ensuite après le transfert.

Le programme assembleur TRANSFOTO dont le code est présent sur la disquette Basic DOS 3.3 PURPLESOFT apporte un très bon exemple en Basic de maniement de la routine AUXMOVE. Son listing est donné au paragraphe VII-5.

Liste des paramètres pour AUXMOVE

nom	adresse	fonction
carry		1 : mémoire principale vers mémoire auxiliaire 0 : mémoire auxiliaire vers mémoire principale
A1L A1H	$3C $3D	partie basse de l'adresse du premier octet à transférer partie haute de l'adresse du premier octet à transférer
A2L A2H	$3E $3F	partie basse de l'adresse du dernier octet à transférer partie haute de l'adresse du dernier octet à transférer
A4L A4H	$42 $43	partie basse de l'adresse de destination (1er octet) partie haute de l'adresse de destination (1er octet)

VII-4.2 *La routine XFER*

La seconde routine permet elle de transférer le contrôle à un sous-programme résidant en mémoire auxiliaire ou de le «rendre» à un sous-programme de la mémoire principale.

Il est alors nécessaire de préciser si la mémoire principale « passe la main » à la mémoire auxiliaire ou vice-versa, il faut aussi,préciser si le programme auquel on donne le contrôle utilisera la pile et page zéro en mémoire principale ou celles de la mémoire auxiliaire, enfin il faudra évidemment donner l'adresse du début de ce sous-programme.

— Le positionnement de la retenue (carry) détermine le sens dans lequel s'effectue le transfert de contrôle : si la retenue est à 1 (SEC), la mémoire principale passe le contrôle à la mémoire auxiliaire. Si la retenue est à 0 (CLC) c'est alors la mémoire auxiliaire qui transfère le contrôle à la mémoire principale.

— Le positionnement de l'indicateur de dépassement (overflow) détermine quant à lui la pile et la page zéro utilisées par le sous-programme : si cet indicateur est à 1, la pile et la page zéro de la mémoire auxiliaire seront utilisées. S'il est à 0, la pile et la page zéro de la mémoire principale seront utilisées.

— Enfin l'adresse du sous-programme auquel on donne le contrôle doit être placée en $3ED pour la partie basse et $3EE pour la partie haute.

Remarque très importante : il est préférable de transférer le contrôle à votre sous-programme par l'instruction JMP XFER (JMP 0C314) et non par un CALL (JSR 0C314), ceci essentiellement pour des raisons de gestion de pile..... Au moment de l'appel, une pile est activée mais au retour... (à méditer, croyez-nous, les auteurs de ces quelques lignes en ont fait la triste expérience).

Liste des paramètres pour XFER

nom et adresse	fonction
Carry	0 : mémoire principale vers mémoire auxiliaire 1 : mémoire auxiliaire vers mémoire principale
Overflow	1 : page zéro et pile en mémoire auxiliaire 0 : page zéro et pile en mémoire principale
$3ED $3EE	partie basse de l'adresse de début du programme partie haute de l'adresse de début du programme

VII-5 EXEMPLES D'UTILISATION DE LA MÉMOIRE AUXILIAIRE

VII-5.1 *Exemple d'utilisation de la routine AUXMOVE en Basic*

Le programme listé ci-dessous s'appelle TRANSFOTO et permet le transfert d'images 16K utilisant les pages graphiques 1 et 1X aux adresses $4000 à $7FFF (buffer1) ou aux adresses $8000 à $BFFF (buffer2) en mémoire auxiliaire. Les transferts peuvent s'effectuer dans les deux sens. Il est ici écrit pour fonctionner en Basic, et le code est fourni sur la disquette Basic DOS 3.3 PURPLESOFT.

Par exemple, l'appel du petit sous-programme suivant permettra si I=1 de sauver les pages graphiques 1 et 1X dans le buffer 1 et si I=2 de charger le contenu du buffer 2 en page 1 et 1X.

```
10 D$=CHR$(4)
20 PRINT D$;« BLOAD TRANSFOTO,A$300 »
30 POKE 768,I:CALL 769
40 RETURN
```

Listing de TRANSFOTO

Si CHOIX (adresse $300 ou 768 dec) contient :

0 : les adresses de $4000 a $7FFF en aux seront chargées en PAGE1 et 1X

1 : les PAGE1 et 1X seront sauvegardées de $4000 à $7FFF en aux

2 : les adresses de $8000 a $BFFF en aux seront chargées en PAGE 1 et 1X

3 : les PAGE1 et 1X seront sauvegardées de $8000 à $BFFF en aux après l'appel du sous-programme TRANSFOTO en $301 ou 769 dec

```
          .ORG 300

AUXMOVE   .EQU 0C311

A1L       .EQU 3C     ;adresse de départ, partie basse
A1H       .EQU 3D     ;adresse de départ, partie haute

A2L       .EQU 3E     ;dernière adresse a transférer, partie basse
A2H       .EQU 3F     ;derniere adresse a transférer, partie haute


A4L       .EQU 42     ;adresse de destination, partie basse
A4H       .EQU 43     ;adresse de destination, partie haute


CHOIX     .BYTE 00

TRANSFOTO PHA             ;sauvegarde A
          TXA
          PHA             ;sauvegarde X
          PHP             ;blocage des interruptions
          SEI


          LDA 0C01D
          PHA             ;sauvegarde de l'état de HIRES
          LDA 0C01B
          PHA             ;sauvegarde de l'état de 80STORE
          LDA 0C01C
          PHA             ;sauvegarde de l'état de PAGE2

          LDX CHOIX       ;CHOIX dans X

          STA 0C057       ;HIRES on
          STA 0C001       ;80STORE on

          LDA #0
          STA A1L
          STA A4L         ;initialisation des parties basses des pointeurs
          LDA #0FF        ;A1, A2, A3
          STA A2L
```

```
            LDA START1,X    ;chargement dans A1H de la partie haute de l'adresse
            STA A1H         ;de départ à transférer
            LDA END1,X      ;chargement dans A2H de la partie haute de la
            STA A2H         ;dernière adresse a transférer
            LDA DEST1,X     ;chargement dans A4H de la partie haute de l'adresse
            STA A4H         ;de destination



            STA 0C054       ;accès PAGE1

            TXA
            ROR A           ;on fait tomber le bit de parité de X dans la carry
            JSR AUXMOVE     ;transfert PAGE1 <--> AUX

            LDA #0
            STA A1L
            STA A4L         ;réinitialisation des parties basses des pointeurs
            LDA #0FF        ;A1, A2, A4
            STA A2L

            LDA START2,X    ;chargement dans A1H de la partie haute de l'adresse
            STA A1H         ;de départ à transférer
            LDA END2,X      ;chargement dans A2H de la partie haute de la
            STA A2H         ;dernière adresse à transférer
            LDA DEST2,X     ;chargement dans A4H de la partie haute de l'adresse
            STA A4H         ;de destination

            STA 0C055       ;accès PAGE1X

            TXA
            ROR A           ;on fait tomber le bit de parité de X dans la carry
            JSR AUXMOVE     ;transfert PAGE1X <--> AUX

            PLA
            BMI OLDSTORE
            STA 0C054       ;remise en état de PAGE2

OLDSTORE    PLA
            BMI OLDHIRES
            STA 0C000       ;remise en état de 80STORE

OLDHIRES    PLA
            BMI OLDX
            STA 0C056       ;remise en état de HIRES

OLDX        PLP
            PLA
            TAX
            PLA             ;remise en état X et A
            RTS


          ;tableau des adresses hautes

START1      .BYTE 40,20,80,20
START2      .BYTE 60,20,0A0,20

END1        .BYTE 5F,3F,9F,3F
END2        .BYTE 7F,3F,0BF,3F

DEST1       .BYTE 20,40,20,80
DEST2       .BYTE 20,60,20,0A0
```

VII-5.2 *Exemple d'utilisation de la routine AUXMOVE en Pascal*

```
procedure MAINAUX (var X;TAILLE,ADDR:
INTEGER);external;

procedure AUXMAIN (ADDR,TAILLE:INTEGER;
var X);external;


(* à placer dans le programme utilisateur *)
```

Nous donnons ici un exemple de procédures externes bien pratique permettant la sauvegarde de variables en mémoire auxiliaire puis leur récupération. En tant que procédures externes, il faut pour les utiliser compiler le programme, puis lier le code obtenu avec le fichier TRANSPORT.CODE à l'aide du LINKER afin d'obtenir le code exécutable.(TRANSPORT.CODE contient le code des procédures MAINAUX et AUXMAIN et est disponible sur PURPLE:)

Ces procédures peuvent être fatales pour votre programme et vos données si elles sont mal employées. En particulier, aucun test de validité sur les paramètres passés n'est effectué.

1er cas. Vous voulez sauvegarder en mémoire auxiliaire une variable X de n'importe quel type à l'adresse ADDR en mémoire auxiliaire. Vous ferez alors l'appel de procédure suivant :

MAINAUX(X,SIZEOF(X),ADDR);

2e cas. Vous voulez récupérer dans une variable X les données disponibles à l'adresse ADDR en mémoire auxiliaire. Vous ferez alors l'appel de procédure suivant :

AUXMAIN(ADDR,SIZEOF(X),X);

La gestion des adresses en mémoire auxiliaire est laissée entièrement à l'utilisateur, ADDR est une adresse qui DOIT être comprise entre $200 et $BFFF (attention éventuellement aux pages graphiques et à la position des commutateurs associés).

ADDR en décimal est donc soit compris entre 512 et 32767 ou compris entre -32768 et -16385 puisqu'un entier en Pascal est nécessairement compris entre -32768 et 32767.

