Table 6-5. I/O Memory Switches

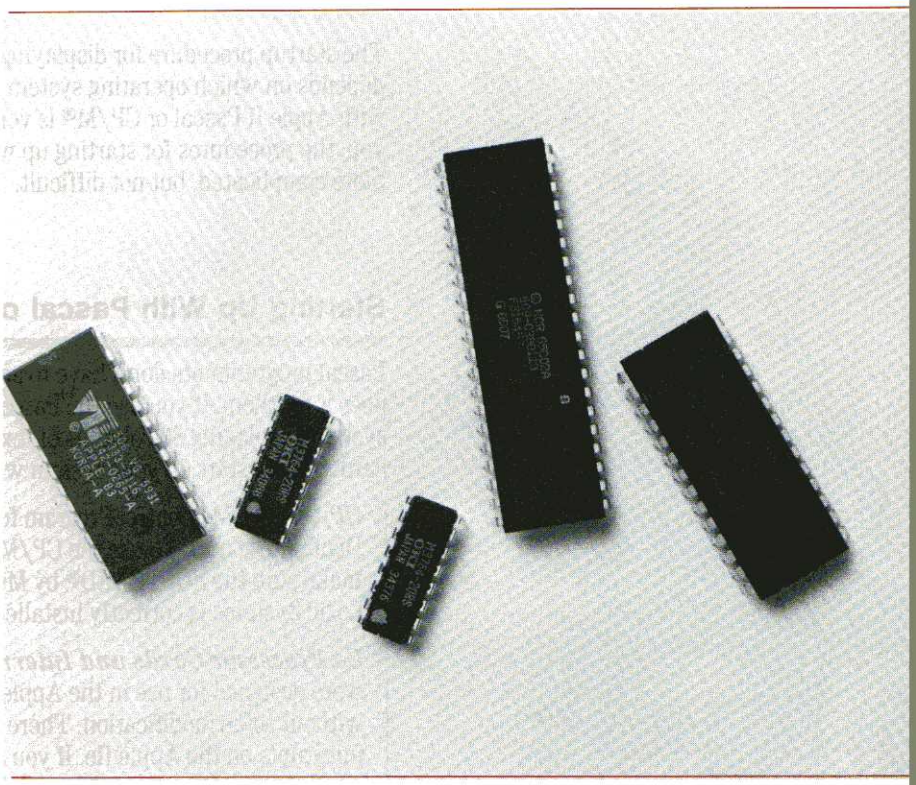
Name	Function	Location Hex	Location Decimal		Notes
SLOTC3ROM	Slot ROM at $C300	$C00B	49163	-16373	Write
	Internal ROM at $C300	$C00A	49162	-16374	Write
	Read SLOTC3ROM switch	$C017	49175	-16361	Read
SLOTCXROM	Slot ROM at $Cx00	$C006	49159	-16377	Write
	Internal ROM at $Cx00	$C007	49158	-16378	Write
	Read SLOTCXROM switch	$C015	49173	-16363	Read

Table 6-7. I/O Routine Offsets and Registers Under Pascal 1.1 Protocol

Addr.	Offset for	X Register	Y Register	A Register
$Cs0D	Initialization			
	On entry	$Cs	$s0	
	On exit	Error code	(unchanged)	(unchanged)
$Cs0E	Read			
	On entry	$Cs	$s0	
	On exit	Error code	(unchanged)	Character read
$Cs0F	Write			
	On entry	$Cs	$s0	Char. to write
	On exit	Error code	(unchanged)	(unchanged)
$Cs10	Status			
	On entry	$Cs	$s0	Request (0 or 1)
	On exit	Error code	(changed)	(unchanged)

Appendix G

Using an 80-Column Text Card



This appendix explains how to use 80-column text cards with high-level languages. Information about using 80-column text cards with assembly language programs through the Apple IIe Monitor firmware is found in Chapter 3 of this manual. The information in this appendix applies to the Apple IIe 80-Column Text Card and the Apple IIe Extended 80-Column Text Card.

If you are using Applesoft, ProDOS, or DOS you can choose to leave the 80-column text card inactive after installing it. You will want to do this when running software that does not take advantage of the 80-column display capability.

