Chapter 6

I/O Configuration

The SoftCard version of CP/M can be modified for use with different I/O devices and software. This chapter describes the following areas of CP/M that can be modified:

The screen function interface

The Keyboard Character Definition Table

Patch areas for I/O software

All three areas can be changed or examined with the CONFIGIO utility program.

CONFIGIO

CONFIGIO is a utility program that changes designated areas of the BIOS. CONFIGIO consists of a series of menus that allow you to perform the following functions:

Examine and modify the screen function interface for use with an external terminal

Redefine keyboard characters

Load user I/O driver software into designated user patch areas

Save changes made with CONFIGIO on a system disk

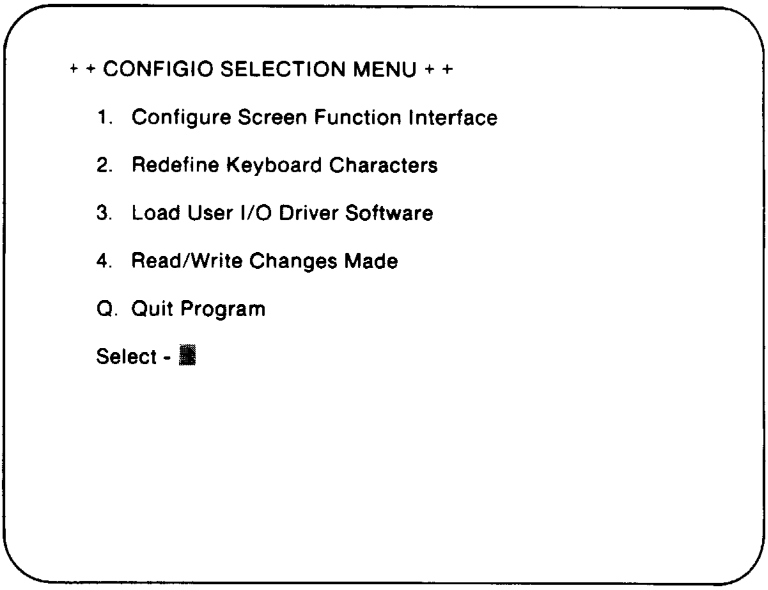
Running the CONFIGIO Program

The CONFIGIO program is on the Premium SoftCard IIe Master disk. To run it, insert a CP/M system disk that contains CONFIGIO.BAS and GBASIC.COM into drive A:. Load CP/M with a cold start. When you see the CP/M command level prompt A>, type

```
GBASIC CONFIGIO
```

and press the RETURN key.

When CONFIGIO has been loaded into memory, the screen displays a menu, as shown below. Each selection allows you to perform the task named. To select a task, press the number key corresponding to the task you wish to perform.



```
++ CONFIGIO SELECTION MENU ++

   1. Configure Screen Function Interface

   2. Redefine Keyboard Characters

   3. Load User I/O Driver Software

   4. Read/Write Changes Made

   Q. Quit Program

   Select - █
```

CONFIGIO Menu Selections

1. Configure Screen Function Interface

   This selection allows you to specify the control sequences required for an external terminal or application program to execute specific screen functions. Instructions for configuring the screen function interface for an external terminal are provided in the "Configuring the Screen Function Interface" section of this chapter.

2. Redefine Keyboard Characters

   This selection allows you to redefine the ASCII value assigned to any particular key on the keyboard, such as a seldom-used control character. Instructions for redefining keyboard characters are given in the "Redefining Keyboard Characters With CONFIGIO" section of this chapter.

3. Load User I/O Driver Software

   This option allows you to load the necessary I/O driver software into the patch areas for use with nonstandard Apple I/O devices or I/O software. If you are adding an I/O device that requires special I/O software, the technical manual for that device should give explicit instructions on how to load the I/O software into memory. If it does not, contact the manufacturer of the I/O device.

   If you are planning to add your own I/O software to the patch areas, read "Adding Nonstandard I/O Devices and User Software" in this chapter.

4. Read/Write Changes Made

   This option allows you to save the changes made with CONFIGIO menu selections 1 through 3. Instructions for using menu selection 4 are listed with instructions on using the other menu selections in this chapter.

Q. Quit Program

   Pressing *Q* exits the CONFIGIO program and returns to the CP/M operating system.

Screen Function Interface

The screen function interface controls how characters are displayed on the Apple screen or on the screen of an external terminal. Screen functions (also called screen attributes) are special control sequences that govern the display characteristics of the screen monitor or terminal. Some application programs are written for more than one computer and must be modified to display characters on the screen correctly.

Most popular terminals, including the standard Apple screen monitor, support special screen functions such as direct cursor addressing, screen clear, and highlighted text. Many CP/M application programs, such as word processing packages and business software, use these functions as part of the application display. The character sequences, however, often differ from terminal to terminal.

The screen function interface is configured for the standard Apple screen monitor. The Soroc IQ™ 120/IQ 140, Hazeltine™ 1500/1510, and Datamedia terminals can be used as external terminals without any modifications to the screen function interface. If you use an external terminal that is not compatible with your application software, special assembly language subroutines must be written to resolve the differences.

Screen Function Tables

The screen function interface solves the compatibility problem by translating the functions (as they are received from the user software) into the corresponding functions expected by the screen display's circuits. This is carried out by two translation tables: the Software Screen Function Table and the Hardware Screen Function Table.

The Software Screen Function Table recognizes an incoming screen function sequence and translates it into the corresponding sequence found in the Hardware Screen Function Table. This sequence is then sent to the terminal device.

Screen Functions Supported

The screen function interface recognizes and translates the following screen functions:

Clear Screen

Clears the entire screen, fills the screen with spaces, and places the cursor in the home position.

Clear to End-of-Page

Clears all information from the cursor (including the cursor position) to the end of the page.

Clear to End-of-Line

Clears all information from the cursor (including the cursor position) to the end of the line.

Set Normal (lowlight) Text Mode

Sets the normal video display mode; characters are displayed as white characters on a black background.

Set Inverse (highlight) Text Mode

Sets the inverse video display mode; characters are displayed as black characters on a white background.

Home Cursor

Moves the cursor to the first character position on the first line.

Address Cursor

Sets the cursor address for a specified printer offset.

Move Cursor Up

Moves the cursor up one line. If the cursor reaches the top line of the screen, it remains there and no scrolling occurs.

Move Cursor Forward

Moves the cursor one cursor position to the right, but does not destroy the character in that position. If the cursor is at the right end of the line, it will remain there.

In addition, there are two other screen functions which are used on all terminals: backspace and linefeed. The backspace character (ASCII 8) function moves the cursor backwards, and the linefeed character (ASCII 10) function moves the cursor down one line.

The control sequences for screen functions are a single control character or an ASCII character preceded by a single lead-in character. Control sequences consisting of three or more characters are not supported.

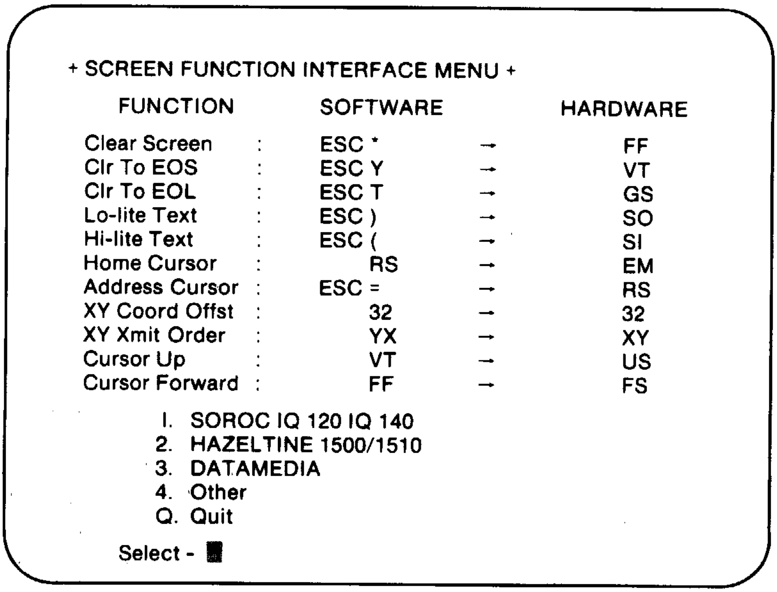
Configuring the Screen Function Interface

Load and run the CONFIGIO program as instructed in the "CONFIGIO" section at the beginning of this chapter.

Note

Before configuring the screen function interface for an external terminal, ensure that there is no accessory board installed in slot 3. If there is, turn the power off, remove the board, and use the standard Apple video screen monitor (see Figure 2.2 in Chapter 2 of the *Microsoft Premium SoftCard IIe System Installation and Operation Manual*). Once the configuration process is complete, you can reinstall the board and use its screen monitor as before.

When the CONFIGIO selection menu appears (see page 162), press the *1* key. CONFIGIO will display the Hardware and Software Screen Function Tables as shown below:



```
+ SCREEN FUNCTION INTERFACE MENU +
    FUNCTION          SOFTWARE                  HARDWARE
 Clear Screen    :    ESC *          →            FF
 Clr To EOS      :    ESC Y          →            VT
 Clr To EOL      :    ESC T          →            GS
 Lo-lite Text    :    ESC )          →            SO
 Hi-lite Text    :    ESC (          →            SI
 Home Cursor     :        RS         →            EM
 Address Cursor  :    ESC =          →            RS
 XY Coord Offst  :        32         →            32
 XY Xmit Order   :        YX         →            XY
 Cursor Up       :        VT         →            US
 Cursor Forward  :        FF         →            FS
            1.  SOROC IQ 120 IQ 140
            2.  HAZELTINE 1500/1510
            3.  DATAMEDIA
            4.  Other
            Q.  Quit
    Select - █
```

The previous menu shows the default values of the Hardware and Software Screen Function Tables. Items in the SOFTWARE column are the default control sequences of the Software Screen Function Table. Items in the HARDWARE column are the ASCII codes needed by the terminal hardware to perform the stated screen function. A NUL (ASCII 00) entry in either table indicates that the function is not available.

Three of the numbered entries in the lower section of the screen are for terminals for which CONFIGIO has data. To configure the screen function interface for any of the terminals listed, type the menu number corresponding to the terminal. For terminals not listed, or for application programs requiring modifications to the screen function interface, press the *4* key. To return to the main CONFIGIO menu, press the *Q* key.

For application programs requiring changes to the screen function interface, the Software Screen Function Table is modified. External terminals will usually require modifications to the Hardware Screen Function Table.

The Software Screen Function Table must match sequences sent by the application program to perform screen functions. The Hardware Screen Function Table must have non-zero entries in all of the nine functions. We recommend setting up the Software Screen Function Table to emulate a Soroc IQ 120/IQ 140 terminal. This is a common configuration that is supported by most CP/M software.

Configuring for an External Terminal

For Soroc IQ 120 or IQ 140 terminals, no changes are needed for either screen function table. However, when you first turn on Soroc terminals, text is shown in the "highlight" mode. CP/M will reset the screen to display in a normal "lowlight" mode whenever a cold start is performed.

