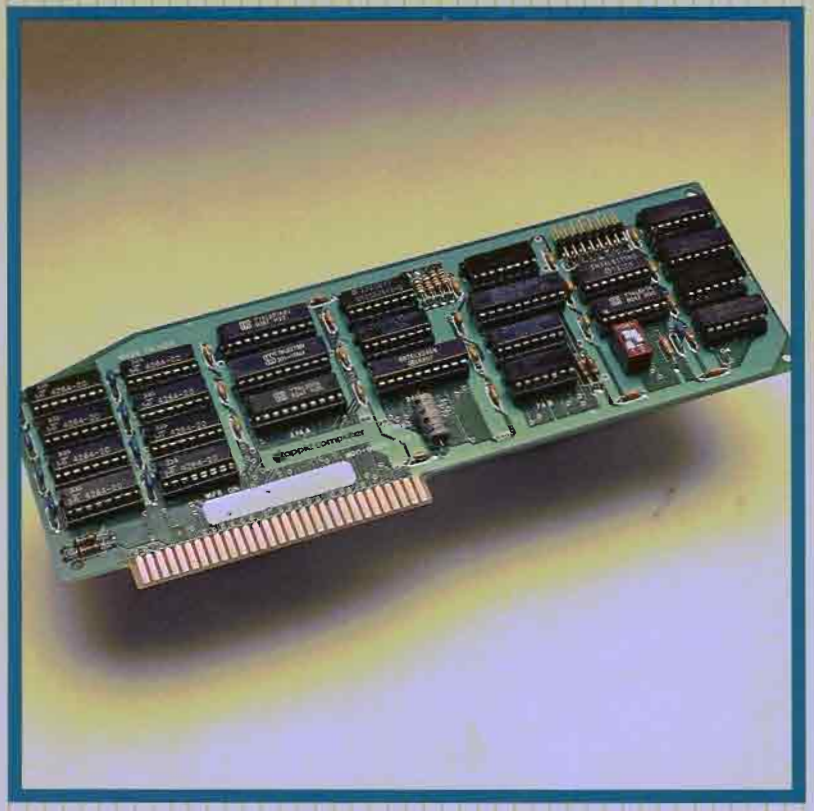
Apple

Extended 80-Column Text/ AppleColor Adaptor Card

Manual



rare
$10

Customer Satisfaction

If you discover physical defects in the manuals distributed with an Apple product or in the media on which a software product is distributed, Apple will replace the documentation or media at no charge to you during the 90-day period after you purchased the product.

In addition, if Apple releases a corrective update to a software product during the 90-day period after you purchased the software, Apple will replace the applicable diskettes and documentation with the revised version at no charge to you during the six months after the date of purchase.

In some countries the replacement period may be different; check with your authorized Apple dealer. Return any item to be replaced with proof of purchase to Apple or an authorized Apple dealer.

Limitation on Warranties and Liability

Even though Apple has tested the software described in this manual and reviewed its contents, neither Apple nor its software suppliers make any warranty or representation, either express or implied, with respect to this manual or to the software described in this manual, their quality, performance, merchantability, or fitness for any particular purpose. As a result, this software and manual are sold "as is," and you the purchaser are assuming the entire risk as to their quality and performance. In no event will Apple or its software suppliers be liable for direct, indirect, incidental, or consequential damages resulting from any defect in the software or manual, even if they have been advised of the possibility of such damages. In particular, they shall have no liability for any programs or data stored in or used with Apple products, including the costs of recovering or reproducing these programs or data. Some states do not allow the exclusion or limitation of implied warranties or liability for incidental or consequential damages, so the above limitation or exclusion may not apply to you.

Copyright

Product Revisions

Apple cannot guarantee that you will receive notice of a revision to the product described in this manual, even if you have returned a registration card received with the product. You should periodically check with your authorized Apple Dealer.

Apple Computer, Inc. 1984
20525 Mariani Avenue
Cupertino, California 95014
(408) 996-1010

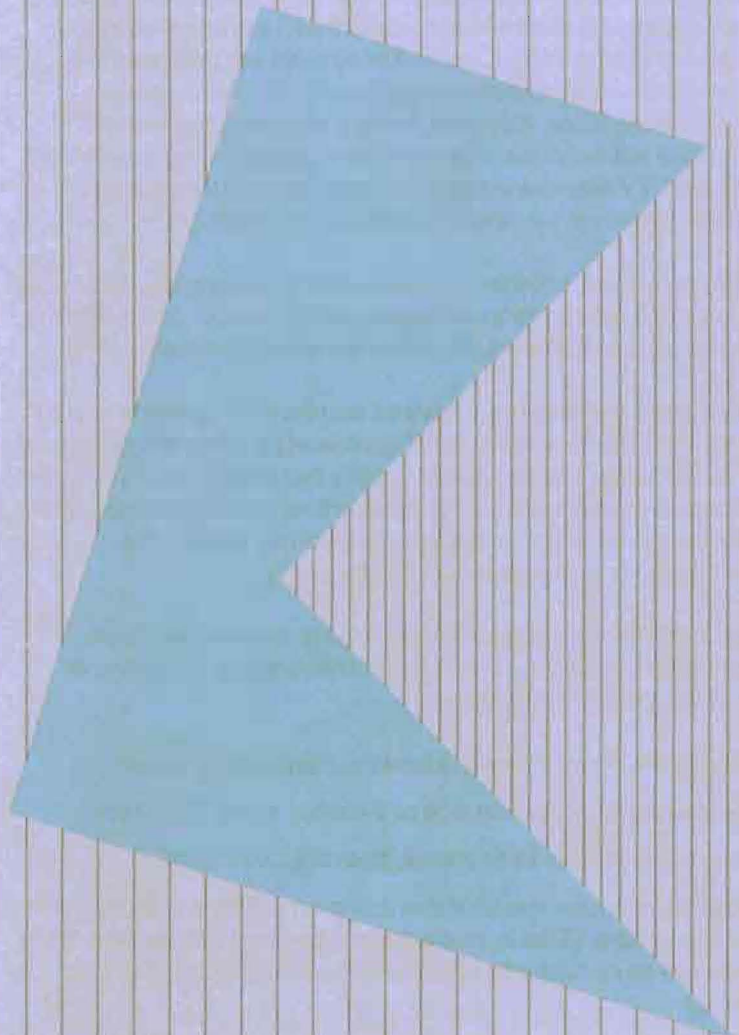
Warning

This equipment has been certified to comply with the limits for a Class B computing device, pursuant to Subpart J of Part 15 of FCC Rules. Only peripherals (computer input/output devices, terminals, printers, etc.) certified to comply with the Class B limits may be attached to this computer. Operation with non-certified peripherals is likely to result in interference to radio and TV reception.

Apple

Extended 80-Column Text/ AppleColor Adaptor Card

Manual



Radio and Television Interference

The equipment described in this manual generates and uses radio frequency energy. If it is not installed and used properly, that is in strict accordance with our instructions, it may cause interference to radio and television reception.

This equipment has been tested and complies with the limits for a Class B computing device in accordance with the specifications in Subpart J of Part 15 of FCC rules. These rules are designed to provide reasonable protection against such interference in a residential installation. However, there is no guarantee that the interference will not occur in a particular installation, especially if a "rabbit ear " TV antenna is used. (A "rabbit ear" antenna is the telescoping rod type usually contained on TV receivers.)

You can determine whether your computer is causing interference by turning it off. If the interference stops, it was probably caused by the computer or its peripherals. To further isolate the problem:

- Disconnect the peripheral devices and their I/O cables one at a time. If the interference stops, it is caused by either the peripheral or its I/O cable. These devices usually require shielded I/O cables. For Apple peripherals, you can obtain the proper shielded cable from your dealer. For non-Apple peripherals, contact the manufacturer or dealer for assistance.

If your computer does cause interference to radio or television reception, you can try to correct the interference by using one or more of the following measures:

- Turn the TV or radio antenna until the interference stops.
- Give the computer to one side or the other of the TV or radio.
- Move the computer farther away from the TV or radio.
- Plug the computer into an outlet that is on a different circuit from the TV or radio. (That is, make certain the computer and the TV or radio are on circuits controlled by different circuit breakers or fuses.)
- Consider installing a rooftop TV antenna with coaxial cable lead-in between the antenna and TV.

If necessary, you should consult your dealer or an experienced radio/ television technician for additional suggestions. You may find helpful the following booklet, prepared by the Federal Communications Commission:

"How to Identify and Resolve Radio-TV Interference Problems"

This booklet is available from the U.S. Government Printing Office, Washington, DC 20402, Stock number 004-000-00345-4.

Table of Contents

List of Figures and Tables

About This Manual

This manual describes the basic operations and use of the Extended 80-Column Text/AppleColor Card.

What This Manual Contains

This manual is arranged so that you can find the information you need, quickly and easily. Here is an overview of what this manual contains.

- Chapter 1, Getting Acquainted, describes the Extended 80-Column Text/AppleColor Card. Read it to familiarize yourself with your new peripheral controller.
- Chapter 2, Card Installation, gives you step-by-step instructions for properly installing your new card.
- Chapter 3, Card Activation and Deactivation, explains why, when, and how to activate and deactivate the card.
- Chapter 4, Memory Usage With Card, describes how the main and the auxiliary memories can be utilized with the card.
- Chapter 5, Video Display Modes, describes the various display modes.
- Chapter 6, New Double High Resolution Graphics Modes, describes how to control the new graphics modes available with the card.
- Chapter 7, Double High Resolution Graphics Software, describes the new software driver routines that support the new double-hi-res graphics modes.
- Chapter 8, Programming Examples, presents listings of the programs available on the demonstration diskette.
- Appendix A, Composite Cursor Chart, lists the various forms of the cursor with the different operational modes of the Extended 80-Column Text/AppleColor Card.

- Appendix B, Display Features, lists the effects of the different operational modes of the Extended 80-Column Text/AppleColor Card on the various display features.
- Appendix C, Escape Features, lists the standard escape features.
- Appendix D, Control-Character Features, lists the standard control-character features.
- Appendix E, ASCII, Binary, and Hexadecimal Codes, lists the standard ASCII character set with corresponding binary byte patterns and their hexadecimal and decimal equivalents.
- A glossary and index at the end of the manual help you quickly locate information about the Extended 80-Column Text/ AppleColor Card.

Who Should Read What

The following chart shows which chapters are recommended for various kinds of readers. You may fall into more than one of the categories (for example, you may be a first-time user setting up your own system).

	Chapters							
Reader	1	2	3	4	5	6	7	8
Person setting up the card	●	●						
First-time user	●	●	●					
Experienced Apple computer user		●	●	●	●	●	●	●
Programmer						●	●	●
Business person	●		●					

Aids to Understanding

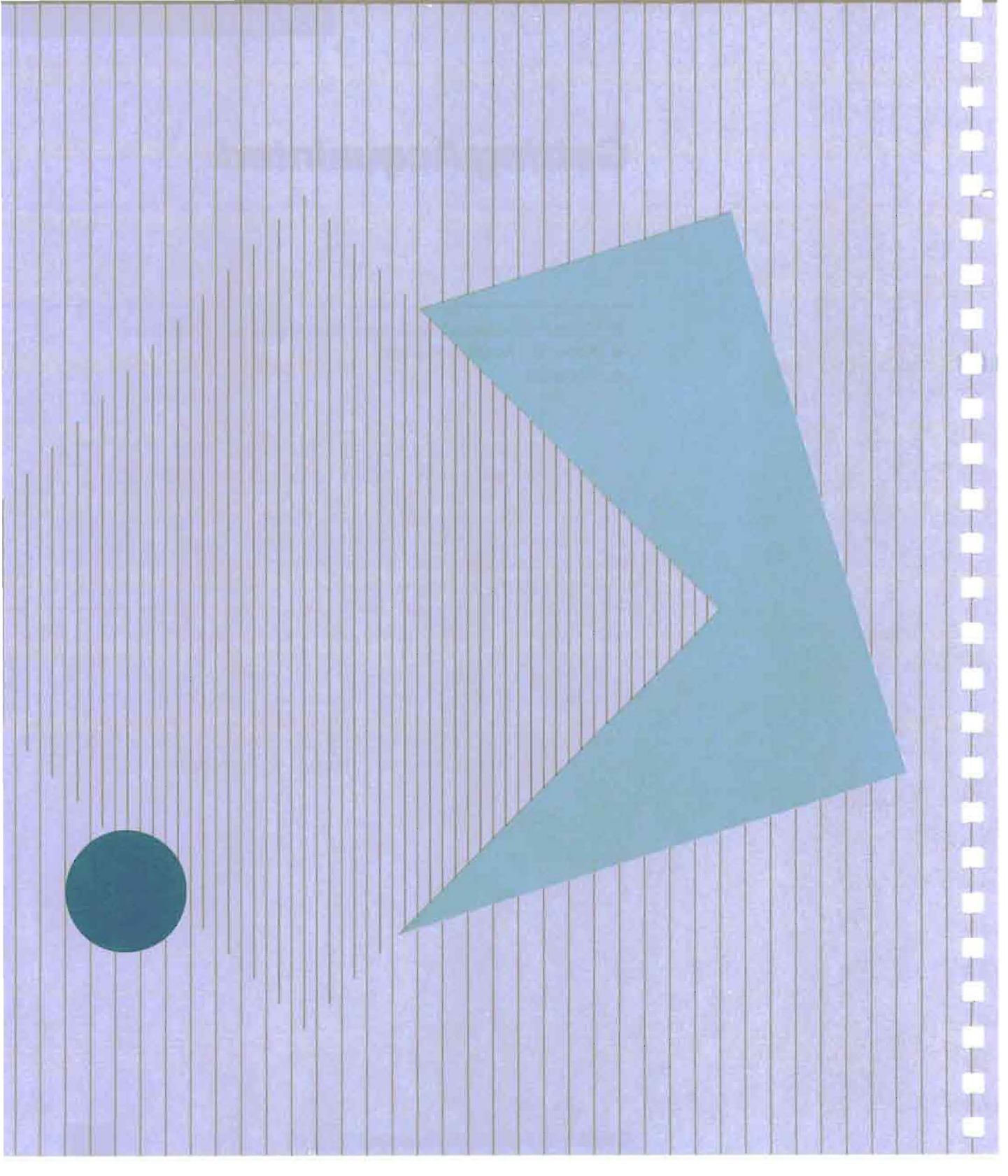
Computer jargon and words with which you may be unfamiliar are printed in *italics* when first used, and are defined in the glossary. In addition, look for these visual aids:

Gray Boxes: Gray boxes contain incidental information you may find useful.

Warning

Warning boxes indicate potential problems or disasters.

Getting Acquainted



Getting Acquainted

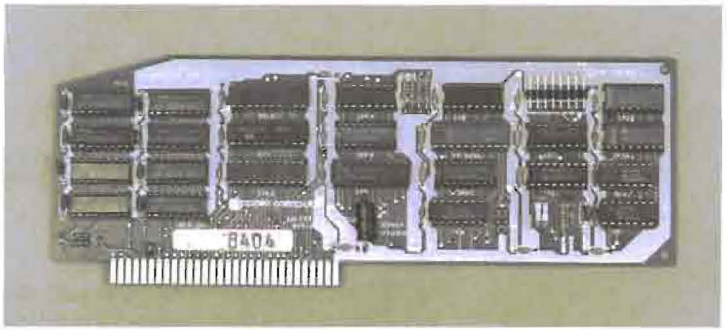
This chapter introduces you to your new Extended 80-Column Text/AppleColor Card. Several brief comparisons are made between this new card and its predecessor, and a brief description of the newly available features is given.

What the Extended 80-Column Text/AppleColor Card Does

The Apple IIe Extended 80-Column Text/AppleColor Card (see Figure 1-1) provides all the features of the Extended 80-Column Text Card; plus some new video modes and the ability to interface your Apple IIe computer to a National Television Standards Committee (NTSC) or a Red, Gren, Blue (RGB) type monitor, or to both at the same time. It also allows you to select the color of your monitor's text by the simple setting of two on-board mini-slide switches. The RGB output of your new card improves the video quality of the Apple IIe by removing the extraneous colors that occur during color transitions in the *hi-res* graphics and mixed modes.

Like the Extended 80-Column Text Card, your new card also uses the resident Apple IIe 80-column firmware, and has been designed to be software compatible with all existing and future Apple IIe software. A unique software aspect of the Extended 80-Column Text/AppleColor Card is the new Applesoft *Ampersand (&) driver routines* which it supports.

Figure 1-1. Extended 80-Column Text/AppleColor Card



About the Auxiliary Memory

Also like the Extended 80-Column Text Card, the Extended 80-Column Text/AppleColor Card has 64K bytes of *auxiliary RAM memory*. A *1K* byte area of the *auxiliary memory* is used to expand the text display to 80 columns, leaving the other 63K bytes for programs or data storage. When you use only 40 columns for your text display, then all 64K bytes of auxiliary memory are available for programs or data. With either extended 80-column card, your Apple IIe can access 128K bytes of memory. Special circuits, under software control, allow the *logical address space* provided by the 6502 microprocessor within the Apple IIe to be switched between the main and auxiliary memory.

Warning

Careless switching between main and auxiliary memory is almost certain to crash your programs. If you want to use auxiliary memory in your own programs, be sure to study the rest of this manual and the relevant information in the Apple IIe Reference Manual.

Features

For NTSC type monitors, the video output signal from your computer is through the standard video output jack. For RGB type monitors, the video signal output is through a video signal output connector which will be attached to the back of the Apple IIe during the installation of the Extended 80-Column Text/AppleColor Card. Although both types of monitors can be driven by the Extended 80-Column Text/AppleColor Card, even at the same time, the full capabilities of the card cannot be appreciated unless an RGB type monitor is used.

Any one of the following standard display format options may be selected with either of the extended 80-column cards:

- 40 column by 24 line text
- 80 column by 24 line text
- 16 color lo-res with option of mixing 40 column text
- 16 color lo-res with option of mixing 80 column text
- 6 color hi-res with option of mixing 40 column text
- 6 color hi-res with option of mixing 80 column text
- Monochrome 560 pixel X 192 pixel with option of mixing 80 column text in bottom four lines.

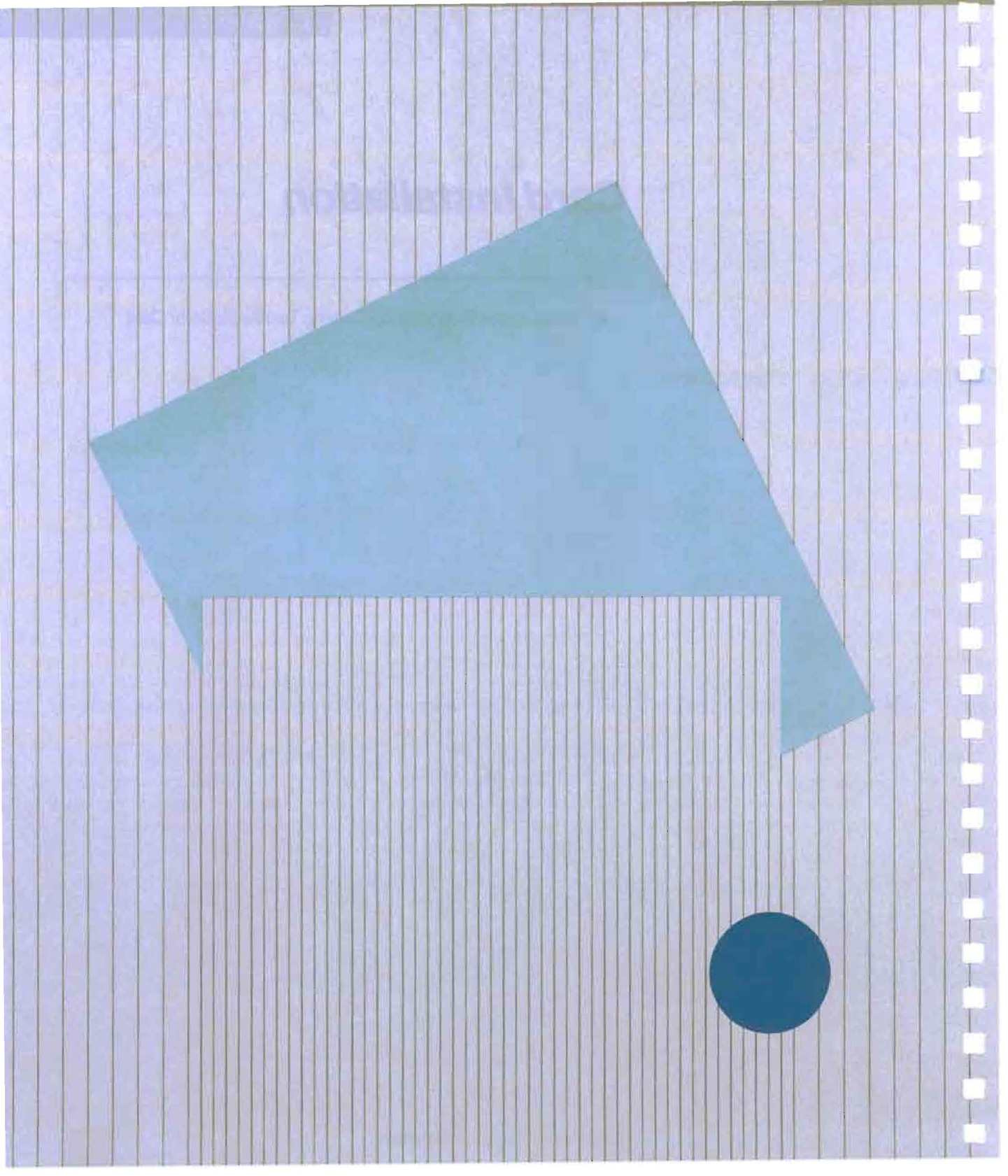
The Extended 80-Column Text/AppleColor Card, unlike the Extended 80-Column Text Card, also provides the following new display format options for RGB type monitors:

- Mode 1: 560 pixels X 192 pixels with monochrome only
- Mode 2: 140 pixels X 192 pixels with full 16-color capability
- Mode 3: mix 560x192 pixels and 140x192 pixels anywhere on screen

Your Extended 80-Column Text/AppleColor Card can also simulate monitors of different color phosphors when used with an RGB type monitor. If you are accustomed to doing your word processing in green, blue, amber, or white, you can select the desired text color by setting two mini-slide switches. However, highlighting of text (i.e., inverse video) is always white.

A special set of Applesoft BASIC Drivers has been implemented as ampersand (&) routines to enhance the usefulness of the Extended 80-Column Text/AppleColor Card. They are described in Chapter 7 of this manual.