The startup procedure for displaying 80 columns of text on your Apple IIe depends on which operating system you plan to use. Starting up the system with Apple II Pascal or CP/M® is very easy; the operating system does it for you; the procedures for starting up with ProDOS or DOS 3.3 are slightly more complicated, but not difficult.

Starting Up With Pascal or CP/M

Pascal programmers don't have to activate the text card because Pascal does it for them. If you use the Pascal language or the CP/M operating system, displaying 80 columns of text is automatic once you've installed the card. Simply start up your system with any Pascal or CP/M startup disk.

CP/M: CP/M (Control Program for Microprocessors) is a trademark of Digital Research. To use the CP/M operating system with your Apple IIe, make sure the SOFTCARD® by Microsoft or the Z-Engine™ by Advanced Logic Systems is correctly installed before you start up the computer.

Co-Processor Cards and Interrupts: Some co-processor cards that were designed for use in the Apple II Plus may not work with an Apple IIe without some modification. There could be problems if you want to use interrupts on the Apple IIe. If you are having problems with a coprocessor card, check with the card's manufacturer for their recommendations.

Refer to the operating system reference manual for your version of Apple Pascal for more information.

When using Apple II Pascal 1.1, you'll probably want to run the program SETUP to make the ↑ and ↓ keys functional. SETUP is a self-documenting program on the Pascal disk APPLE3. Pascal versions 1.2 and later are already configured to use the ↑ and ↓ keys.

Starting Up With ProDOS or DOS 3.3

ProDOS and DOS 3.3 both look for a startup program on the startup (boot) disk as soon as the operating system has been loaded and begins executing. If the operating system finds the program, called STARTUP on a ProDOS disk and usually called HELLO on a DOS 3.3 disk, it will execute the program.

You can write a customized startup program that will set up the 80-column text card in any state you need. Just be sure it is on your startup disk and has the startup filename.

Here is a sample Applesoft startup program that works with both ProDOS and DOS 3.3:

```
10 HOME:D$=CHR$(4)
20 PRINT D$;"PR#3"
30 END
```

You can do whatever you wish with the program from line 20 on. Note that the screen will have switched to 80-column text mode after line 20.

> *By the Way:* If you arrange to have the card active automatically, you will still, of course, be able to switch into 40-column mode.

Using the GET Command

The presence of an active 80-column text card in the IIe requires that BASIC programmers use some alternate to Applesoft's INPUT command if their programs are to be userproof. Applesoft programmers should use either the GET command or the RDKEY or GETLN subroutines.

This is because the escape sequences used to switch back and forth between modes or to deactivate the card sometimes make it necessary to accept escape sequences in INPUT mode when using an 80-column card. Because the program accepts escape sequences typed from the keyboard, your program will not be userproof against accidental sequences typed in response to an INPUT command.

To get around this problem, you can use the GET command instead. The program does not read escape sequences typed from the keyboard in response to a GET command. This means that your users can err in their responses without endangering the display.

When to Switch Modes Versus When to Deactivate

When using BASIC, deactivate the text card whenever a previous (BASIC) program has left the card active (leaving a solid cursor on the screen) or whenever you want to send output to a peripheral device.

Switch back and forth between 40-column and 80-column displays for visual appeal. For full use of the control characters described later, your card must be active, although it can display in either 40-column or 80-column mode.

Original IIe

Tabbing in Applesoft: You must switch to a 40-column display to use Applesoft comma tabbing or the HTAB command.

Display Features With the Text Card

With an active 80-column card you can issue BASIC and PRODOS commands in lowercase characters. You can also issue commands in lowercase from the keyboard, that is, in immediate mode. This is particularly convenient because REM statements and data within quotes remain in lowercase as they were typed.

If you are using DOS 3.3, you must issue commands in uppercase whether or not your card is active.

INVERSE, FLASH, NORMAL, HOME

There are several commands you can give your computer from Applesoft BASIC to affect the appearance of text on the screen. All of these features are described in the *Applesoft BASIC Programmer's Reference Manual*.