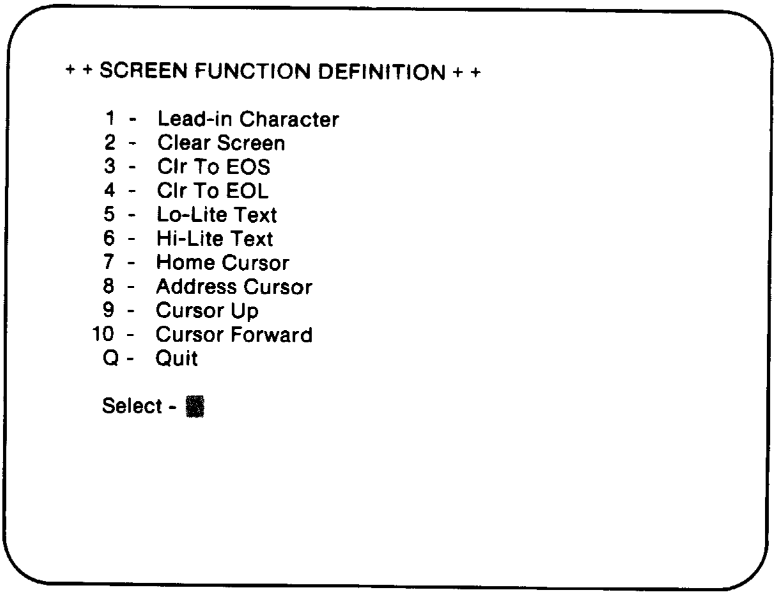
For Hazeltine 1500/1510 terminals, use the Hardware Screen Function Table only. (CP/M translates the Hazeltine cursor-addressing function with no XY coordinate offset.) We do not recommend using the Hazeltine screen function sequences in the software table. It is best to set up the hardware table for the Hazeltine, and the software table for another common terminal, such as the Soroc IQ 120/IQ 140.

For Datamedia terminals, set up the Hardware Screen Function Table only. (Datamedia terminal control sequences are not usually supported by CP/M software.) Set the hardware table for use with a 24x80 video board, and the software table for another common terminal type, such as the Soroc IQ 120/ IQ 140.

Note

> Highlight text and lowlight text screen functions (GBASIC commands INVERSE and NORMAL) are not supported by Datamedia terminals. Thus, the table entries specified for these functions are set to an arbitrary value to ensure that these two entries will have non-zero values.

To configure the screen interface for a terminal not listed in the menu, press the *4* key when the Screen Function Interface menu appears. CONFIGIO will load and display the list of configurable screen functions shown in the following figure.



```
+ + SCREEN FUNCTION DEFINITION + +

     1 -  Lead-in Character
     2 -  Clear Screen
     3 -  Clr To EOS
     4 -  Clr To EOL
     5 -  Lo-Lite Text
     6 -  Hi-Lite Text
     7 -  Home Cursor
     8 -  Address Cursor
     9 -  Cursor Up
    10 -  Cursor Forward
     Q -  Quit

     Select - █
```

You can now change any of the values in the Terminal Screen Function Definition Table.

Note

> The appropriate screen function command characters for your terminal are described in the technical manual for that terminal. To find out which codes are transmitted by a particular program (for example, a word-processing program), consult the manual for that program.

Select a number (1 through 10) to define the character sequences for any of the functions listed in Table 6.1.

Table 6.1.
Screen Function Descriptions

Number	Title	Description
1	Lead-in character	Defines the lead-in character: the character (usually an escape sequence) that precedes the screen function command character. A particular screen function may or may not require a lead-in character.
2	Clear screen	Clears the screen and places the cursor at the top left corner of the screen.
3	Clear to EOS	Clears the screen from the cursor to the end of the screen.
4	Clear to EOL	Clears the screen from the cursor to the end of the line.
5	Lowlight text	Sets the normal video mode for displaying text.
6	Highlight text	Sets inverse or double intensity video mode, depending on which mode your terminal supports.
7	Home cursor	Puts the cursor at the top left corner of the screen, but does not clear the screen.
8	Address cursor	Tells the terminal to go to a cursor address defined by the next two characters entered.
	XY coordinate offset	Defined as part of selection 8. The XY coordinate offset is the number that is added to the X and Y coordinates when they are sent to the terminal (usually 32).
	XY transmit order	Also defined as part of selection 8. Establishes the order in which coordinates are transmitted. Must be either XY or YX (usually YX).
9	Cursor up	Moves the cursor up one line on the screen.
10	Cursor forward	Move the cursor forward on a line without deleting the character under the cursor.

To assign an escape sequence to any of these functions, type the corresponding number and press the RETURN key.

For example, press the *1* key if you wish to specify a screen function lead-in character. The program will display:

```
LEAD-IN CHAR:
```

Enter the lead-in character required. Characters can be typed in any one of the following formats:

aaa	where *aaa* is a 2- or 3-character ASCII name.
c	where *c* is any keyboard character.
CONTROL-*c*	where *c* is any character.
LC-*c*	LC- indicates that the following character is lowercase. Type this in place of a lowercase character if your keyboard has no lowercase characters.
&H *hh*	*hh* is the ASCII hexadecimal code (preceded by &H). Use this format if the character cannot be typed. (See "ASCII Character Codes," Appendix H, in the *Microsoft BASIC Interpreter Reference Manual.*)

After you have entered the lead-in character, the program will ask:

```
SOFTWARE OR HARDWARE (S/H)?
```

If the lead-in character is to be used in the Software Screen Function Table, press the *S* key. If the lead-in character is to be used in the Hardware Table, press the *H* key.

To define any of the other screen functions, press the number for that function. The program will prompt you for the command character for that particular function.

The program then returns to the Screen Function Definition menu and waits for you to select another number or *Q*. You can make as many changes to the tables as you wish in this way.

The process for the address cursor function differs somewhat. If you press *8*, address cursor, the process is the same as with the other selections, until you see the prompt:

```
REQUIRE LEAD-IN (Y/N)?
```

After you answer this prompt by pressing *Y* or *N*, the computer displays:

```
XY COORD OFFST :
```

Type a numeral for the number of spaces that are to be added to the X and Y coordinates before they are transmitted. Finally, the program asks:

```
XY XMIT ORDER :
```

If the X and Y coordinates are transmitted in the order Y then X, enter *YX*. If the coordinates have been transmitted X then Y, enter *XY*.

The program then asks

```
SOFTWARE OR HARDWARE (S/H)?
```

and then continues in the same manner as with the other functions.

Configuring for Application Programs

Use the same procedure as that used for external terminals. Most application programs will give explicit instructions on how to configure the screen function interface. If a program requires changes to the screen function interface, but doesn't give instructions, use the following procedure:

1. Load and run the CONFIGIO program as instructed in the "CONFIGIO" section in the beginning of this chapter.

2. When the CONFIGIO selection menu appears (see page 162), press the *1* key. CONFIGIO will display the Screen Function Interface Menu as shown on page 167.

3. Press the *4* key.

4. Select the desired function by pressing the appropriate key or keys.

5. When

   ```
   SOFTWARE OR HARDWARE (S/H)?
   ```

   appears, press the *S* key.

6. Type the appropriate control sequence listed by the application program documentation.

7. Save the changes that you have made in the screen function interface. (See the following section for more information on saving changes.)

Saving the Changes to the Screen Function Interface

Save the changes made in the screen function interface by first pressing the *Q* key. When the main CONFIGIO menu appears, press the *4* key. The program will display:

```
+ READ/WRITE I/O CHANGES MADE +

Read Or Write (R/W)?
```

Press the *W* key. The program will display:

```
Destination Drive (A:-D:)?
```

Press the *A* key to save the changes made in the screen interface on the system disk in drive A:. The program will then display the main CONFIGIO menu.

Using the Screen Function Interface From Within a Program

The screen functions listed in Table 6.1, "Screen Function Descriptions," make it possible to write programs that perform special screen functions. Table 6.2, "Screen Function Interface Addresses," shows the correspondence between the Software and the Hardware Screen Function Tables in memory. It lists the function number and the hexadecimal address of each entry. The internal format of the two 11-byte tables is identical.

Table 6.2.

Screen Function Interface Addresses

Function Number	Software Table Address	Hardware Table Address	Function Description
	0F396H	0F3A1H	Cursor address coordinate offset. Range: 0 to 127. If the high-order bit is 0, the X and Y coordinates are expected to be transmitted Y first, X last. If the high-order bit is 1, the coordinates are sent X first, Y last.
	0F397H	0F3A2H	Lead-in character. This byte is zero if there is no lead-in character.
1	0F398H	0F3A3H	Clear screen.
2	0F399H	0F3A4H	Clear to end-of-page.
3	0F39AH	0F3A5H	Clear to end-of-line.
4	0F39BH	0F3A6H	Set normal (lowlight) text mode.
5	0F39CH	0F3A7H	Set inverse (highlight) text mode.
6	0F39DH	0F3A8H	Home cursor.
7	0F39EH	0F3A9H	Address cursor.
8	0F39FH	0F3AAH	Cursor up.
9	0F3A0H	0F3ABH	Cursor forward.

A NUL character entry in either Screen Function Interface Table will disable that function on the standard Apple screen monitor.

The standard Apple screen monitor supports all nine screen interface functions, independent of the Hardware Screen Function Table. However, if a Software Screen Function Table entry is zero, the function is disabled.

If the lead-in character of the Hardware Screen Function Table is OFF, the entire table is bypassed.

If a numbered table entry is zero, the function is not implemented.

If the entry has 1 as the high-order bit, the function requires a lead-in character.

An entry with the high-order bit set to zero indicates that the function does not require a lead-in character.

To ensure portability, the Hardware Screen Function Table must be set up correctly for the specific terminal. The following example lists a short segment of 8080 assembly language code which illustrates the use of the Screen Function Tables for terminal independent screen programming.

```
;               Terminal Independent Screen I/O
;
;               This routine will execute the screen function specified by E, where E
;               contains the screen function number from one to nine. If the function is
;               not implemented, the subroutine simply returns, and all registers are
;               destroyed.
;
;               Equates:
;
BDOS            EQU    0005H            ;CP/M function call address
SXYOFF          EQU    0F396H           ;Software cursor address XY coordinate offset
SFLDIN          EQU    0F397H           ;Software function lead-in character
SSFTAB          EQU    0F398H           ;Software screen functions
;
SCRFUN:         MVI    D,0              ;Prepare for index
                LXI    H,SSFTAB-1       ;Point to Software Screen
                                        ;Function Table minus one
                DAD    D                ;Index to desired function character
                MOV    A,M              ;Get the character
                ORA    A                ;See if a lead-in is required
                RZ                      ;If the function isn't there, quit
                JP     CONOUA           ;If positive, no
                PUSH   PSW              ;Save character
                LDA    SFLDIN           ;Get software lead-in character
                CALL   CONOUA           ;Output character in A
                POP    PSW              ;Get character again
CONOUA:         MOV    E,A              ;Put character in its place
CONOUE:         MVI    C,2              ;Console output function
                JMP    BDOS             ;Call CP/M BDOS at 0005H
;
;               This routine will position the cursor at the X,Y coordinates in HL.
;
GOTOXY:         PUSH   H                ;Save coordinates while we do sequential
                                        ;addressing
                MVI    E,7              ;Do an address cursor function
                CALL   SCRFUN
                POP    H                ;Get coordinates back
                LDA    SXYOFF           ;Get software XY coordinate offset
                ORA    A                ;Set CC's on A
                JP     NORVS            ;Reverse coordinates if negative
                MOV    E,L              ;Reverse H and L
                MOV    L,H
                MOV    H,E
NORVS:          MOV    E,A              ;Save offset
                ADD    H                ;Add offset
                MOV    H,A              ;Save for later
                MOV    A,E              ;Get offset again
                ADD    L
                PUSH   H                ;Save all this
                CALL   CONOUA           ;Output first coordinate
                POP    H                ;Restore coordinates
                MOV    E,H              ;Output second coordinate and return
                JMP    CONOUE
```

Keyboard Character Definition

Some CP/M application programs require the use of keys which are not available on the Apple keyboard. For example, the Apple IIe keyboard does not have a RUBOUT key. This can be resolved by redefining specific keys in the Keyboard Character Definition Table located at memory locations F3ACH through F3B7H.