Listing des procédures externes MAINAUX et AUXMAIN

```
A1L .EQU 3C
A1H .EQU 3D
A2L .EQU 3E
A2H .EQU 3F
A4L .EQU 42
A4H .EQU 43

AUXMOVE  .EQU 0C311

SAUVE    .EQU 0

         .PROC MAINAUX,3

         .DEF TRANSPORT

          LDY #1                ;sens de transfert

TRANSPORT PLA
          STA SAUVE
          PLA                   ;sauvegarde retour Pascal



          STA SAUVE+1

          LDX #7
$01       LDA A1L,X
          STA SAUVE+2,X         ;sauvegarde A1,A2 ,A4 (A3 est sauvé inutilement)
          DEX
          BPL $01

          CLD                   ;mise a 0 du mode décimal

          PLA
          STA A4L               ;adresse de destination dans A4
          PLA
          STA A4H

          PLA
          TAX
          STA PLUSB+1           ;on dépile le nombre d'octets à transférer
          PLA
          STA PLUSH+1

          TXA
          BNE $02
          DEC PLUSH+1           ;nbre d'octets-1 à transférer dans PLUSB+1 et
$02       DEC PLUSB+1           ;PLUSH+1
          PLA
          STA A1L
          CLC
PLUSB     ADC #0
          STA A2L               ;dépilement de l'adresse source à
          PLA                   ;transférer dans A1 et calcul de la dernière
          STA A1H               ;adresse à transférer que l'on sauve dans A2
PLUSH     ADC #0
          STA A2H

          PHP                   ;empêche les interruptions
          SEI

          TYA                   ;carry à 0 transfert vers la mémoire principale
          ROR A                 ;carry à 1 transfert vers la mémoire auxiliaire

          JSR AUXMOVE           ;éxécution du transfert

          PLP

          LDX #7
$01       LDA SAUVE+2,X         ;restauration de A1,A2,A3 et A4
          STA A1L,X
          DEX
          BPL $01

          LDA SAUVE+1
          PHA
          LDA SAUVE             ;retour Pascal
          PHA
          RTS                   ;fin de TRANSPORT

         .PROC AUXMAIN,3

         .REF TRANSPORT

          LDY #0                ;sens de transfert
          JMP TRANSPORT

         .END
```

VII-6 IDENTIFICATION DU MATÉRIEL

Ce paragraphe s'adresse aux concepteurs de logiciels, il leur donne deux exemples de programmes, le premier en Basic, le second en Pascal qui permettent de reconnaître la nature du matériel sur lequel va fonctionner leur logiciel.

VII-6.1 *Exemple de programme de reconnaissance de la carte ève en Basic*

Le programme listé ci-dessous s'appelle IDENTIFICATION et permet de reconnaître le matériel utilisé. Le code est fourni sur la disquette Basic DOS 3.3 PURPLESOFT.

Par exemple, l'appel du programme suivant permettra d'imprimer 0 s'il est exécuté sur un Apple II ou un Apple II Plus, 1 s'il est exécuté sur un Apple *//e* sans carte texte 80 colonnes dans le slot auxiliaire, 2 s'il est exécuté sur un Apple *//e* avec une carte texte 80 colonnes sans extension 64K dans le slot auxiliaire, 3 avec une carte texte 80 colonnes étendue (64K), 4 avec la carte RVB 80 colonnes étendue (64K) ève du Chat Mauve.

```
10 TEXT:HOME:D$=CHR$(4)
20 PRINT D$;« BLOAD INDENTIFICATION,A$300 »
30 CALL 768:PRINT(PEEK(975))
40 END
```

Listing de IDENTIFICATION

```
        .ORG 300

CODE    .EQU 3CF        ;contient apres l'appel de IDENTIFIE :
                        ; 0 --> si non Apple //e
                        ; 1 --> pas de carte texte 80 colonnes dans le slot
                        ;       auxiliaire
                        ; 2 --> carte texte 80 colonnes dans le slot
                        ;        auxiliaire sans extension 64K
                        ; 3 --> carte texte 80 colonnes étendue (64K)
                        ; 4 --> carte RVB 80 colonnes étendue ève du Chat
                        ;       Mauve (64K)

IDENTIFIC PHP           ;sauvegarde registre d'état
          CLD           ;mode décimal off
          SEI           ;empêche toute interruption

          JSR SOMCALC   ;calcul de D000+D100+D200+....FF00
          STA SOMME     ;partie haute dans Y, basse dans A
          STY SOMME+1   ;résultat dans SOMME

          LDX #0        ;X sert d'index de reconnaissance
          LDA 0C081
          LDA 0C081     ;lecture de la ROM

          LDA 0FBB3     ;on teste la signature de l'Apple //e
          CMP #6
          BNE ROM       ;si résultat diff de 6, ce n'est pas un Apple //e

          INX           ;X=1
          LDA 0C017     ;si lors de l'initialisation par RESET aucune carte
          BMI ROM       ;n'a été reconnue

CARTE80   INX           ;X=2
          LDA 0C01D
          PHA           ;sauvegarde de l'état de HIRES
          LDA 0C018
          PHA           ;sauvegarde de l'état de 80STORE
          LDA 0C01C
          PHA           ;sauvegarde de l'état de PAGE2

ECRAN     LDA 0C019     ;on boucle tant qu'on est visualisé
          BMI ECRAN

          STA 0C057     ;HIRES on
          STA 0C001     ;80STORE on
          STA 0C055     ;accès PAGE1X

          LDA 400
          PHA           ;sauvegarde de l'octet situé en 400
          LDA 2400
          PHA           ;sauvegarde de l'octet situé en 2400

          LDA #0EE      ;on charge EE en 400 PAGE1X
          STA 400
          LDA 2400      ;2400 est confondu avec 400 par le hardware si une
          CMP #0EE      ;carte texte sans extension 64K est présente
          BNE ETENDUE
          ASL 2400
          LDA 400       ;on recommence avec une valeur différente au cas ou ..
          CMP 2400
          BEQ OLD24

ETENDUE   INX           ;X=3
          LDA #0F
          STA 0C0B9     ;TEXT16 on et registre automatique a 0F
          STA 0C054     ;PAGE1
          LDA 400
          STA 400
          STA 0C0B8     ;TEXT16 off
          STA 0C055     ;PAGE1X
          LDA 400
          BMI OLD24     ;0EE ou 0F en 400 PAGE1X
          INX           ;X=4

OLD24     PLA
          STA 2400      ;remise en état 2400

          PLA
          STA 400       ;remise en état de 400

          PLA
          BMI OLDSTORE
          STA 0C054     ;remise en état de PAGE2

OLDSTORE  PLA
          BMI OLDHIRES
          STA 0C000     ;remise en état de 80STORE

OLDHIRES  PLA
          BMI ROM
          STA 0C056     ;remise en état de HIRES
```

```
        LDA 400
        PHA             ;sauvegarde de l'octet situé en 400
        LDA 2400
        PHA             ;sauvegarde de l'octet situé en 2400

        LDA #0EE        ;on charge EE en 400 PAGE1X
        STA 400
        LDA 2400        ;2400 est confondu avec 400 par le hardware si une
        CMP #0EE        ;carte texte sans extension 64K est présente

        BNE ETENDUE
        ASL 2400
        LDA 400         ;on recommence avec une valeur differente au cas ou ..
        CMP 2400
        BEQ OLD24

ETENDUE INX             ;X=3
        LDA #0F
        STA 0C0B9       ;TEXT16 on et registre automatique a 0F
        STA 0C054       ;PAGE1
        LDA 400
        STA 400
        STA 0C0B8       ;TEXT16 off
        STA 0C055       ;PAGE1X
        LDA 400
        BMI OLD24       ;$400 contient EE ou 0F
        INX             ;X=4

OLD24   PLA
        STA 2400        ;remise en état 2400

        PLA
        STA 400         ;remise en état de 400

        PLA
        BMI OLDSTORE
        STA 0C054       ;remise en état de PAGE1-PAGE2

OLDSTORE PLA
        BMI OLDHIRES
        STA 0C000       ;remise en état de 80STORE

OLDHIRES PLA
        BMI FIN
        STA 0C056       ;remise en état de HIRES

FIN     LDA 0C08B
        LDA 0C08B

        PLP             ;débloque les interruptions

        LDA #0
        PHA
        TXA
        PHA             ;résultat dans la pile

        LDA 1
        PHA
        LDA 0           ;retour Pascal
        PHA
        RTS

        .END            ;fin de SIGNATURE
```

Zones mémoire d'affichage

modes de visualisation	Page	Adresse de début		Adresse de fin	
Texte 40 colonnes et Graphisme basse résolution	1 2	$400 $800	1024 2048	$7FF $BFF	2047 3071
Texte 80 colonnes		$400	1024	$7FF	2047
Graphisme haute résolution standard	1 2	$2000 $4000	8192 16384	$3FFF $5FFF	16383 24575
Haute résolution étendue		$2000	8192	$3FFF	16383

Tableau VII-1

Commutateurs d'affichage

Nom	Fonction	adresse Hex	adresse Décimal	lecture écriture
TEXT	"on": écran texte	$C051	-16303	
	"off": écran graphique	$C050	-16304	
	lire l'état de TEXT	$C01A	-16358	lire
MIXED	"on": texte sous graphique	$C053	-16301	
	"off": écran tout graphique	$C052	-16302	
	lire l'état de MIXED	$C01B	-16357	lire
PAGE2	"on": Affichage de la page 2	$C055	-16299	
	"off": Affichage de la page 1	$C054	-16300	
	lire l'état de PAGE2	$C01C	-16356	lire
HIRES	"on": Graphisme haute résolution	$C057	-16297	
	"off": Graphisme basse résolution	$C056	-16298	
	lire l'état de HIRES	$C01D	-16355	
80COL	"on": texte 80 colonnes	$C00D	-16371	écrire
	"off": texte 40 colonnes	$C00C	-16372	écrire
	lire état de 80COL	$C01F	-16353	lire
80STORE	"on": modifie la fonction de PAGE2	$C001	-16383	écrire
	"off": fonction usuelle pour PAGE2	$C000	-16384	écrire
	lire l'état de 80STORE	$C01B	-16360	lire