Card Installation



Card Installation

This chapter describes and depicts the installation of your Extended 80-Column Text/AppleColor Card.

Installing the Extended 80-Column Text/ AppleColor Card

Your Extended 80-Column Text/AppleColor Card can be installed in only a few minutes. Read the following directions and do as they say, step by step. Be sure to fill out and mail the warranty card that came packed in the box with your Extended 80-Column Text/AppleColor Card (if you have not already done so).

1. Take the Extended 80-Column Text/AppleColor Card out of its protective bubble wrap and place it on a secure flat surface, chip side up. Look at the card and carefully identify those parts of it that are identified in Figure 2-1.

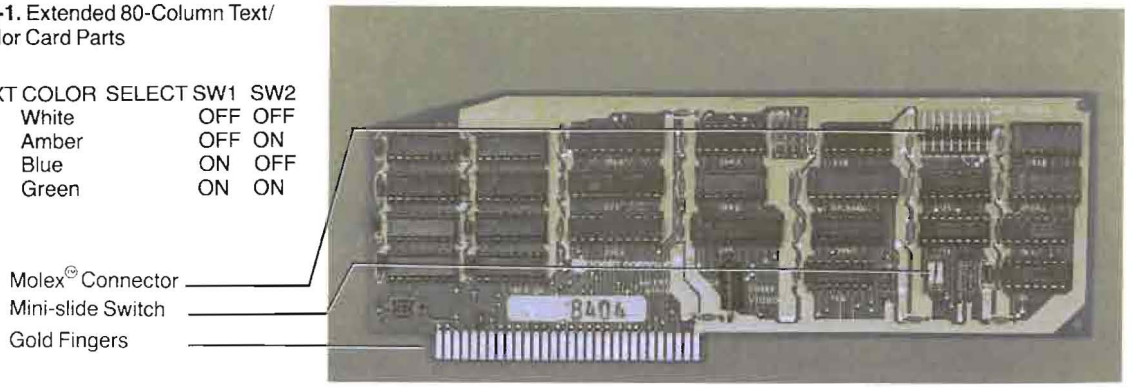
   Along the bottom edge of the card is a row of gold fingers. This is the edge that you will insert into the 60-pin auxiliary slot which is inside your Apple IIe. Do not touch any of these gold fingers; moisture left by your hands will attract dust, which can cause a defective connection.

2. At this time you can set the mini-slide switches SW1 and SW2 on the Extended 80-Column Text/AppleColor Card for the color of text that you prefer. Refer to the chart next to Figure 2-1.

3. Before you do anything else, make sure that you turn off the power to your Apple IIe. Do not unplug your Apple IIe, just turn it off. If you unplug the Apple IIe, you isolate it from earth ground and leave it vulnerable to static charge damage.

Figure 2-1. Extended 80-Column Text/ AppleColor Card Parts

FOR TEXT COLOR SELECT	SW1	SW2
White	OFF	OFF
Amber	OFF	ON
Blue	ON	OFF
Green	ON	ON

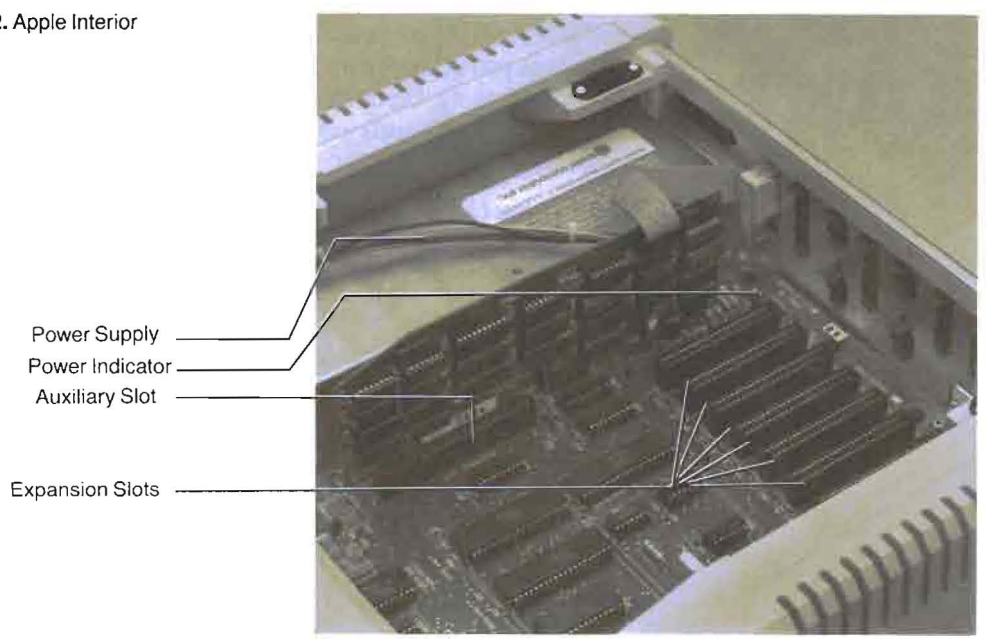


Warning

It is essential that you turn off the Apple IIe's power before installing, removing or disconnecting anything. This prevents possible damage not only to the circuits of your IIe and the Extended 80-Column Text/AppleColor Card, but also to you.

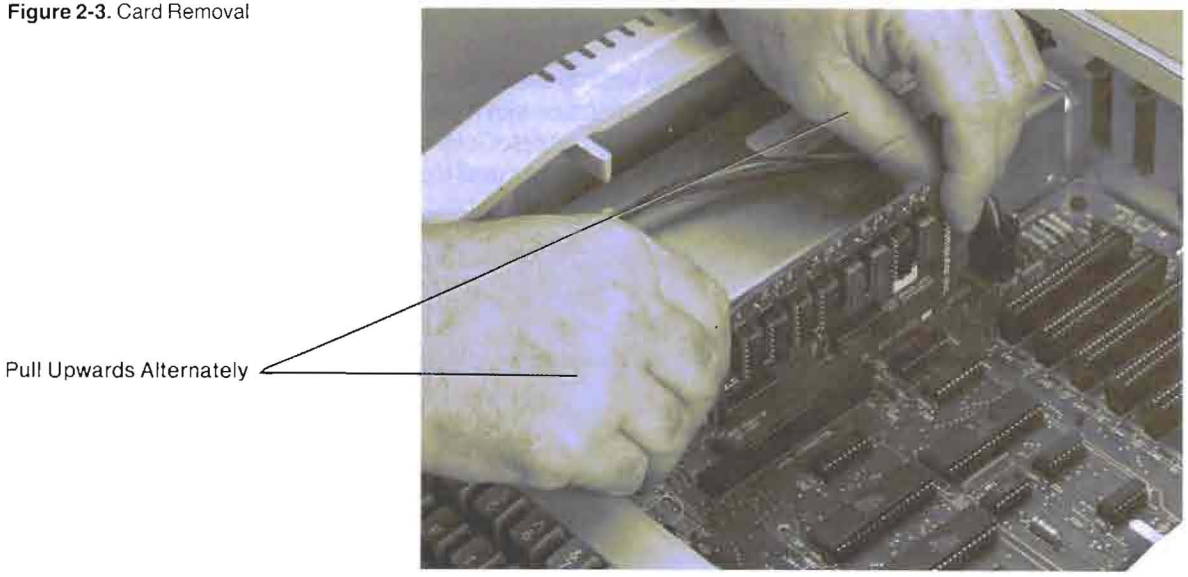
4. Remove the cover from your Apple IIe by pulling up on the rear edges of the cover until it pops off. Slide the cover away from the keyboard and put it in a safe place for a few minutes.
5. Touch the power supply cover (the big gold or silver metal box inside the Apple IIe, refer to Figure 2-2) to discharge any static electricity you may be carrying on your clothes or body.

Figure 2-2. Apple Interior



6. Check inside the Apple IIe to be sure that the little red power indicator in the back isn't glowing, refer to Figure 2-2. If it is, the power is on (be sure to turn it off). This light provides an extra warning which is helpful if you get interrupted in the middle of removing or installing something, and forget to turn off the power.
7. Find the auxiliary expansion slot inside your Apple IIe. This slot is labeled AUX. CONNECTOR. When you are facing the keyboard, this slot is located to the left front side of the main logic board, next to the power supply. Refer to Figure 2-2.
8. If you already have an 80-Column or an Extended 80-Column Text Card in the auxiliary expansion slot, you may now remove it by grasping it firmly on both its nearest and farthest upper edge and gently pulling upward, first on one edge, then on the other. Refer to Figure 2-3.

Figure 2-3. Card Removal



9. The internal cable connector and screw plate assembly, shown in Figure 2-4, provides the connection between the Extended 80-Column Text/AppleColor Card and your Apple IIe.

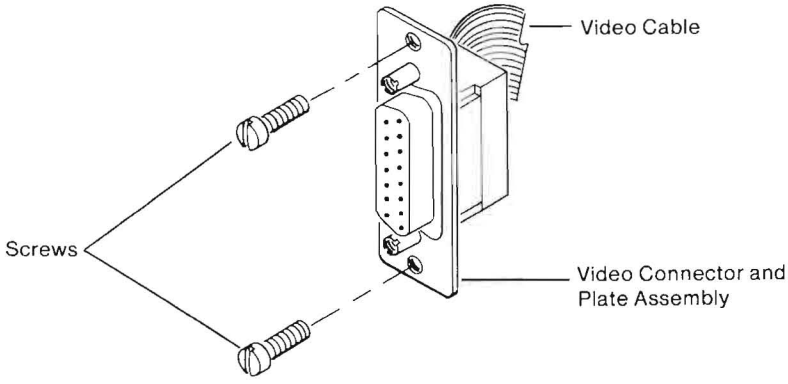


Figure 2-4. Internal Cable Connector/ Screw Plate Assembly

10. Select an appropriate size opening in the Apple IIe back panel near the expansion slot where you'll be installing the Extended 80-Column Text/AppleColor Card. Press on the tab of the opening's cover plate until it pops out. Refer to Figure 2-5.

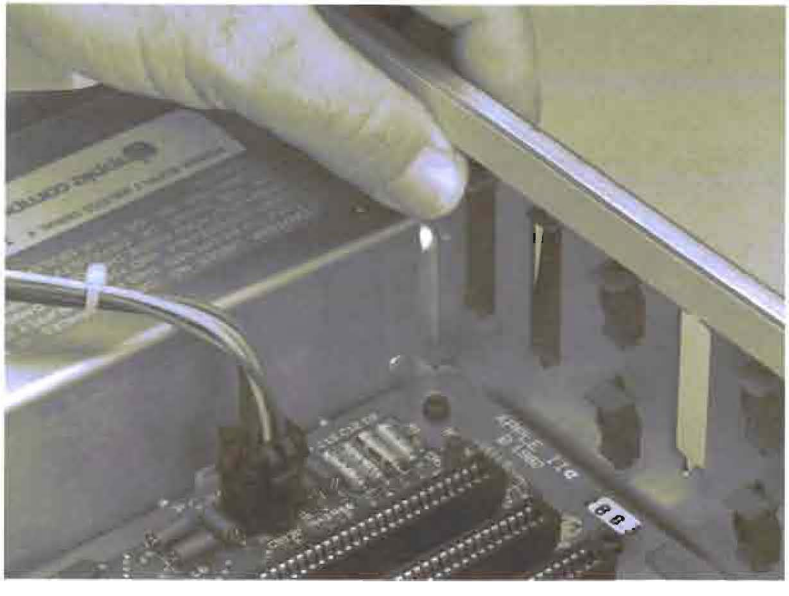
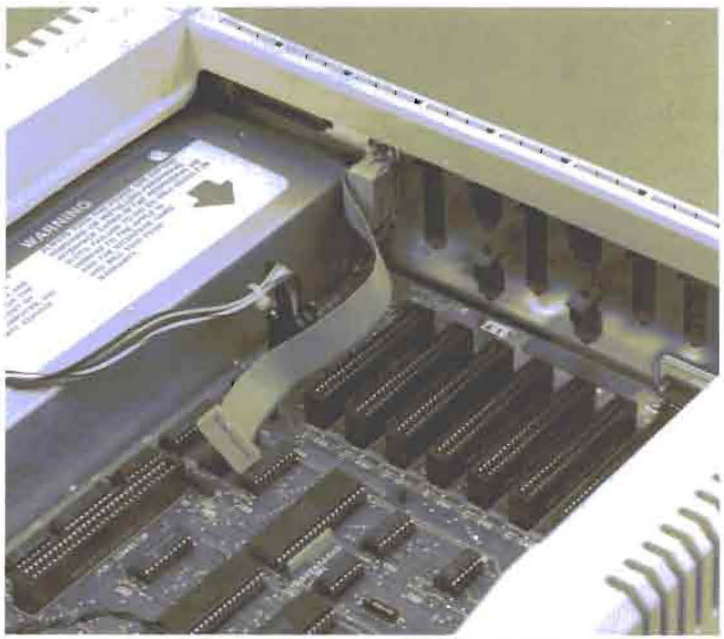


Figure 2-5. Cover Plate Removal

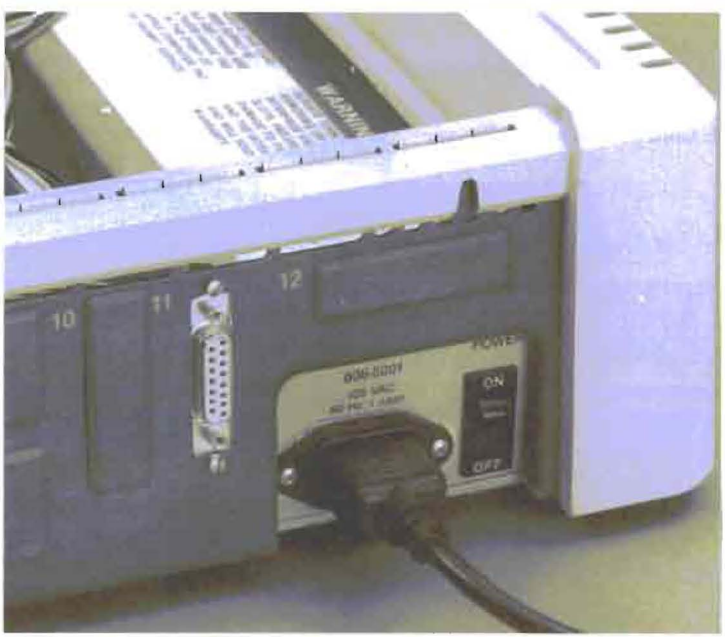
11. Install the internal cable connector and screw plate assembly as shown in Figure 2-6 with one screw in the lower notch of the Apple IIe back panel opening. Line-up the plate so that the upper hole is aligned with the upper notch of the back panel opening, and then insert and tighten the upper screws.

Figure 2-6. Internal Cable Connector and Screw Plate Installation



Note: Figure 2-7 shows the Video Connector Installed in your Apple IIe.

Figure 2-7. Video Connector/ Assembly Installed

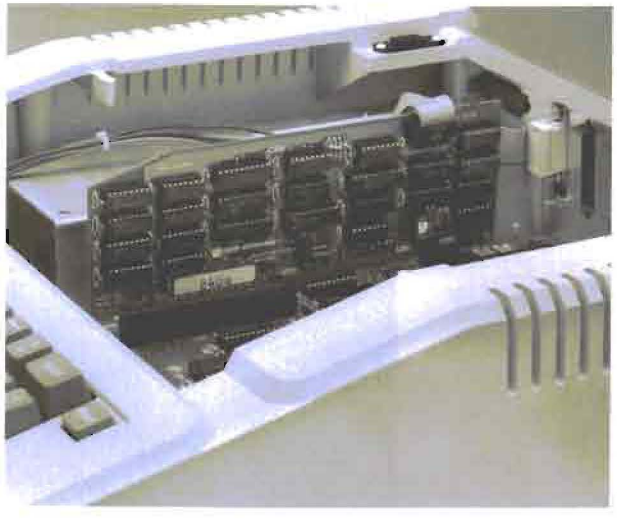


12. Pick up the Extended 80-Column Text/AppleColor Card and, holding it carefully without touching the gold fingers, position it over the auxiliary expansion slot as shown in Figure 2-8. Guide the card carefully into the edge connector by gently rocking it back and forth to get it started, and then firmly pressing down on that portion of the top edge of the card that is directly above the edge connector until the card is fully seated.

Figure 2-8. Install Card

13. Now connect the Molex connector end of the internal cable assembly onto the Molex pins on the card so that all of the pins are connected, and so that the Molex connector is properly installed as shown in Figure 2-9. Be sure that the cable crosses flat over the edge of the card and that there are no kinks in it.

Figure 2-9. Card Installed

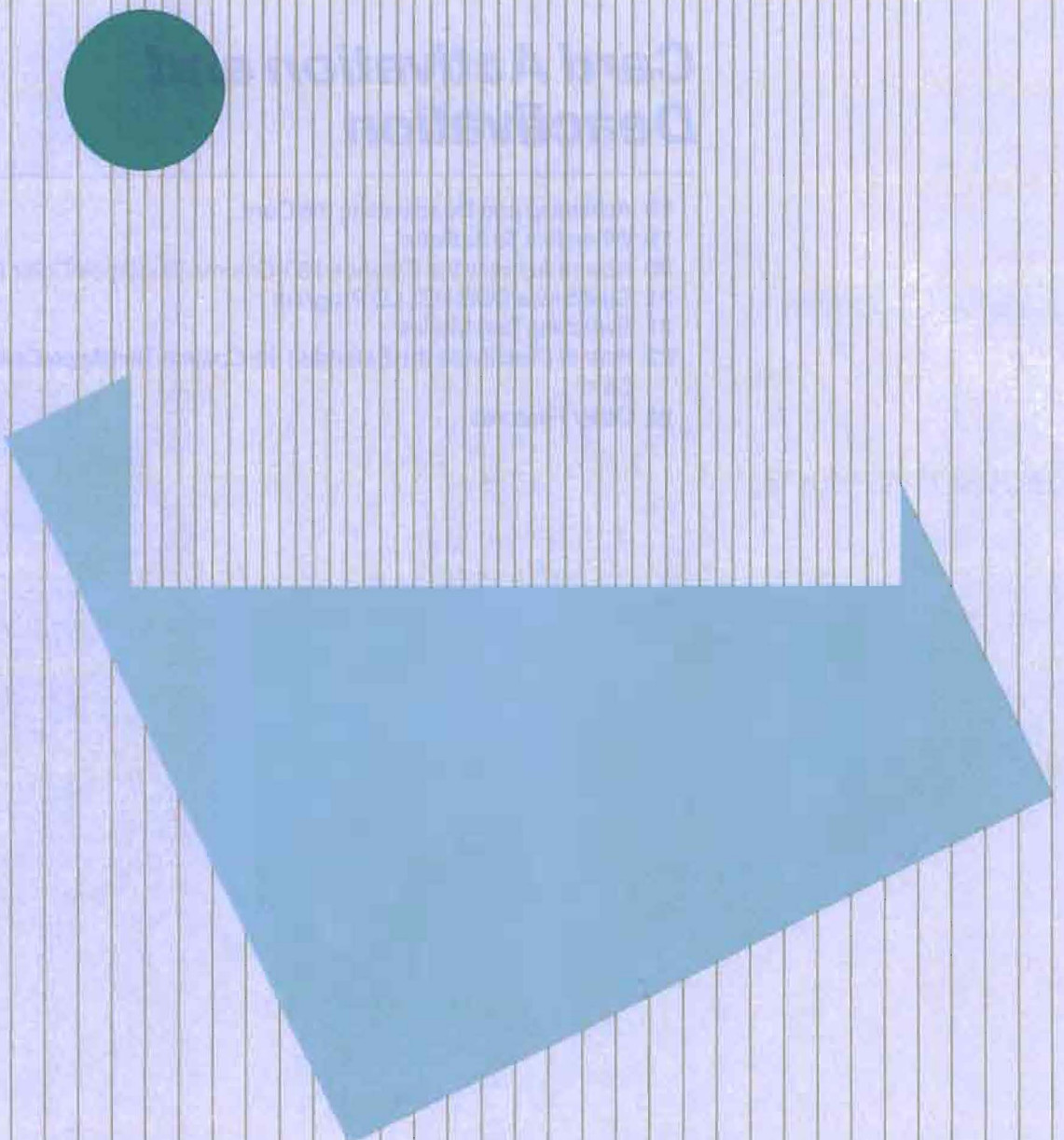


14. The Extended 80-Column Text/AppleColor Card is now completely and properly installed. Now replace the cover on your Apple IIe by inserting the front lip of the cover under that part of the chassis just above the keyboard, guiding the cover into place, and pushing down firmly on its back corners until the lid snaps back into place.

15. You can now connect your RGB type monitor cable to the video signal output connector of the Extended 80-Column Text/ AppleColor Card, turn on the power to your monitor and computer, and start up (i.e., *boot*) your system with the *demo* disk or any startup disk.

When you are satisfied that the Apple IIe Extended 80-Column Text/ AppleColor Card is properly installed, go on to the next chapter to learn about how to use your new card.

Card Activation and Deactivation



Card Activation and Deactivation

This chapter discusses why you should (or should not) activate your new card, and describes when and how to activate or deactivate it.

Activating and Deactivating the Card

The Extended 80-Column Text/AppleColor Card must be activated in order to be used, but you do not have to activate it yourself, although you might have to deactivate it. Read on!

When Not to Activate

So, you've got your Extended 80-Column Text/AppleColor Card installed successfully. Now what? Well, that depends on how you want to use your card. Even though it is plugged in and the power has been turned on, that does not mean that the card is available yet for your use. It must first be activated. However, if you do not intend to do any programming, you do not have to worry about activating it because your application programs that utilize it will automatically do that for you.

In fact, if you do not intend to do any programming at all, you may stop reading this manual now, and still reap the full benefits and joys of your new Extended 80-Column Text/AppleColor Card. It would be advisable though to also read the section in this chapter on how to deactivate the card, because you might need to deactivate it if one of your application programs leaves it on, and because you will have to deactivate it when you want to switch any output from the monitor screen to your printer.

You can always tell when the Extended 80-Column Text/AppleColor Card is deactivated by inspecting the cursor (see Appendix A). If the cursor is a blinking square checkerboard pattern, then the card is deactivated.

If you are going to program with your Extended 80-Column Text/AppleColor Card, but you are going to program in either Pascal or CP/M, you still do not have to concern yourself about how to activate your card because it will also be automatically activated for you when you boot Pascal or CP/M.