- INVERSE tells the computer to display black characters on a white background instead of the normal display of white characters on a black background. This command is normally only available for uppercase characters, but with an active 80-column text card it is available for uppercase and lowercase characters.
- FLASH causes subsequently printed characters to blink quickly between inverse and normal characters. You can turn off the FLASH command by typing the NORMAL command. The FLASH command is normally available only with uppercase characters; it is not available at all while the card is active.

- NORMAL tells the computer to turn off the INVERSE or FLASH command and to display subsequently printed characters normally. It works the same way with the card active or inactive.
- HOME clears the screen and returns the cursor to the upper-left corner of the screen. Both the NORMAL HOME and INVERSE HOME commands are available while the card is active, but INVERSE HOME works a little differently when the card is active.

By the Way: The FLASH and INVERSE commands can be used to highlight important screen messages within a BASIC program.

Important!

If you are using the FLASH command (which means the 80-column text card is inactive) and then type PR#3 to activate the card, the screen turns white as the cursor goes to the HOME position. Whatever you type appears in black characters on the white screen. If you list or run an Applesoft BASIC program, some of the characters will appear as MouseText characters. To avoid this, remember to use either the NORMAL or INVERSE command before you exit the program.

Tabbing With the Original Apple IIe

You cannot use conventional 40-column tabbing in BASIC with the original model Apple IIe with an 80-column display. You do not have to turn off your card, but you must switch out of 80-column mode to use the HTAB command or to use comma tabbing.

When an original Apple IIe is displaying 80-column text, you should use the POKE 1403 command for horizontal tabbing in the right half of the screen instead of the HTAB command.

Comma Tabbing With the Original Apple IIe

In BASIC you can use commas in PRINT statements to instruct the computer to display all or part of your output in columns. This is known as comma tabbing. You can use this method of tabbing as long as the screen is displaying 40 columns (that is with the card inactive or after issuing an ESCAPE-4 command to switch to 40-column mode). You cannot use this method of tabbing with an 80-column display. If you try to do so, characters will be placed in memory outside the screen area and may change programs or data in memory.

HTAB and POKE 1403

The VTAB (vertical tab) and HTAB (horizontal tab) statements can be used to place the cursor at a specific location on the screen before printing characters. The largest value you can use with the VTAB statement is 24; the largest for HTAB is 255. The VTAB command works just the same in an 80-column display as it does in a 40-column display.

On the original Apple IIe, the HTAB command causes the cursor to wrap around to the next line after it reaches the 40th column, so you cannot use this command to position the cursor in the last 40 columns while the screen is displaying 80 columns.

POKE 1403 is specifically designed to solve this problem. Using the POKE 1403 command allows you to tab horizontally across the extra 40 columns provided by the 80-column text card.

If you want to tab past column 40 while the card is active and the screen is displaying 80 columns, use the following, where n is a number from 0 to 79:

```
POKE 1403, n
```

When you use the HTAB command, HTAB 1 places the cursor at the leftmost position on the screen. When you use the POKE 1403 command, POKE 1403,0 places the cursor at the leftmost position on the screen.

Using Control Characters With the Card

Using BASIC with an active 80-column text card increases the number of functions you can perform with control characters. Originally control-character commands were so named because they were given from the keyboard by pressing the CONTROL key in conjunction with another key. You can perform the same functions from your programs by using an equivalent control-character code. Commands based on these two-key combinations are called control-character commands even when they must be issued from a program.

Control Characters and Their Functions

Table G-1 lists the control-character commands supported by BASIC with an 80-column card. The table includes the corresponding command code, its function and whether a given command can be executed from the keyboard as well as from a program.