Tableau VII-2

Commutateurs mémoires

Nom	Fonction	adresse Hex	adresse Décimal	lecture écriture
RAMRD	"on": lecture 48K aux.	$C003	-16381	écrire
	"off": lecture 48K //e	$C002	-16382	écrire
	lire l'état de RAMRD	$C013	-16365	lire
RAMWRT	"on": écriture 48K aux.	$C005	-16379	écrire
	"off": écriture 48K //e	$C004	-16380	écrire
	lire l'état de RAMWRT	$C014	-16364	lire
ALTZP	"on": pile,page 0,bank switched aux.	$C009	-16375	écrire
	"off": pile,page 0,bank switched //e	$C008	-16376	écrire
	lire l'état de ALTZP	$C016	-16362	lire
80STORE	"on": accès à la page 1X par PAGE2	$C001	-16383	écrire
	"off": utiliser RAMRD,RAMWRT	$C000	-16384	écrire
	lire l'état de 80STORE	$C018	-16360	lire
PAGE2	"on": accès mém. aux. (si 80STORE on)	$C055	-16299	
	"off": accès mém. //e	$C054	-16300	
	lire l'état de PAGE2	$C01C	-16356	lire
HIRES	"on": accès possible page graph. 1X	$C057	-16297	
	"off": utiliser RAMRD,RAMWRT	$C056	-16298	
	lire l'état de HIRES	$C01D	-16355	lire

Tableau VII-3

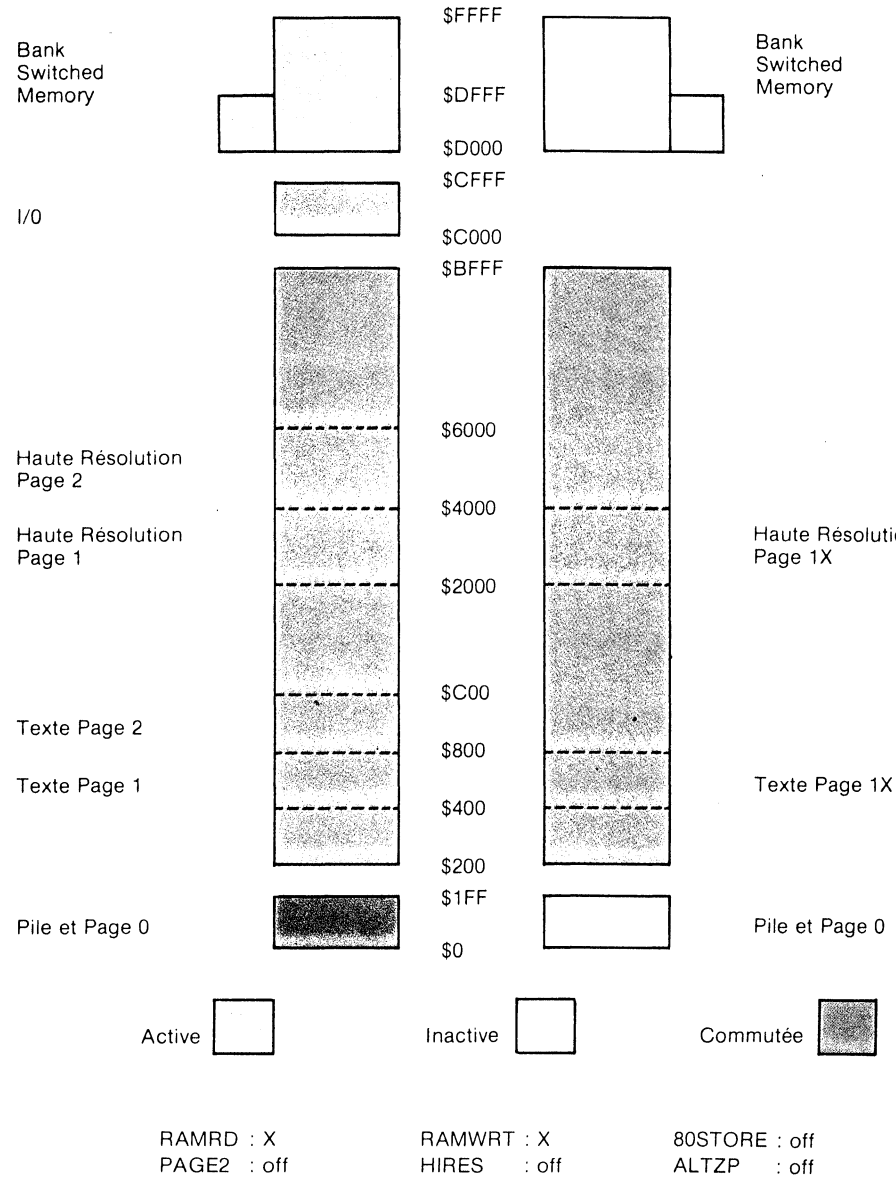


Figure VII-1 :
Commutations provoquées par RAMRD et RAMWRT lorsque 80STORE est « on »

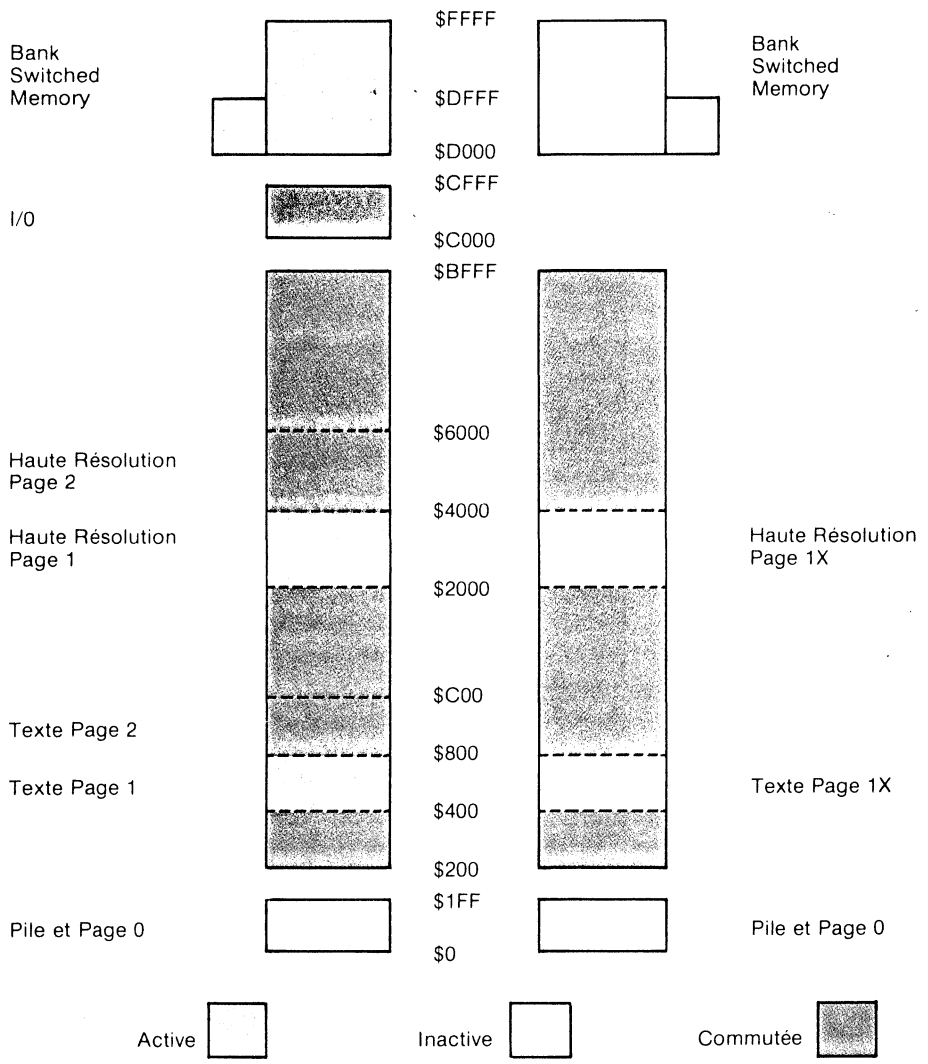


Figure VII-2 :
Commutations provoquées par RAMRD et RAMWRT lorsque 80STORE et HIRES sont « on »