How to Activate the Extended 80-Column Text/AppleColor Adapter Card

After all this fuss, you might be surprised to learn just how very easy it is to activate your Extended 80-Column Text/AppleColor Card. The only thing you have to do is enter the following command:

```
PR#3
```

"PR" stands for *peripheral* and "#3" stands for *expansion slot* number three, so the command PR#3 means to the computer, "Activate the card in the peripheral expansion slot number three." In fact, entering PR#n into the Apple IIe, where n can be any number from one to seven (representing one of the seven peripheral expansion slots), will activate the card in that expansion slot.

By The Way: Perhaps you've already recalled that the Extended 80-Column Text/AppleColor Card is in the auxiliary slot, not expansion slot number three. Good! You see, the 80-column text cards for the original Apple II were designed to be installed in slot three for compatibility with the Apple Pascal Operating System, and so the resident 80-column firmware references that slot. In order to maintain software compatibility between newer products and the older system resident firmware, it was necessary to design the auxiliary slot as a substitute slot for expansion slot three, but to reference the auxiliary slot as though it was expansion slot three. This allows you to use either the auxiliary slot or expansion slot three, but not both.

Warning

Whenever you have a peripheral card in the auxiliary slot, you must not also have a card in expansion slot three, and conversely, or you may damage both cards. Always deactivate the Extended 80-Column Text/AppleColor Card before sending output to a printer or before activating any other expansion slot; otherwise, unpredictable results may occur and your program may be damaged.

When the Extended 80-Column Text/AppleColor Card is activated, the cursor changes shape to a solid lined rectangle about half its original width and you can now type 80 characters to each line.

Customize DOS Hello Program

If you do not care to have to activate your Extended 80-Column Text/AppleColor Card every time you want to use it, you can automatically activate it every time you boot the Apple IIe by simply adding the PR#3 command to the DOS HELLO program. To add the PR#3 command to the DOS HELLO program, follow these very easy steps:

1. Boot the Apple IIe with the DOS 3.3 System Master disk and make an unprotected copy of the DOS 3.3 System Master disk.
2. Insert your write-non-protected copy of the DOS 3.3 System Master disk into drive 1, and enter the following command:

   ```
   LOAD HELLO
   ```

3. After the HELLO program has been loaded, enter the following BASIC commands (be sure to use line numbers that have not already been used):

   ```
   1 D$ = CHR$(4)
   2 PRINT D$; "PR#3"
   ```

4. Now you'll need to alter the security of the HELLO file on disk, so that you can overwrite it with your newly modified HELLO program. Then you'll have to actually overwrite that older HELLO program with your new one. Finally, you'll need to resecure the new program which you just saved on the disk so that it is not inadvertently modified at some later time. To do all of these things, simply enter the following commands:

   ```
   UNLOCK HELLO
   SAVE HELLO
   LOCK HELLO
   ```

That's it! Nothing to it! Now whenever you boot your Apple IIe with this disk, you will automatically activate the Extended 80-Column Text/AppleColor Card. You can forget about having to enter PR#3!

Switching Text Modes

While the Extended 80-Column Text/AppleColor Card is active, you may switch between 80-column and 40-column displays without deactivating it by the use of two simple commands.

What you enter...	What you get...
ESC 4	Switches an 80-column display to a 40-column display
ESC 8	Switches a 40-column display to an 80-column display

By The Way: Notice that when you hit the ESC key, a + sign appears in the center of the cursor. This indicates that the Apple IIe is in the escape mode and is awaiting a legitimate escape command (Table C-1 lists the escape commands in Appendix C). Also notice that when you enter the 40-column display mode by switching, the cursor doubles in size to form a square of solid lines, rather than a blinking checkerboard. This indicates that the Extended 80-Column Text/AppleColor Card is still activated, even though you are not in the 80-column text mode (see Appendix A).

How to Deactivate the Extended 80-Column Text/AppleColor Card

To deactivate the Extended 80-Column Text/AppleColor Card, enter the following command:

ESC CONTROL-Q

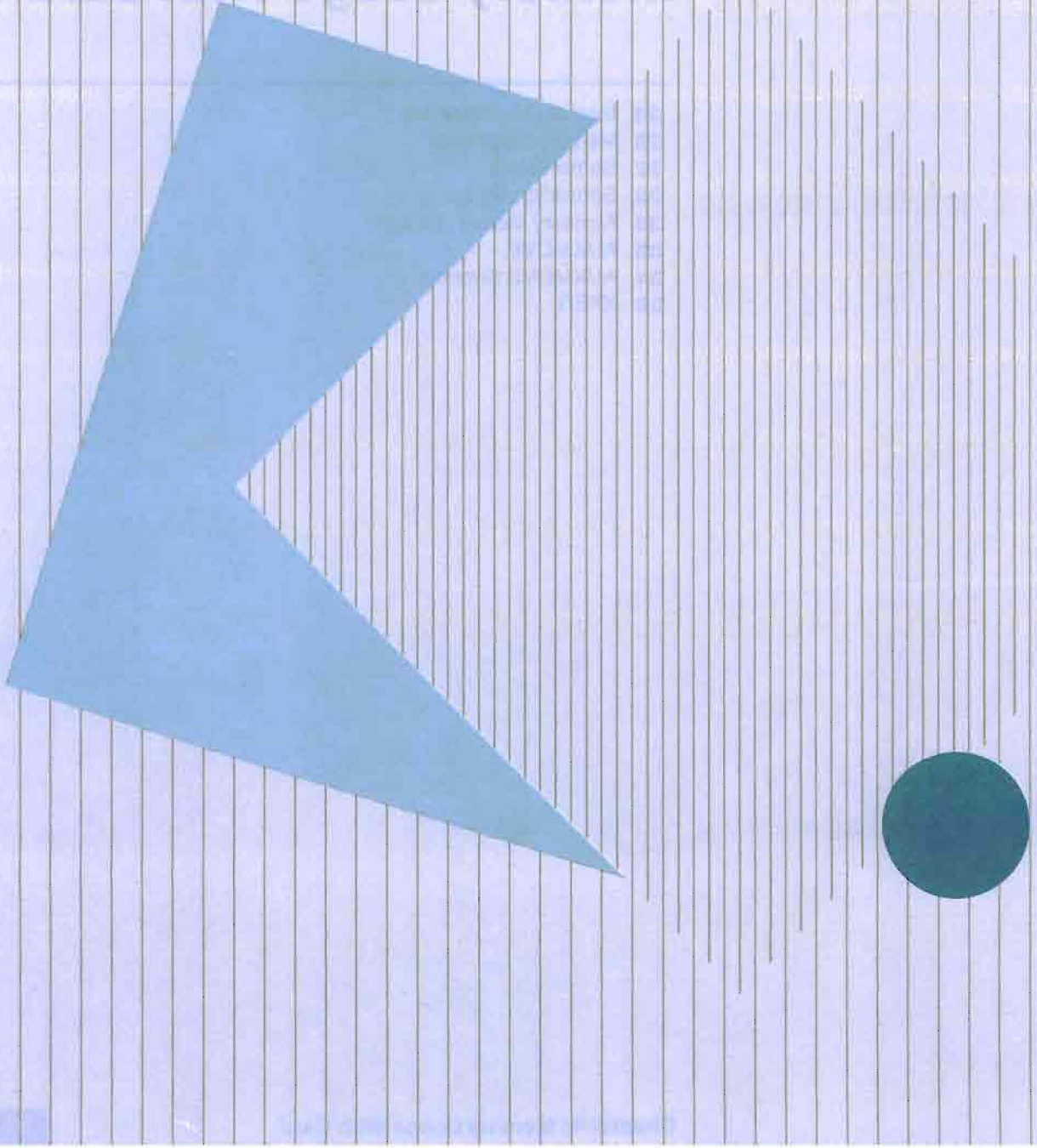
By The Way: Notice that the cursor now returns to a blinking checkerboard square, indicating not only that the 40-column display mode is active, but also that the Extended 80-Column Text/AppleColor Card has been deactivated. Table A.1 in appendix A is a convenient summary of the various forms of the cursor.

Another method by which to deactivate the Extended 80-Column Text/AppleColor Card is to enter CONTROL-RESET, but this is not recommended because it can possibly damage any program you may happen to have in *read-write* memory. The first method of deactivating the Extended 80-Column Text/AppleColor Card (i.e., ESC CONTROL-Q) will never affect any programs in memory, and is therefore the recommended method.

Other Features

Refer to appendixes B, C, and D for descriptions of other display, escape, and control function features of the Extended 80-Column Text/AppleColor Card.

Memory Usage With Card



Memory Usage With Card

This chapter describes the various ways that memory can be utilized with the Extended 80-Column Text/AppleColor Card.

Memory Management

It is important to realize that, in general, each byte of *physical memory* must have a unique address in order to be correctly accessed. The CPU of the Apple IIe computer, the 6502 microprocessor, can address only 64K bytes of memory. To access more than 64K bytes of memory in the Apple IIe, certain memory segments are overlapped with each other as indicated in Figure 4-1. Customized integrated circuits, namely a Memory Management Unit (MMU), an Input/Output Unit (IOU), and a Programmed Array Logic Unit (PAL), prevent ambiguous addressing by controlling the logical address space in such a way that only one byte of physical memory is uniquely accessed by any logical address. Refer to *Chapter 7* of the *Apple IIe Reference Manual* for details.

Operationally, the MMU and the IOU provide several *softswitches* that allow you to control just which part of memory you are going to use, and, to a certain extent, just how you are going to use it. From a user's point of view, a softswitch is simply a pair of reserved addresses which, when read from or written to, cause one of the customized integrated circuits to switch part of the logical address space from one segment of physical memory to another.

Memory Segments

The *memory map* of Figure 4-1 shows the functions of the major segments of the memory address space for each part of physical memory when an extended 80-column card, with 64K bytes of auxiliary memory, has been installed in your Apple IIe computer. As the map indicates, certain different segments of memory are

specified by the same set of addresses. It is important to realize that these overlapping segments of physical memory cannot all be addressed at the same time. Their logical address space must be switched from one segment of physical memory to another by reading or writing an appropriate combination of softswitches, as described in the next section.

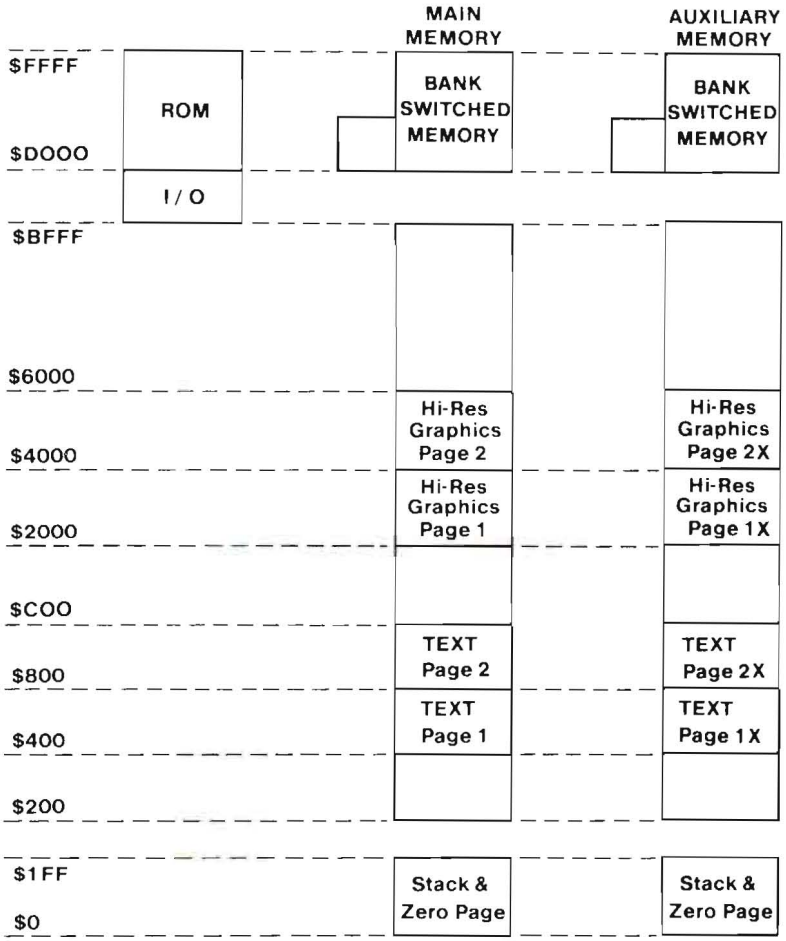
Notice the 4K byte discontinuity in both the main and auxiliary read-write memory from address $C000 to $CFFF. As indicated in Figure 4-1, that 4K byte address segment has been dedicated to input/output (I/O) control functions, implemented as *firmware* in ROM memory. The two 4K byte segments of main and auxiliary read-write memory which would have been assigned the addresses from $C000 to $CFFF, have instead been assigned to the set of addresses from $D000 to $DFFF as indicated by the small blocks to the left of the both bank switched memories in Figure 4-1. This displacement makes those 8K bytes of physical memory accessible through a 4K byte address space that has already assigned to 4K bytes of system ROM and to another 8K bytes of main and auxiliary read-write memory. The control of that address space is described in detail in *Chapter 4* of the *Apple IIe Reference Manual*.

In summary, the top 12K bytes of address space from $D000 to $FFFF initially provide access to the system ROMs that contain the system *monitor* and the *Applesoft interpreter*. Special softswitches permit access to the different banks of physical memory. For example, when the Pascal operating system is loaded, it is placed in the top 12K bytes of main read-write memory, and the system ROMs in that address space are disabled. The top 12K bytes of auxiliary memory are normally disabled, but they can be accessed as indicated in the section on softswitches.

The bulk of physical memory is read-write memory that can be accessed through the 48K byte address space from $200 to $BFFF. Softswitches control which physical memory occupies this address space (main or auxiliary) and also just which section of that address space will provide support for the display screen at any given time (refer to Figure 4-1). Those softswitches are also described in the next section.

Finally, as also indicated in Figure 4-1, address space $0 to $1FF is dedicated to read-write memory which supports certain special dynamic requirements of the system. Again, the softswitches that control which physical memory is accessed, main or auxiliary, are described in the next section.

Figure 4-1. Memory Segments



Softswitches

Each softswitch has at least two addresses associated with it. There is always one to turn it on, and one to turn it off. Sometimes there is one to relate its status—to tell you whether it is on or off (bit 7 is a 1 if the switch is on). All of the softswitches available with the Extended 80-Column Text Card are also available with the Extended 80-Column Text/AppleColor Card. Table 4-1 lists the names of those softswitches that allow you to switch different parts of the logical address space between the various segments of physical memory, along with their activating, deactivating, and status addresses, and a brief description of the functions they provide.

Figures 4-2 through 4-5 graphically depict the effect of each softswitch on the available physical memory, and illustrate the interactive effects of these softswitches; study them carefully. Table 4-2 summarizes both the effects of these softswitches on the different segments of physical memory and their interdependence.

Table 4-1. Memory Softswitches

Name	Function	Location Hex	Location Decimal	Notes
RAMRD	Read auxiliary memory	$C003	49155 -16381	Write
	Read main memory	$C002	49154 -16382	Write
	Read RAMRD switch	$C013	49171 -16365	Read
RAMWRT	Write auxiliary memory	$C005	49157 -16379	Write
	Write main memory	$C004	49156 -16380	Write
	Read RAMWRT switch	$C014	49172 -16354	Read
80STORE	On access display page	$C001	49153 -16383	Write
	Off use RAMRD RAMWRT	$C000	49152 -16384	Write
	Read 80STORE switch	$C018	49176 -16360	Read
PAGE2	Page 2 on (Aux memory)	$C055	49237 -16299	1
	Page 2 off (Main memory)	$C054	49236 -16300	1
	Read PAGE2 switch	$C01C	49180 -16356	Read
HIRES	On access hi-res pages	$C057	49239 -16297	2
	Off use RAMRD RAMWRT	$C056	49238 -16298	2
	Read HIRES switch	$C01D	49181 -16355	Read
ALTZP	Auxiliary stack & z. p.	$C009	49161 -16373	Write
	Main stack & zero page	$C008	49160 -16374	Write
	Read ALTZP switch	$C016	49174 -16352	Read

(1) When 80STORE is on, the PAGE 2 switch selects main or auxiliary display memory.

(2) When 80STORE is on, the HIRES switch enables you to use the PAGE 2 switch to switch between the high-resolution page-1 area in main memory or auxiliary memory.