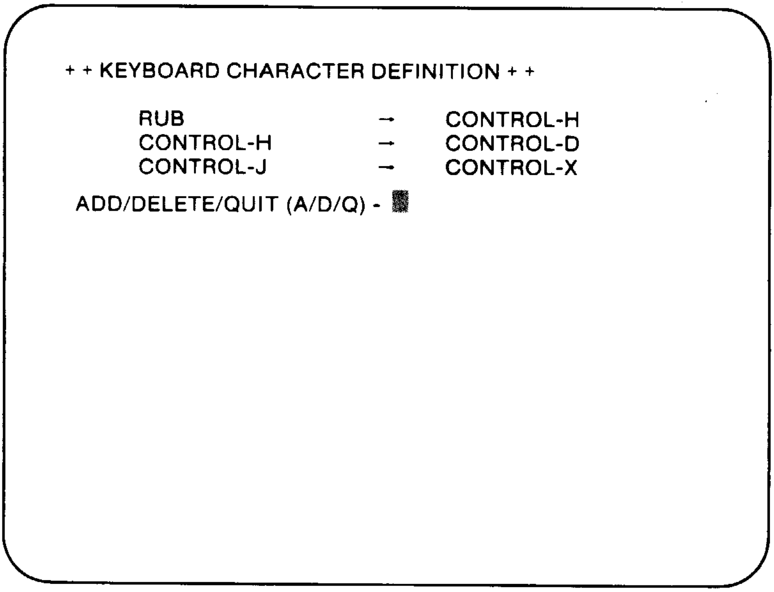
Keyboard Character Definition Table

The Keyboard Character Definition Table supports up to six character redefinitions. Entries in the table consist of two bytes: the first byte is the ASCII value of the keyboard character to be redefined, and the second byte is the desired ASCII value of the character. Both bytes must have their high-order bits cleared.

If there are fewer than six entries in the Keyboard Character Definition Table, a byte with the high-order bit set is put at the end of the table.

Redefining Keyboard Characters With CONFIGIO

Load and run the CONFIGIO program as instructed in the "CONFIGIO" section in the beginning of this chapter. When the first CONFIGIO selection menu appears, press the *2* key. CONFIGIO will display the Keyboard Character Definition menu as shown below:



```
+ + KEYBOARD CHARACTER DEFINITION + +

        RUB                 →    CONTROL-H
        CONTROL-H           →    CONTROL-D
        CONTROL-J           →    CONTROL-X

 ADD/DELETE/QUIT (A/D/Q) - █
```

To redefine a character response for a key, press the *A* key. To delete an entry from the table, press the *D* key. Press the *Q* key to return to the main CONFIGIO menu.

When you press the *A* key, the CONFIGIO program displays:

```
CHAR:
```

Type the character or character sequence to be defined. The table entry can be typed in one of the following formats:

aaa	where *aaa* is a 2- or 3-character ASCII name.
c	where *c* is any character.
CONTROL-*c*	where *c* is any keyboard character.
LC-*c*	LC- indicates that the following character (*c*) is lowercase. Type this in place of a lowercase character if your keyboard has no lowercase characters.
&H *hh*	*hh* is the ASCII hexadecimal code (preceded by &H). Use this format if the character cannot be typed. See "ASCII Character Codes," Appendix H, in the *Microsoft BASIC Interpreter Reference Manual.*

Save the changes made to the Keyboard Character Definition Table by pressing the *Q* key. When the main CONFIGIO menu appears, press the *4* key. The program will display:

```
+ READ/WRITE I/O CHANGES MADE +

READ OR WRITE (R/W)?
```

Press the *W* key. The program will display:

```
DESTINATION DRIVE (A:-D:)?
```

Press the *A* key to save the changes made in the screen function interface on the system disk in drive A:. The program will then display the main CONFIGIO menu.

Example

CONTROL-C can be redefined as a NUL character (ASCII code 00) to prevent the user from exiting a BASIC program. This is accomplished by running the CONFIGIO program and selecting "2. Redefine Keyboard Characters" from the main CONFIGIO menu.

When the Keyboard Character Definition menu appears, press the *A* key. When the CHAR: prompt appears, type:

```
CONTROL-C
```

and press the RETURN key. If the character is acceptable, the program prompts you to enter the new definition of the character with an arrow as shown:

```
CONTROL-C →
```

Now type

```
NUL
```

and press the RETURN key. If your entry is not acceptable, the computer will erase what you have just entered and wait for an acceptable character entry.

If the entry is acceptable, the Keyboard Character Definition menu is displayed again with the new definitions added to the menu.

Note

If you have followed the example, you will find that you cannot exit the CONFIGIO program with CONTROL-C.

To delete the entry just made, type *D*. CONFIGIO will display the CHAR: prompt again. Now type

```
CONTROL-C
```

and press the RETURN key. The list is displayed again with the CONTROL-C → NUL entry deleted.

Type *Q* to return to the main menu.

Notes on Keyboard Character Definition

We recommend that you delete entries to the Keyboard Character Definition Table if they do not apply to your keyboard. For example, if your keyboard has a RUBOUT key, you should delete the DEL entry.

Redefining CONTROL-C as a NUL character to prevent exiting BASIC programs with CONTROL-C is useful, but it can cause problems at CP/M command level. CONTROL-C is used by CP/M for a warm start.

Certain terminals and 80-column display boards perform their own character redefinitions. For example, the Videx™ Videoterm™ display board uses CONTROL-A to switch between uppercase and lowercase input mode. Since CONTROL-A is also used in BASIC to enter edit mode, we recommend redefining another character as CONTROL-A (such as CONTROL-W).

Adding Nonstandard I/O Devices and User Software

The user patch areas and the I/O Vector Table provide a means of using nonstandard I/O devices with CP/M or adding special I/O software. I/O devices include printers, communication interface boards, modems, and other physical devices in addition to terminals. I/O software can be either *substitution* routines or *filter* routines.

Note

> Most Apple I/O interface boards contain 6502 ROM drivers. The easiest way to interface these board types to SoftCard CP/M is to call the 6502 subroutines in the ROM. This should be sufficient to interface most common I/O devices to SoftCard CP/M. (See "0 CALLSUB" in Chapter 4 for more information on calling subroutines.)

Substitution routines are the assembly language routines which allow CP/M to communicate with nonstandard I/O devices. ("Nonstandard" applies to any device that is not normally configured for CP/M or Apple Pascal). Most accessory boards will have an accompanying substitution routine for interfacing the board to CP/M.

Substitution routines also include routines that change the normal format of I/O data (from the I/O device) with which the BIOS communicates. The SoftCard version of CP/M treats all substitution routines as "type 1" vector patches. Type 1 vector patches are user-written assembly language routines that are not dependent on the standard BIOS routines.

Filter routines are assembly language routines that change the input data before sending it to the standard BIOS I/O routines. They are called filter routines because they filter the incoming data. Filter routines are considered "type 2" vector patches.

Any I/O routines added to CP/M must be written into the designated user patch areas of the BIOS. I/O routines must have code that alters the BIOS vector so that the BIOS vector points to the user-written routine instead of the standard I/O routine. If your I/O routine is a substitution type of routine, no further action is necessary. If, however, it is a filter type, the normal BIOS vector must be saved and placed in your routine.

User Patch Areas

The SoftCard version of CP/M provides four 64-byte areas for user-written I/O assembly language routines. Three of the areas are for a certain slot. The fourth is for general usage. Table 6.3 shows the memory location of each patch area, the slot assignment, and the assigned logical device for the patch area.

Table 6.3.

User Patch Areas

Address Range	Assigned Slot	Assigned Logical Device
0FE00H—0FE3FH	1	LST:
0FE40H—0FE7FH	2	PUN: and RDR:
0FE80H—0FEBFH	3	TTY:
0FEC0H—0FEFFH	None	Use for filter routines or to continue a substitution routine.

If there is no board installed in a particular slot, its allocated 64-byte space in the patch area can be used for other purposes relating to its assigned logical device.

I/O Vector Table

All of the "primitive" character I/O functions used by the Apple I/O system are performed through the I/O Vector Table. These vectors point to the standard I/O subroutine located in the CP/M BIOS, but can be altered by the CONFIGIO program to point to user-installed I/O driver subroutines.

I/O driver subroutines are "patched" to CP/M by adding the appropriate I/O vector which points to the specified subroutine. Table 6.4 lists vector locations and their purposes.

Table 6.4.

I/O Vector Table Description

Number	Address	Name	Description
1	0F3C0H	Console Status	If a character is ready, the console status returns 0FFH in register A. If not, 00H is returned.
2	0F3C2H	Console Input vector 1	Reads a character from the console into the A register with the high-order bit clear.
3	0F3C4H	Console Input vector 2	Same as Console Input vector 1.
4	0F3C6H	Console Output vector 1	Sends the ASCII character in register C to the console device.
5	0F3C8H	Console Output vector 2	Same as Console Output vector 1.
6	0F3CAH	Reader Input vector 1	Reads a character from the paper tape reader device into register A.
7	0F3CCH	Reader Input vector 2	Same as Reader Input vector 1.
8	0F3CEH	Punch Output vector 1	Sends the character in register C to the paper tape punch device.
9	0F3D0H	Punch Output vector 2	Same as Punch Output vector 1.
10	0F3D2H	List Output vector 1	Sends the character in register C to the line printer device.
11	0F3D4H	List Output vector 2	Same as List Output vector 1.

Note

> If a Console Output vector is specified, the B register will contain a number corresponding to a screen function output. (The B register contains zero during normal character output.) The B register will also contain a non-zero number during the output of the address cursor function (X and Y coordinates) after executing screen function number 7.

Adding I/O Software to the User Patch Areas

To add I/O software to the user patch areas, you must first create an executable COM file with the ASM and LOAD programs. Then, load the file into the patch area with the CONFIGIO program. CONFIGIO will also save the changes to a CP/M system disk.

When creating the COM file, the first 11 bytes of the actual routine must be in the format shown in Table 6.5. Only one patch routine can be written into a patch area per COM file. You can use as many vectors in the I/O Vector Table as desired. Examples of patch routines are given in "Substitution I/O Routine Example," and "Filter I/O Routine Example," at the end of this section.

Table 6.5.
Format for User-Written Patch Routines

Byte	Contents
1	The number of patches to I/O Vector Table to be made.
2 and 3	The destination address of the patch routine.
4 and 5	The length of the routine.
6*	Vector patch type which is either type 1 or type 2. 1 = substitution patch 2 = filter patch
7*	The vector number (1—11) to be patched.
8 and 9*	If the routine is a type 1 patch, bytes 8 and 9 contain the address to be patched into the vector. The address points to the user's code. If the routine is a type 2 patch, bytes 8 and 9 contain the address where the current contents of the specified vector are placed. (This can be the address field of a JMP instruction, etc.)
10 and 11*	The new address to be placed in the specified vector.