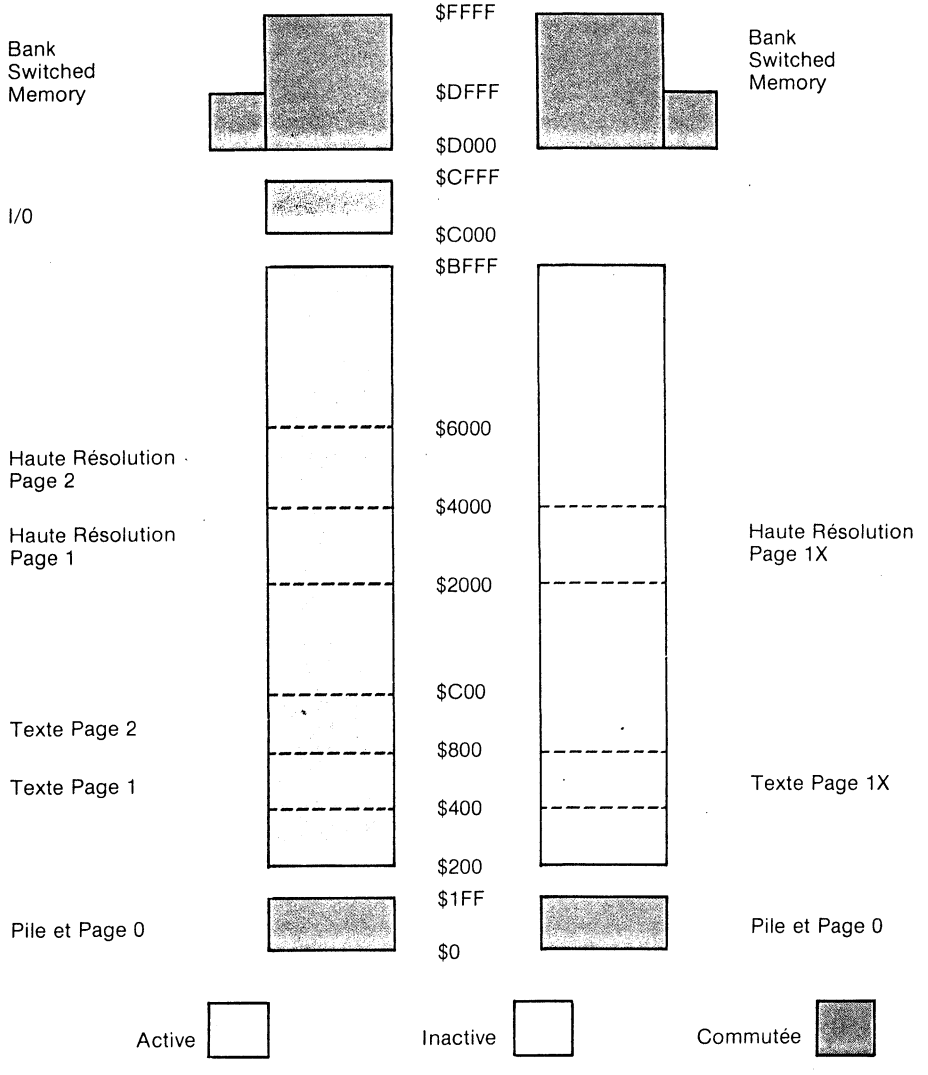


Figure VII-3 :
Commutations provoquées par ALTZP

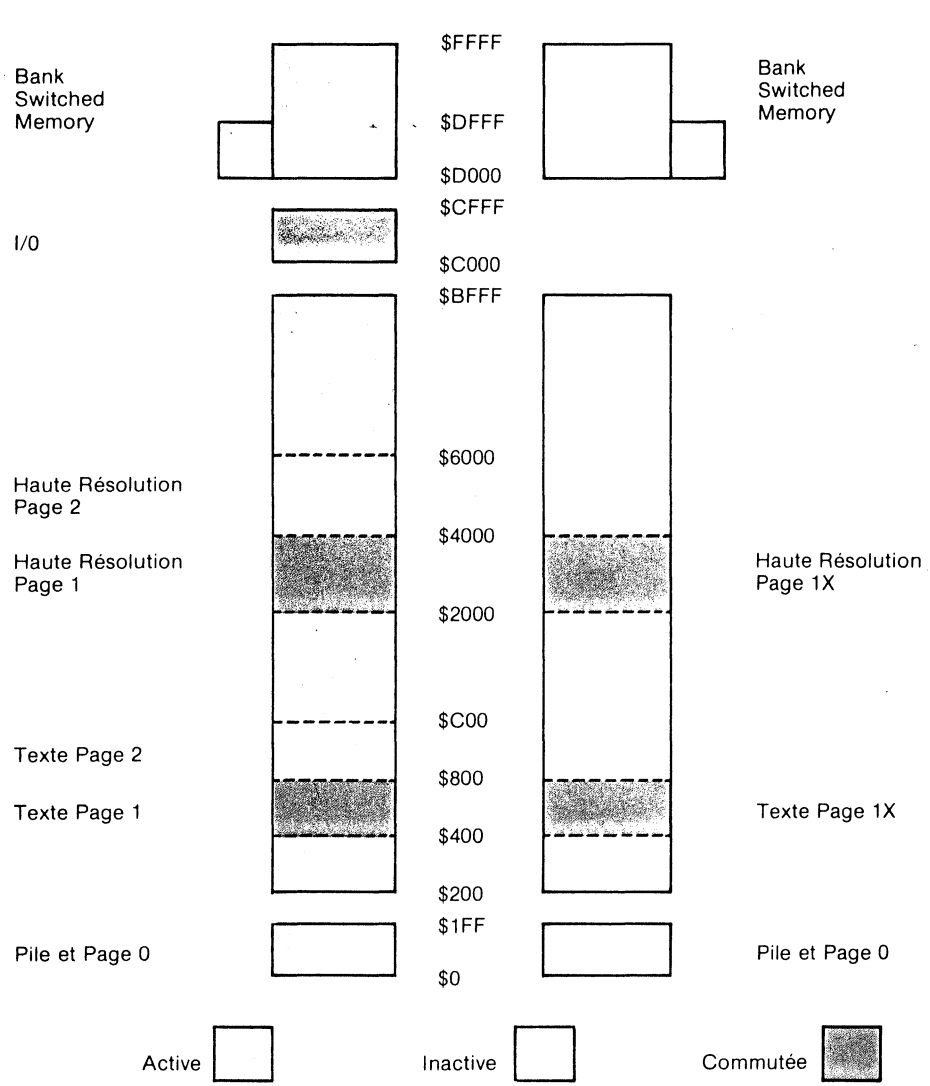


Figure VII-4 :
Commutations provoquées par PAGE2 lorsque 80STORE et HIRES sont « on »

Eve et le langage machine

CHAPITRE VIII

Eve et le langage machine

VIII-1 OCCUPATION MÉMOIRE DU LOGICIEL GRAPHIQUE PURPLESOFT

VIII-1.0 *Introduction*

Le plus grand soin a été apporté à cette gestion mémoire. En effet, la plus grande partie de ce logiciel réside en mémoire auxiliaire se contentant d'une place de 1K-octets en mémoire principale et laissant donc ainsi le maximum de place en mémoire principale pour les programmes de l'utilisateur.

Ce logiciel est entièrement rédigé dans le langage assembleur du 6502; il se compose de 3 parties :

a) L'interface

Cette première partie occupe 1K-octets en mémoire principale. Elle se situe entre le DOS et ses buffers en Basic et juste au-dessous de la page haute résolution protégée en Pascal. Les variables globales, les appels de sous-programmes, l'interpréteur Basic y résident.

b) Les caractères graphiques

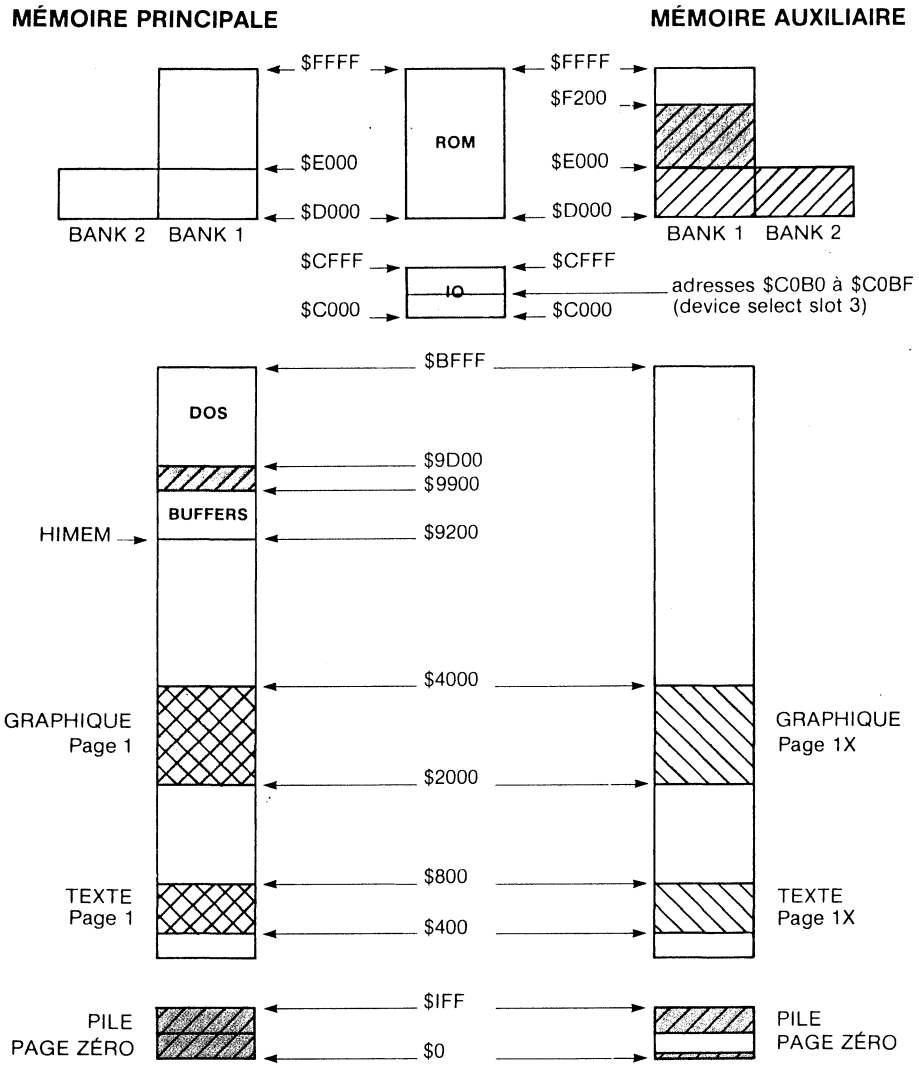
Cette seconde partie occupe 8K-octets en mémoire auxiliaire dans les deux banques parallèles de 4K situées entre les adresses $D000 et $DFFF. Elle peut contenir ainsi jusqu'à 8 ensembles de 128 caractères.

Le premier est chargé dans la banque 1 en $D000.
Le second est chargé dans la banque 1 en $D400.
Le troisième est chargé dans la banque 1 en $D800.
Le quatrième est chargé dans la banque 1 en $DC00.
Le cinquième est chargé dans la banque 2 en $D000.
Le sixième est chargé dans la banque 2 en $D400.
Le septième est chargé dans la banque 2 en $D800.
Le huitième est chargé dans la banque 2 $DC00.