Figure 4-2. Effect of Switching RAMRD and RAMWRT With 80STORE Off

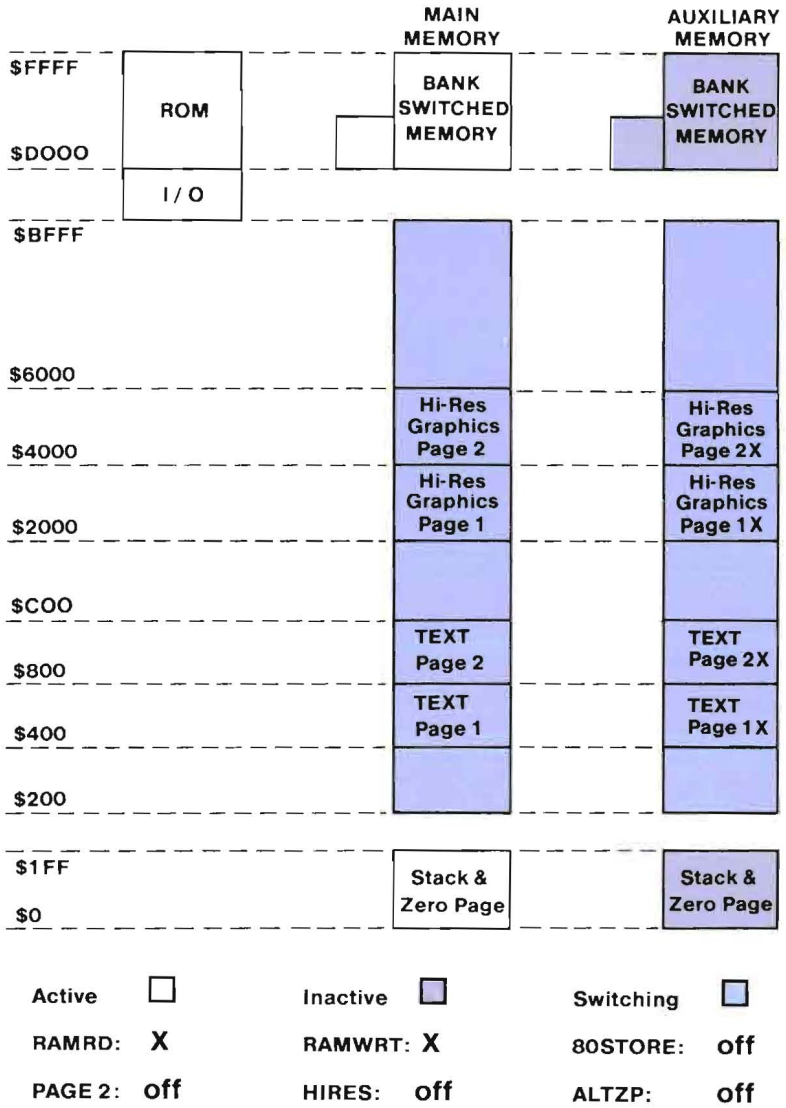


Figure 4-3. Effect of Switching RAMRD and RAMWRT With 80STORE and HIRES On

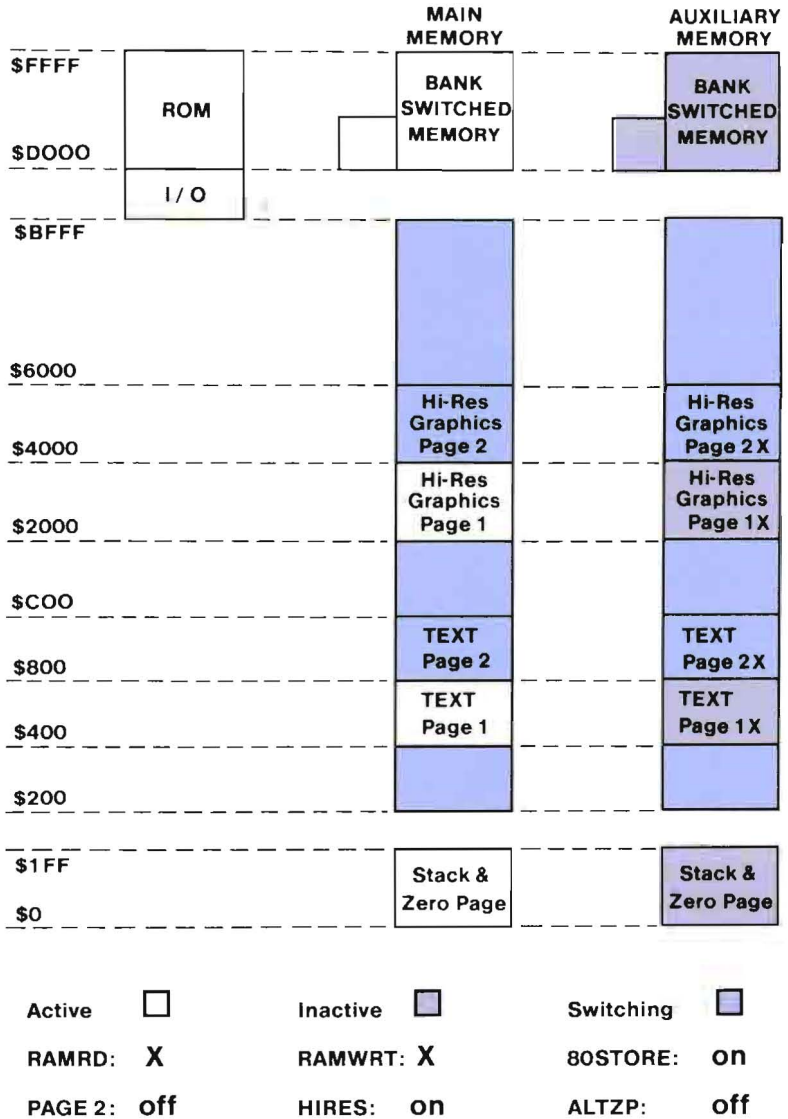


Figure 4-4. Effect of Switching ALTZP

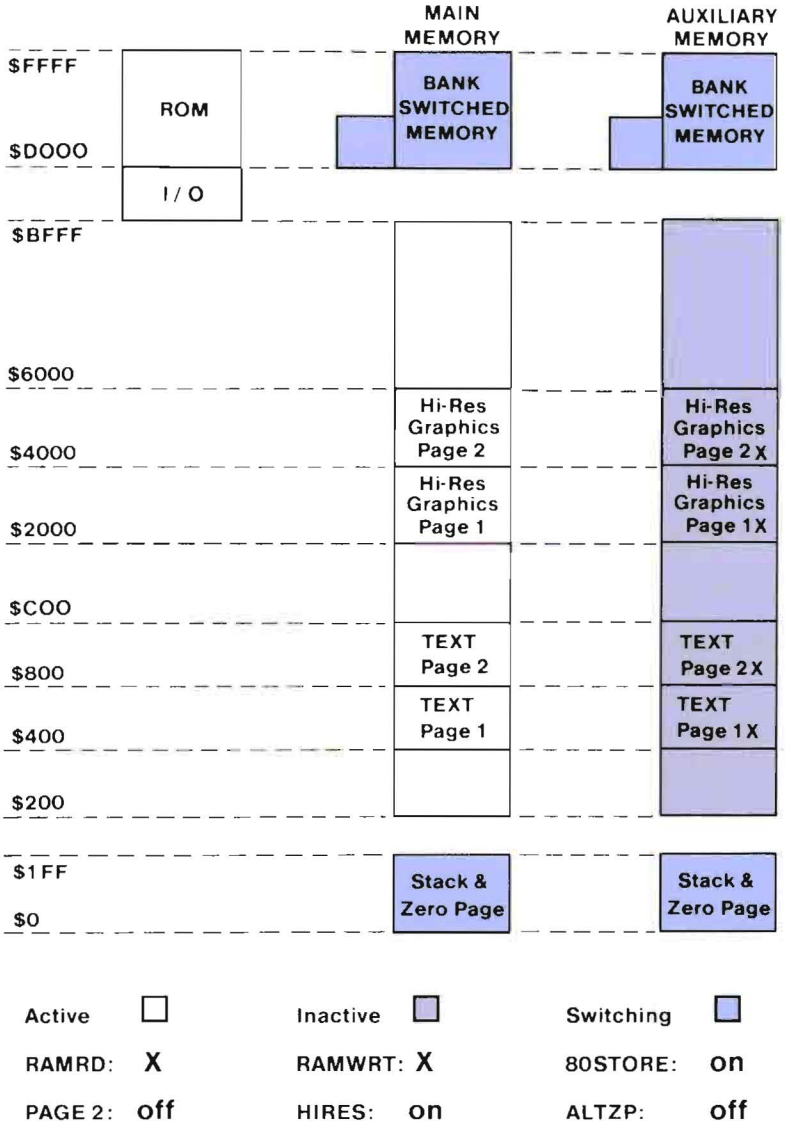


Figure 4-5. Effect of Switching PAGE2 With 80STORE and HIRES On

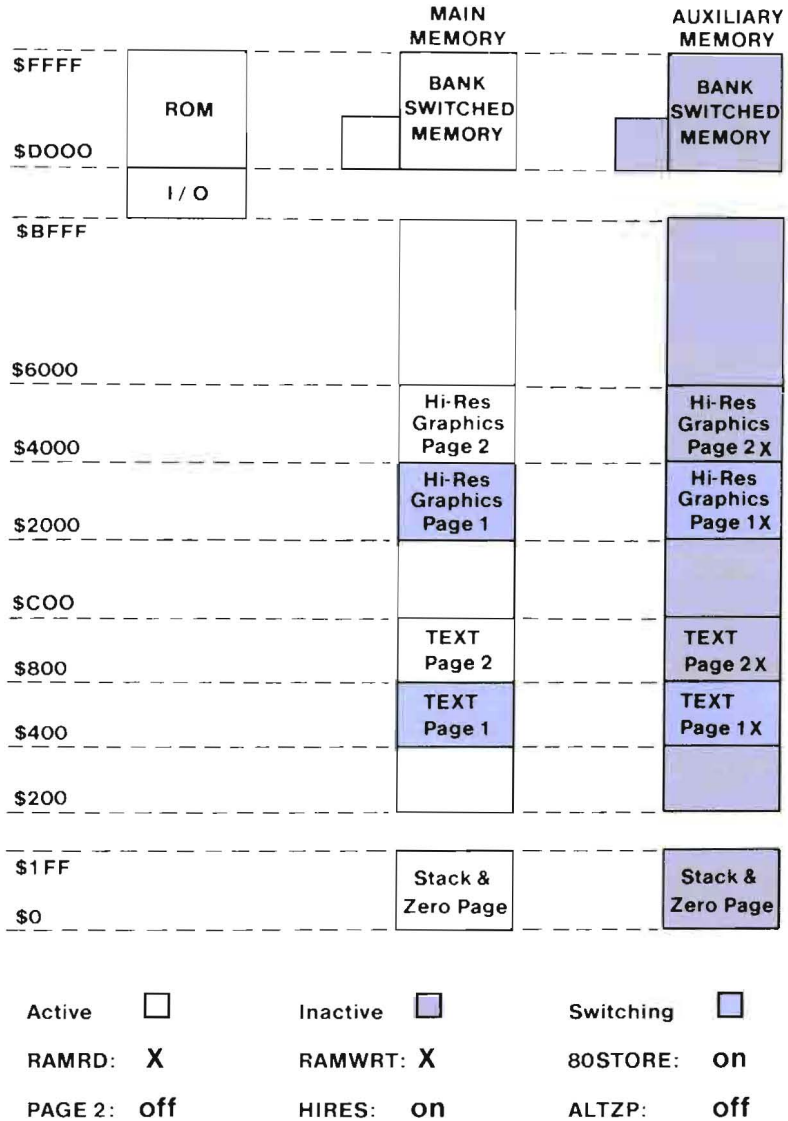


Table 4-2. Interrelated Softswitch Controls Summary

80STORE		ON		OFF
HIRES		ON	OFF	Use softswitches RAMRD and RAMWRT to switch read and write activities, respectively, between auxiliary memory (softswitch ON) and main memory (softswitch OFF).
	O N	Switch-in AUX $2000 - $3FFF & $400 - $7FF	Switch-in AUX $400 - $7FF	
PAGE2				
	O F F	Switch-in main $2000 - $3FF & $400 - $7FF	Switch-in main $400 - $7FF	
		Hi-res Page 1X or Page 1 & Text Page 1X or Page 1	Text Page 1X or Text Page 1	
		NOTE: With 80STORE on, neither RAMRD nor RAMWRT has any effect on the display page porttions of either auxiliary or main memory.		

Softswitch Usage

The 80STORE, HIRES, and PAGE2 softswitches are used primarily for addressing display data. Since these switches override the general purpose switches, they must be properly set even when your program does not use them (i.e., they must be off or reset if you intend to use either RAMRD or RAMWRT). In fact, the correct usage of the available softswitches is the key to designing properly functioning programs that utilize the 80-column or extended 80-column cards.

Auxiliary Memory Usage

Although there are various softswitches that switch the logical address space between selectable segments of physical memory, that does not mean that you can simply write into the selected memory from the keyboard. For example, if you were to deactivate the 80STORE softswitch and then activate the RAMWRT softswitch to write to the auxiliary memory from the keyboard, you would be greatly disappointed because you would have instead simply lost control of your computer.

To utilize auxiliary memory, you must first place meaningful data into it, and then transfer control of your computer to that data. But to achieve either of those goals, you must first carefully prepare your system environment so that you do not lose control over it. Certain system parameters must be saved and later restored so that your program does not crash.

Two utility *subroutines*, AUXMOVE and XFER, have been provided in the 80-column firmware to make things easy for you. All you have to do is type your data into main memory, set up a few subroutine parameters, and then perform either a call to AUXMOVE or a jump to XFER. All other system environment requirements will be handled for you automatically.

AUXMOVE

AUXMOVE is a built-in subroutine that allows you to copy blocks of data between main and auxiliary memory by making a simple subroutine call to address 44937. Prior to executing a call to AUXMOVE, you must set up the four parameters given in Table 4-3.

Table 4-3. Parameters for AUXMOVE Subroutine

Name	Location	Parameter passed
Carry		1=Move from main to auxiliary memory 0=Move from auxiliary to main memory
A1L	$3C	Source starting address, low-order byte
A1H	$3D	Source starting address, high-order byte
A2L	$3E	Source ending address, low-order byte
A2H	$3F	Source ending address, high-order byte
A4L	$42	Destination starting address, low-order byte
A4H	$43	Destination starting address, high-order byte

Simply place into A1 and A2, respectively, the address of the first and last bytes of the section of memory to be copied, and into A4 the starting address of the section of memory where the copy should begin. Then set, or reset, the carry bit to specify the direction of the copy operation (from main to auxiliary memory, or from auxiliary to main memory), and make the subroutine call to AUXMOVE. For example, to transfer the contents of main memory locations $2000 to $3FFF to auxiliary memory beginning at location $4000, enter the following command sequence into your Applesoft program (without the REM statements):

- REM Store the low-order byte of the source starting address into A1
- POKE 60,00

- REM Store the high-order byte of the source starting address into A1
- POKE 61,32
- REM Store the low-order byte of the source ending address into A2
- POKE 62,255
- REM Store the high-order byte of the source ending address into A2
- POKE 63,63
- REM Store the low-order byte of the destination starting address into A4
- POKE 67,00
- REM Store the high-order byte of the destination starting address into A4
- POKE 68,64
- REM Store a 1 to set (or a 0 to reset) the carry flag into address $48
- POKE 72,1
- REM Call the subroutine to set (or reset) the carry flag
- CALL -193
- REM Call the subroutine to perform the desired copy operation between the selected physical memory segments
- CALL AUXMOVE

AUXMOVE Shortcut

Although AUXMOVE is generally used in a program, it can be used directly from the keyboard. To do this, use the following procedure:

1. Prepare the carry flag to specify the direction of the transfer: from main to auxiliary memory, or from auxiliary to main memory by first entering the proper value at memory location 72 with a POKE command, and then transferring it to the carry flag with a CALL -193 command.
2. Access the system software monitor by entering a CALL -151 command.

3. Turn on 80-column ROM to access AUXMOVE from the system software monitor by entering C00A:0 from the monitor prompt.
4. Turn off 80STORE so that the AUXMOVE subroutine can properly utilize the RAMRD and RAMWRT softswitches by entering C000:0 from the monitor prompt.
5. Set up the monitor's USER command to jump to AUXMOVE. To set up the monitor's USER command for this, simply place a jump instruction into memory location $3F8, followed by the starting address of the AUXMOVE subroutine.
6. Set up the destination starting address and the source starting and ending addresses in the format of a normal monitor move command.
7. Actuate the USER command by entering a CONTROL-Y key combination. A copy of your data will be transferred between main and auxiliary memory.

The following machine language example demonstrates how this procedure works by copying some numbers from main memory to auxiliary memory, clearing main memory, and then copying the numbers back from auxiliary memory. Be careful to enter all addresses and data correctly. If you make an entry error while attempting to run this example, exit the system monitor by entering 3 D0G (RETURN) from the monitor prompt (*), and then begin again. Although the following example illustrates the above procedure, step one from above is not presented first, because it must instead be used wherever it is needed in order to specify the desired direction of the data transfers.

You type	You get	Purpose
CONTROL ⓞ RESET	]	Reset your computer
CALL -151 RETURN	*	Enter the system software monitor
C00A:0 RETURN	*	Turn on 80-column ROM
C000:0 RETURN	*	Turn off 80STORE
8000:1 2 3 4 5 6 7 8 RETURN	*	Place some numbers into main memory
7FFF RETURN	7FFF- nn *	Some number, nn, is displayed, followed by a monitor prompt
RETURN	8000- 01 02 03 04 05 06 07 08	Verify that your numbers are in main memory. The actual display will look like this: 8000- 01 02 03 04 05 06 07 08
3F8:4C 11 C3 RETURN	*	Set up the USER command location to jump to the start of the AUXMOVE subroutine
8000<8000.8007 CTRL Y (not visible) RETURN	*	Set up the address parameters and actuate the USER command to enable the AUXMOVE subroutine to copy your data to auxiliary memory beginning at location 8000 (note that you did not have to set the carry flag this time because it is automatically set when you enter the AUXMOVE routine)

You type	You get	Purpose
8000:AA AB AC AD AE AF BB BC (RETURN)	*	Place new data into main memory starting at location 8000
7FFF (RETURN)	7FFF- nn *	Some number, nn, is displayed, followed by a monitor prompt
(RETURN)	8000- AA AB AC AD AE AF BB BC	Verify that the new data is in main memory. The display will actually look like this: 8000- AA AB AC AD AE AF BB BC
300:00 (RETURN)	*	Prepare to clear the carry flag
301:AD 0 3 (RETURN)	*	Continue to prepare to clear the carry flag by loading the accumulator with the contents of main memory location 300 (i.e., the previously stored value of 0)
304:2A (RETURN)	*	Clear the carry by rotating the accumulator sign bit into the carry flag, thereby setting the direction of the data move from auxiliary to main memory
305:4C 11 C3 (RETURN)	*	Prepare for a jump to AUXMOVE
3F8:4C 1 3 (RETURN)	*	Set up the USER command location to jump to the start of this routine at main memory location 301

You type	You get	Purpose
8000<8000.8007 CTRL Y (not visible) RETURN	*	Set up the address parameters and actuate the USER command so that AUXMOVE subroutine will this time copy the previously stored contents of auxiliary memory back to the main memory
7FFF RETURN	7FFF- nn *	Some number, nn, is displayed, followed by a monitor prompt
RETURN	8000- 01 02 03 04 05 06 07 08	Verify that your original data are once again back in main memory. Again, the display will actually look like this: 8000- 01 02 03 04 05 06 07 08

By The Way: Notice that you must implement some additional instructions to clear the carry flag, but you did not use any to set it. That was because, as stated above, the monitor always enters AUXMOVE with the carry flag set. If you did want to set the carry flag, you would use the same routine as used here to clear it, but you would load main memory location #300 with FF instead of 00. In case you were wondering: the reason why this somewhat unusual form of the AUXMOVE which you used works, is that addresses for the monitor's move command, which this form of the AUXMOVE uses, are stored in exactly the same locations as those required by the AUXMOVE subroutine. So when AUXMOVE is ready to pick up the required addresses, it picks up those that you just placed there with the special move command format. Finally, this procedure may seem like a rather long "shortcut", but explanations are generally more tedious than techniques.

XFER

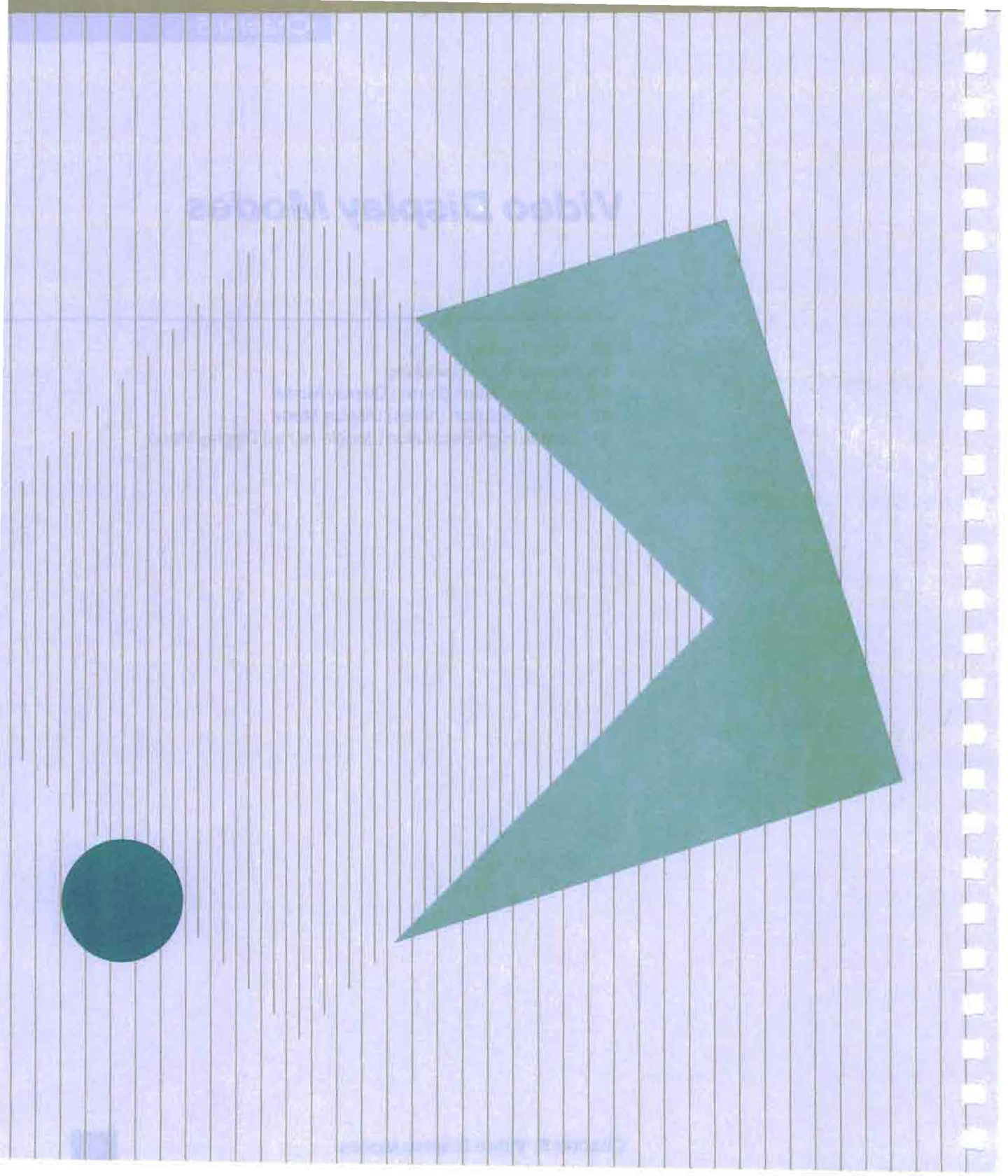
XFER is a built-in subroutine that allows you to transfer control to and from program segments previously moved into auxiliary memory. Prior to executing a jump to XFER, you must set up the three parameters specified in Table 4-4. These parameters can be set up in

a manner similar to that used to set up the parameters required for the AUXMOVE subroutine. A slight wrinkle in the procedure is that here you must carefully consider the desired values of the carry and the overflow flags as a unit since that is how they will be treated by a CALL -193 command. For example, if the carry flag is to be reset to zero and the overflow flag set to one, then you must POKE memory location 72 with the value of 64.

Table 4-4. Parameters for XFER Subroutine

Name or Location	Parameter passed
Carry	1=Transfer from main to auxiliary memory 0=Transfer form auxiliary to main memory
Overflow	1=Use page zero and stack in auxiliary memory 0=Use page zero and stack in main memory
$3ED	Program starting address, low-order byte
$3EE	Program starting address, high-order byte

Video Display Modes



Video Display Modes

This chapter describes the various ways that the video display can be utilized with the Extended 80-Column Text/AppleColor Card.

Video Display

As shown in the memory map of Figure 5-1, certain parts of the memory, known as the *display pages*, are dedicated to displaying different data types: text and graphics. Table 5-1 identifies the various available display modes with their page and memory address locations.

Figure 5-1. Display Pages

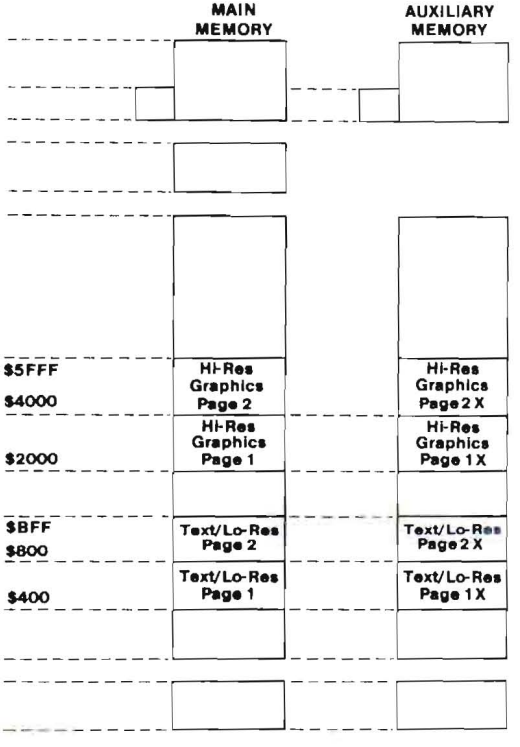


Table 5-1. Video Display Page Location

Note: 80-column mode uses the 1024-byte page-1 locations in both main and auxiliary memory. The PAGE 2 switch is used to select one or the other for storing data.

Display mode	Page	Lowest Address		Highest Address	
40-column Text,	1	$400	1024	$7FF	2047
Low-resolution Graphics	2	$800	2048	$BFF	3071
80-column Text	1	$400	1024	$7FF	2047
High-resolution Graphics	1	$2000	8192	$3FFF	16383
	2	$4000	16384	$5FFF	24575

Display Mode Switching

Table 5-2 lists the names of the softswitches that control the various display modes, along with their activating, deactivating, and status addresses and a brief description of the functions they provide.

Table 5-2. Display Softswitches

(1) This mode is only effective when graphics-mode switch is ON.

(2) This switch has a different function when the 80-column text card's auxiliary text page is enabled for writing.

(3) This switch changes the function of the PAGE 2 switch for addressing the auxiliary text memory on the extended 80-column text card.

Notice that the 80STORE,HIRES and PAGE2 softswitches which first appeared in Table 4-1 are also listed here. These are the same softswitches and, except for softswitch PAGE2, they function in the same way as stated earlier. They are listed in both places because they control both the activation of selected memory segments and the video display format, since the contents of the memory display pages map directly to the display screen.