* Bytes 1 through 5 are repeated for each I/O vector patch made. If there is more than one patch made, then bytes 7 through 11 will be offset by the number of times you repeat bytes 1 through 5.

The actual program code follows the patch information described in Table 6.5. Conversion restricts the size of the program code to 64 bytes per slot-dependent patch area. Use the patch area appropriate for your application and slot use. (See Table 6.3 for more information on user patch areas.)

Steps for Adding I/O Software to the User Patch Areas

If the software already exists in a disk file, start the procedure at step 3. If you are entering a program from the keyboard, continue with the next step.

1. Use the DDT "S" command to enter the program into memory at location 100H.

2. Save the program with the SAVE command by typing:

   ```
   SAVE 1 filespec
   ```

   and then pressing the RETURN key to execute the command.

3. Run the CONFIGIO program. When the main menu appears, press the *3* key. The program will display:

   ```
                ++ LOAD USER I/O DRIVER SOFTWARE ++
   SOURCE FILE NAME?
   ```

4. Type the *filespec* of the file containing the I/O software and press the RETURN key. The program will display

   ```
   LOADING...
   ```

   as it loads the routines from file into the user patch area. After the routines are loaded, the program returns to the main CONFIGIO menu.

5. Press the *4* key. CONFIGIO will display:

```
                + READ/WRITE I/O CHANGES MADE +

          READ OR WRITE (R/W)?
```

6. Press the *W* key to write the I/O software into the BIOS in memory. The program will display:

```
          DESTINATION DRIVE (A:-D:)?
```

   To save the changes to the BIOS on the system disk that is in active drive, press the *A* key. For any other CP/M system disk, go to step 4.

7. Insert a CP/M system disk into the appropriate drive. Type the corresponding drive letter and press the RETURN key. The BIOS on the disk is replaced with the one currently in memory. When the BIOS is copied into the CP/M system tracks on the destination disk, the program returns to the main CONFIGIO menu.

Note

Pressing the *R* key in step 6 permits the BIOS to be read from the CP/M system disk and loaded into memory. When the operation is complete, the program returns to the I/O Configuration Program menu.

Substitution I/O Routine Example

```
;
; Substitution routine
;
; This is a substitution routine for a second printer installed in the slot defined by "SLOT".
; This routine assumes there is a CCS 7710 interface board installed in SLOT and that all
; protocol is done elsewhere.
;
; SLOT is the value used to build the addresses used for status and data for the CCS 7710.
; Change it to whatever slot the CCS board is in.
;
; Miscellaneous definitions:
;
0002 =          SLOT     EQU   2
C0A0=           STAT     EQU   0C080H+(SLOT SHL 4)
C0A1=           DATA     EQU   STAT+1
FE00 =          DEST     EQU   0FE00H
;
; 6502 Subroutine call definitions
;
0040 =          X6502    EQU   40H                       ;6502 transfer address
0045 =          AREG     EQU   45H                       ;6502 register A pass area
004A =          ADDR     EQU   4AH                       ;Address to PEEK or POKE
0049 =          CMD      EQU   49H                       ;Command pass area
0002 =          PEEK     EQU   2                         ;The PEEK command
0003 =          POKE     EQU   3                         ;The POKE command
;
                OFFSET   SET   DEST-UL1
0100                     ORG   100H
;
; Information block for CONFIGIO
;
0100 01                  DB    1                         ;Type 1 patch
0101 00FE                DW    DEST                      ;Tells CONFIGIO where to put it
0103 2900                DW    LENGTH                    ;How many bytes to store
0105 01                  DB    1                         ;Type 1 patch
0106 0B                  DB    11                        ;Patch list out vector 2 (UL1 :)
0107 00FE                DW    DEST                      ;Address to patch into vector
                                                         ;table
;
; The actual driver
;
0109 3E02       UL1:     MVI   A,PEEK                    ;Do a PEEK command
010B 324900              STA   CMD                       ;Store the command
010E 21A0C0              LXI   H,STAT                    ;This is the address we want to
                                                         ;read
0111 224A00              SHLD  ADDR                      ;Store the address
0114 CD4000              CALL  X6502                     ;Go read the 6502 memory
                                                         ;location
0117 3A4500              LDA   AREG                      ;Get the result
011A E602                ANI   2                         ;Mask status bit
;
```

```
011C CA00FE              JZ     UL1+OFFSET          ;If zero, then character not ready
                                                    ;to send
011F 79                  MOV    A,C                 ;Get the character
0120 324500              STA    AREG                ;Store it for the 6502
0123 3E03                MVI    A,POKE              ;POKE this byte
0125 324900              STA    CMD                 ;Store the command
0128 21A1C0              LXI    H,DATA              ;This is the address we want to
                                                    ;write
012B 224A00              SHLD   ADDR                ;Store the address
012E CD4000              CALL   X6502               ;Go POKE the output byte
0131 C9         RET                                 ;And go home
;
0029 =          LENGTH   EQU    $-UL1
0132                     END
```

Filter I/O Routine Example

```
;
; nolf - eliminate extra linefeeds
;
; This program removes linefeeds from a CR-LF sequence sent to the printer.
;
0100                     ORG    100H
                ORIGIN   SET    0FE3FH-LENGTH       ;Origin for ASM
                OFFSET   SET    ORIGIN-NOLF         ;True origin:end
                                                    ;
0100 01                  DB     1                   ;Number of patches
0101 2BFE                DW     ORIGIN              ;Place to put code
0103 1400                DW     LENGTH              ;Length of code
0105 02                  DB     2                   ;Type of patch
0106 0A                  DB     10                  ;Vector to change
0107 3DFE                DW     NOTCR+OFFSET+1      ;Place to put old vector contents
0109 2BFE                DW     ORIGIN              ;New vector contents
                                                    ;
010B 00         CRFLAG   DB     0                   ;Last character to pass
                                                    ;through
                                                    ;
010C 212AFE     NOLF:    LXI    H,CRFLAG+OFFSET     ;Point to crflag
010F 7E                  MOV    A,M                 ;Get last character
0110 71                  MOV    M,C                 ;Save current character
0111 FE0D                CPI    13                  ;Last char a cr?
0113 C23CFE              JNZ    NOTCR+OFFSET        ;No...just pass through
0116 79                  MOV    A,C                 ;Get current character...
0117 FE0A                CPI    10                  ;Is it a line feed?
0119 C8                  RZ                         ;Yes...don't print it
011A FE0D                CPI    13                  ;Is it another cr?
011C C8                  RZ                         ;Yes...don't print it
011D C30000     NOTCR:   JMP    0000                ;This address patched
                                                    ;by CONFIGIO
;
0014 =          LENGTH   EQU    $-NOLF
0120                     END
```

I/O Device Protocols for Assembly Language Programs

The I/O device protocol is similar to that supported by the initial release of Apple Pascal, which requires the installation of accessory board types in specific accessory slots. Table 6.5 shows the required slot assignments. In addition to the standard Apple I/O devices, the SoftCard implementation of CP/M supports many other I/O devices.

Note

> Contact your dealer or Microsoft Corporation to determine which I/O devices are compatible with the Premium SoftCard IIe System.

Table 6.6.

Accessory Slot Addresses and Assignments

Slot	Acceptable Board Type	Slot Address
1	2,3,4,6	C100H—C1FFH
2	2,3,4,6 (input) 1,2,3,4,6 (output)	C200H—C2FFH
3	2,3,4,6	C300H—C3FFH
4	Any type	C400H—C4FFH
5	2	C500H—C5FFH
6	2	C600H—C6FFH
7	Any type	C700H—C7FFH

Type 1 is an unknown board type.

Type 2 is Apple II Disk Controller.

Type 3 is an Apple Communications Interface board or California Computer Systems® 7710A™ Serial Interface board.

Type 4 is an Apple High-Speed Serial Interface board or Apple Silentype® interface board.

Type 5 is an Apple Parallel Printer Interface board.

Type 6 is an Apple Firmware board.

Slots Type Table

The programmer may access the Slots Type Table from within a program. The table is located at addresses F3B9H through F3BFH.

When CP/M is loaded into memory with a cold start, each of the Apple I/O accessory slots is checked to see if a standard Apple peripheral board is installed. This is done by checking to see if there is ROM present in the slot-dependent address allocated for accessory board ROMs and then comparing two signature bytes to those of the standard Apple I/O peripheral boards.

This information is then stored in the Slots Type Table, which is located at 3B8H in the I/O Configuration Block. There are seven bytes in the Slots Type Table, each byte corresponding to the seven slots from one to seven. The value of a table entry may range from zero to six. The meaning of each value is as follows:

Value	Explanation
0	No peripheral board ROM was detected which usually means that no board is installed in the slot.
1	A peripheral board ROM was detected, but it is of an unknown type.
2	An Apple Disk II Controller board is installed in the slot.
3	An Apple Communications Interface board or CCS 7710A Serial Interface board is installed in the slot.
4	An Apple High-Speed Serial Interface, Videx Videoterm, M&R Sup'R Terminal or Apple Silentype printer interface is installed in the slot.
5	An Apple Parallel Printer Interface is installed in the slot.
6	An Apple Firmware Card is installed in the slot.

Disk Drive Byte

Disk drive byte is a single byte for monitoring the number of disk drives in the system. The byte is equal to the number of disk controller boards in the system multiplied by two. This value does not reflect an odd number of disk drives such as only one drive connected to a controller board. The disk drive byte is located at Z80 address F38BH.

Chapter 7

Using the SoftCard With Apple Programs

The SoftCard display and memory features can be used with Apple DOS, Apple Pascal and Apple BASIC programs as well as CP/M. This chapter describes how to use the Microsoft Premium Softcard IIe System display and memory features with your Apple programs.

SoftCard Features Under Apple DOS

When Apple DOS, Apple Pascal, or Apple BASIC (Applesoft® or Integer BASIC) programs are running, the SoftCard circuit board can be used exactly like the Apple 80-Column Text Card or Extended 80-Column Text Card. The SoftCard recognizes all commands for either board.

The display circuitry of the SoftCard supports all special features and commands associated with Apple IIe 80-column firmware contained on the Apple IIe motherboard. Display firmware is described in Chapter 3 of the *Apple IIe Reference Manual*. Instructions for using the SoftCard display features are described in the next section, "Using the SoftCard Display Features."

The SoftCard contains 64K bytes of random access memory that can be accessed by the 6502 microprocessor. In the 40-column display mode, the entire SoftCard memory is available for data storage. In the 80-column display mode, 63K bytes are available for storage and the remaining 1K byte is reserved for the 80-column display.

Using the SoftCard Display Features

When running Apple DOS or Apple BASIC programs, the SoftCard display features can be switched between 40-column and 80-column display modes. The Apple Pascal operating system, like CP/M, uses the 80-column display mode only.

Switching Between Displays

There are two methods for switching display modes when running either Apple DOS or Apple BASIC programs. The first method is activating and deactivating the display circuitry of the SoftCard. The second method is a temporary switch between display modes using escape key sequences. Most often, the display features will be activated when Apple DOS or Apple BASIC is loaded into memory. Then, the escape key sequences are used to switch between display modes. Deactivating the SoftCard display features is performed only when required. For example, the display features are turned off when sending output to certain I/O devices such as an Apple Silentype printer. Table 7.1 shows the commands for switching display modes.

Table 7.1.