Table G-1. Control Characters With 80-Column Firmware On

Control Character	ASCII Code	Apple IIe Name	Action Taken by BASICOUT
Control-G	BEL	bell	Produces a 1000 Hz tone for 0.1 second.
Control-H	BS	backspace	Moves cursor position one space to the left; from left edge of window, moves to right end of line above.
Control-J	LF	line feed	Moves cursor position down to next line in window; scrolls if needed.
Control-K †	VT	clear EOS	Clears from cursor position to the end of the screen.
Control-L †	FF	home and clear	Moves cursor position to upper-left corner of window and clears window.
Control-M	CR	return	Moves cursor position to left end of next line in window; scrolls if needed.
Control-N †	SO	normal	Sets display format normal.
Control-O †	SI	inverse	Sets display format inverse.
Control-Q †	DC1	40-column	Sets display to 40-column.
Control-R †	DC2	80-column	Sets display to 80-column.
Control-S *	DC3	stop-list	Stops listing characters on the display until another key is pressed.
Control-U †	NAK	quit	Deactivates 80-column video firmware.
Control-V †	SYN	scroll	Scrolls the display down one line, leaving the cursor in the current position.
Control-W †	ETB	scroll-up	Scrolls the display up one line, leaving the cursor in the current position.
Control-X	CAN	disable MouseText	Disable MouseText character display; use inverse uppercase.

Table G-1—Continued. Control Characters With 80-Column Firmware On

Control Character	ASCII Code	Apple IIe Name	Action Taken by BASICOUT
Control-Y †	EM	home	Moves cursor position to upper-left corner of window (but doesn't clear).
Control-Z †	SUB	clear line	Clears the line the cursor position is on.
Control-[	ESC	enable MouseText	Map inverse uppercase characters to MouseText characters.
Control-\ †	FS	forward space	Moves cursor position one space to the right; from right edge of window, moves it to left end of line below.
Control-] †	GS	clear EOL	Clears from the current cursor position to the end of the line (that is, to the right edge of the window).
Control-_	US	up	Moves cursor up a line, no scroll.

* Only works from the keyboard.

† Doesn't work from the keyboard.

How to Use Control-Character Codes in Programs

To issue a control-character command from a program, use the ASCII decimal code that corresponds to the control-character. (See Table G-1.)

The following example shows how to use ASCII decimal codes in an Applesoft BASIC program. Type

```
HOME [?]
NEW
10 PRINT CHR$(15): PRINT "MAKE HAY"
20 PRINT CHR$(14): PRINT "WHILE THE SUN SHINES"
RUN
```

(CHR$ is the Applesoft BASIC command that signifies that a control-character function is to be performed.)

You will get

```
]NEW

]10 PRINT CHR$(15): PRINT "MAKE HAY"

]20 PRINT CHR$(14): PRINT "WHILE THE SUN SHINES"

]RUN

MAKE HAY

WHILE THE SUN SHINES

]
```

See Chapter 3 in this manual for a description of control-character functions.

The ASCII decimal codes for inverse video (Control-O) and normal video (Control-N) are 15 and 14. When the PRINT statements in the example are executed, the display switches to inverse and prints MAKE HAY, then switches back to a normal display and prints WHILE THE SUN SHINES.