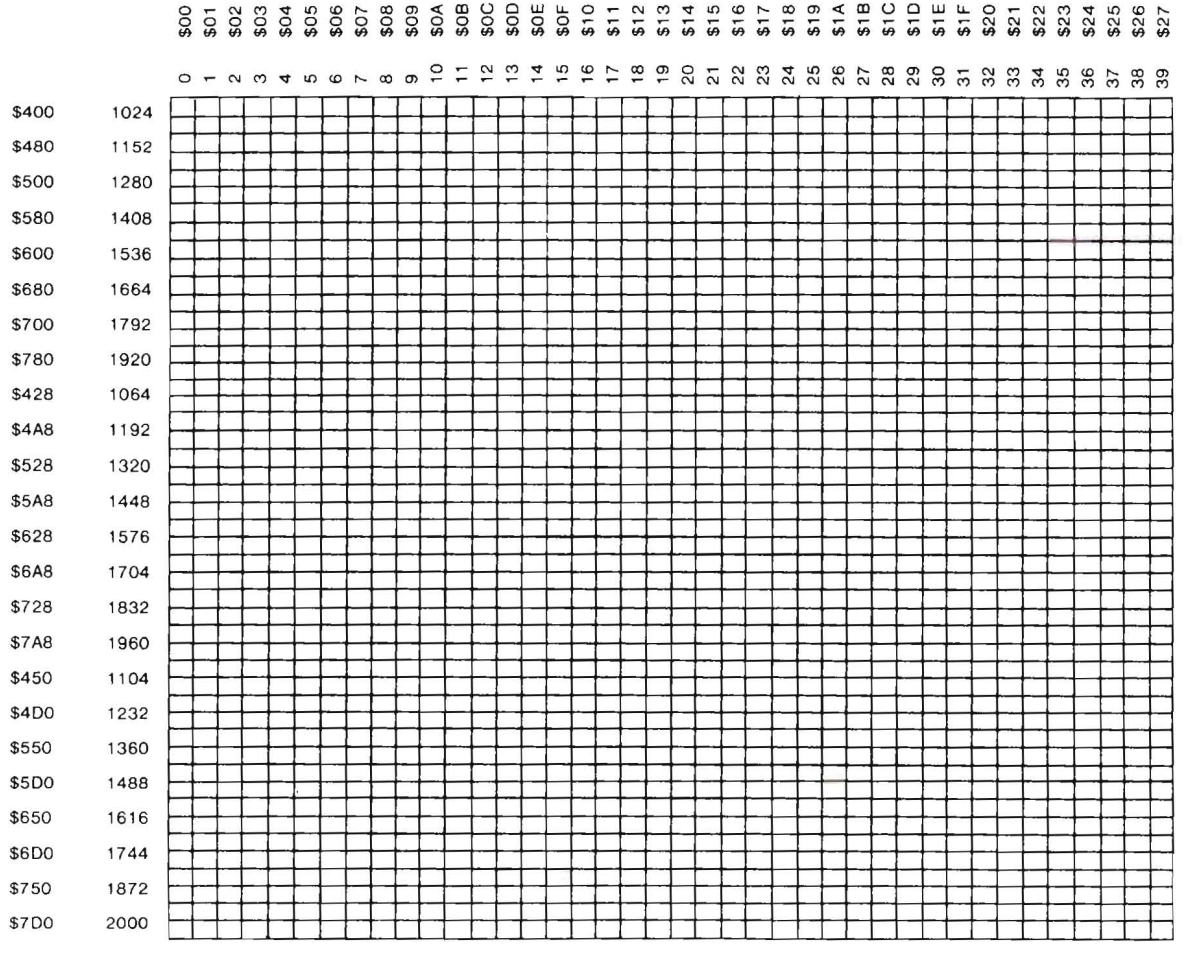
Name	Function	Location Hex	Location Decimal	Notes
TEXT	Text mode on	$C051	49233 -16303	
	Text mode off (graphics)	$C050	49323 -16304	
	Read TEXT switch	$C01A	49178 -16358	Read
MIXED	Mixed-mode on	$C053	49235 -16301	1
	Mixed-mode off	$C052	49234 -16302	1
	Read MIXED switch	$C01B	49179 -16357	Read
PAGE 2	Page 2 on	$C055	49237 -16299	2
	Page 2 off (Page 1)	$C054	49236 -16300	2
	Read PAGE 2 switch	$C01C	49180 -16356	Read
HIRES	Hi-res mode on	$C057	49239 -16297	1
	Hi-res mode off	$C056	49238 -16298	1
	Read HIRES switch	$C01D	49181 -16355	Read
80COL	80-column display on	$C00D	49165 -16371	Write
	80-column display off	$C00C	49164 -16372	Write
	Read 80COL switch	$C01F	49183 -16353	Read
80STORE	Store in auxiliary memory	$C001	49153 -16383	Write, 3
	Store in main memory	$C000	49152 -16384	Write, 3
	Read 80STORE switch	$C018	49176 -16360	Read

Low Resolution (lo-res) Display Mode

In the lo-res mode, the video display can present either text or graphics data alone, or mix the two by displaying graphics data in the upper portion of the display screen and text on the bottom (four display lines are available for text). Through the proper use of the appropriate softswitches, the contents of the various memory display pages may be viewed.

When the text option is chosen in the lo-res display mode, each byte from the selected segment of memory is displayed in the area of the display screen associated with the address of the byte as indicated in Figure 5-2.

Figure 5-2. Memory/Display Map of 40-column Text



When the Extended 80-Column Text/AppleColor Card is activated and the text option is chosen in the lo-res display mode, successive bytes from both auxiliary and main memory are displayed alternately in the area of the display screen associated with the address of a byte, as illustrated in Figure 5-3.

Figure 5-3. Memory/Display Map of 80-column Text

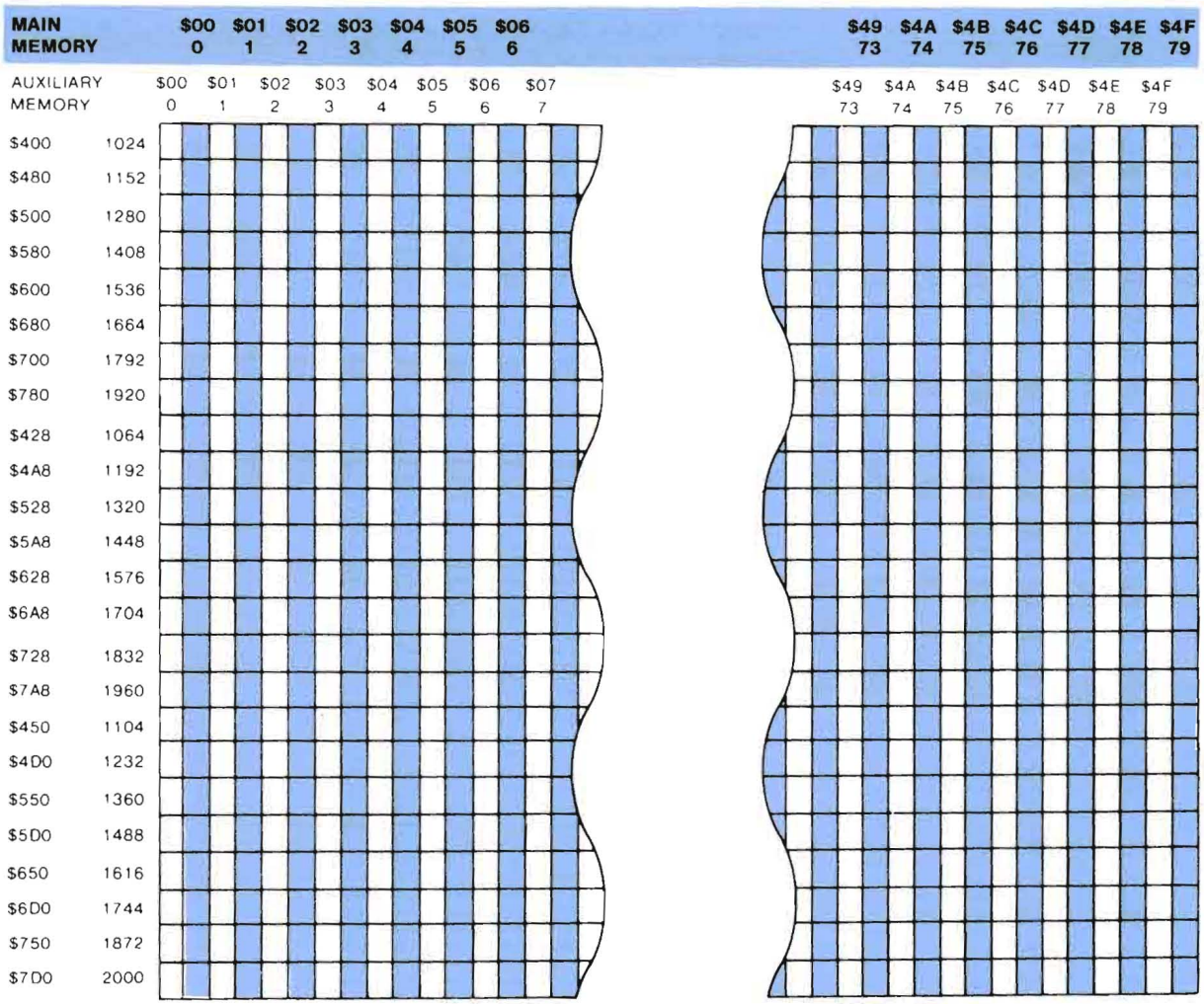


Figure 5-4. Memory/Display Map of Low Resolution Graphics

When the graphics data option is chosen in the lo-res display mode, the low-order *nibble* of each byte is positioned above the high-order nibble of the same byte in the area of the display screen associated with the address of the byte, as indicated in Figure 5-4. The numeric value of each nibble specifies a color, and so one color is positioned above another in each area of the display screen. By carefully selecting the memory addresses according to Figure 5-4, and by specifying the numeric value of each nibble for the selected memory addresses from Table 5-3, various colored forms may be created at any selected screen position.

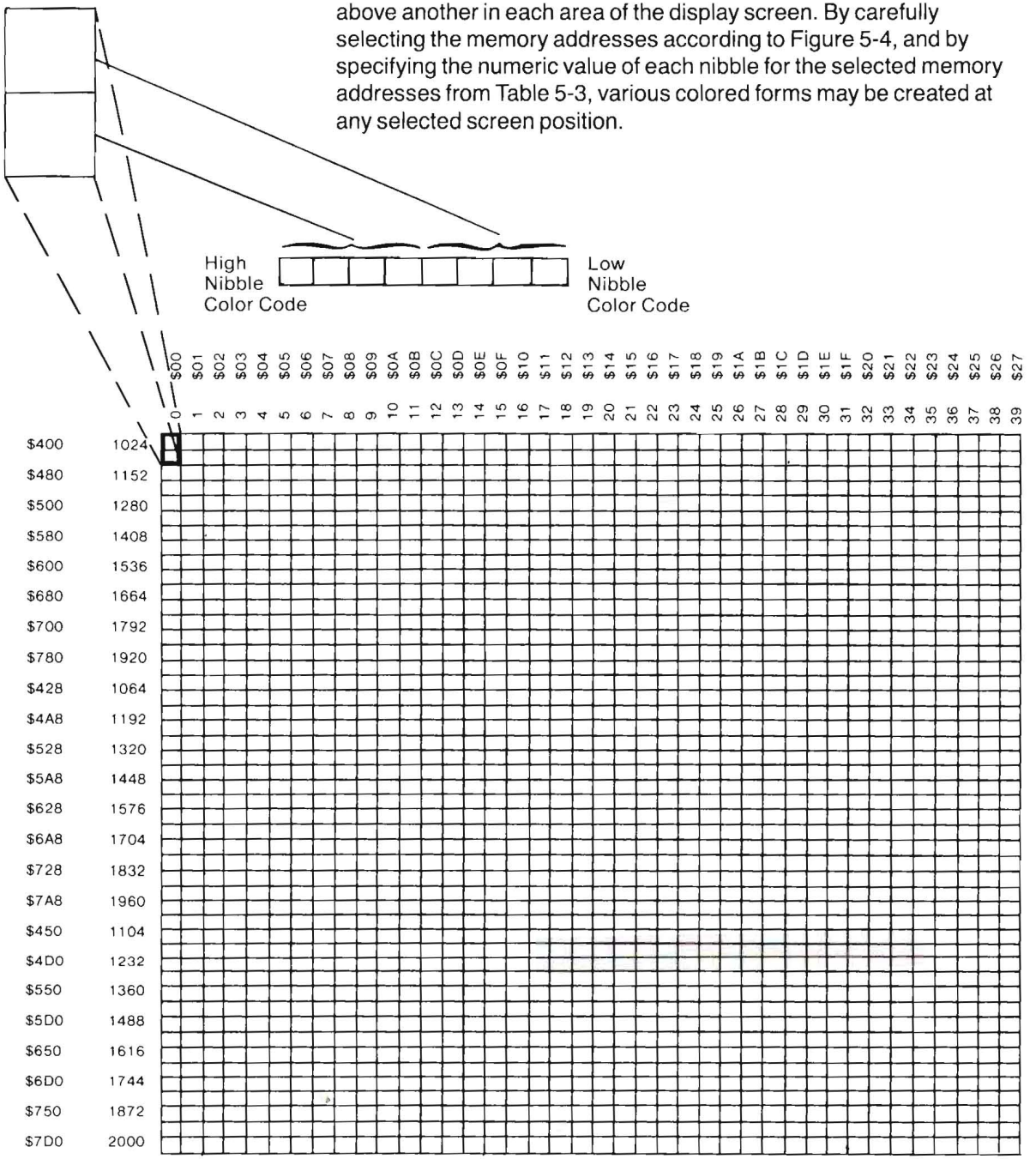


Table 5-3. Low Resolution Graphics Colors

Colors may vary, depending upon the controls on the monitor or television set.

Nibble Value Decimal	Hex	Color	Nibble Value Decimal	Hex	Color
0	$0	Black	8	$8	Brown
1	$1	Magenta	9	$9	Orange
2	$2	Dark Blue	10	$A	Gray 2
3	$3	Purple	11	$B	Pink
4	$4	Dark Green	12	$C	Light Green
5	$5	Gray 1	13	$D	Yellow
6	$6	Medium Blue	14	$E	Aquamarine
7	$7	Light Blue	15	$F	White

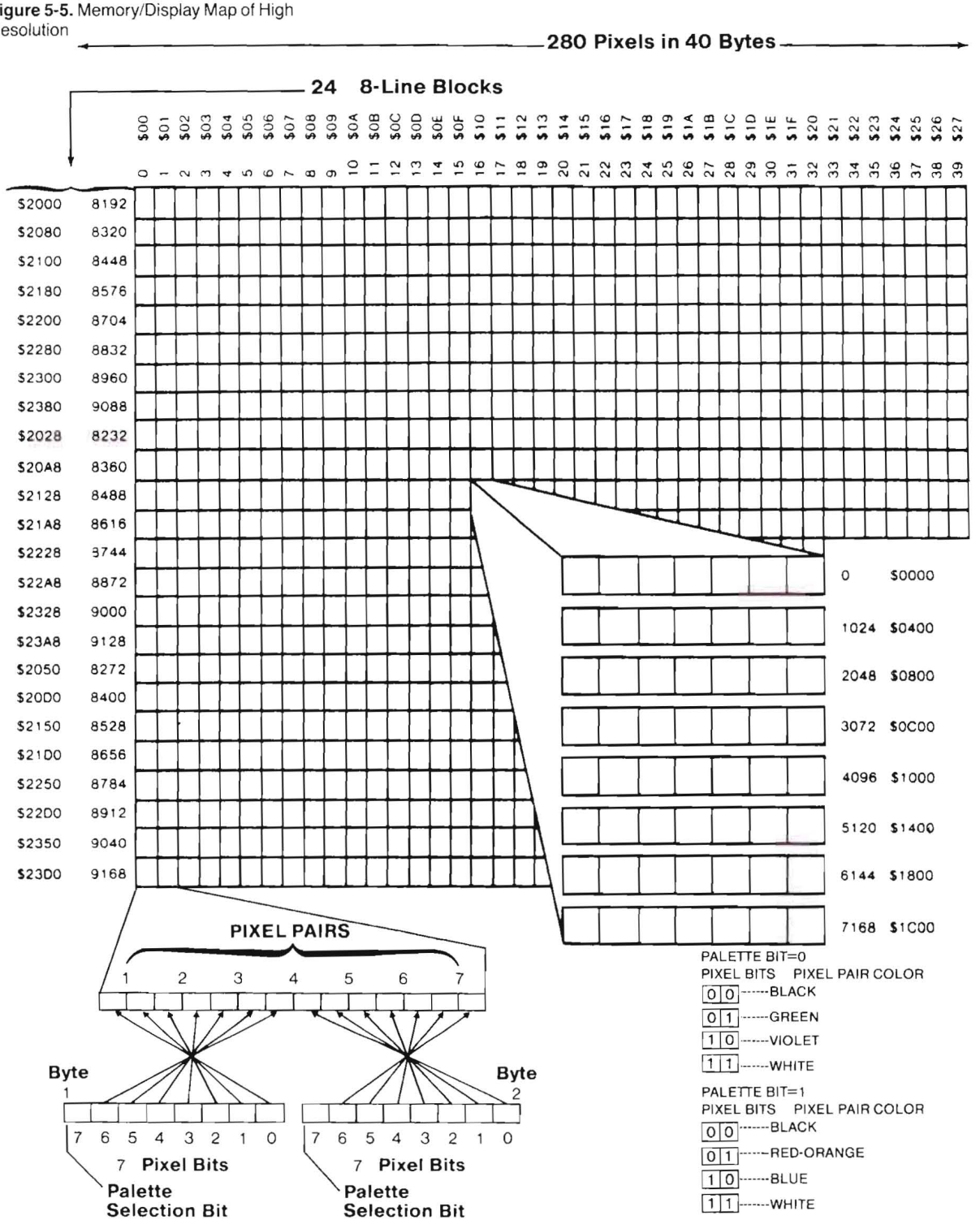
Since the value of any nibble can range between zero and fifteen, each nibble can select any one of sixteen colors, and each selectable color has been assigned the numeric value which produces it. Notice that black and white are considered as colors.

High Resolution (hi-res) Display Mode

The hi-res display mode is primarily a graphics display mode. It divides the screen area into a grid of 280 by 192 individually addressable *pixels*. In mixed mode, the bottom 31 rows of pixels are used to provide four rows of text. These text rows are very useful during the *debug* phases of program development. Through the proper use of the appropriate softswitches, the contents of various memory display pages may be viewed.

In full screen graphics, individual bits of each byte are displayed on a small area of the display screen, as shown in the *memory/display* map of Figure 5-5. Each small rectangle represents both a display area seven pixels wide by eight pixels high and an associated group of hi-res memory locations. One of the rectangles is blown-up and broken-down as a typical example of the mapping of memory address bits to their display screen pixels. Each little square in any row of the blown-up, broken-down rectangle represents one of the seven least significant bits of a byte of memory and its corresponding pixel location on the display screen. The hexadecimal numbers, along the sides, all represent hi-res memory addresses.

Figure 5-5. Memory/Display Map of High Resolution



To find where any byte of hi-res memory will appear on the display screen, just find the address of the byte on the memory/display map. To find any such address on the memory/display map, first look down the row of addresses on the left side of the map for the address that is closest to, but not greater than that of the hi-res memory byte you are trying to locate. The memory byte will be displayed somewhere in that row. Now look for an address across the top of the memory/display map which when added to the address of the row is still not greater than the address of the memory byte. These two address portions, that from the row and that from the column, will uniquely identify a rectangle that represents a small area of the display screen. The memory byte will be displayed somewhere within that area. Finally, to approximate the actual pixel locations on the display, look down the row of addresses to the right of the blown-up, broken-down rectangle for the address which when added to the sum of the first two addresses equals the address of the memory byte you are trying to locate. This address sum will specify the row of squares, within the identified rectangle, which will be uniquely associated with the memory byte. The location of that row of squares will allow you to approximate the display screen location of the pixels associated with the memory byte.

The most significant bit (bit seven) of each memory byte is not displayed in the hi-res graphics display mode, but is used instead to aid in the selection of a color (as shown in Table 5-4). Bit seven is sometimes referred to as a "palette bit" since its value determines which of two groups of colors (palettes) are available. Bit seven causes color changes by shifting the time frame of the lower bits with respect to the frequency of the color subcarrier signal used in the NTSC system. This shifting causes color interactions between adjacent pixels on the display screen.

In order to program a desired color at a particular location on the display screen, it is convenient to think in terms of pairs of adjacent pixels whose color is selected by their corresponding memory bits, from a palette specified by bit seven of their memory byte. Figure 5-5 illustrates the relations that exist between a pixel pair, their corresponding memory bits (referred to as "pixel bits"), the palette bit, and the selected color for the pixel pair. Notice the way that the pixel bits are matched with their corresponding pixels, and the way that certain pixel pairs are formed from the most significant pixel bit of one memory byte and the least significant pixel bit of an adjacent memory byte; these are important facts to keep in mind when you design your graphics programs. The apparent swapping of pixel bit

order and pixel location is also illustrated in Figure 5-6, where the pixel bits of Figure 5-5 are more correctly represented as memory bits that give rise to their corresponding pixels.

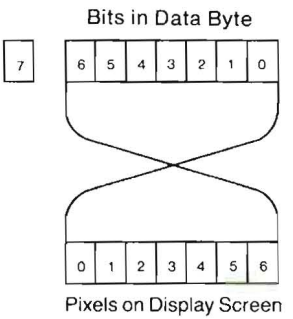
Table 5-4. High Resolution Graphics Colors

Colors may vary, depending upon the controls on the monitor or television set.

Bits 0-6	Bit 7 Off	Bit 7 On
Adjacent columns off	Black 1	Black 2
Even columns on	Purple	Blue
Odd columns on	Green	Orange
Adjacent columns on	White 1	White 2

Notice that there are only six colors available in the hi-res graphics display mode. Also notice that because of the bit mapping nature of the hi-res graphics display mode, eight times as much memory is required to address the same area of the display screen; which is, of course, what accounts for the higher resolution.

Figure 5-6. Display of High Resolution Memory Bits



Double High Resolution (double-hi-res) Display Mode

The double-hi-res display mode is also primarily a graphics mode. In a sense, the double-hi-res mode does for the hi-res mode what the 80-column text mode does for the 40-column text mode; i.e., it doubles both the data storage and the display capabilities of the Apple IIe, and it does this in a manner similar to that used by the lo-res 80-column text mode. Just as the 80-column text mode *interleaves* auxiliary and main memory (addresses $400 through $800), the double-hi-res mode interleaves the hi-res graphics display pages 1X and 1 of auxiliary and main memory (addresses $2000 through $4000).

The addresses on the memory/display maps for the hi-res (Figure 5-5) and the double-hi-res (Figure 5-7) modes are identical.

However, each address of any rectangle of the double-hi-res memory/display map represents the seven least significant bits of both auxiliary and main memory, and their corresponding 14 pixels, as shown in the blown-up, broken-down rectangle of Figure 5-7.