SoftCard Display Mode Commands

Command	**Purpose**
PR#3	Activates the 80-column display features of the SoftCard circuit board.
ESC-4	Changes to the 40-column display mode.
ESC-8	Changes to the 80-column display mode.
ESC-CONTROL-Q	Deactivates the 80-column display features of the SoftCard circuit board.
CONTROL-RESET	Deactivates the 80-column display features and clears certain portions of memory.

Changing the Default Display Mode

The 40-column display is the default display mode for Apple DOS or Apple BASIC programs. By modifying the HELLO program, you can automatically switch to the 80-column display mode whenever Apple DOS is loaded into memory. To modify the HELLO program, perform the following steps:

1. Insert a copy of the DOS master disk into drive 1 and load the HELLO program by typing

   ```
   LOAD HELLO
   ```

2. When you see the Apple DOS "]" prompt, type

   ```
   LIST
   ```

3. When you see the "]" prompt again, type the following program lines:

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

4. Now type

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

This procedure can be repeated to change the default display mode for other Apple DOS system disks.

Deactivating the 80-Column Display Mode

There are two commands for deactivating the 80-column display mode: ESC-CONTROL-Q and CONTROL-RESET. The preferred method of deactivating the 80-column display mode is by typing ESC-CONTROL-Q. Typing CONTROL-RESET, under certain conditions, could erase your program from memory. Using the CONTROL-RESET method will also fill the screen with random characters when the system changes display modes. The random characters are in the video memory only and will scroll off the screen when entering new data.

The only time you should deactivate the 80-column display mode is when certain application programs require you to do so, or when sending the output of the program to another device such as a printer.

Cursor Symbols

Whenever you change display modes, the cursor will change shape. Table 7.2 shows the different shapes of the cursor and the corresponding display mode.

Table 7.2.

Cursor Symbols

Cursor	Mode	Remarks
▮	Active	The cursor is always a rectangle whenever CP/M or Pascal operating systems are running. Under Apple DOS or Apple BASIC, the cursor will be a rectangle after you activate the 80-column display.
▒	Inactive	When Apple DOS or BASIC is loaded into memory at system startup time, the cursor is shown as a flashing checkerboard.
⊞	Active	The cursor is shown as in inverse cross in a rectangle when ESC-8 is typed.*
⊞	Inactive	The cursor is shown as an inverse cross in a square when ESC-4 is typed.*

* Applicable in Apple DOS or BASIC only.

80-Column Operation

80-column operation is defined when the 80-column display circuits of the SoftCard are active. When active, the operation of certain display commands are modified. Table 7.3 lists the Applesoft BASIC display commands that are affected when 80-column display mode is active.

Table 7.3.

Restrictions for Using Applesoft BASIC Commands in 80-Column Display

Command	Action
FLASH	*80-column active mode*: Not available in this mode. *80-column inactive mode*: FLASH performs as described in Chapter 4 of the *Applesoft BASIC Programming Reference Manual*; characters flash between normal and inverse video.
INVERSE	*80-column active mode*: INVERSE performs as described in Chapter 4 of the *Applesoft BASIC Programming Reference Manual*; black characters on a white screen. Both uppercase and lowercase characters are allowed. The HOME command clears the screen to a white background. *80-column inactive mode*: Performs the same way as in the 80-column active mode but restricted to uppercase characters only. The HOME command clears the screen to a black background.

Table 7.3. *(continued)*

Command	**Action**
NORMAL	*80-column active mode*: NORMAL performs as described in Chapter 4 of the *Applesoft BASIC Programming Reference Manual*; it clears the screen to a black background. *80-column inactive mode*: Performs the same way as in the 80-column active mode.
HTAB	*80-column active mode*: HTAB performs as described in Chapter 4 of the *Applesoft BASIC Programming Reference Manual* when using the ESC-4 sequence to display 40 columns. When displaying 80 columns, HTAB is unavailable. (Use the POKE 36,*xx* command instead.) *80-column inactive mode*: HTAB performs as described in Chapter 4 of the *Applesoft BASIC Programming Reference Manual.*
Comma Tabbing	*80-column active mode*: Comma tabbing performs as described in "Print Format" in Chapter 1 of the *Applesoft BASIC Programming Reference Manual* when using the ESC-4 sequence to display 40 columns. Unavailable in the 80-column display mode. *80-column inactive mode*: Comma tabbing performs as described in Chapter 1 of the *Applesoft BASIC Programming Reference Manual.*

Escape Key Sequences

Table 7.4 lists the escape key sequences that can be used when the SoftCard display features are active.

Table 7.4.

Escape Key Sequences for Display Modes

Sequence	Action
ESC-@	Clears the window and moves the cursor to its HOME position.
ESC-A	Moves the cursor up one line.
ESC-B	Moves the cursor right one position.
ESC-C	Moves the cursor left one position.
ESC-D	Moves the cursor down one line.
ESC-E	Clears to the end of the line.
ESC-F	Clears to the bottom of the window.
ESC-I	Moves the cursor up one line and turns on escape mode.
ESC-↑	Same as ESC-I.
ESC-J	Moves the cursor left one position and turns on escape mode.
ESC-←	Same as ESC-J.
ESC-K	Moves the cursor right one position and turns on escape mode.
ESC-→	Same as ESC-J.
ESC-M	Moves the cursor down one line and turns on escape mode.
ESC-↓	Same as ESC-J.
ESC-R	Turns on uppercase restrict mode.
ESC-T	Turns off uppercase restrict mode.
ESC-4	Turns on the 40-column display mode.
ESC-8	Turns on the 80-column display mode.
ESC-CONTROL-Q	Deactivates the 80-column circuits of the SoftCard.

Control Key Sequences

Some of the control key sequences that work with Apple BASIC programs work differently in the active 80-column display mode. Table 7.5 lists the control key sequences and their actions for both 80-column active and inactive modes.

Table 7.5.

Control Key Sequences

Sequence	Apple IIe Name	ASCII Code	Action
CONTROL-G	Bell	7	Produces a tone (1000 Hz) for 0.1 seconds.
CONTROL-H	Backspace	8	Moves the cursor one position to the left.
CONTROL-J	Linefeed	10	Moves the cursor down to the next line.
CONTROL-K	Clear EOS	11	Clears from the cursor position to the end of the screen.
CONTROL-L	Clear	12	Moves the cursor to the left corner of the screen and clears the entire screen.
CONTROL-M	Return	13	Moves the cursor to the left-most column of the next line.
CONTROL-N	Normal	14	Sets the display format to normal mode.
CONTROL-O	Inverse	15	Sets the display format to inverse mode.
CONTROL-Q	40-column	17	Sets the display to the 40-column mode of operation.
CONTROL-R	80-column	18	Sets the display to 80 columns.

Table 7.5. *(continued)*

Sequence	Apple IIe Name	ASCII Code	Action
CONTROL-S	Stop list	19	Stops the output to the display until another key is pressed. This command will only work at Apple DOS command level and not from within a program.
CONTROL-U	Quit	21	Deactivates the 80-column circuits, places the cursor in the HOME position and clears the screen.
CONTROL-V	Scroll	22	Scrolls the display down one line with the cursor in the same column as before.
CONTROL-W	Scroll-up	23	Scrolls the display up one line with the cursor in the same column as before.
CONTROL-Z	Home	25	Moves the cursor to the upper left corner of the screen.
CONTROL-	Forward space	28	Moves the cursor right one position. If the cursor is on the left side of the screen, it will be moved to the left end of the line below.
CONTROL-]	Clear EOL	29	Clears to the end of the line from the cursor.
CONTROL-^	GOTO xy	30	Moves the cursor to the coordinate given. This key sequence is not supported by Apple BASIC programs.

With the exception of CONTROL-Q, CONTROL-H, CONTROL-J, and CONTROL-M, control key sequences are only available when the SoftCard display features are active.

CONTROL-N, CONTROL-O, CONTROL-Q, CONTROL-R, and CONTROL-U are available only from BASIC programs and will not work at Apple DOS command level.

Using the SoftCard 64K-Byte Memory

The Premium SoftCard IIe System has 64K bytes of additional memory available for data storage. In Apple nomenclature, any memory not contained on the Apple motherboard is referred to as *auxiliary memory*. Therefore, all of the SoftCard memory is considered to be auxiliary memory.

Of the 64K bytes of SoftCard memory, one kilobyte (addresses 0400H—7FFH) is reserved for the 80-column video display. The other 63K bytes can be used for data storage. If the SoftCard display circuits are inactive, all 64K of the SoftCard's memory can be used for data storage.

SoftCard Video Memory Operation

When the SoftCard 80-column features are active, half of the data for the 80-column display is stored in main memory on text page 1 and the remaining data is stored in SoftCard memory. The 80-column display circuits receive data from both memory areas simultaneously and display them as two adjacent characters.

Both memory areas are connected to the address bus in parallel. This allows access to both memory areas during a display cycle. When in the 40-column display mode, the SoftCard display circuits use every other clock cycle to get data from memory. In 80-column display mode, the remaining clock cycles are for processing the additional display data from SoftCard memory.

In 80-column mode operation, a byte of display data from main memory is sent to a buffer on the main logic board, and the display data from SoftCard memory goes into a buffer on the SoftCard. The data bytes from these buffers are then switched onto the video data bus on alternate clock cycles. (The first byte is from the SoftCard and the next byte is from main memory and so on.) The main memory displays characters in odd columns and the auxiliary SoftCard memory displays characters in even columns.

Because the 80-column display contains twice as many characters as the 40-column, it has to output twice as many pixels across the screen horizontally. Pixels are the dots that compose screen characters or graphics. This doubled output increases the data transmission rate to the screen from 7 MHz to 14 MHz, effectively making the characters more narrow and therefore more dim on a normal video monitor. Therefore, to produce a satisfactory 80-column display, a monitor with a bandwidth of at least 14 MHz is required.

This video data transmission scheme applies only to the video display. Access of data in the SoftCard memory is accomplished by switching the data bus to Read/Only from the SoftCard as described in the previous paragraphs. For more information about how the Apple IIe display memory operates, see Chapter 2 and Chapter 7 of the *Apple IIe Reference Manual.*

Addressing SoftCard Memory

Because the 6502 microprocessor can only address 64K bytes of memory at a time, special circuitry has been built into the Apple motherboard to access auxiliary memory. Thus, the locations in the 64K address space can be either in the main memory (Apple motherboard) or in auxiliary memory (SoftCard circuit board). Figure 7.1, "Memory Mapping Under Apple DOS," shows the address mapping of both memory areas.