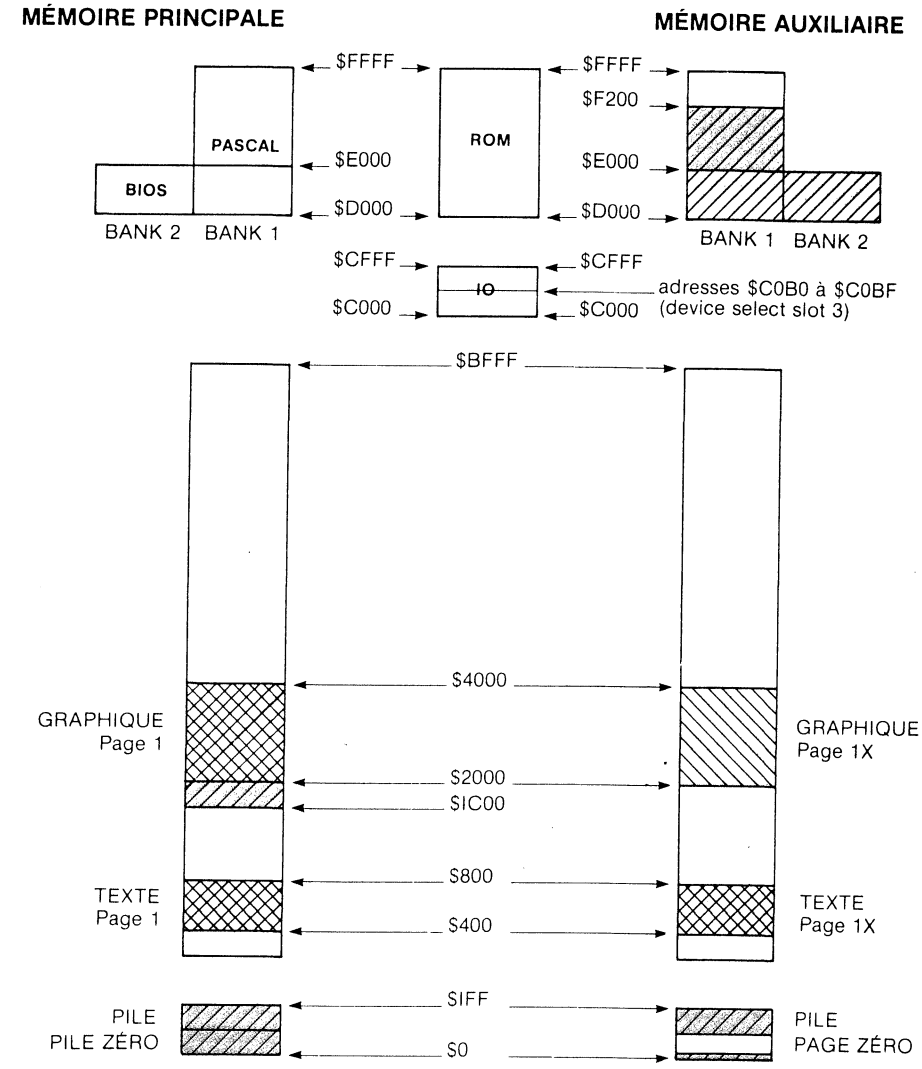
c) L'implémentation

Cette troisième partie contient toutes les procédures internes graphiques et réside en mémoire auxiliaire entre les adresses $E000 et $F1FF.

VIII-1.1 *Cartographie de la gestion mémoire*



GESTION MÉMOIRE EN BASIC



GESTION MÉMOIRE EN PASCAL

REMARQUES IMPORTANTES

— Les adresses $00 à $0B dans la page 0 de la mémoire auxiliaire sont utilisées de manière fugitive par ce logiciel. Vous pouvez donc les utiliser de manière provisoire mais n'espérez pas y retrouver vos pointeurs après l'appel d'un sous-programme de PURPLESOFT.

— La pile auxiliaire (page 1) est évidemment utilisée.

— Tous les modes graphiques « 16K » utilisent la page 1X (adresses $2000 à $3FFF) en mémoire auxiliaire.

— En Basic, les buffers du DOS sont translatés de 1K-octets vers le bas en mémoire principale pour dégager une place pour l'interface.

— En Pascal, la pile des variables pointeurs est remontée à l'adresse $4000 pour protéger les adresses inférieures donc en particulier la page graphique en mémoire principale. L'interface est alors placé en $1C00.

VIII-2 DESCRIPTION DE L'INTERFACE

VIII-2.0 *Avertissement*

Pour mieux comprendre le fonctionnement des sous-programmes et des variables décrits ci-dessous, il est conseillé de lire au préalable le chapitre V décrivant le logiciel PURPLESOFT.

VIII-2.1 *Variables destinées à la lecture et à l'écriture*

a) Table de commande des couleurs (16x16 octets)

— COLORSTAB: adr. basic $9900 à $99FF hexa (-26368 à -26113 déc.)
adr. pascal $1C00 à $1CFF hexa (7168 à 7423 déc.)

Le p-ème octet (où p=N+16x0) contient un entier compris entre 0 et 15 qui correspond à la couleur effectivement affichée lors d'un traçage dans la couleur d'encre N sur la couleur d'écran 0 à condition toutefois que la variable OPTTABLE soit non nulle.

— OPTTABLE: adr. basic $9A26 hexa (-26074 déc.)
adr. pascal $1D26 hexa (7462 déc.)

Si cet octet est à 0, la table est inactive sinon elle est activée.

b) Options lors d'un changement de mode graphique

— OPTCLEAN: adr. basic $9A2E hexa (-26066 déc.)
adr. pascal $1D2E hexa (7470 déc.)

— SWITXTGR: adr. basic $9A30 hexa (-26064 déc.)
adr. pascal $1D30 hexa (7472 déc.)

c) Variables de passage de paramètres

— PARA1: adr. basic $9A00 / $9A01 hexa (-26112 / -26111 déc.)
adr. pascal $1D00 / $1D01 hexa (7424 / 7425 déc.)

— PARA2: adr. basic $9A02 / $9A03 hexa (-26110 / -26109 déc.)
adr. pascal $1D02 / $1DO3 hexa (7426 / 7427 déc.)

— PARA3: adr. basic $9A04 / $9A05 hexa (-26108 / -26107 déc.)
adr. pascal $1D04 / $1D05 hexa (7428 / 7429 déc.)

— PARA4: adr. basic $9A06 / $9A07 hexa (-26106 / -26105 déc.)
adr. pascal $1D06 / $1D07 hexa (7430 / 7431 déc.)

— ABS1: adr. basic $9A08 / $9A09 hexa (-26104 / -26103 déc.)
adr. pascal $1D08 / $1D09 hexa (7432 / 7433 déc.)

— ORD1: adr. basic $9A0A / $9A0B hexa (-26102 / -26101 déc.)
adr. pascal $1D0A/$1D0B hexa (7434 / 7435 déc.)

— ABS2: adr. basic $9A0C / $9A0D hexa (-26100 / -26099 déc.)
adr. pascal $1D0C / $1D0D hexa (7436 / 7437 déc.)

— ORD2: adr. basic $9A0E / $9A0F hexa (-26098 / -26097 déc.)
adr. pascal $1D0E / $1D0F hexa (7438 / 7439 déc.)

d)Abscisse et ordonnée du curseur graphique

— XCURS: adr. basic $9A10 / $9A11 hexa (-26096 / -26095 déc.)
adr. pascal $1D10 / $1D11 hexa (7440 / 7441 déc.)

— YCURS: adr. basic $9A12 / $9A13 hexa (-26094 / -26093 déc.)
adr. pascal $1D12 / $1D13 hexa (7442 / 7443 déc.)

e) Extensions de l'utilisation du symbole AMPERSAND

— NEWINTERP: adr. basic $9AC6 / $9AC7 hexa (-25914 / -25913 déc.)

Ces deux octets contiennent usuellement l'adresse $DEC9 (SYNTAX ERROR) on peut y mettre l'adresse d'une nouvelle extension d'interpréteur utilisant l'ampersand.

VIII-2.2 *Variables uniquement destinées à la lecture*

a) Limites de la fenêtre graphique

— LEFT: adr. basic $9A16 / $9A17 hexa (-26090 / -26089 déc.)
adr. pascal $1D16 / $1D17 hexa (7446 / 7447 déc.)

Ces deux octets contiennent la limite à gauche de la fenêtre qui est comprise entre 0 et la résolution maximale-1 du mode graphique en vigueur.

— RIGHT: adr. basic $9A18 / $9A19 hexa (-26088 / -26087 déc.)
adr. pascal $1D18 / $1D19 hexa (7448 / 7449 déc.)

Ces deux octets contiennent la limite à droite de la fenêtre qui est comprise entre 1 et la résolution maximale du mode graphique en vigueur.

— BOTTOM: adr. basic $9A1A / $9A1B hexa (-26086 / -26085 déc.)
adr. pascal $1D1A / $1D1B hexa (7450 / 7451 déc.)

Le premier octet contient le bas de la fenêtre qui est compris entre 0 et 191, le second doit être à 0.

— TOP: adr. basic $9A1C / $91D hexa (-26084 / -26083 déc.)
adr. pascal $1D1C / $1D1D hexa (7452 / 7453 déc.)

Le premier octet contient le haut de la fenêtre qui est comrpis entre 1 eet 192, le second doit être à 0.

b) Echelle et caractères graphiques

— CHARADRESS: adr. basic $9A1E / $9A1F hexa (-26082 / -26081 déc.)
adr. pascal $1D1E / $1D1 F hexa (7454 / 7455 déc.)

Ces deux octets contiennent l'adresse du charset courant.

— XFACTEUR: adr. basic $9A20 hexa (-26080 déc.)
adr. pascal $1D20 hexa (7456 déc.)

Cet octet contient le facteur de répétition de colonnes qui est compris entre 0 et 127.

— YFACTEUR: adr. basic $9A22 hexa (-26078 déc.)
adr. pascal $1D22 hexa (7458 déc.)

Cet octet contient le facteur de répétition de lignes qui est compris entre 0 et 127.

— CHARSET: adr. basic $9A24 hexa (-26076 déc.)
adr. pascal $1D24 hexa (7460 déc.)