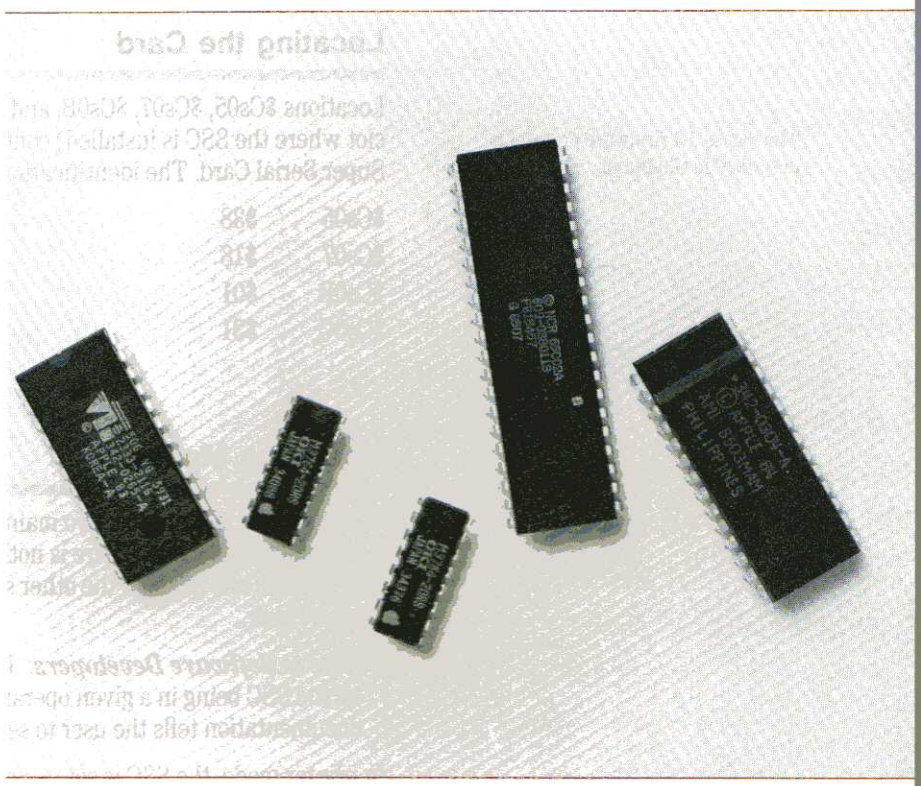
A Word of Caution to Pascal Programmers

Avoid writing Control-U or Control-Q to the console from a Pascal program. Either one puts the system into a state that will cause Pascal to eventually crash.

You can't send control characters from the keyboard to the 80-column firmware when using Pascal. The only exceptions to this rule are Control-M (CR) and Control-G (BEL).

Appendix H

Programming With the Super Serial Card



For more information about the installation and operation of the SSC, see the Super Serial Card manual.

This appendix briefly describes how to use the Apple II Super Serial Card (SSC) from programs, how to find the SSC through software, and the commands supported by the SSC.

The SCC is one of the most common serial interface cards used with the Apple IIe, and the Apple IIc's serial ports operate very much like the Super Serial Card. This similarity should make it easier for you to write programs for both the Apple IIe and Apple IIc.

Locating the Card

The Pascal 1.1 firmware protocol is described in Chapter 6.

Locations $Cs05, $Cs07, $Cs0B, and $Cs0C (where *s* is the number of the slot where the SSC is installed) contain the identification bytes for the Super Serial Card. The identification byte's values are

$Cs05	$38
$Cs07	$18
$Cs0B	$01
$Cs0C	$31

Operating Modes

The Super Serial Card has two main operating modes: printer mode and communications mode. There is nothing you can do from software to change from one mode to the other since they are set by the position of the jumper block.

> *Note to Software Developers:* If you are writing software that depends on the SSC being in a given operating mode, make sure that your documentation tells the user to set up the SSC in the proper way.

In printer mode, the SSC is set to send data to a printer, local terminal, or other serial device. In communications mode, the SSC is set to operate with a modem. From communications mode, the SSC can enter a special mode called terminal mode. In terminal mode the Apple IIe acts like an unintelligent terminal.

Operating Commands

For each of the operating modes, you can control many aspects of data transmission such as baud rate, data format, line feed generation, and so forth.

Your program can change these aspects by sending control codes as commands to the card. All commands are preceded by a command character and followed by a carriage return character ($0D).

The command character is usually Control-I in printer mode and Control-A in communications mode and terminal mode. In the command examples in the following sections, Control-I is used unless the command being described is available only in communications mode or terminal mode. A carriage return character is represented by its ASCII symbol, CR.

There are three types of command formats:

- A number, represented by n, followed by an uppercase letter with no space between the characters (for example, 4D to set data format 4).
- An uppercase letter by itself (for example, R to reset the SSC).
- An uppercase letter followed by a space and then either E to enable or D to disable a feature (for example, L D to disable automatic insertion of line feed characters).

The allowable range of n is given in each command description that follows.

The choice of enable or disable is indicated with E/D. The underscore character (_) before the E/D in commands that allow enable/disable is to remind you that a space is required there.

The SSC checks only numbers and the first letters of commands and options. (All such letters must be uppercase.) Further letters, which you can add to assist your memory, have no effect on the SSC. For example, XOFF Enable is the same as X E. The SSC ignores invalid commands.

Important! The spaces in command examples are there for clarity; generally you will not use spaces in a command string. Where a space is required in a command string, an underscore (_) character will appear in the text as a reminder.

The Command Character

The normal command character is Control-I (ASCII $09) in printer mode, or Control-A (ASCII $01) in communications mode. If you want to change the command character from Control-I to Control-something else, send Control-I Control-something else. For example, to change the command character to Control-W, send Control-I Control-W. To change back, send Control-W Control-I. No return character is required after either of these commands.