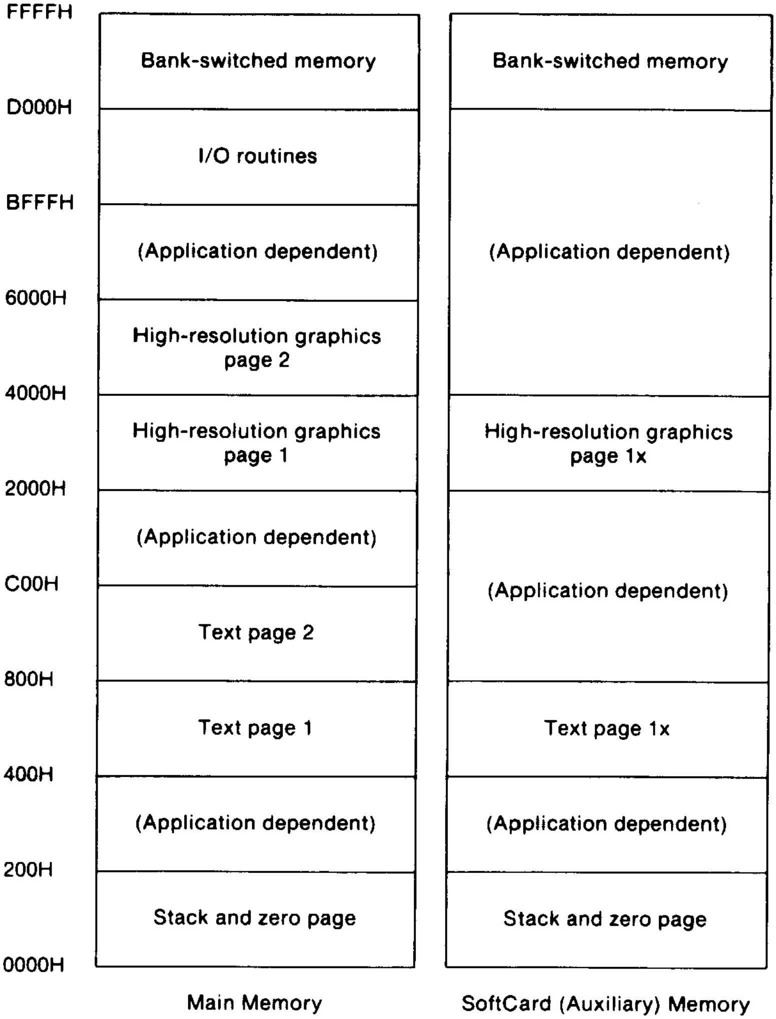


Figure 7.1. Memory Mapping Under Apple DOS

The 6502 addresses the SoftCard memory (or any other auxiliary memory) through the Apple address bus. To use the SoftCard memory for data storage, the 6502 switches its data bus so that it accesses SoftCard memory instead of the main memory. To expand the display for 80-column operation, the 6502 gets data from both memory areas as explained in "SoftCard Video Memory Operation," earlier in this chapter.

The components that control the bus switching are called the Memory Management Unit (MMU). The MMU works in conjunction with the 6502. It also contains firmware (soft switches) that can be set by programs to monitor the address bus and switch the address bus to the appropriate memory area.

The SoftCard memory is divided into three segments. The largest area is referred to as the 48K segment and is for data storage. The bank-switched segment is a 16K-byte section of memory that replaces the main memory in the upper address range (D000H to FFFFH). If you plan to use this part of the SoftCard memory, read the section entitled "Bank-Switched Memory" in Chapter 4 of the *Apple IIe Reference Manual.*

Note

> The switching of the ROM and D000H bank is independent of the the auxiliary RAM switching, so the bank switches have the same effect on the SoftCard RAM as they do on the main memory.

The lowest addresses of the SoftCard memory are reserved for the 6502 stack and zero page. (Zero page is used for system parameters.) Any time the addresses are switched to the bank-switched SoftCard memory, the addresses in zero page and page 1 (the 6502 stack) are also switched.

Doubling the Resolution of Graphic Displays

As described in "SoftCard Video Memory Operation" in this chapter, high-resolution graphics in the 80-column display mode will have the same resolution as in the 40-column display mode. A mixed mode text display in 80-column display mode can be used. *Mixed mode text display* is defined as a graphic screen display with the bottom four lines reserved for text.

To increase the horizontal resolution of graphic or mixed mode displays, the Annunciator 3 soft switch is turned on. The SoftCard can be modified to double normal graphics resolution with 80-column text.

Modify the SoftCard by changing the jumper on the SoftCard circuit board to the position shown in Figure 7.2. Use the installation procedure in the *Microsoft Premium SoftCard IIe Installation and Operation Manual* to remove and reinstall the circuit board.

The AN3 (Annunciator 3) switch is turned on by writing to location C05FH and turned off by writing to location C05EH. When you change the jumper 560 x 192 position, turn on the AN3 switch, and select the high-resolution graphics, mixed text display mode.

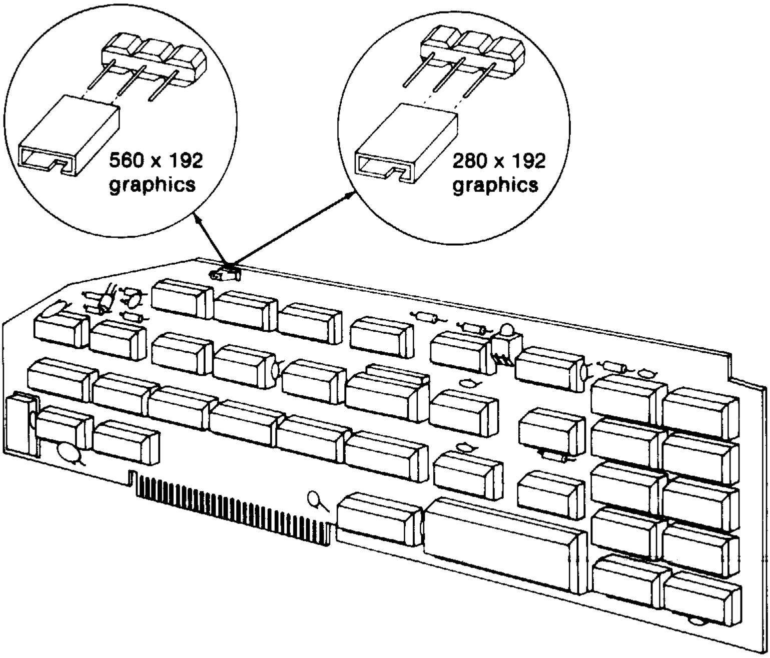


Figure 7.2. Changing the High-Resolution Graphics Jumper

Note

> The European version of the SoftCard circuit board differs slightly from the one illustrated in Figure 7.2, but the jumper configurations remain the same.

When the jumpers have been changed, the soft switches described in "Display Mode Switching" in this chapter must be turned on to select the 80-column display mode.

Warning

> Doubling the resolution of the 80-column display by the method just described is only applicable to Rev B and later Apple IIe computers. If you have a Rev A Apple IIe, changing the jumpers on the SoftCard circuit board will cause unpredictable results.
>
> The revision letter for the Apple IIe is located on the Apple IIe motherboard near the back panel.

When the SoftCard jumper is changed and the high-resolution graphics display mode is selected, the Apple IIe generates a display using high-resolution page 1 addresses in both main memory and SoftCard memory.

Memory mapping for the high-resolution graphics display doubles the columns as does 80-column text display mode, but uses high-resolution page 1 instead of text page 1. In 80-column text mode, a pair of data bytes is displayed as pairs of characters. In double high-resolution mode, however, the data byte pair is displayed as adjacent screen pixels, seven for each byte. As in 80-column text mode, there are twice as many pixels across the display screen. Therefore, the pixels are only half as wide.

Existing Apple II graphics programs do not support the doubled high-resolution graphics display. Until new programs become available, you will have to write your own plotting routines.

The Extended Display

To take advantage of the additional memory, you must set up your programs to operate in one part of memory while they switch the other part between main and auxiliary RAM. Your program can perform the memory switching by means of the soft switches described in the section "Display Mode Switching" later in this chapter, or by using the AUXMOVE and XFER subroutines described later in this section. Except for these subroutines, most existing Apple II system software (DOS version 3.3, Pascal version 1.1) doesn't support the additional memory.

Warning

Do not use SoftCard memory directly from a program running under interpreter languages such as Apple BASIC or Pascal. Interpreters map the main memory areas differently and will abort if switched to an auxiliary memory area. When restarted, programs and data can be lost.

Display Pages

The Apple IIe video display is generated from data stored in specific memory areas called display pages. The 40-column text and low-resolution display modes use text page 1 (400H—7FFH) and text page 2 (800H—BFFH) in main memory.

The 80-column text display uses a combination of text page 1 in main memory and page 1x in the SoftCard memory. Text page 1x uses the same address range as text page 1, but resides in the SoftCard memory rather than main memory. To store data in page 1x, you must use the appropriate soft switches to enable the 80-column display routines in ROM. The display modes and the corresponding display page addresses are listed in Table 7.6.

Table 7.6.

Display Page Addresses

Display Mode	Display Page	Address Range
40-column text	Page 1	400H—7FFH
Low-resolution graphics (40x48)	Page 2	800H—BFFH
80-column text	Page 1	400H—7FFH
Standard high-resolution graphics (280x192)	Page 1 Page 2	2000H—3FFFH 4000H—5FFFH
Double high-resolution graphics (560x192)	Page 1	2000H—3FFFH

Display Mode Switching

You can select the display mode that is appropriate for your application by accessing the Apple soft switches. The switches have three addresses for enabling, disabling, and checking the status of the switch.

Table 7.7 shows the addresses of the soft switches that control the display modes. The table gives the switch locations in three forms: hexadecimal, decimal, and negative decimal. You can use the hexadecimal (hex.) values in your assembly language programs and the decimal (dec.) values in PEEK and POKE commands in Applesoft BASIC programs. Negative decimal (neg. dec.) values are used in Integer BASIC programs.

Important

> Make sure that only the indicated operations are used for the soft switches. If you read a switch marked WRITE, erroneous data is returned.

Table 7.7.

Display Mode Soft Switches

Name and State	Purpose	Memory Addresses		
		Hex.	Dec.	Neg. Dec.
TEXT				
ON	Display text	C051	49233	-16303
OFF	Display graphics	C050	49232	-16304
READ	TEXT status	C01A	49178	-16358
MIXED				
ON	Text with graphics	C053	49235	-16301
OFF	Graphics only	C052	49234	-16302
READ	MIXED status	C01B	49179	-16357
PAGE2				
ON	Display page 2	C055	49237	-16299
OFF	Display page 1	C054	49236	-16300
READ	PAGE2 status	C01C	49180	-16356
HIRES				
ON	High-resolution graphics	C057	49297	-16297
OFF	Low-resolution graphics	C056	49298	-16298
READ	HIRES status	C01D	49181	-16355
80COL				
ON	Display 80-column	C00D	49165	-16371
OFF	Display 40-column	C00C	49164	-16372
READ	80COL status	C01F	49183	-16353
80STORE				
ON	Store in auxiliary page	C001	49153	-16383
OFF	Store in main page	C002	49152	-16384
READ	80STORE status	C018	49176	-16360

80-Column Display Memory

Data for 80-column display mode is stored in the same memory locations in text page 1 (main memory) and in text page 1x (SoftCard memory). Figure 7.3 shows the memory mapping for 80-column text display. Odd bytes are stored in SoftCard memory and even bytes are stored in main memory. When the 80-column display mode is active, the SoftCard reads data from both memory areas simultaneously, but displays data sequentially. The byte from SoftCard memory is displayed first, followed by the byte from the main memory.

You can store data directly into SoftCard memory by setting the 80STORE switch to ON. Setting the 80STORE switch to ON enables the PAGE2 soft switch to select between display stored in page 1 and page 1x.

Important

When you change the 80STORE and PAGE2 switches from the Apple monitor program, it will change the switch setting back to the original setting when you display the commands you type.