Cet octet contient le numéro du charset courant entre 0 et 7.

c) Option de transfert, texte et grahique en vigueur

— TRANSFERT: adr. basic $9A28 hexa (-26072 déc.)
adr. pascal $1D28 hexa (7464 déc.)

Cet octet contient le numéro de l'option de transfert courante comprise entre 0 et 7.

— TEXTWRK: adr. basic $9A2A hexa (-26070 déc.)
adr. pascal $1D2A hexa (7466 déc.)

Octet contenant le numéro du mode texte courant dans 1..6 (non utilisé en pascal).

— GRAFWRK: adr. basic $9A2C hexa (-26068 déc.)
adr. pascal $1D2C hexa (7468 déc.)

Octet contenant le numéro du mode graphique courant dans 0..9.

d) Informations couleur

— COLOR: adr. basic $9A32 hexa (-26062 déc.)
adr. pascal $1D32 hexa (7474 déc.)

Octet contenant la couleur de l'encre utilisée dans 0..16.

— FILLCOL: adr. basic $9A34 hexa (-26060 déc.)
adr. pascal $1D34 hexa (7476 déc.)

Octet contenant la couleur de fond utilisée dans 0..16.

— REG: adr. basic $9A36 hexa (-26058 déc.)
adr. pascal $1D36 hexa (7478 déc.)

Octet contenant l'état du registre couleur de la carte ève.

— COLRETURN: adr. basic $9A38 hexa (-26056 déc.)
adr. pascal $1D38 hexa (7480 déc.)

Octet où est retournée la couleur d'un point de l'écran.

— BACKGROUND: adr. basic $9A3C hexa (-26052 déc.)
adr. pascal $1D3C hexa (7484 déc.)

Octet où est retournée la couleur des bits à 0 de la cellule concernée par le dernier appel de SCREENCALL en mode CP280.

— CPBYTE: adr. basic $9A3E hexa (-26050 déc.)
adr. pascal $1D3E hexa (7486 déc.)

Octet où est retourné le contenu de la cellule concernée par le dernier appel de SCREENCALL en mode CP280.

— CPCELL: adr. basic $9A40 hexa (-26048 déc.)
adr. pascal $1D40 hexa (7488 déc.)

Octet où est retourné le numéro dans 0..39 de la cellule concernée par le dernier appel de SCREENCALL en mode CP280.

— CPBIT: adr. basic $9A42 hexa (-26046 déc.)
adr. pascal $1D42 hexa (7490 déc.)

Octet où est retourné le numéro du bit dans 0..6 concerné par le dernier appel de SCREENCALL en mode CP280.

VIII-2.3 *Appels des sous-programmes graphiques*

Toutes les opérations de traçage et de remplissage sont éventuellement affectées par l'usage de la table de commande des couleurs et l'option de transfert. Elles ne fonctionnent pas dans les modes "8K".

a) COLORCALL: Fixe la couleur d'encre à n.

Basic: JSR $9A44 ou CALL-26044 / Pascal: JSR $1D44 ou CALL (7492)

Un seul paramètre n compris entre 0 et 16 est passé dans le mot PARA1 (attention !!! 16 bits).

b) BACKCALL: Fixe la couleur de fond à n.

Basic: JSR $9A4A ou CALL-26038 / Pascal: JSR $1D4A ou CALL (7498)
Un seul paramètre n compris entre 0 et 16 est passé dans le mot PARA1 (attention !!! 16 bits).

c) TEXTCALL: Exécute l'ordre &TEXT n (inutilisable en Pascal)

Basic: JSR $9A50 ou CALL-26032
Un seul paramètre n compris entre 0 et 6 est passé dans le mot PARA1 (attention !!! 16 bits).

d) GRCALL: Exécute l'ordre &GR n

Basic: JSR $9A56 ou CALL-26026 / Pascal: JSR $1D56 ou CALL (7510)
Un seul paramètre n compris entre 0 et 10 est passé dans le mot PARA1 (attention !!! 16 bits).

Remarque importante : Tout appel du sous-programme GRCALL place le switch HIRES dans la position on ($C057) pour des raisons liées au fonctionnement du switch 80STORE (voir APPLE //e reference manual).
Il est donc impossible par exemple de visualiser la page basse résolution et de travailler simultanément dans les modes graphique "16K".

e) DOTCALL: Trace un point en x,y

Basic: JSR $9A5C ou CALL-26020 / Pascal: JSR $1D5C ou CALL (7516)
Deux paramètres x et y respectivement passés dans le mot ABS1 et le mot ORD1 (attention !!! 2x16 bits).

f) LINECALL: Trace une ligne de xdep,ydep à xarr,yarr

Basic: JSR $9A62 ou CALL-26014 / Pascal: JSR $1D62 ou CALL (7522)
Quatre paramètres xdep,ydep,xarr et yarr respectivement passés dans les mots ABS1,ORD1,ABS2 et ORD2 (attention !!! 4x16 bits).

g) SCREENCALL: Retourne la couleur du point de coordonées x,y

Basic: JSR $9A68 ou CALL-26008 / Pascal: JSR $1D68 ou CALL (7528)
Deux paramètres x et y respectivement passés dans le mot PARA1 et le mot PARA2 (attention !!! 2x16 bits).
La couleur du point de coordonnées x, y est retournée dans COLRETURN.

h) FILLCALL: Remplit la fenêtre de la couleur de fond

Basic: JSR $9A6E ou CALL-26002 / Pascal: JSR $1D6E ou CALL (7534)

i) CPCALL: Ne fonctionne qu'en mode CP280; remplit la page 1X de la couleur d'encre et de la couleur de fond courantes.

Basic: JSR $9A74 ou CALL-25996 / Pascal: JSR $1D74 ou CALL (7540)

j) WINDCALL: Fixe les limites de la fenêtre graphique g,d,b et h

Basic: JSR $9A7A ou CALL-25990 / Pascal: JSR $1D7A ou CALL (7546)
Quatre paramètres g,d,b et h respectivement passés dans les mots PARA1,PARA2,PARA3 et PARA4 (attention aux limites et aux 4x16 bits).

k) XFRCALL: Fixe l'option de transfert à n.

Basic: JSR $9A80 ou CALL-25984 / Pascal: JSR $1D80 ou CALL(7552)
Un seul paramètre n compris entre 0 et 7 est passé dans le mot PARA1.

l) SCALECALL: Fixe les échelles de reproduction des caractères graphiques et des formes (tableau de booléens)

Basic: JSR $9A89 ou CALL-25975 / Pascal: JSR $1D89 ou CALL(7561)
Deux paramètres xfact et yfact compris entre 1 et 128 respectivement passés dans le mot PARA1 et le mot PARA2 (attention !!! 2x16 bits).

m) DRAWCALL: Trace le tableau de booléen qui commence à l'adresse placée dans le mot PARA1

Basic: JSR $9A8F ou CALL-25969 / Pascal: JSR $1D8F ou CALL(7567)

n) CHARCALL: Trace le caractère dont le code ASCII est passé dans PARA1 à l'emplacement du curseur graphique puis déplace le curseur pour tracer éventuellement un nouveau caractère

Basic: JSR $9A98 ou CALL-25960 / Pascal: JSR $1D98 ou CALL(7576)

o) CHRSETCALL: Fixe l'ensemble de caractères de travail au charset numéro n

Basic: JSR $9AAE ou CALL-25938 / Pascal: JSR $1DAE ou CALL(7598)
Un seul paramètre n compris entre 0 et 7 est passé dans le mot PARA1 (attention !!! 16 bits).

p) TRANSCHAR: Transfère un nouveau charset préalablement chargé à l'adresse placée dans PARA1 à la place à la place du charset courant.

Basic: JSR $9AB4 ou CALL-25932 / Pascal: JSR $1DB4 ou CALL(7604)

q) P1P1XCALL: Passage d'une page texte à l'autre (1 sur 1X) ou vice-versa avec sauvegarde des données d'entrées-sorties..

Basic: JSR $9ABA ou CALL-25926 / Pascal: JSR $1DBA ou CALL(7610)

VIII-3 CARACTÈRES GRAPHIQUES

VII-3.0 *Introduction*

Huit ensembles de 128 caractères (charsets) sont disponibles. Le sous-programme CHRSETCALL permet de choisir le numéro de l'ensemble de caractères avec lequel on veut travailler. Le sous-programme CHARCALL trace le caractère graphique dont le code ASCII se trouve dans PARA1 à la position du curseur à l'échelle fixée par SCALECALL. Le sous-programme TRANSCHAR permet de remplacer l'ensemble de caractères de travail par un autre personnel à l'utilisateur (caractères espagnols, suédois, etc...) chargé au préalable en mémoire.

VIII-3.1 *Structure d'un ensemble de caractères graphiques (charset)*

Chaque caractère est organisé comme une matrice de 7x8 points et utilise 8 octets en mémoire, le bit 7 de poids fort de chaque octet étant ignoré. Un ensemble de 128 caractères utilise donc 128x8=1024 octets (1K-octet) de mémoire.

exemple :	n° bit	7	6	5	4	3	2	1	0	
	octet n° 7	0	0	1	1	1	1	1	0	
	octet n° 6	0	0	0	0	0	0	1	0	
	octet n° 5	0	0	0	0	0	0	1	0	
	octet n° 4	0	0	1	1	1	1	1	0	le bit 7 est ignoré
	octet n° 3	0	0	0	0	0	0	1	0	
	octet n° 2	0	0	0	0	0	0	1	0	
	octet n° 1	0	0	1	1	1	1	1	0	
	octet n° 0	0	0	0	0	0	0	0	0	← bit correspondant au premier point traçé à la position du curseur

De quel caractère bien connu s'agit-il sachant qu'à l'écran le point correspondant au bit de poids faible est le plus à gauche ?

Il sera codé en mémoire par les 8 octets : 00 3E 02 02 3E 02 02 3E

Vous pouvez donc créer ainsi vos propres caractères ou mieux créer un programme qui le fasse !!!..ou plus facile utiliser les charsets disponibles sur vos disquettes.