You can send the command character itself through the SSC by sending it twice in a row: Control-I Control-I; no return character is required after this command. This special command allows you to transmit the command character without affecting the operation of the SSC, and without having to change to another command character and then back again later.

Here is how to generate this character in BASIC and Pascal:

Applesoft BASIC:
```
PRINT CHR$(9);"command"
```
Pascal:
```
WRITELN (CHR(9),'command');
```

Baud Rate, nB

You can use this command to override the physical settings of switches SW1-1 through SW1-4 on the SSC. For example, to change the baud rate to 135, send Control-I 4B CR to the SSC.

Table H-1. Baud Rate Selections

n	SSC Baud Rate	n	SSC Baud Rate
0	use SW1-1 to SW1-4	8	1200
1	50	9	1800
2	75	10	2400
3	109.92 (110)	11	3600
4	134.58 (135)	12	4800
5	150	13	7200
6	300	14	9600
7	600	15	19200

Data Format, nD

You can override the settings of switch SW2-1 with this command. The table below shows how many data and stop bits correspond to each value of n. For example, Control-I 2D CR makes the SSC transmit each character in the form one start bit (always transmitted), six data bits, and one stop bit.

Table H-2. Data Format Selections

n	Data Bits	Stop Bits
0	8	1
1	7	1
2	6	1
3	5	1
4	8	2 *
5	7	2
6	6	2
7	5	2 †

* 1 with Parity options 4 through 7

† 1½ with Parity options 0 through 3

Parity, nP

You can use this command to set the parity that you want to use for data transmission and reception. There are five parity options available, described in Table H-3.

Table H-3. Parity Selections

n	Parity to Use
0, 2, 4 or 6	None (default value)
1	Odd parity (odd total number of ones)
3	Even parity (even total number of ones)
5	MARK parity (parity bit always 1)
7	SPACE parity (parity bit always 0)

For example, the command string Control-I 1P CR makes the SSC transmit and check for odd parity. Odd parity means that the high bit of every character is 0 if there is an odd number of 1 bits in that character, or 1 if there is an even number of 1 bits in the character, making the total number of 1 bits in the character always odd. This is an easy (but not foolproof) way to check data for transmission errors. Parity errors are recorded in a status byte.

Set Time Delay, nC, nL, and nF

Some printers can't keep up with the Apple IIe when they are doing certain operations. You may need to change default settings on the SSC to give a printer the time it needs.

The nC command overrides the setting of switch SW2-2 on the SSC. That switch provides two choices: either no delay or a 250 millisecond delay after the SSC sends a carriage return character.

The nL command allows time after a line feed character for a printer platen to turn so the paper is vertically positioned to receive the next line.

The nF command allows time after a form feed character for the printer platen to move the paper form to the top of the next page (typically a longer time than a line feed).

Table H-4. Time Delay Selections

n	Time Delay
0	none
1	32 milliseconds
2	250 milliseconds (1/4 second)
3	2 seconds

Consult the user manual for a given printer to find out how much time it takes to move its print head and platen so you can determine an appropriate set of values for these three delays. The idea is to have at least enough time for the printer parts to move the required distance, but not so much time that overall printing speed is slowed down drastically. Many printers require no delays because they have a buffer built in to keep accepting characters even while they are doing form feeds and so on.

A typical setup for a *very* slow printer would be Control-I 2C CR, Control-I 2L CR, Control-I 3F CR; that is, the SSC waits 250 milliseconds after transmitting carriage returns, 250 milliseconds after transmitting line feeds, and 2 seconds after transmitting form feed characters.

Echo Characters to the Screen, E_E/D

For the Apple IIe, as for most computers, displaying (echoing) a character on the video screen during communications is a separate step from receiving it from the keyboard, though we tend to think if these as one step, as on a typewriter. For example, if you send Control-A E_D CR, the SSC does not forward incoming characters to the Apple IIe screen. This can be used to hide someone's password entered at a terminal, or to avoid double display of characters.

This command is used in communications mode only.