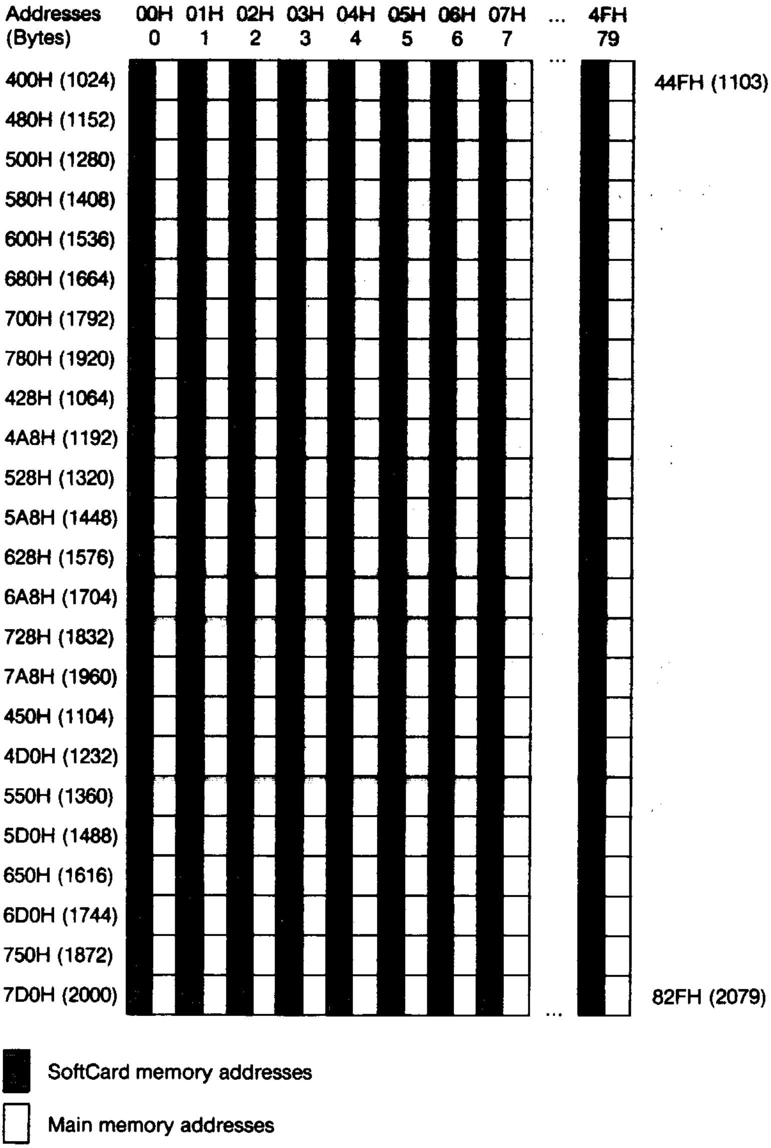


Figure 7.3. 80-Column Text Display Map

To use the 560x192 high-resolution graphics, store data directly on high-resolution page 1x in the SoftCard memory as described earlier in this section. Set both 80STORE and HIRES to ON and use PAGE2 to switch from page 1 in main memory to page 1x in the SoftCard memory.

Memory mapping for 560x192 high-resolution graphics is similar to the standard high-resolution graphics mapping which is described in Chapter 2 of the *Apple IIe Reference Manual*, with the addition of the column doubling produced by the 80-column display. Like the 80-column text mode, the double high-resolution graphics displays two bytes in the time normally required for one, but uses high-resolution graphics page 1 and page 1x instead of text page 1 and page 1x.

In 560x192 high-resolution graphics mode, each pair of data bytes is displayed as 14 adjacent pixels, seven for each byte. The high-order bit (color-select bit) of each byte is ignored. In this mode, the SoftCard memory byte is displayed first to allow data from the SoftCard memory to appear in every other column segment (seven columns each) starting with column segment 0—6 and ending with column segment 547—552. Data from the main memory appears in remaining column segments (7—13, 21—27, etc.) up to column segment 553—559.

As in 80-column text mode, there are twice as many pixels across the display screen making the pixels only half as wide. On a low-bandwidth monitor or a TV set, single pixels are dimmer than normal.

SoftCard Memory Switching

The SoftCard memory can be switched with the soft switches listed in Table 7.8. The sections following the table describe how to use the switches.

Table 7.8.

Memory Select Switches

		Memory Addresses		
Name and State	**Purpose**	**Hex.**	**Dec.**	**Neg. Dec.**
RAMRD				
ON	Read auxiliary 48K	C003	16155	-16381
OFF	Read main 48K	C002	16154	-16382
READ	RAMRD status	C013	16171	-16365
RAMWRT				
ON	Write auxiliary 48K	C005	16157	-16379
OFF	Write main 48K	C004	16156	-16380
READ	RAMWRT status	C014	16172	-16354
ALTZP				
ON	Access auxiliary memory areas*	C009	16373	-16373
OFF	Access main memory areas*	C008	16374	-16374
READ	ALTZP status	C016	16352	-16352
80STORE				
ON	Access page 1x	C001	16383	-16383
OFF	Use RAMRD, RAMWRT	C000	16384	-16384
READ	80STORE status	C018	16360	-16360
PAGE2				
ON	Access auxiliary memory	C055	16299	-16299
OFF	Access main memory	C054	16300	-16300
READ	PAGE2 status	C01C	16356	-16356
HIRES				
ON	Access high-resolution page 1x	C057	16297	-16297
OFF	Use RAMRD, RAMWRT	C056	16298	-16298
READ	HIRES status	C01D	16355	-16355

* Includes zero page, stack and bank-switched memory.

Note

> To retain compatibility with Apple II software, the soft switches in Table 7.8 list their memory locations with the Apple keyboard functions listed in Table 2.2 of the *Apple IIe Reference Manual.* The read and write operations for keyboard functions are different from the read and write functions listed in Table 7.8. For more information about the keyboard functions, see Chapter 2 of the *Apple IIe Reference Manual.*

Switching the 48K Bank

Switching the 48K-byte section of memory is performed by the RAMRD and the RAMWRT switches. RAMRD selects memory for reading and RAMWRT selects memory for writing. Setting the switches independently makes it possible for a program which has instructions that are being fetched from one 48K-byte memory bank to store data in the 48K-byte bank.

SoftCard memory locations corresponding to text page 1 and high-resolution graphics page 1 can be used as part of the 48K bank by using RAMRD and RAMWRT. These SoftCard memory areas can also be controlled separately by using the display-page switches 80STORE, PAGE2, and HIRES described in "80-Column Display Memory" in this chapter. Figures 7.4 and 7.5 show which areas of memory are active or switched for the different settings of RAMRD and RAMWRT.

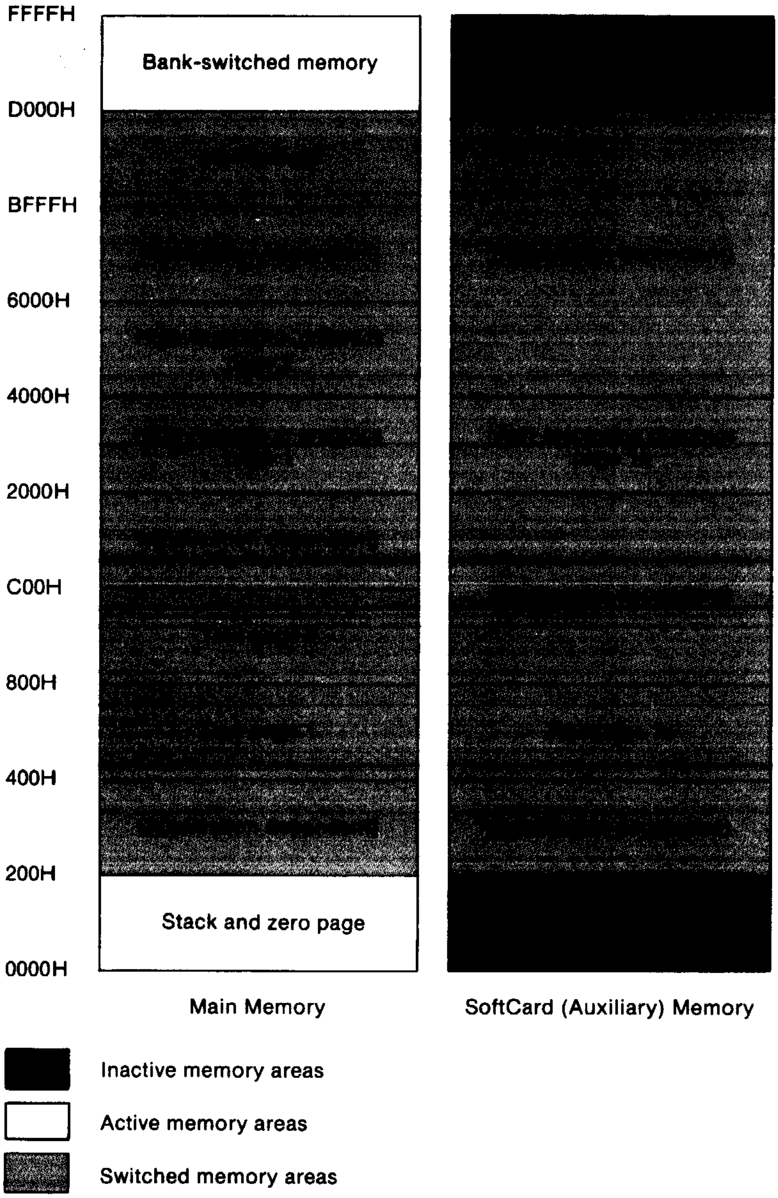


Figure 7.4. Memory Mapping With RAMRD and RAMWRT Switches On, 80STORE Off

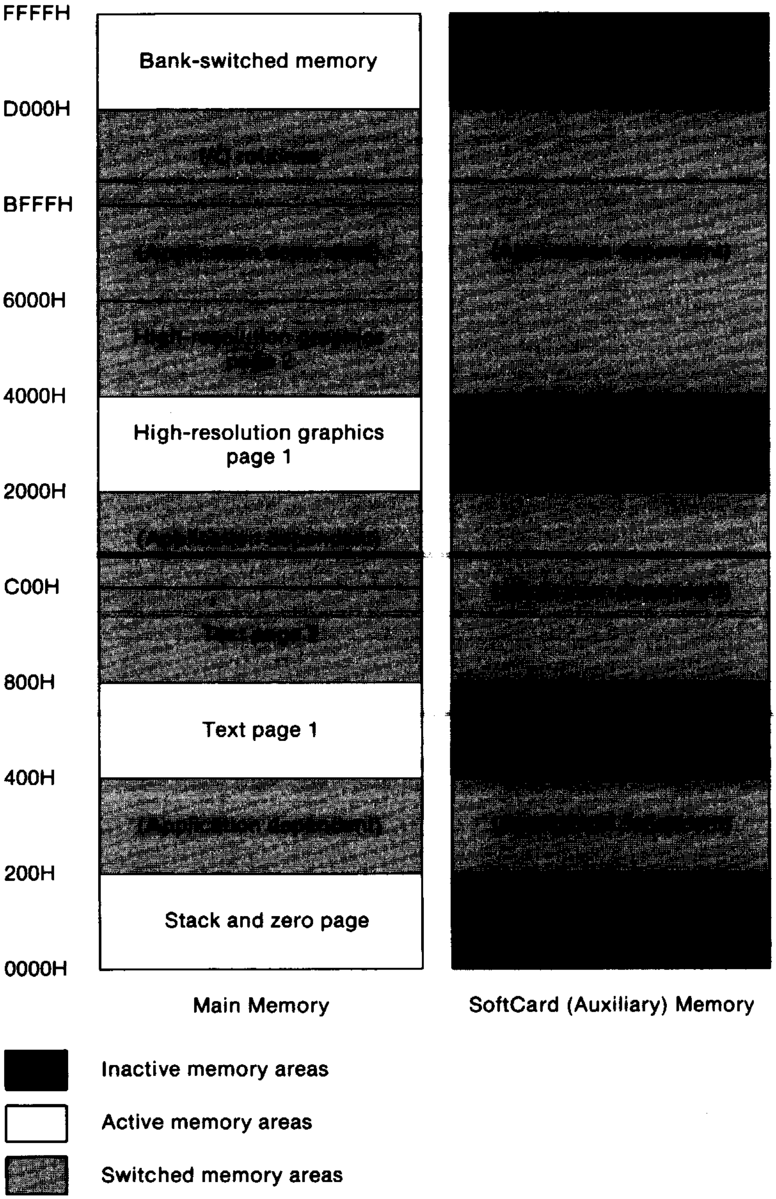


Figure 7.5. Memory Mapping With RAMRD, RAMWRT, 80STORE, and HIRES Switches On

As shown in Table 7.8, the 80STORE switch enables memory switching. When 80STORE is on, PAGE2 can select any memory bank. If the HIRES switch is off, the PAGE2 switch selects main or SoftCard memory in text display page 1 (0400H—07FFH). If HIRES is on, the PAGE2 switch selects main or auxiliary memory in text page 1 and high-resolution graphics page 1 (2000H—3FFFH).