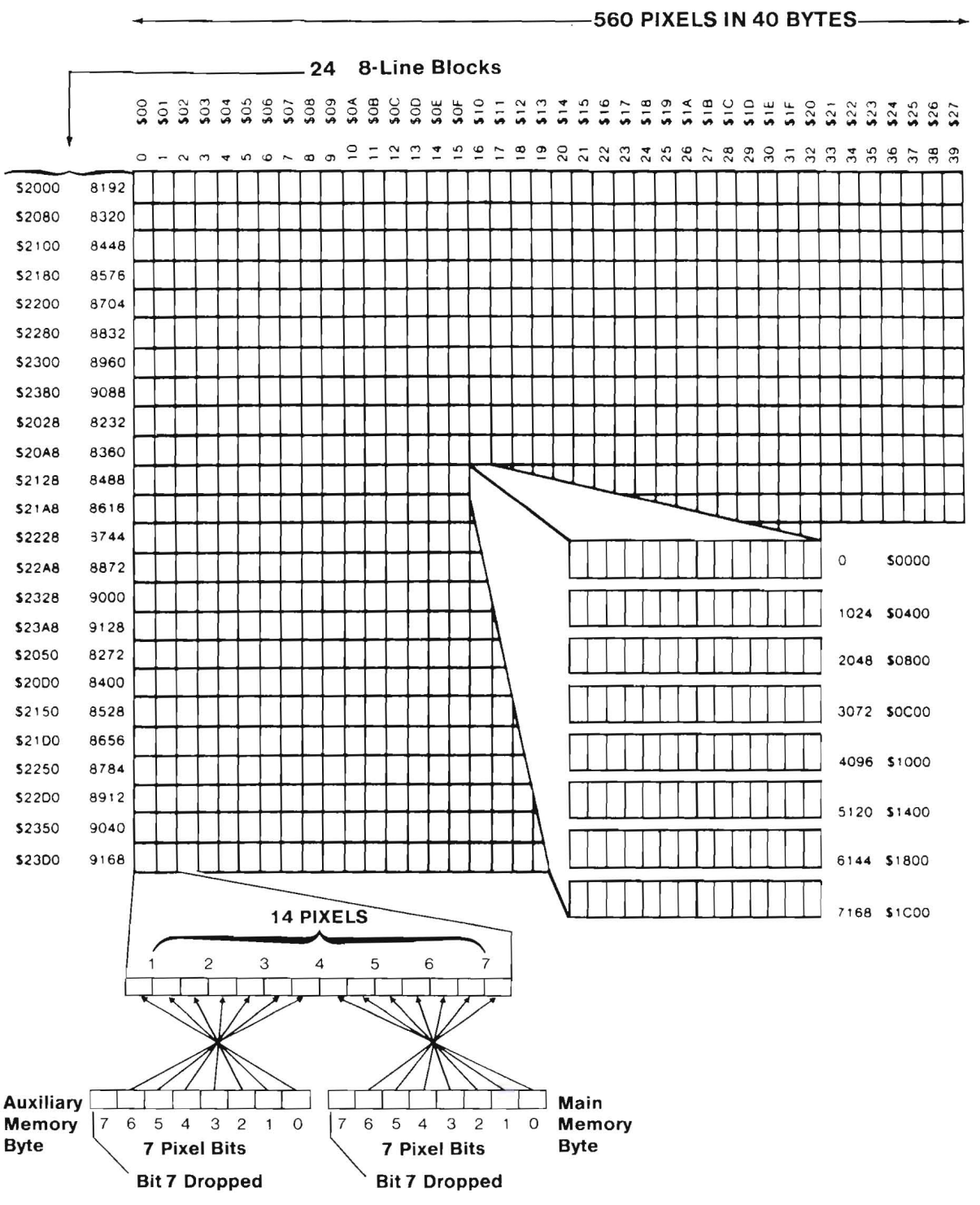
The method for locating the pixels of any double-hi-res memory byte on the display screen is identical to the method explained earlier for the hi-res display mode. Of course in this case, any memory/display map address now specifies 14 pixel locations: seven associated with an address from auxiliary memory, followed by seven from the same address of main memory.

A truly fantastic aspect of the double-hi-res mode is that we now have all 16 of the colors which we previously had with only the lo-res graphics mode. So, we now have more than the best of both worlds: all the colors of lo-res plus double the resolution of hi-res!

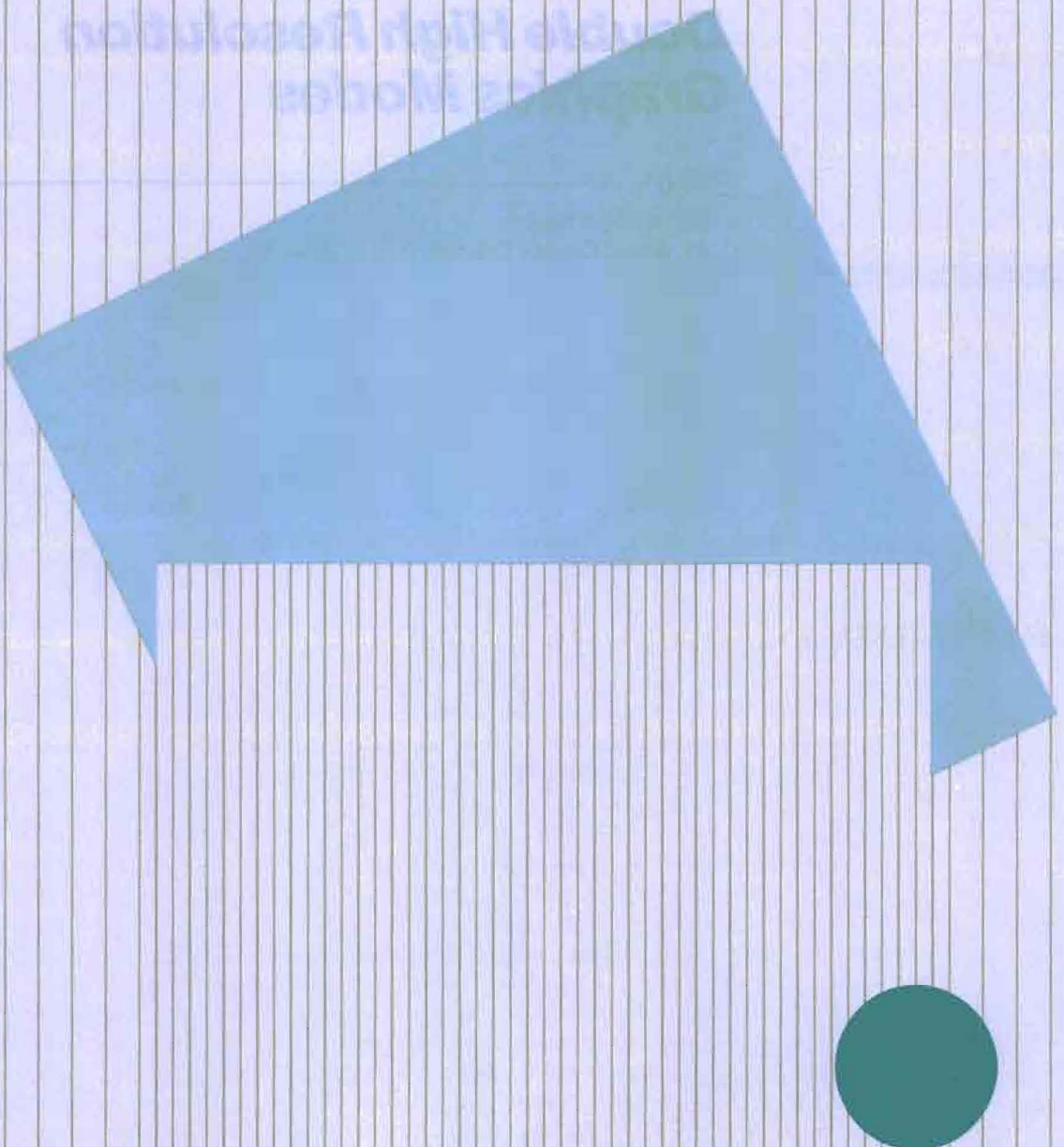
Colors for the Apple IIe computer are generated within a TV or an NTSC monitor by the interaction of nibble patterns, from the computer, with the color subcarrier signal of the TV or the monitor. The nibble patterns for the double-hi-res colors are identical to those used for the lo-res graphics colors, listed in Table 5-3.

Since nibble color patterns do not neatly overlap the 14-dot pixel patterns of any double-hi-res memory/display map address, it is necessary to perform certain *masking* operations of bits in order to draw with a desired color. Fortunately, appropriate subroutines are now available to alleviate such difficulties, and all you really have to do is to specify the desired colors. User commands that support this capability, and others, are presented in Chapter 7 of this manual.

Figure 5-7. Memory/Display Map of Double High Resolution



Double High Resolution Graphics Modes



Double High Resolution Graphics Modes

This chapter describes how to select one of the double high resolution (double-hi-res) graphics modes.

AN3 Softswitch

AN3 is the name of a new softswitch on the Apple IIe revision B, and later, computers. This softswitch allows the Apple IIe to generate double-hi-res graphics when it is reset in conjunction with other softswitches as described in the next section. The control addresses for this softswitch are listed in Table 6-1.

Table 6-1. AN3 Addresses

NAME	FUNCTION	LOCATION	NOTES
AN3	On: Disables double-hi-res	$C05F	Read or Write
	Off: Enables double-hi-res	$C05E	Read or Write

Video Display Control

Three double-hi-res modes are available: Mode 1 (which displays 560 X 192 pixels in monochrome), Mode 2 (which displays 140 X 192 pixels in any of 16 colors), and Mode 3 (also known as the "mixed mode", which allows the mixing of Modes 1 and 2 anywhere on the display screen). Only an RGB color monitor can properly display all of these modes, and the Extended 80-Column Text/AppleColor Card must first be placed in the desired mode.

To set any mode, a *software driver* must set a couple of flags. If your program uses only one of the double-hi-res modes, then the software driver is needed only at the start of your program. If your program uses more than one of mode, then the appropriate software driver must be used everytime your program switches from one mode to another. Your software drivers should parallel the following procedure:

1. First, place the system in the full screen hi-res graphics mode by performing each of the following READs from the system monitor, or PEEKs from the BASIC interpreter:

FUNCTION	MONITOR	BASIC
Full Screen (not mixed mode)	$C052	49234 -16302
Hi-res	$C057	49239 -16297
Graphics	$C050	49232 -16304

2. Then select Mode 1 (560 X 192 monochrome), or Mode 2 (140 X 192 color), or Mode 3 (mixed mode) by performing the following WRITEs and READs from the system monitor, or POKEs and PEEKs from the Applesoft BASIC interpreter:

   A. MODE 1 (MONOCHROME 560 X 192)

FUNCTION	MONITOR	BASIC	NOTE
80COL off	$C00C	49164 -16372	WRITE or POKE
AN3 off	$C05E	49246 -16290	double-hi-res on
AN3 on	$C05F	49247 -16289	double-hi-res off
AN3 off	$C05E	49246 -16290	*
AN3 on	$C05F	49247 -16289	*
80COL on	$C00D	49165 -16371	WRITE or POKE
AN3 off	$C05E	49246 -16290	

   B. MODE 2 (COLOR 140 X 192)

FUNCTION	MONITOR	BASIC	NOTE
80COL on	$C00D	49165 -16371	WRITE or POKE
AN3 off	$C05E	49246 -16290	
AN3 on	$C05F	49247 -16289	
AN3 off	$C05E	49246 -16290	
AN3 on	$C05F	49247 -16289	
AN3 off	$C05E	49246 -16290	

C. MODE 3 (MIXED MODE)

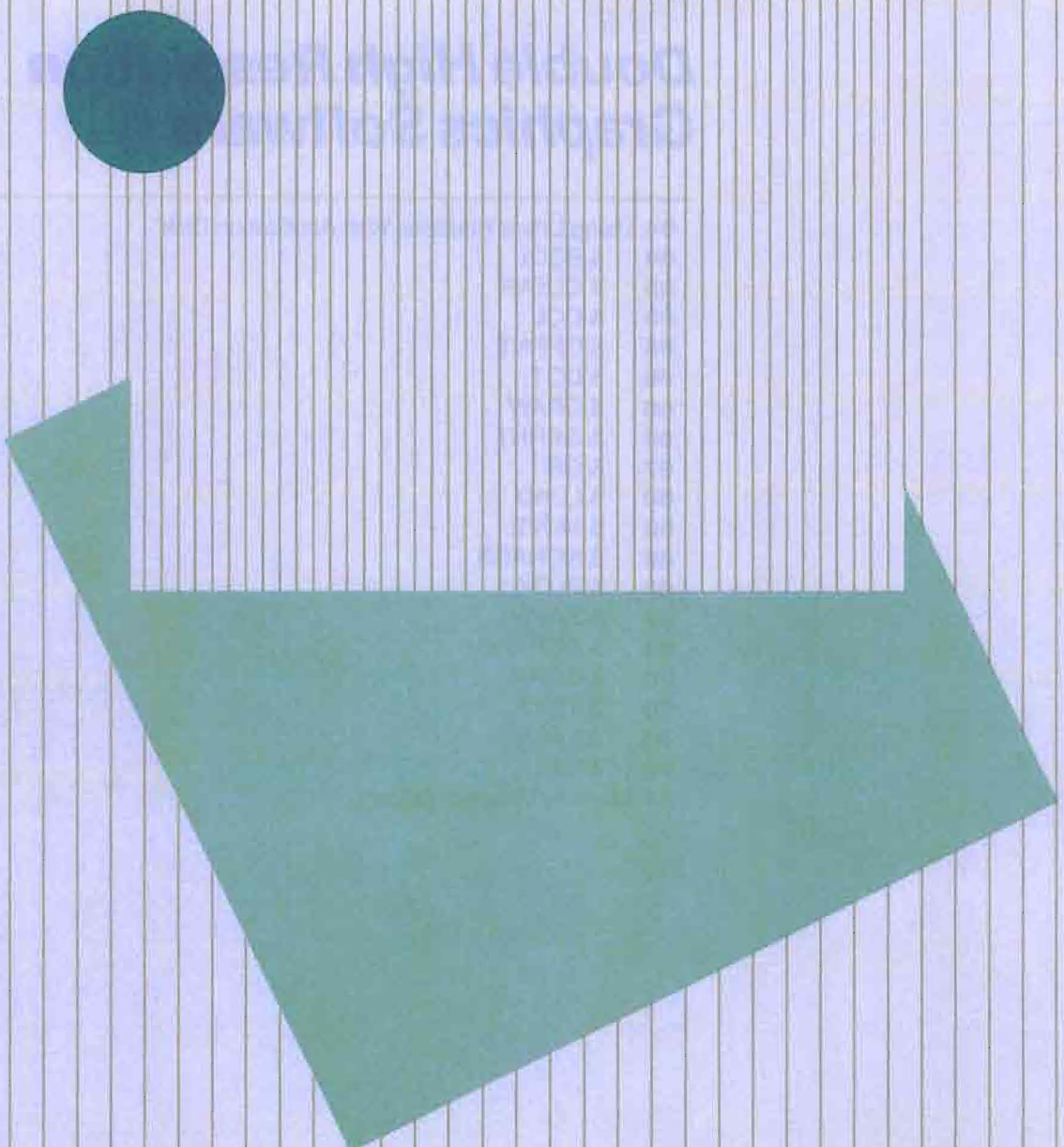
FUNCTION	MONITOR	BASIC	NOTE
80COL off	$C00C	49164 -16372	WRITE or POKE
AN3 off	$C05E	49246 -16290	
AN3 on	$C05F	49247 -16289	
80COL on	$C00D	49165 -16371	WRITE or POKE
AN3 off	$C05E	49246 -16290	
AN3 on	$C05F	49247 -16289	
AN3 off	$C05E	49246 -16290	

*Alternate READs, or PEEKs, turn AN3 off and on.

Warning

Step one command sequence should always be performed prior to the execution of any set of step two commands, in order for the desired double-hi-res mode to be properly selected.

Double High Resolution Graphics Software



Double High Resolution Graphics Software

This chapter describes the new Applesoft routines that drive the double-hi-res graphics modes available with your Extended 80-Column Text/AppleColor Card.

Commands to perform many essential functions are part of Applesoft BASIC and are built into your Apple IIe computer. All of the graphics and text display modes, and commands of Applesoft BASIC are supported by the Extended 80-Column Text/AppleColor Card. In addition to supporting an 80-column display, the card also supports the new double-hi-res graphics modes; however, special software must first be loaded into your computer to enable them. This software consists of machine language routines, commonly referred to as *drivers*. These Applesoft drivers have been implemented as a series of ampersand (&) routines that allow considerable flexibility in the command formats, and provide a very easy-to-use interface. These routines are called ampersand routines because they utilize the Applesoft *ampersand vector*, which allows you to invoke *machine language* subroutines. When the Applesoft *interpreter* encounters an & character at the beginning of a command, it jumps to the memory location specified by that *vector*. This feature gives you access from the Applesoft interpreter to the commands described in this chapter.

These Applesoft ampersand routines always use the hi-res graphics pages 1 and 1X; pages 2 and 2X are not supported by them. They become available when the software driver HIRES is brought into memory with a BRUN command. The first ampersand command that is used after they are BRUN, must be the & GR command, which initializes a selected double-hi-res graphics mode.

Using Driver Routines With Applications Disks

Familiarity with the disk format and copy functions of your computer's operating system (DOS 3.3 or ProDOS) is assumed in the following instructions (refer to the appropriate DOS manual, if necessary):

For DOS 3.3

1. Format a disk for DOS 3.3 using the INIT command.
2. Copy the file HIRES from the DOS 3.3 side of the demonstration/ drivers disk to your DOS 3.3 formatted disk.

For ProDOS

1. Format a disk for ProDOS using the FILER on your ProDOS utilities disk.
2. Copy the ProDOS and BASIC.SYSTEM files from your ProDOS utilities disk to your ProDOS formatted disk.
3. Copy The file HIRES from the ProDOS formatted side of the demonstration/drivers disk to your ProDOS formatted disk.

You can now use the BRUN command to load and run the driver file HIRES from BASIC programs on your formatted disk.

The following commands have been implemented in the double-hi-res modes:

& BCOL	& MOVE
& CLEAR	& NCHARS
& COL	& PLOT
& CPRNT	& SAVE
& DOT	& SCHARS
& DRAW	& SCRN
& GPRNT	& TEXT
& GR	& TPRNT
& LOAD	& VFILL

& BCOL

This command specifies the background or fill color. The color specified by this command is used by the & CLEAR and the & VFILL commands, and also as the background color when displaying

characters or bit mapped shapes. The color parameter specification is the same as for the & COL command, except that the default color for all options is black.

For example:

```
& BCOL = 1
```

& CLEAR

This command clears the screen to the color specified by the last & BCOL command, or to black if no & BCOL command has been issued since the last & GR command. It has the same effect as an & VFILL command routine (which specifies the whole drawing area), except that it is much faster. No command parameters are required.

& COL

This command specifies the pen color to be used for drawing. The default color is white. Select the appropriate number from the following list for a desired color:

Number	Color	Number	Color
0	Black	8	Brown
1	Magenta	9	Orange
2	Dark Blue	10	Gray 2
3	Purple	11	Pink
4	Dark Green	12	Light Green
5	Gray 1	13	Yellow
6	Medium Blue	14	Aquamarine
7	Light Blue	15	White

For example, for dark blue:

```
& COL = 2
```

& CPRNT

This command controls the control character trap in the double-hi-res graphic print routines, and allows the user to output control characters instead of performing the action specified under the & GPRNT command. If the required parameter is zero, control characters will not be printed; any other value will enable control

character printing. The user must ensure that suitable characters are defined in the font being used. The default font has definitions for all control characters.

For example:

```
& CPRNT 1
& CPRNT 0
```

& DOT

This command places a dot in the foreground color at the current pen position. No parameters are required for this command.

& DRAW

This command draws a predefined shape on the screen. It transfers a specified portion of a block of bits to the screen, placing them below and to the right of the current pen position, which is not altered. Each bit set to one in the source block generates a pixel using the current pen color, while each bit set to zero generates a pixel using the current fill color.

Six parameters are required: the first is an address pointer to the first element of the source block, the second specifies the number of bytes in each row of the source block, the third specifies the number of bits to skip in each source row before beginning a transfer, the fourth specifies the number of source rows to skip prior to the transfer, and the fifth and sixth specify the bit width and height of the block portion to be transferred to the screen.

The following example specifies the bit array starting at location $300 as the source, that each row should be three bytes long, that the drawing should begin at the start of the array, and that the area to be drawn should be 24 bits wide and eight rows in height:

```
& DRAW (768, 3, 0, 0, 24, 8)
```

& GPRNT

This command replaces the standard screen output *hooks* by a *pointer* to a routine in the graphics package which places text on the graphics screen. The default character font is very similar to that used for standard text output by the Apple IIe. This command

disconnects the Disk Operating System while it is active. All output from the standard Applesoft print statements will be directed to the graphics screen until a command such as & TPRNT is given to restore the previous output hooks.

All characters are printed in a matrix of seven horizontal by eight vertical pixels (the same as the standard text). After each character is written, the pen position is moved to the right by seven pixels. Control characters are ignored except for the linefeed and carriage return characters. Linefeed characters cause the current pen position to move down the screen by eight pixels, the size of a character row on the text screen. If the downward movement results in an attempt to move off the bottom of the screen, the pen wraps around to the top of the screen. The carriage return control character moves the pen position horizontally to the left edge of the screen and moves the pen vertically as described for the linefeed character. This is the exact equivalent of a carriage return.

The following command sequence illustrates how characters may be placed on the screen:

```
& GR 2                  Initialize double-hi-res mode option 2
& BCOL = 2              Set fill color to dark blue
& CLEAR                 Clear the screen
& COL = 13              Set pen color to yellow
& MOVE (70, 96)         Move to center of the screen
& GPRNT                 Redirect output to screen
PRINT "Some test text"  Print text onto the screen
& TPRNT                 Restore normal output
```

& GR

This is the initial entry point for the graphics routines. It must be used with a mode option number from the following list:

MODE OPTION	DESCRIPTION
1	560 X 192 in monochrome
2	140 X 192 with 16 colors
3	Mixed mode, 560 X 192 in monochrome mixed with 140 X 192 16 colors

For example:

```
& GR 2
```

When this command is executed, the current pen position is moved to location 0,0 (i.e., the top left corner of the screen), the pen itself is set to white, the background or fill color is set to black, and hi-res graphics pages 1 and 1X are selected for display. If the option parameter is given as 0, the graphics routines are disconnected and the space allocated to them is returned to the system.

& LOAD

This command loads a file from disk which has been saved using the & SAVE command. A filename must be specified in the same manner as for the & SAVE command.