Remarque importante : La structure des caractères graphiques n'est pas la même que celle des tableaux de booléens utilisée par DRAWCALL qui est compactifiée (tous les bits sont utilisés).

Informations techniques

CHAPITRE IX

Informations techniques

IX-1 LES COMMUTATEURS DE LA CARTE ÈVE

L'utilisation de ces commutateurs suppose une bonne connaissance de l'Apple *//e* et de ses commutateurs. Une mauvaise utilisation peut déclencher à l'écran une visualisation surprenante.

Les commutateurs de la carte ève et ceux de l'Apple *//e* fonctionnent de la même façon : une adresse sert à les brancher (« on »), une autre à les débrancher (« off »). Il n'est pas possible de relire l'état de ces commutateurs; à l'allumage de votre Apple *//e*, ils sont tous débranchés. Ils occupent les adresses $C0B0 à $C0BF (-16208 à -16193) réservées à la carte occupant le slot 3 de votre Apple *//e*. Il faudra écrire à ces adresses car, à chaque fois la donnée sera mémorisée dans un registre de la carte ève :CPREG.

CPREG mémorise la couleur des points (en partie basse) et la couleur de fond (en partie haute) utilisées dans le mode texte couleur et dans le mode graphique CP280. Si CPREG est en fonction (voir à ce sujet la description des autres commutateurs) son contenu est automatiquement inscrit en mémoire auxiliaire lors d'une écriture en mémoire principale aux adresses $400-$7FF (page écran texte) et $2000-$3FFF (page écran graphique).

AN3 :	on	$C05F	-16289
	off	$C05E	-16290

Le commutateur AN3 est implémenté sur votre Apple *//e* et permet lorsqu'il est branché de visualiser des écrans graphiques « double haute résolution » (utilisant 16K de mémoire écran) avec la carte ève. La visualisation de certains modes graphiques « 16K » nécessite le branchement de 80COL; consulter à ce sujet le tableau IX-1.

HR1 :	on	$C0B3	-16205
	off	$C0B2	-16206
HR2 :	on	$C0B5	-16203
	off	$C0B4	-16204
HR3 :	on	$C0B7	-16201
	off	$C0B6	-16202

Le fonctionnement de ces trois commutateurs dépend de l'état de AN3:

— Si AN3 est branché, l'affichage graphique haute résolution est la haute résolution standard et HR1, HR2 et HR3 en sélectionnent le mode de décodage. Lorsque HR1 est branché les points isolés de couleur sur fond blanc (usuellement) apparaissent alors noir; ceci permet une meilleure lisibilité des caractères sur fond blanc sans beaucoup altérer le reste de l'image. Lorsque HR2 est branché les points isolés de couleur sur fond noir (usuellement) apparaissent alors blanc; ceci permet une meilleure lisibilité des caractères sur fond noir mais peut modifier de façon importante le reste de l'image.

Lorsque HR3 est branché les surfaces de couleur (usuellement uniformes) apparaissent pointillées horizontalement. On se rapproche alors de l'image obtenue sur un moniteur monochrome et ceci peut être plus agréable lorsqu'il n'y a pas de surface colorée mais uniquement des structures filaires (pour des animations en trois dimensions par exemple). Un cas particulier : si HR2 et HR3 sont branchés et HR1 débranché l'image est alors visualisée en noir et blanc.

— Si AN3 est débranché l'affichage graphique haute résolution utilise alors 16K de mémoire (le double) : c'est la haute résolution étendue. Et les commutateurs HR1, HR2 et HR3 permettent de choisir le type d'affichage selon le tableau IX-1 suivant :

Choix d'un mode graphique

AN3	HR1	HR2	HR3	Mode grahique	Note
on	off	off	off	Haute résolution standard	
on	on			HR std + pts noirs/fond blanc	
on		on		HR std + pts blancs/fond noir	
on			on	HR std + horizontales pointillées	
on	off	on	on	HR std en noir et blanc	
off	off	off	off	COL140 (140 pts/ligne, 16 coul.)	1
off	on	off	off	COL280A (280 pts/ligne, 4 coul.)	1
off	off	on	off	COL280A (280 pts/ligne, 4 coul.)	1
off	on	on	off	écran bloqué au noir	2
off	off	on	on	BW560 (560 pts/ligne, noir et blanc)	1
off	on	on	on	CP280 (280 pts/ligne, 16 coul.,lim.)	2

Note 1 : 80COL doit être branché (on) pour obtenir cette visualisation.
Note 2 : Si LOCKCPREG est débranché et ENHRCPREG branché, CPREG est alors en fontion.

Tableau IX-1

TXT16 : on $C0B9 -16199
off $C0B8 -16200

Si TXT16 est branché et 80COL débranché, l'écran texte 40 colonnes usuel est alors remplacé par un écran texte couleur où peuvent être choisies individuellement la couleur de chaque caractère et celle du fond sur lequel il est affiché. A chaque caractère de la page écran texte correspond un octet en mémoire auxiliaire (à la même adresse) qui représente alors ses couleurs (points et fond) à l'écran. Si TEXT16 est branché et LOCKCPREG débranché, le contenu de CPREG est automatiquement inscrit en mémoire auxiliaire lors de l'envoi à l'écran texte d'un caractère.

LOCKCPREG : on $C0BD -16195
off $C0BC -16196

LOCKCPREG, lorsqu'il est branché bloque totalement le fonctionnement de CPREG (plus d'écriture automatique en mémoire auxiliaire en texte couleur et en mode graphique CP280).

ENHRCPREG : on $C0B1 -16207
off $C0B0 -16208

ENHRCPREG lorsqu'il est débranché bloque le fonctionnement de CPREG mais uniquement sur l'espace mémoire $2000-$3FFF correspondant à l'écran haute résolution graphique. Si TXT16 est branché (et LOCKCPREG débranché) CPREG continue de fonctionner sur l'espace mémoire $400-$7FF (écran texte).
ATTENTION : ENHRCPREG doit être débranché lorsque AN3 est branché.

TXTGREEN : on $C0BB -16197
off $C0BA -16198

TXTGREEN lorsqu'il est branché remplace la visualisation monochrome blanche des écrans texte 40 et 80 colonnes par une visualisation monochrome verte.

LOCKRES : on $C0BF -16193
off $C0BE -16194

Lorsque LOCKRES est débranché tout <Ctrl-Reset> débranche tous les commutateurs de la carte ève (la carte étant ainsi remise dans son état à l'allumage de l'Apple *//e*). Brancher LOCKRES permet de soustraire la carte ève à cette action <Ctrl-Reset> .

IX-2 LES CONNECTEURS DE LA CARTE ÈVE

Sur la carte ève il y a trois connecteurs :

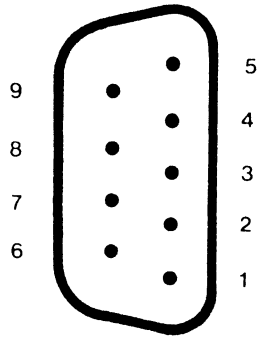
— un connecteur 8 broches destinée au raccordement sur la prise PÉRITEL d'un téléviseur

haut
- pin 1 : Commutation rapide (2 Volts., 75 Ohms)
- pin 2 : Signal audio (son)
- pin 3 : Synchronisation composite négative (75 Ohms, 0.5 Volts)
- pin 4 : Rouge (75 Ohms, ajustable de 0 à 0.7 Volts)
- pin 5 : Vert (75 Ohms, ajustable de 0 à 0.7 Volts)
- pin 6 : Bleu (75 Ohms, ajustable de 0 à 0.7 Volts)
- pin 7 : Masse (0 Volts)
- pin 8 : Alimentation 12 Volts (commutation lente)

bas

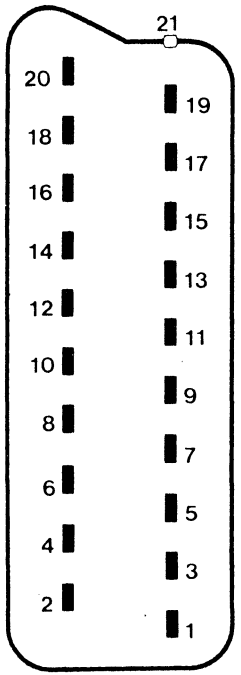
Ces signaux sont compatibles avec certains moniteurs RVB à entrées analogiques.

Bien vérifier tout de même si vous désirez brancher un moniteur RVB, particulièrement au niveau de la synchronisation. Il ne restera alors qu'à trouver ou fabriquer le câble nécessaire. Deux branchements sont possibles : sur notre câble à l'aide d'une prise PÉRITEL femelle ou sur le connecteur D 9 broches utilisé sur le panneau arrière; tous les signaux décrits ci-dessus s'y retrouvent de la façon suivante :



pin 1 : Non connecté
pin 2 : Masse
pin 3 : Vert
pin 4 : Synchronisation
pin 5 : Commutation rapide
pin 6 : 12 Volts
pin 7 : Bleu
pin 8 : Rouge
pin 9 : Audio (son)

Connecteur D 9 broches



pins 2 et 6 : Audio (son)

pin 7 : Bleu

pin 8 : 12 Volts

pin 11 : Vert

pin 15 : Rouge

pin 16 : Commutation rapide

pin 17 : Masse

pin 20 : Synchronisation

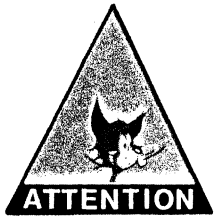
Prise PÉRITEL

— un connecteur 10 broches regroupant toutes les tensions d'alimentation, la synchronisation composite négative à un niveau T.T.L.LS et l'information couleur sous la forme de 4 bits (niveau T.T.L.LS). Ces sorties sont identiques aux sorties RVB de l'Apple *III* et permettent le branchement sur tout moniteur qui peut se connecter sur l'Apple *III*. Un câble permettant de passer du connecteur 10 broches de la carte ève à un connecteur D 15 broches identique à celui de l'Apple *III* peut alors être nécessaire.

haut
- pin 1 : + 12 Volts (alimentation)
- pin 2 : – 12 Volts (alimentation)
- pin 3 : – 5 Volts (alimentation)
- pin 4 : Synchronisation (niveau T.T.L.LS)
- pin 5 : + 5 Volts (alimentation)
- pin 6 : Masse
- pin 7 : XRGB1 (information couleur, bit de poids faible)
- pin 8 : XRGB2 (information couleur)
- pin 9 : XRGB4 (information couleur)
- pin 10 : XRGB8 (information couleur, bit de poids fort)

bas

— un connecteur 2 broches destiné au raccordement du son de votre Apple *//e* à la carte ève pour une éventuelle sortie audio sur votre téléviseur.