If you use both the bank control switches and the display-page switches, the display-page switches take priority. That is, if 80STORE is off, RAMWRT and RAMRD control the entire memory space from 0200H to BFFFH. If 80STORE is on, RAMWRT and RAMRD have no effect on the display page and PAGE2 controls the text page. If both 80STORE and HIRES are on, then PAGE2 controls both text page 1 and high-resolution graphics page 1 regardless of the settings of RAMRD and RAMWRT. Figure 7.6 shows which segments of memory are used when PA6E2 is on.

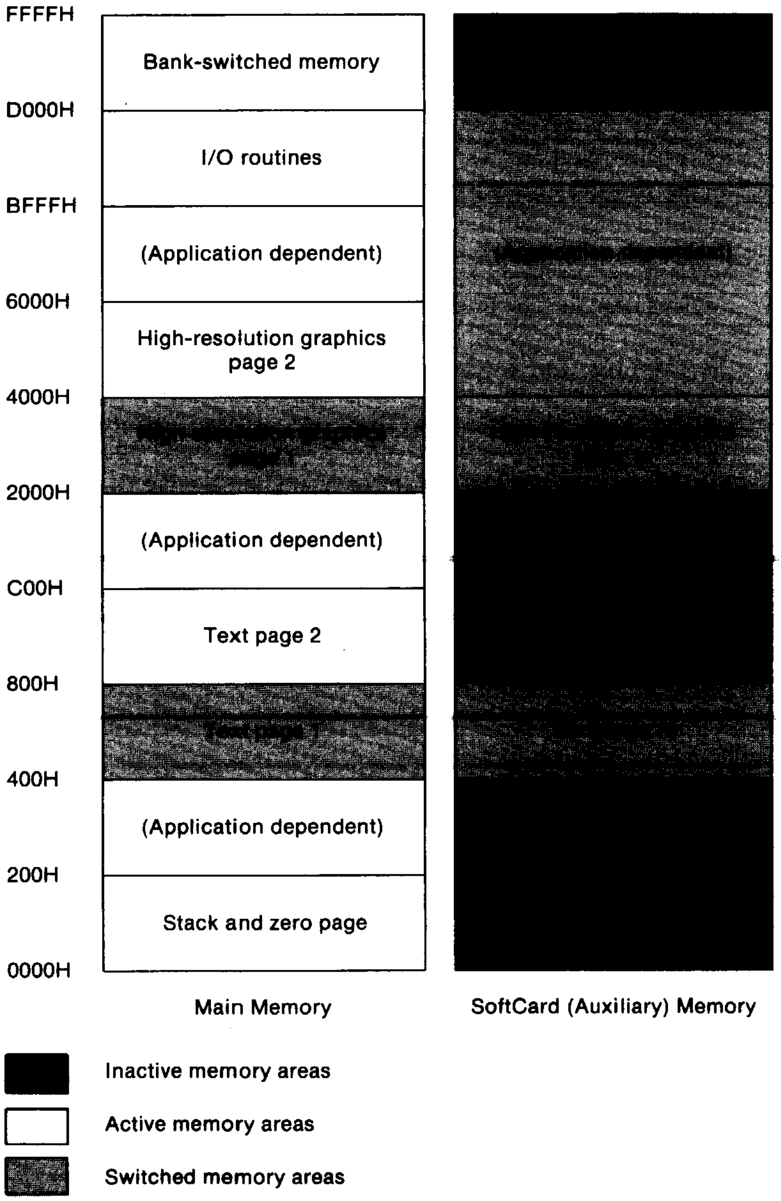


Figure 7.6. Memory Mapping With RAMRD, and RAMWRT Switches Off, and 80STORE, HIRES, and PAGE2 Switches On

To check the status of any of the switches, read the indicated status byte from Table 7.8. If the byte has the high bit set to a one, the switch is turned on. If the high bit is zero, then the switch is off.

Switching Other Segments of Memory

Soft switch ALTZP controls the access to the bank-switched memory, associated stack, and zero page areas. The following section, "Auxiliary Memory Subroutines," describes the firmware that can be called for switching between the main and SoftCard memory areas.

The status of the ALTZP switch is read from location C016H. If the sign bit (bit 7) is set to one, ALTZP is on. If zero, ALTZP is off. Figure 7.7 shows the affects of the ALTZP switch.

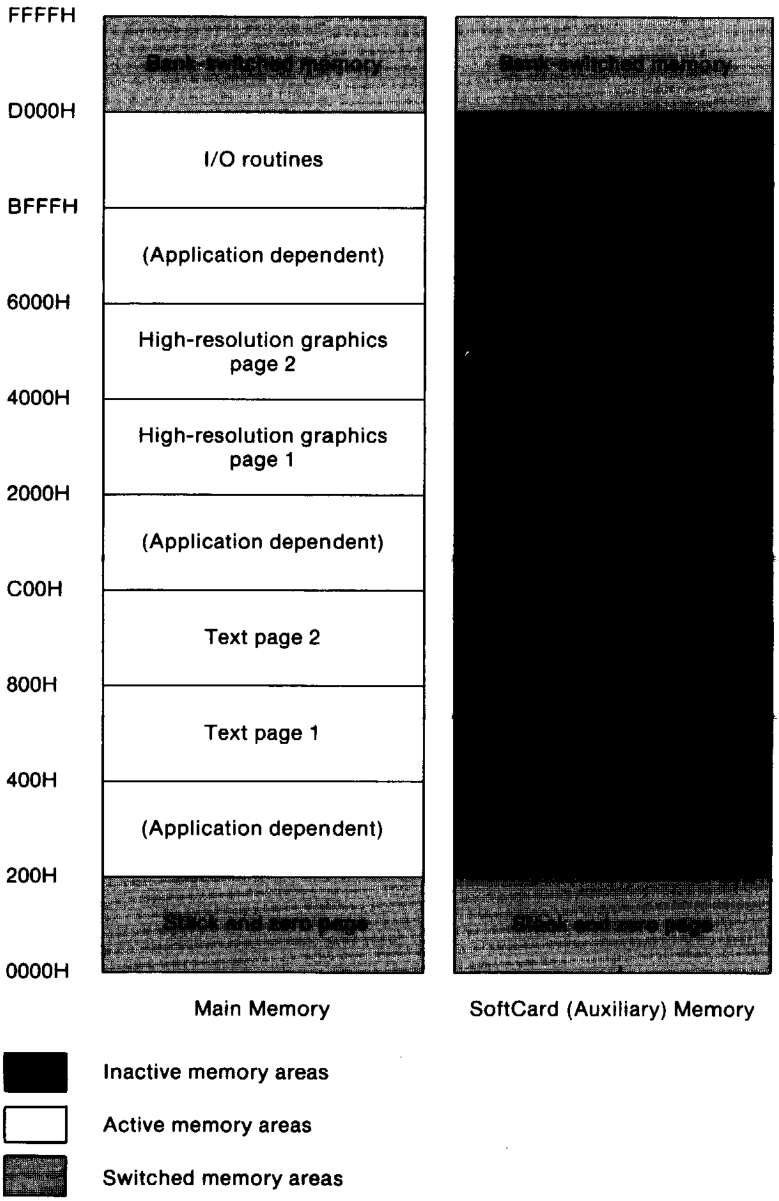


Figure 7.7. Memory Mapping With ALTZP Switch On

Auxiliary Memory Subroutines

You can use the SoftCard memory for assembly language programs or routines without the soft switches described by accessing the AUXMOVE and XFER subroutines stored in ROM. AUXMOVE and XFER make it easier to use the SoftCard memory but don't provide error protection.

The AUXMOVE subroutine copies data between memory areas and XFER transfers control between memory areas. Instructions for using AUXMOVE and XFER are given in Chapter 3 of the *Apple IIe Reference Manual.* AUXMOVE starts at address C311H and XFER starts at address C314H.

Copying Data Between Main Memory and the SoftCard

For 6502 assembly language programs and subroutines, AUXMOVE can be used to copy data between memory areas. Before AUXMOVE can be used, the data addresses must be stored in zero page by byte pairs and the carry bit of the 6502 Processor Status Word (PSW) must be set to one in order to select the direction of the data transfer.

The address byte pairs are called A1, A2, and A4. Their purpose and locations are given in Table 7.9.

Table 7.9.
AUXMOVE Parameters and Locations

Name	Address	Parameters
A1L	3CH	Low-order byte of the source start address.
A1H	3DH	High-order byte of the source start address.
A2L	3EH	Low-order byte of the source ending address.
A2H	3FH	High-order byte of the source ending address.
A4L	42H	Low-order byte of the destination address.
A4H	42H	High-order byte of the destination address.

Put the addresses of the first and last bytes of the block of data you want to copy into address byte pairs A1 and A2. The starting address of the destination memory area is put into address byte pair A4.

The AUXMOVE routine uses the carry bit to select the direction to copy the data. To copy from main memory to SoftCard memory, set the carry bit to one. Set the carry bit to zero to copy data from SoftCard memory to main memory.

When calling AUXMOVE, the subroutine copies the block of data as specified by the 6502 register A and the carry bit. The contents of the accumulator and the index registers (IX and IY) remain the same after the AUXMOVE routine has run.

Transferring Control to SoftCard Memory

Control is transferred between main memory and SoftCard memory through the XFER routine. To use XFER, set the XFER parameters to the following values: the carry and the overflow bits are set to one, and the program starting address is put in locations 3EDH—3EEH. (Setting carry and overflow to zero transfers control back to main memory.) When the parameters are set, use a a jump instruction to pass control to the XFER subroutine. XFER will preserve the contents of the accumulator and transfer address in the program stack. It then sets up the soft switches for the selected parameters and jumps to the new program.

Note

> You must save the current stack pointer in the current program memory space before using the XFER routine. XFER writes over the current contents when running. When XFER transfers control back to the main program, it will restore the original stack pointer contents.