For example:

```
& LOAD "PIE.PIC"

& LOAD "MY GRAPH, S6, D2"

& LOAD FILES
```

& MOVE

This command moves the current pen position without drawing on the screen. The command routine requires two coordinates, which may be *literals*, *variables*, or *expressions*. The first coordinate is the horizontal position in terms of pixels from the left edge of the screen, and the second is the vertical row position from the top of the screen.

For example:

```
& MOVE (20, 50)

& MOVE (X, Y)
```

& NCHARS

This command changes the font used for displaying text on the double-hi-res display to a user defined font. It requires one parameter which specifies the disk file that contains the new font.

For example:

```
& NCHARS "ROMAN.FONT"
or
& NCHARS "BYTE.FONT, S6,D2,V12"
or
& NCHARS F$
```

& PLOT

This command draws a line in the foreground color from the current pen position to a new one as specified in the command parameter coordinates. For example, the following statements will cause a brown line to be drawn across the top of the screen in double-hi-res option 2:

```
& GR 2
& MOVE (0, 0)
& COL = 8
& PLOT (139, 0)
```

& SAVE

This command saves the high resolution graphics page 1 areas of the main and auxiliary memory to a specified disk file. The disk file name may be specified as a literal or in a *string* variable. Slot, drive and volume parameters may be specified as described in the Apple II DOS User's Guide. The file will appear to be a binary format file in the disk directory, but is not in the standard binary format and can only be loaded using the & LOAD command.

For example:

```
& SAVE "TEST PICTURE"
or
& SAVE FN$
```

& SCHARS

This command changes the font for displaying text on the double-hi-res display back to the system standard font. No parameters are required.

& SCRN

This command returns a value to the user representing the color at the current pen position. An integer or real variable parameter must be specified to accept the returned value that represents the current color as specified by the & COL command. The returned value will be stored in the specified variable.

For example:

```
& SCRN (VA)
```

& TEXT

This command switches the display back to the text screen, and ensures that the DOS print hooks are correctly set. It is advised that this routine always be called prior to terminating a program which uses any of the graphics commands. No command parameters are required.

& TPRNT

This command restores the print hook to normal when text is no longer to be sent to the graphics screen. It has no effect if the & GPRNT command is not active. The & TEXT command contains an implicit & TPRNT call.

& VFILL

This command clears a portion of the screen to the background color specified in the last & BCOL command, or to black if that command has not been specified since the last & GR command. Four parameters are required to specify the screen boundary to be filled. The first pair parameters specify the left and right horizontal coordinates, and the second pair specify the top and bottom vertical coordinates. The values in each pair may be specified in any order. As with all other coordinate specifications, the bounds may be specified as literals, variables, or expressions.

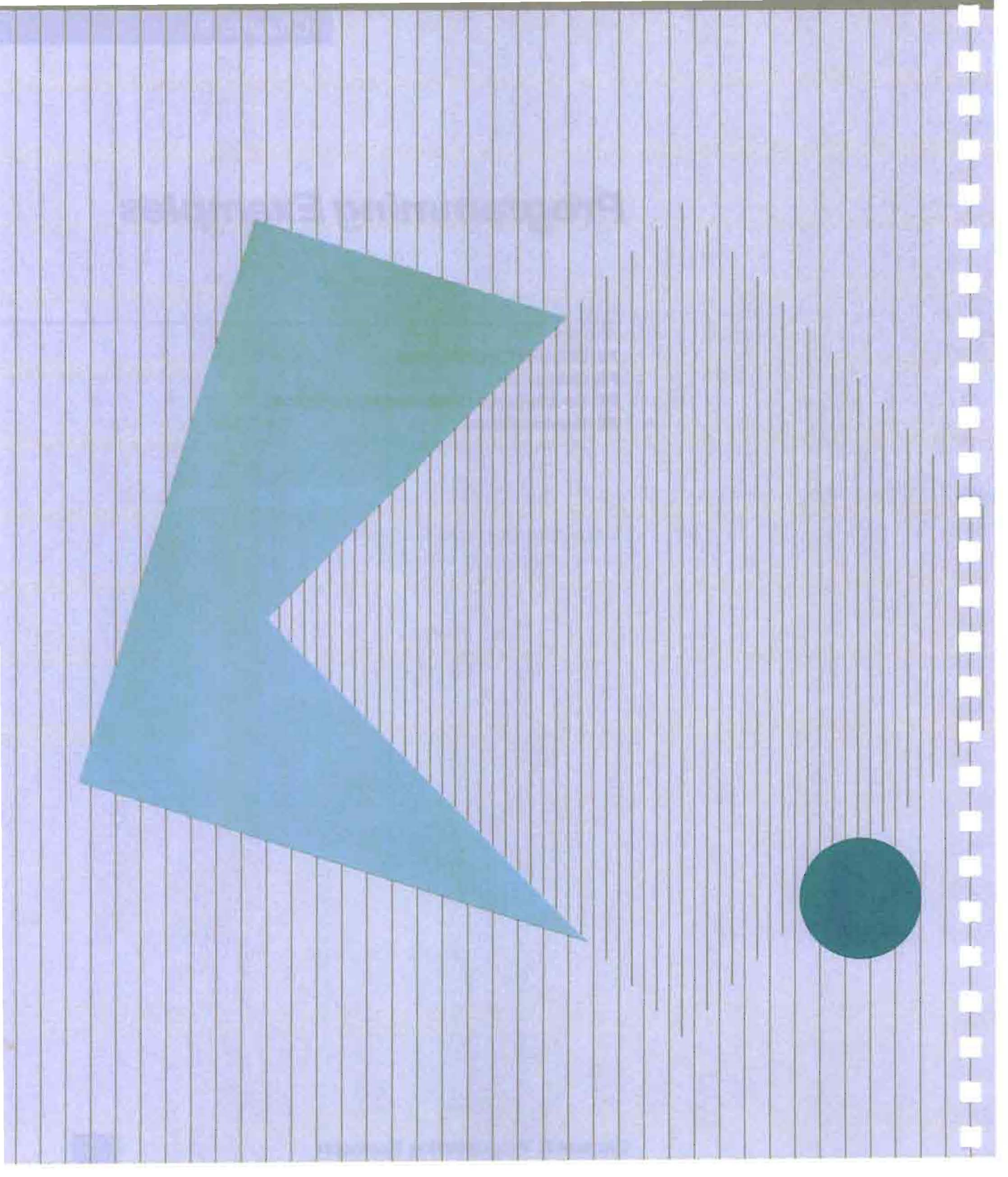
For example:

```
& VFILL (X1, X2, Y1, Y2)

& VFILL (20, 70, 10, 30)

& VFILL (XL, XL + 20, YL, YL + 30)
```

Memory Usage of Drivers

The routines just descibed have been implemented in machine language segments located in main and auxiliary memory. When the software driver file HIRES is BRUN, the first block is loaded at address $2000. The rest of the file is loaded from $2100 to #3FFF using a fast loader. The routines common to all options occupy the first 2K bytes from $2100 to $2800, followed by 1K byte for the standard font, and then 5K bytes for option specific routines (only 1K byte of these are in memory at any time). A 4K byte space is allocated for these routines below DOS and the DOS file buffers are reallocated below these routines. The option specific routines are loaded into the auxiliary memory from $4400 to $5FFF, and an extra copy of the standard font is placed below them at $4000 to $43FF.

Programming Examples



Programming Examples

This chapter contains examples to illustrate the use of some of the new Applesoft ampersand drivers. It is intended that these examples serve as representative examples, showing some of the graphics effects that may be obtained with the new ampersand commands. You will probably benefit most from them if you study them on your computer where you can experiment with them and view your results. These programs are included on your Demo/Drivers disk.

Dithered Color Squares

This program will give you some idea of the range and quality of the colors that you can obtain using the double-hi-res graphics option.

Figure 8-1. Dithered Color Squares Program Listing

```
3   GOSUB 110: POKE - 16368,0
4   & GR 2: & BCOL = 0: & CLEAR
5   FOR X = 0 TO 15
6   & BCOL = X
7   FOR Y = 0 TO 15
8   & COL = Y: & MOVE(X * 8 + 6,Y * 11 + 6): & DRAW
(20000,1,0,0,7,10)
9   NEXT Y
10 S = PEEK ( - 16384): IF S > 127 THEN 21
12 NEXT X
19 POKE - 16368,0
20 GET K$
21 & TEXT : HOME
22 PRINT CHR$ (4);"RUN MENU"
110 FOR X = 1 TO 10
115 READ K: POKE 19999 + X,K
120 NEXT X
130 RETURN
200 DATA 85,170,85,170,85,170,85,170,85,170
```

Cubes

You can experiment with this program to investigate color interactions on the perimeters of adjacent display objects.

Figure 8-2. Cubes Program Listing

```
100 REM    3-D CUBES
101 POKE - 16368,0
110 & GR 2
120 & CLEAR
200 FOR XZ =0 TO 4
220 FOR YZ = 0 TO 3
240 LET X0 = XZ * 20 + YZ * 10
250 LET Y0 = 20 + YZ * 40
260 READ C,D,E
270 IF C < 99 THEN 300
280 RESTORE : READ C,D,E
300 FOR I = 0 TO 9
310 LET J = I + I:K = J + J
320 LET X = X0 + I:Y = Y0 - I
330 & COL = C
340 & MOVE(X,Y)
350 & PLOT (X,Y + J)
360 & COL = D
370 & PLOT (X,Y + J + 30)
380 NEXT
390 LET X0 = X0 + 10
410 FOR I = 0 TO 9
420 LET J = I + I:K = J + J
430 LET J = 20 - J
440 LET X = X0 + I
450 LET Y = Y0 - 10 + I
460 & COL = C
470 & MOVE(X,Y)
480 & PLOT (X,Y + J)
490 & COL = E
500 & PLOT (X,Y + J + 30)
519 S = PEEK ( - 16384): IF S > 127 THEN 601
520 NEXT
540 NEXT : NEXT
600 POKE - 16368,0: GET K$
601 & TEXT : HOME
602 PRINT CHR$ (4);"RUN MENU"
1000 DATA 7,6,2,13,9,8,11,3,1,14,12,4
1010 DATA 11,3,1,14,12,4,7,6,2,13,9,8
1020 DATA 99,99,99
```

Double-hi-res Colors and Mixed Options

This program not only displays the 16 available colors, it also demonstrates the use of the new double-hi-res display mode option 3 with text in the graphics area of the screen.

Figure 8-3. Double-hi-res Colors Program Listing

```
5  GOTO 1000
10 X = 1:W = 14:G = 3
11 Y1 = 88:Y2 = 163
20 FOR I = 0 TO 7
21 & BCOL = I
22 & VFILL(X,X + W,Y1,Y1 - 40)
23 X = X + W + G
24 NEXT I
25 X = 1
30 FOR I = 8 TO 15
31 & BCOL = I
32 & VFILL(X,X + W,Y2,Y2 - 40)
33 X = X + W + G
34 NEXT I
100 & GPRNT: & BCOL = 0: & COL = 9
101 & MOVE(23,12): PRINT "Apple //e RGB "
110 & GR 3: & GPRNT: & MOVE(230,24)
111 PRINT "The Apple Colors "
120 & MOVE(22,Y1 + 3): PRINT "Black "
121 & MOVE(84,Y1 + 3): PRINT "Magenta "
122 & MOVE(161,Y1 + 3): PRINT "Dark "
123 & MOVE(161,Y1 + 11): PRINT "Blue "
124 & MOVE(224,Y1 + 3): PRINT "Violet "
125 & MOVE(297,Y1 + 3): PRINT "Dark "
126 & MOVE(293,Y1 + 11): PRINT "Green "
127 & MOVE(358,Y1 + 3): PRINT "Gray 1 "
128 & MOVE(426,Y1 + 3): PRINT "Medium "
129 & MOVE(432,Y1 + 11): PRINT "Blue "
130 & MOVE(498,Y1 + 3): PRINT "Light "
131 & MOVE(501,Y1 + 11): PRINT "Blue "
132 & MOVE(22,Y2 + 3): PRINT "Brown "
133 & MOVE(86,Y2 + 3): PRINT "Orange "
134 & MOVE(154,Y2 + 3): PRINT "Gray 2 "
135 & MOVE(229,Y2 + 3): PRINT "Pink "
136 & MOVE(293,Y2 + 3): PRINT "Green "
137 & MOVE(358,Y2 + 3): PRINT "Yellow "
138 & MOVE(434,Y2 + 3): PRINT "Aqua "
139 & MOVE(498,Y2 + 3): PRINT "White "
150 & TPRNT
155 POKE - 16368,0: GET K$: & TEXT : HOME
156 PRINT CHR$ (4);"RUN MENU"
1000 REM ************************
1001 REM * BLOAD HIRES IF USING *
1002 REM * THIS PROGRAM WITHOUT *
1003 REM * THE STARTUP PROGRAM. *
1004 REM ************************
1005 & GR 2: & BCOL = 0: & CLEAR
1010 GOTO 10
```

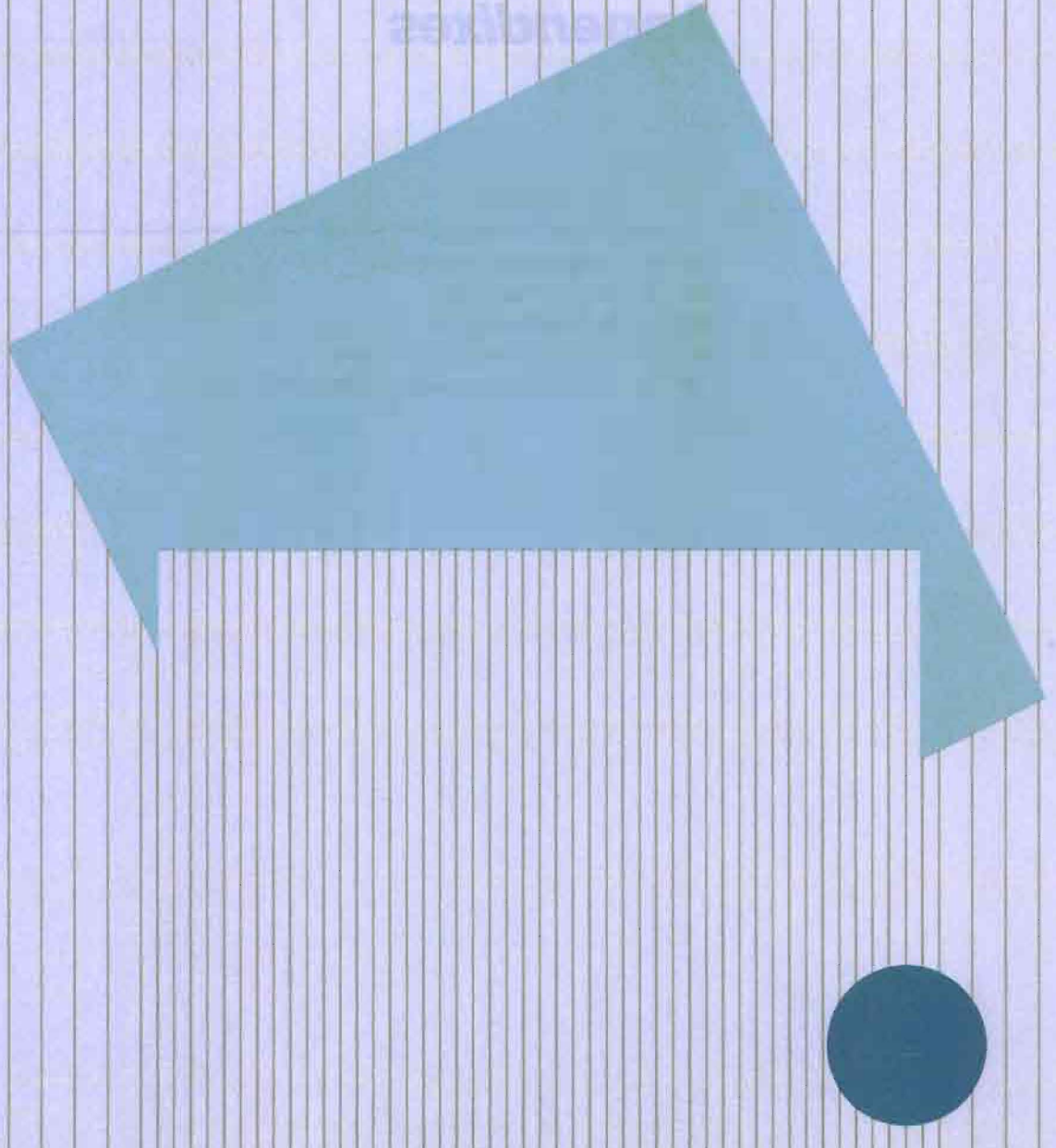
Random Colors

This program demonstrates the use of 140 X 192 with 16 colors to plot lines in random colors.

Figure 8-4. Color Theme Program Listing

```
90 POKE - 16368,0
100 & GR 2
110 & CLEAR
120 T = 2
130 OF = 70
140 EI = 95
150 N = 60
160 OX = 70
170 OY = 94
180 & COL = 2
190 M = 65
200 FOR K = 1 TO 100
210 X = INT ( RND (1) * M + OF)
220 Y = INT ( RND (1) * N + EI)
230 FOR I = 1 TO N STEP T
240 X = X - T
250 Y = Y - T
260 RY = T * EI - Y
270 RX = T * OF - X
280 & MOVE(OX,OY)
290 & PLOT (X,Y)
300 & MOVE(XO,OY)
310 & PLOT (RX,Y)
320 & MOVE(XO,YO)
330 & PLOT (RX,RY)
340 & MOVE(OX,YO)
350 & PLOT (X,RY)
355 S = PEEK ( - 16384): IF S > 127 THEN 450
360 NEXT I
370 & COL = 15 * RND (1)
380 OX = X
390 OY = Y
400 YO = RY
410 XO = RX
420 NEXT K
450 & TEXT : HOME
452 PRINT CHR$ (4);"RUN MENU"
```

Appendixes



Composite Cursor Chart

This appendix lists the various forms the cursor takes to indicate the current operating system state of the Apple IIe computer.

Table A-1. Composite Cursor Chart

Cursor	Card	Language/Operating	When
	active	Pascal and CP/M	always
	inactive	BASIC/DOS	at system startup
	active	BASIC/DOS	after you type PR#3
	active: 80-column display	BASIC/DOS	escape mode
	40-column display		

Display Features

This appendix lists how the various common display features are affected by an active Apple IIe 80-column card.

Table B-1. Display Features

Command	Card Inactive	Card Active
INVERSE	• white screen • black characters • only available for uppercase characters	• white screen • black characters • available for upper-and lowercase characters
NORMAL HOME	• clears to black screen • white characters	• clears to black screen • white characters
INVERSE HOME	• clears to black screen • inverse characters	• clears to white screen • black characters
FLASH	• characters blink between inverse and normal • only available for uppercase characters	FLASH not available

Escape Features

This appendix lists the standard escape features that are available with all 80-column cards.

Table C-1. Standard Escape Features

Escape Feature	Function
ESC @	Clears the window and moves the cursor to its HOME position.
ESC A	Moves the cursor right one space.
ESC B	Moves the cursor left one space.
ESC C	Moves the cursor down one line.
ESC D	Moves the cursor up one line.
ESC E	Clears to the end of the line.
ESC F	Clears to the bottom of the window.
ESC I ESC ^	Moves the cursor up one line and turns on escape mode.
ESC J ESC <	Moves the cursor left one space and turns on escape mode.
ESC K ESC >	Moves the cursor right one space and turns on escape mode.
ESC M ESC v	Moves the cursor down one line and turns on escape mode.
ESC 4	Switches an 80-column display to a 40-column display.
ESC 8	Switches a 40-column display to an 80-column display.
ESC CONTROL-Q	Deactivates the 80-Column Text card.

Control-Character Functions

This appendix lists the standard control-key functions that are available on all 80-column cards.

Table D-1. Standard Control-Character Functions

Control Character	ASCII Name	ASCII Decimal Code	What is Executed
CONTROL -G	(BEL)	7	Produces a 100 Hz tone for 0.1 second.
CONTROL -H	(BS)	8	Moves cursor to the previous cursor position.
CONTROL -J	(LF)	10	Moves cursor position down to next line in window; scrolls if needed.
CONTROL -K	(VT)	11	Clears from cursor position to the end of the window.
CONTROL -L	(FF)	12	Moves cursor position to upper-left corner of window and clears window.
CONTROL -M	(CR)	13	Moves cursor position to left end of next line in window; scrolls if needed.
CONTROL -N	(SO)	14	Sets display format normal.
CONTROL -O	(SI)	15	Sets display format inverse.
CONTROL -Q	(DC1)	17	Sets dispaly to 40 columns.
CONTROL -R	(DC2)	18	Sets display to 80 columns.
CONTROL -S	(DS3)	19	Stops sending characters to the display until a key is pressed.
CONTROL -U	(NAK)	21	Deactivates 80-Column Text Card, homes cursor, and clears screen.
CONTROL -V	(SYN)	22	Scrolls the display down one line, leaving the cursor in the current position.

Control Character	ASCII Name	ASCII Decimal Code	What is Executed
CONTROL -W	(ETB)	23	Scrolls the display up one line, leaving the cursor in the current position.
CONTROL -Y	(EM)	25	Moves cursor position to upper-left corner of window (but doesn't clear).
CONTROL -Z	(SUB)	26	Clears the line the cursor position is on.
CONTROL -\	(FS)	28	Moves cursor position one space to the right; from right edge of window, moves it to left end of line below.
CONTROL -]	(GS)	29	Clear line from cursor position to the right edge of window.
CONTROL -^	(RS)	30	Using the next two characters, minus 32, as one-byte X and Y values, moves the cursor position CH = X, CV = Y.

ASCII, Binary, and Hexadecimal Codes

There are 256 possible eight-bit binary numbers, from 00000000 to 11111111. Of these, the first 128 (from 00000000 to 01111111) have been assigned to characters and commands used in data processing and communication. Their standard assignment forms the ASCII character set. (ASCII stands for the American Standard Code for Information Interchange.)

The remaining 128, which differ from the first 128 only because their most significant binary digit (first digit) is 1 instead of 0, are not officially assigned. Nevertheless, they are often called high ASCII characters.

The following chart lists the 128 standard ASCII character assignments. For each character it gives the binary, decimal, and hexadecimal equivalents for both standard and high versions. Note that the first 27 codes are control characters set by pressing the CONTROL key simultaneously with the desired character key.