ATTENTION : Ne pas relier à un autre connecteur que le J18 SPEAKER de votre Apple *//e*.

APPENDICE

Compatibilité entre les unités Pascal Purplegraphics février 83 et juillet 83

A-1 PRÉSENTATION

La nouvelle unité PURPLEGRAPHICS (juillet 83) est plus performante :
— Elle occupe en mémoire principale une place réduite permettant l'élaboration de programmes beaucoup plus long.
— Certaines erreurs ont été corrigées.
— Certaines procédures ont été supprimées, de nouvelles ont été créées.
— La structure des images sauvegardées sur disque a été modifiée pour être directement compatibles avec les images conçues sur l'ordinateur Apple *///*.
— Huit ensembles de caractères graphiques sont disponibles sans accès disque particulier.

A-2 NOUVELLES PROCÉDURES

```
procedure CHARSET(N:INTEGER);
procedure LOADCHARSET(S:STRING);
procedure CLEANOPTION(C:BOOLEAN);
procedure TEXTBG;
procedure TEXTBW;
```

Cinq variables prédéfinies ont été ajoutées :

```
CPFOREGND:COLORTYPE
CPBACKGND:COLORTYPE
CPCELL:INTEGER
CPBIT:INTEGER
CPBYTE:PACKED ARRAY [0..6] OF BOOLEAN
```

A-3 PROCÉDURES SUPPRIMÉES

ERASE	:	n'a plus aucune utilité, la procédure GRAFIXMODE ayant été améliorée (l'enlever de vos programmes)
INITGRAFIX	:	toutes les fonctions de cette procédure sont accessibles séparément par d'autres procédures
SYSTCHARSET	:	équivalente à CHARSET (0)
ALTCHARSET	:	équivalente à CHARSET (1)

A-4 MODIFICATIONS DE FONCTIONNEMENT

La fonction DOTCOLOR en CP280 retourne la couleur du point d'écran et initialise les variables prédéfinies CPFOREGND,CPBACKGND, CPCELL,CPBIT et CPBYTE.

Les procédures GLOAD et GSAVE manipulent des fichiers structurés différemments : les 16 premiers blocs correspondent à la mémoire écran graphique située en mémoire auxiliaire et les 16 autres à la partie située en mémoire principale. Avant la sauvegarde GSAVE place en $2078 et $2079 des codes correspondants au mode graphique utilisé et en $207A une signature du logiciel PURPLEGRAPHICS. Ces fichiers sont compatibles avec les fichiers de type FOTO créés sur ordinateur Apple /// dans les modes graphiques COL140 et BW560.
Note : Le programme MODFOTO sur PURPLE: permet de modifier vos anciens fichiers de type FOTO pour les rendre compatibles avec la nouvelle unité.

La procédure WINDOW n'agit plus sur la fenêtre s'il y a erreur dans les valeurs des paramètres.

A-5 RECOMPILATION

Pour être utilisable avec la nouvelle unité (juillet 83), tous les programmes créés avec l'ancienne (février 83), doivent être recompilés avec cette nouvelle unité dans la bibliothèque.

Quelques légères modifications peuvent être nécessaires dans le texte du programme pour tenir compte des changements décrits ci-dessus.

Addentum

1. *Emploi du commutateur PAGE2*

Il est déconseillé d'utiliser directement le commutateur PAGE2 pour accéder à une page 1X, ceci posant souvent des problèmes au niveau des entrées-sorties. A ce propos, le petit programme au chapitre VII est destiné à expliquer le fonctionnement de PAGE2 mais est un très mauvais exemple de programmation.
Le sous-programme P1P1XCALL (cf chapitre VIII) permet la commutation d'une page mémoire à l'autre avec sauvegarde des données d'entrées-sorties.

2. *Informations complémentaires sur PURPLESOFT*

Si vous utilisez le logiciel PURPLESOFT, l'instruction PR#3 doit être évitée au profit de &TEXT5 ou &TEXT6.

Au chargement du logiciel PURPLESOFT par le programme HELLO ou par une instruction BRUN PURPLESOFT, l'espace mémoire entre les adresses $1E00 et $5600 est utilisé de manière provisoire. Cette initialisation peut donc détruire une partie de vos programmes et de vos données. De plus, la page graphique 1X (en mémoire auxiliaire) est alors nettoyée.

3. *Informations complémentaires sur l'unité PURPLEGRAPHICS*

La valeur 17 est retournée dans la variable prédéfinie IOERROR si le fichier utilisé par les procédures GLOAD, GSAVE ou LOADCHARSET n'a pas une taille normale.

Les fichiers de type FOTO créés sur l'Apple /// par l'unité PGRAF sont directement compatibles avec l'unité PURPLEGRAPHICS s'ils ont été créés dans le mode BW560. Les images créées en mode CP280 ou COL140 peuvent être rendues compatibles en exécutant le programme MODFOTO.

4. *Caractères en mode graphique CP280.*

Compte tenu des limitations de ce mode, il est fortement conseillé de positionner le curseur avec une abscisse multiple de 7 pour dessiner des caractères ou des chaînes de caractères et de ne pas utiliser l'option de transfert.

5. *Tabulation en mode texte 80 colonnes*

Certains problèmes de tabulation peuvent apparaître en 80 colonnes HTAB ne doit pas être utilisé et POKE 36,X peut être remplacé (en 80 colonnes seulement) par POKE 1403,X.

6. *Appels de sous-programmes graphiques*

Les paramètres des sous-programmes documentés aux chapitres V et VIII doivent être initialisés JUSTE AVANT leur appel. Par exemple en Basic, le(s) POKE(s) doivent être placés juste avant le CALL.

7. *Programme Information*

Nous vous rappelons que le programme basic INFORMATION, sur la disquette PURPLESOFT (DOS 3.3), est conçu pour vous présenter toutes les précisions n'ayant pu être imprimées dans notre documentation. En juillet 1983, ce programme expliquait ceci :

— Si vous utilisez directement le commutateur PAGE2, notez bien que toutes les instructions et routines de PURPLESOFT débranchent ce commutateur (position « off »). Tenez-en compte pour l'élaboration de vos programmes.

— Le fichier PURPLESOFT est doté d'un chargement rapide. PURPLESOFT effectue le chargement rapide puis exécute PURPLESOFT* qui contient toutes les routines graphiques. Les fichiers PURPLESOFT et PURPLESOFT* ne doivent pas être copiés autrement que par le programme COPYA. Vous pourrez ensuite effacer par un « DELETE » les fichiers qui ne vous sont pas nécessaires.

A ce propos, les huit premiers « charset » sont déjà contenus dans le fichier PURPLESOFT* et leur présence sur la disquette n'est pas nécessaire pour l'exécution de vos programmes (il en est de même en Pascal).

— Les programmes basic GRLOAD et GRSAVE, destinés au chargement et à la sauvegarde d'images graphiques sur disque, peuvent être utilisés du clavier avec la séquence suivante :

```
LOAD GRLOAD ou LOAD GRSAVE
10 N$ = « nom de votre image »
RUN
```

Mais vous pouvez aussi insérer les quelques lignes de ces programmes dans les votres. Ces sous programmes utilisant le DOS 3.3 sont assez lents. Si vous désirez un chargement rapide de vos images graphiques, programmez plutôt en langage Pascal (chargement environ dix fois plus rapide).

8. *Différences entre les révisions A et B de la carte ève*

Des récentes informations sur le comportement de la **Double** Haute Résolution graphique sur des Apple *IIe* américains, **en couleur,** par la sortie vidéo NTSC, nous ont amenés à modifier très légèrement la carte ève et ses logiciels.

La visualisation couleur aux États-Unis, en Double Haute Résolution, est voisine du mode graphique 16 couleurs COL140 de la carte ève révision A mais avec une mauvaise correspondance des couleurs. La carte ève révision B respecte cette correspondance et assure ainsi une compatibilité plus grande avec de futurs logiciels conçus aux États-Unis pour la double haute résolution couleur.

Cette modification concerne uniquement la répartition des couleurs dans le mode graphique COL140. Les clients possesseurs d'une carte ève révision A et désireux de modifier celle-ci peuvent nous contacter pour acheter un kit de modification.

Pour assurer une totale compatibilité avec des images conçues en mode COL140 sur Apple *III* ou sur Apple *IIe* avec nos logiciels révision A (Juillet et Février 83) le programme Pascal MODFOTO a été complété. En Basic, vous pourrez taper et utiliser les quelques lignes suivantes pour modifier les couleurs d'une image affichée à l'écran en mode COL140 :

```
10 FORI=0 TO 15
20 READ A
30 POKE-26368+16*I,A
40 NEXT I
50 POKE-26074, 1
60 & BACK=0
70 & CLEAR
80 POKE-26074,0
90 FOR I=0 TO 15 : POKE-26368+16*I, I : NEXT
100 DATA 0, 8, 1, 9, 2, 10, 3, 11, 4, 12, 5, 13, 6, 14, 7, 15
```

Ces lignes de programme donnent un exemple d'utilisation de la table de commande des couleurs.

9. *A l'attention des concepteurs de logiciel*

Les concepteurs de logiciels désireux de commercialiser un produit utilisant le mode graphique COL140 peuvent nous contacter au sujet de ces problèmes de couleurs entre les cartes ève révision A et B.