Table E-1. ASCII, Binary, and Hexadecimal Codes

	Low ASCII				High ASCII		
ASCII		**Dec**	**Hex**	**76543210**	**Dec**	**Hex**	**76543210**
CONTROL-@	NUL	0	00	00000000	128	80	10000000
CONTROL-A	SOH	1	01	00000001	129	81	10000001
CONTROL-B	STX	2	02	00000010	130	82	10000010
CONTROL-C	ETX	3	03	00000011	131	83	10000011
CONTROL-D	EOT	4	04	00000100	132	84	10000100
CONTROL-E	ENQ	5	05	00000101	133	85	10000101
CONTROL-F	ACK	6	06	00000110	134	86	10000110
CONTROL-G	BEL	7	07	00000111	135	87	10000111
CONTROL-H	BS	8	08	00001000	136	88	10001000
CONTROL-I	HT	9	09	00001001	137	89	10001001
CONTROL-J	LF	10	0A	00001010	138	8A	10001010
CONTROL-K	VT	11	0B	00001011	139	8B	10001011
CONTROL-L	FF	12	0C	00001100	140	8C	10001100
CONTROL-M	CR	13	0D	00001101	141	8D	10001101
CONTROL-N	SO	14	0E	00001110	142	8E	10001110
CONTROL-O	SI	15	0F	00001111	143	8F	10001111
CONTROL-P	DLE	16	10	00010000	144	90	10010000
CONTROL-Q	DC1	17	11	00010001	145	91	10010001

ASCII	Low ASCII	Dec	Hex	76543210	High ASCII Dec	Hex	76543210
CONTROL-R	DC2	18	12	00010010	146	92	10010010
CONTROL-S	DC3	19	13	00010011	147	93	10010011
CONTROL-T	DC4	20	14	00010100	148	94	10010100
CONTROL-U	NAK	21	15	00010101	149	95	10010101
CONTROL-V	SYN	22	16	00010110	150	96	10010110
CONTROL-W	ETB	23	17	00010111	151	97	10010111
CONTROL-X	CAN	24	18	00011000	152	98	10011000
CONTROL-Y	EM	25	19	00011001	153	99	10011001
CONTROL-Z	SUB	26	1A	00011010	154	9A	10011010
	ESC	27	1B	00011011	155	9B	10011011
	FS	28	1C	00011100	156	9C	10011100
	GS	29	1D	00011101	157	9D	10011101
	RS	30	1E	00011110	158	9E	10011110
	US	31	1F	00011111	159	9F	10011111
	SP	32	20	00100000	160	A0	10100000
	!	33	21	00100001	161	A1	10100001
	"	34	22	00100010	162	A2	10100010
	#	35	23	00100011	163	A3	10100011
	$	36	24	00100100	164	A4	10100100
	%	37	25	00100101	165	A5	10100101
	&	38	26	00100110	166	A6	10100110
	'	39	27	00100111	167	A7	10100111
	(	40	28	00101000	168	A8	10101000
	)	41	29	00101001	169	A9	10101001
	*	42	2A	00101010	170	AA	10101010
	+	43	2B	00101011	171	AB	10101011
	,	44	2C	00101100	172	AC	10101100
	-	45	2D	00101101	173	AD	10101101
	.	46	2E	00101110	174	AE	10101110
	/	47	2F	00101111	175	AF	10101111
	0	48	30	00110000	176	B0	10110000
	1	49	31	00110001	177	B1	10110001
	2	50	32	00110010	178	B2	10110010
	3	51	33	00110011	179	B3	10110011
	4	52	34	00110100	180	B4	10110100
	5	53	35	00110101	181	B5	10110101
	6	54	36	00110110	182	B6	10110110
	7	55	37	00110111	183	B7	10110111
	8	56	38	00111000	184	B8	10111000
	9	57	39	00111001	185	B9	10111001
	:	58	3A	00111010	186	BA	10111010
	;	59	3B	00111011	187	BB	10111011
	<	60	3C	00111100	188	BC	10111100
	=	61	3D	00111101	189	BD	10111101
	>	62	3E	00111110	190	BE	10111110
	?	63	3F	00111111	191	BF	10111111
	@	64	40	01000000	192	C0	11000000
	A	65	41	01000001	193	C1	11000001
	B	66	42	01000010	194	C2	11000010
	C	67	43	01000011	195	C3	11000011
	D	68	44	01000100	196	C4	11000100
	E	69	45	01000101	197	C5	11000101
	F	70	46	01000110	198	C6	11000110
	G	71	47	01000111	199	C7	11000111
	H	72	48	01001000	200	C8	11001000
	I	73	49	01001001	201	C9	11001001

ASCII	Low ASCII Dec	Hex	76543210	High ASCII Dec	Hex	76543210
J	74	4A	01001010	202	CA	11001010
K	75	4B	01001011	203	CB	11001011
L	76	4C	01001100	204	CC	11001100
M	77	4D	01001101	205	CD	11001101
N	78	4E	01001110	206	CE	11001110
O	79	4F	01001111	207	CF	11001111
P	80	50	01010000	208	D0	11010000
Q	81	51	01010001	209	D1	11010001
R	82	52	01010010	210	D2	11010010
S	83	53	01010011	211	D3	11010011
T	84	54	01010100	212	D4	11010100
U	85	55	01010101	213	D5	11010101
V	86	56	01010110	214	D6	11010110
W	87	57	01010111	215	D7	11010111
X	88	58	01011000	216	D8	11011000
Y	89	59	01011001	217	D9	11011001
Z	90	5A	01011010	218	DA	11011010
[	91	5B	01011011	219	DB	11011011
\	92	5C	01011100	220	DC	11011100
]	93	5D	01011101	221	DD	11011101
^	94	5E	01011110	222	DE	11011110
_	95	5F	01011111	223	DF	11011110
`	96	60	01100000	224	E0	11100000
a	97	61	01100001	225	E1	11100001
b	98	62	01100010	226	E2	11100010
c	99	63	01100011	227	E3	11100011
d	100	64	01100100	228	E4	11100100
e	101	65	01100101	229	E5	11100101
f	102	66	01100110	230	E6	11100110
g	103	67	01100111	231	E7	11100111
h	104	68	01101000	232	E8	11101000
i	105	69	01101001	233	E9	11101001
j	106	6A	01101010	234	EA	11101010
k	107	6B	01101011	235	EB	11101011
l	108	6C	01101100	236	EC	11101100
m	109	6D	01101101	237	ED	11101101
n	110	6E	01101110	238	EE	11101110
o	111	6F	01101111	239	EF	11101111
p	112	70	01110000	240	F0	11110000
q	113	71	01110001	241	F1	11110001
r	114	72	01110010	242	F2	11110010
s	115	73	01110011	243	F3	11110011
t	116	74	01110100	244	F4	11110100
u	117	75	01110101	245	F5	11110101
v	118	76	01110110	246	F6	11110110
w	119	77	01110111	247	F7	11110111
x	120	78	01111000	248	F8	11111000
y	121	79	01111001	249	F9	11111001
z	122	7A	01111010	250	FA	11111010
{	123	7B	01111011	251	FB	11111011
|	124	7C	01111100	252	FC	11111100
}	125	7D	01111101	253	FD	11111101
~	126	7E	01111110	254	FE	11111110
DEL	127	7F	01111111	255	FF	11111111

Glossary

address: A number used to identify something, such as a storage location in the memory of a computer, or an *expansion slot*. Also the act of specifying a memory or peripheral location.

address space: An interval of numbers that refer to individually accessible *devices*. The devices may be actual memory bytes, or only potential ones, or peripheral devices. For example, the 6502 microprocessor is said to have a 64K logical byte address space capability, meaning that it can access 65,536 bytes of computer memory. If your computer has only 16K bytes of physical memory installed, then not all of your computer's logical address space has been actualized (or filled). Your system still has a 64K byte address space, but only 16K bytes of memory could actually be utilized under such conditions.

ampersand driver routine: See *ampersand vector, driver*, and *routine*.

ampersand vector: The name given to a special memory location; namely, "&" given to memory location $3F5. Whenever Applesoft executes an & command, it makes an unconditional jump to memory location $3F5. If a jump instruction and the address of a system subroutine is located there, then Applesoft will continue on to the execution of that subroutine.

Applesoft: An extended version of the BASIC programming language used with the Apple IIe computer. An *interpreter* for creating and executing programs in Applesoft is built into the Apple IIe system in *firmware*. Also see *interpreter*.

ASCII: Acronym for American Standard Code for Information Interchange. It assigns a unique binary number to each *character*.

assembler: A computer program that translates *assembly language source code* into equivalent *machine language object code*. Also see *interpreter*, and *translator*.

assembly language: A programming language that uses convenient mnemonic symbols for the binary codes that control a computer or *CPU*.

assembly language source code: See *assembly language* and *source code*.

auxiliary memory: In the Apple IIe computer, an additional 64K bytes of memory that can be temporarily substituted for the main memory in the 64K byte address space of the 6502 microprocessor. It is a chunk of physical memory which, although it is accessed through the same logical address space as the main memory, is physically independent of main memory.

auxiliary RAM memory: Same as auxiliary memory, see *auxiliary memory*. Also see *random access memory*.

auxiliary expansion slot: The special expansion slot inside the Apple IIe used for the Apple 80-Column Text Card, the Extended 80-Column Text Card, or the Extended 80-Column Text/AppleColor Card.

background color: The color specified by the & BCOL command in the double-hi-res graphics mode. Also called the *fill color.*

back panel: The rear plate of the Apple IIe computer, which includes the power switch, the power connector, and connectors for a video display device, a cassette tape recorder, and other peripheral devices.

BLOAD: A system command which loads a binary encoded *file* from disk into the computer's *read-write memory*.

binary: The base-2 numbering system consisting of the two digits 0 and 1. Most computer storage devices are designed to store binary digits, and computer circuitry is designed to manipulate characters coded in a binary form.

binary file: A *file* whose contents are in binary format.

bit: A single binary digit, consisting of either a zero or a one.

board: Short for printed circuit, or PC, board. It is a board with copper circuits etched on it and electronic components soldered to those etches.

boot: To start a computer by loading a program into memory from an external storage medium, such as a diskette. It is often accomplished by first loading a small program from *ROM* that reads the larger program into memory. The small ROM program is usually loaded when a reset switch is hit, and that program is said to "pull itself in by its own bootstraps"; hence the term *bootstrapping* or *booting*.

booting: See *boot*.

bootstrapping: See *boot*.

BRUN: A system command that causes the execution of a binary encoded program after it has been BLOADed into memory.

byte: The fundamental unit of *character* representation in a piece of electronic equipment. One byte usually consists of eight bits, and can either hold any value from 0 to 255, or can uniquely represent any one of 256 different characters.

cathode-ray tube: An electronic device, such as a television picture tube, that produces images on a screen coated with phosphors that emit light when struck by a focused beam of electrons.

character: Any letter, number, punctuation mark, or control code that can be acted upon by the Apple IIe computer.

character font: The design, shape, and size of a set of display or print characters.

command: A communication from the user to a computer system to perform some action.

compiler: A computer program that translates *high-level language* programs as a whole into their equivalent *machine language* programs. See also *interpreter*, and *translator*.

composite video: A video signal that includes both display information and the synchronization signals needed to display it.

computer: An electronic device for performing predefined (programmed) computations at high speed and with great accuracy.

computer system: A *computer* and its associated *hardware*, *firmware*, and *software*.

connector: A physical device such as a plug, socket, or jack, used to connect one hardware component of a system to another.

constant: A symbolic entity whose value does not change.

control character: A character that controls or modifies the way information is printed or displayed. Control characters have ASCII codes between 0 and 31 and are typed from the Apple IIe keyboard by holding down the CONTROL key while typing some other character. For example, the character CONTROL-M (ASCII code 13) means "go to the beginning of the next line", and is equivalent to the RETURN key.

CPU: A central processing unit; usually a microprocessor like the 6502 in the Apple IIe.

CRT: Initials for cathode ray tube; see *cathode ray tube*. Your monitor screen is a CRT.

cursor: A marker or symbol displayed on the screen that marks where the user's next action will take effect or where the next character typed from the keyboard will appear.

data: Information used or operated on by a computer program.

debug: The process of determining and removing software errors from your programs.

demo: An abbreviation for *demonstration disk*.

demonstration disk: A disk supplied with a program or piece of computer equipment which provides the user with a demonstration of its typical operation.

device: A generic term meaning a piece of electronic equipment of an integrated circuit. The 6502 microprocessor is a device, the Apple IIe computer is a device, and so is a printer, a video monitor, disk drives, etc.

DIP switch: DIP stands for dual in-line package, a description of the physical form the switch takes. It is usually a miniature toggle or slide switch.

diskette: A term sometimes used for flexible or floppy disks.

display: (1) Information exhibited visually, especially on the screen of a display device. (2) To exhibit information visually. (3) A display device.

display pages: Those *pages* of read-write memory dedicated to storing data for the *display screen*. There is a one-to-one mapping of memory bytes (lo-res), or bits (hi-res) to display screen areas.

display screen: The glass front of a display device, on which images are displayed.

double-hi-res graphics: The display of graphics on the Apple IIe computer with potentially twice the resolution capability of the standard hi-res mode, and utilizes sixteen colors.

driver: A program that allows the *CPU* to control an *I/O device* such as a disk drive, a keyboard, a terminal, or a printer.

EPROM: A type of *read-only memory* device where data can be written electrically, erased by the action of ultraviolet light, and re-written again. The process of writing, erasing, and re-writing can be done hundreds of times.

escape mode: A state of the Apple IIe computer, entered by pressing the ESC key, in which certain keys on the keyboard take on special meanings for positioning the cursor and controlling the display of text on the screen.

expansion slot: A connector inside the Apple IIe computer in which a peripheral card can be installed; sometimes called peripheral slot.

expression: Any combination of mathematical or logical *variables*, *constants*, and *operators*.

file: A large quantity of related data treated as a unit.

fill color: See *background color*.

firmware: Software of a relatively permanent nature which is stored in some type of *read-only memory* device, such as a *ROM*, a *PROM*, or an *EPROM*.

foreground color: The color specified by the & COL command in the double-hi-res graphics mode. Also known as the *pen color*.

graphics: (1) Data presented in the form of pictures or images. (2) The display of pictures or images on a computer's display screen. Compare *text*.

half-dot shift technique: The technique used in hi-res graphics to produce color. By shifting the pixel pattern one half dot's worth of time, the monitor is caused to generate color.

hardware: Those components of a computer system consisting of physical (electronic or electromechanical) devices. Compare with *software*, and *firmware*.

hexadecimal: The number system to the base 16. It uses the ten numerals 0 through 9 and the six letters A through F, as numerals, to represent numeric values.

high-level language: A problem oriented language, such as Applesoft, which permits a programmer to concern himself more with the details of his problem, by relieving him of *machine language* execution details.

high-order byte: The more significant half of a memory address or other two-byte quantity. In the Apple IIe's 6502 microprocessor, the low-order byte of an address is usually stored first, and the high-order byte second.

high resolution graphics: The display of graphics on the Apple IIe, using the display screen as an area 280 columns wide by 192 rows high. Any of six colors may be used.

HIRES: As used in this manual, the name of the high resolution softswitch. In general use, it means the same as high resolution.

hook: A built-in expansion capability, usually in *firmware*.

icon: A small image, or picture.

interface: In computer hardware, the equipment that accepts electrical signals from one system and renders them into a form that can be used by another.

interleave: To rearrange the parts of several sequences of things or events so that they alternate with each other in a regular manner.

interpreter: A computer program that translates *high-level language* statements into their equivalent *machine language* instructions immediately. See *compiler* .

inverse video: The display of text on the computer's display screen in the form of black dots on a white (or other single phosphor color) background, instead of the usual white dots on a black background.

I/O: (1) An abbreviation for input/output. (2) The transfer of characters into or out of a computer. (3) Devices for transferring characters into or out of a computer.

kilobyte or K byte: Although the prefix "kilo" means 1000 and is generally represented by the letter "K", in computer jargon both conventionally mean 1024 bytes.

literal: A symbol, usually in a program, that names, identifies or defines itself literally and does not represent something else.

logical address space: Same as address space, see *address space*.

low-order byte: The less significant half of a memory address or other two-byte quantity. In the Apple IIe's 6502 microprocessor, the low-order byte of an address is usually stored first, and the high-order byte second.

low resolution graphics: The display of graphics on the Apple IIe using the display screen as an area 40 columns wide by 48 rows high. Any of 16 colors are available.

LORES or lo-res: Low resolution.

machine language object code: See *machine language*, and *object code*.

machine language or code: The set of binary codes that directly control the functions of a computer.

main memory: The memory component of a computer system that is built into the computer itself and whose contents are directly accessible to the processor.

memory: A hardware component of a computer system that can store information for later retrieval; see *main memory, random-access memory, read-only memory, read-write memory, write-only memory*.

memory/display map: A graphical visual aid that uses *video memory* address values as coordinates on a *display screen* diagram, in order to correlate video memory bytes (or bits) with their display screen locations (or *pixels*).

memory location: A unit of main memory that is identified by an address and can hold a single item of information of a fixed size; in the Apple IIe, that size is one byte, or eight bits, in length. See *address*.

memory map: A visual aid for depicting the address structure and use of a computer's memory.

NTSC: (1) National Television Standards Committee; the committee that defined the standard format used for transmitting broadcast video signals in the United States. (2) The standard video format defined by the NTSC.

nibble: A number of *bits* equal to half a byte; usually four bits, that can represent any value from 0 to 15 (sixteen individual values).

object language or code: The machine language equivalent of a *source language* program, usually prepared by a computer as the output of a *compiler* or an *assembler*.

1K: See *kilobyte*.

operator: A mathematical or logical symbol that represents a required operation between two, or more, *constants, variables*, or *expressions*.

pen color: See *foreground color*.

peripheral: Short for peripheral device. A device attached to the computer that can provide input and/or accept output from the computer. Peripherals include printers, cassette drives, and disk drives.

peripheral card: A removable printed-circuit board that plugs into one of the Apple IIe's expansion slots, expanding or modifying the computer's capabilities by connecting a peripheral device to perform some subsidiary or peripheral function.

physical memory: Memory devices that are actually present in an electronic system. See *memory*.

pixel: A picture element; the smallest area of the display screen that can be individually controlled by a bit from the *video memory*. Sometimes called a dot, a pel, or a pixcell.

pixel turn on: The activation of a pixel in the foreground color defined in the & COL command.

pixel turn off: The activation of a pixel in the background color defined in the & BCOL command.

pointer: A memory location whose contents points to some other memory address.

power supply: The hardware component of a computer that draws electrical current from a power outlet and converts it to the forms needed by other hardware components.

PROM: A type of *read-only memory* device where data are electrically "burned-in", by blowing fusible metal links within the device so as to introduce the desired pattern of zeros and ones. Once the fusible metal links are blown, they cannot be repaired, and so the data that they represent are permanent.

RAM: An abbreviation for *random-access memory*. See random-access memory.

random-access memory: Memory in which the contents of individual locations can be accessed in an arbitrary or random order. This term usually refers to read-write memory, but, strictly speaking, both *read-only memory* and *read-write memory* can be accessed randomly.

raster: The pattern of parallel lines making up the image on a video display screen. The image is produced by controlling the brightness of successive dots on the individual lines of the raster.

read: To transfer data into the computer's memory from a source external to the computer (such as a disk drive or modem) or into the computer's processor from a source external to the processor (such as the keyboard or main memory).

read-only memory: Memory whose contents can be read but not normally written; used for storing firmware. Data placed into read-only memory devices remains there either permanently or semi-permanently, even when the computer's power is turned off, and cannot be easily erased or changed. See *ROM, PROM,* and *EPROM*.

read-write memory: Memory whose contents can both be read and written; often misleadingly called random access memory (or RAM) in order to distinguish it from *ROM*. The information contained in read-write memory is volatile—it is erased when the computer's power is turned off, and is permanently lost unless it has been saved on a more permanent storage medium, such as a disk. See also *random access memory*.

RGB: Stands for red, green, blue: an RGB type monitor is one that uses a separate electron source for each of these colors.

ROM: An abbreviation for *read-only memory*. A ROM is a type of read-only memory device in which data are placed during the manufacturing process by special masking techniques that create the desired patterns of zeros and ones that can never be changed. See read-only memory.

routine: A part of a program that accomplishes some task subordinate to the overall task of the program.

screen: See *display screen*.

shape definition: A series of bytes whose bit patterns can selectively turn on or off graphics display pixels in such a way as to generate a desired shape on the graphics display screen.

shape table: A sequence of shape definitions in a *file*.

softswitch: A means of changing some feature of the Apple IIe from within a program; specifically, a location in memory that produces some special effect whenever its contents are read or written.

software: In general, programs and program instructions; as opposed to *hardware*, which is the machinery that is controlled by the software.

software driver: A *subroutine* that controls a *peripheral* device.

source code: A generic term for the statements in which a programmer writes a program.

source language: A generic term for the computer language in which a programmer writes a program.

string: A linear sequence of characters interpreted as a single data item. It is usually alphanumeric, but it often contains special symbols, such as $, with special meanings.

subroutine: A part of a program that can be executed on request from any point in the program, and which returns control to the point of the request on completion.

text: (1) Information presented in the form of characters readable by humans. (2) The display of characters on the Apple IIe's display screen. Compare *graphics*.

translator: A generic term for a *compiler*, an *assembler*, or an *interpreter* that converts *source code* into equivalent *object code* or *machine code*.

utility: A program designed to perform any useful timesaving computer task.

variable: The name assigned to a mathematical or logical symbol that can assume different values during a program run.

vector: A vector is simply another name for a jump instruction. It transfers the CPU control, or vectors it, to some other memory location. See *ampersand vector*.

write: To transfer characters from the computer to a destination external to the computer (such as a disk drive, printer, or modem) or from the computer's processor to a destination external to the processor (such as main memory).

zero page: The first page (256 bytes) of the Apple IIe's memory, also called "page zero." Since the high-order byte of any address in this page is zero, only the low-order byte is needed to specify a zero-page address; this makes zero-page locations more efficient to address, in both time and space, than locations in any other page of memory.

Index

E

F

G

H

I

J

K

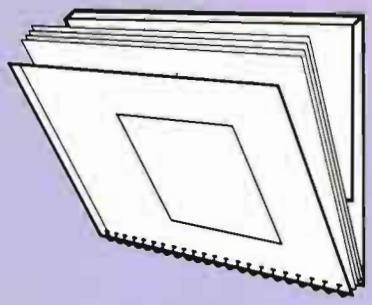
L

M

N

O

Apple Extended 80-Column Text/AppleColor Adaptor Card



Tuck end flap
inside back cover
when using manual.

20525 Mariani Avenue
Cupertino, California 95014
(408) 996-1010
TLX 171-576

030-0960-A