Automatic Carriage Return, C

Sending Control-I C CR to the SSC causes it to generate a carriage return character (ASCII CR) whenever the column count exceeds the current printer line width limit. This command is used in printer mode only.

Important!

Once this option is on, only clearing the high-order bit at location $578+s (where *s* is the slot the SSC is in) can turn this option back off. This option is normally off.

Automatic Line Feed, L_E/D

You can use this command to have the SSC automatically generate and transmit a line feed character after each carriage return character. This overides the setting of switch SW2-5. For example, send Control-I L_E CR to your printer to print listings or double-spaced manuscripts for editing.

Mask Line Feed In, M_E/D

If you send Control-I M_E CR to the SSC, it will ignore any incoming line feed character that immediately follows a carriage return character.

Reset Card, R

Sending Control-I R CR to the SSC has the same effect as sending a PR#0 and an IN#0 to a BASIC program and then resetting the SSC. This command cancels all previous commands to the SSC and puts the physical switch settings back into force.

Specify Screen Slot, S

In communications mode, you can specify the slot number of the device where you want text or listings displayed with this command. (Normally this is slot 0, the Apple IIe video screen.) This allows chaining of the SSC to another card slot, such as an 80-column text card. For the firmware in the SSC to pass on information to the firmware in the other card, the other card must have an output entry point within its $Cs00 space; this is the case for all currently available 80-column cards for the Apple IIe.

For example, let's say you have the SSC in slot 2 with a remote terminal connected to it, and an 80-column card in slot 3. Send Control-A 3S CR to cause the data from the remote terminal to be chained through the card in slot 3, so that it is displayed on the Apple IIe in 80-column format. (Not available in Pascal.)

Translate Lowercase Characters, nT

The Apple IIe Monitor translates all incoming lowercase characters into uppercase ones before sending them to the video screen or to a BASIC program. The nT command has four options, which are shown in Table H-5.

Table H-5. Lowercase Character Display Options

n	Action
0	Change all lowercase characters to uppercase ones before passing them to a BASIC program or to the video screen. This is the way the Apple IIe monitor handles lowercase.
1	Pass along all lowercase characters unchanged. The appearance of the lowercase characters on the Apple II screen is undefined (garbage).
2	Display lowercase characters as uppercase inverse characters (that is, as black characters on a white background).
3	Pass lowercase characters to programs unchanged, but display lowercase as uppercase, and uppercase as inverse uppercase (that is, as black characters on a white background).

Suppress Control Characters, Z

If you issue the Z command described here, all further commands are ignored; this is useful if the data you are transmitting, such as graphics data, contains bit patterns that the SSC can mistake for control characters.

Sending Control-I Z CR to the SSC prevents it from recognizing any further control characters (and hence commands) whether coming from the keyboard or contained in a stream of characters sent to the SSC.

Important! The only way to reinstate command recognition after the Z command is to either reinitialize the SSC, or clear the high-order bit at location $5F8+s (where *s* is the number of the slot in which the SSC is installed).

Find Keyboard, F_E/D

You can use this command to make the SSC ignore keyboard input.

For example, you can include Control-I F_D CR in a program, followed by a routine that retrieves data through the SSC, followed by Control-I F_E CR to turn the keyboard back on.

XOFF Recognition, X_E/D

Sending Control-I X_E CR to the SSC causes it to look for any XOFF ($13) character coming from a device attached to the SSC, and to respond to it by halting transmission of characters until the SSC receives an XON ($11) from the device, signalling the SCC to continue transmission. In printer mode, this function is normally turned off.

Caution | In printer mode, full duplex communication may not work with XOFF recognition turned on, so be careful.

Tab in BASIC, T E/D

In printer mode only, if you send Control-I T_E CR to the SSC, the BASIC horizontal position counter is left equal to the column count. All tabs work, including back-tabs. Tabs beyond column 40 require a POKE to location 36. Commas only work as far as column 40, and BASIC programs will be listed in 40-column format.

Note that this use of tabbing is specific to the SSC—it doesn't go through the 80-column firmware.

Terminal Mode

From communications mode, the SSC can enter terminal mode and make the Apple IIe act like an unintelligent terminal. This is useful for connecting the Apple IIe to a computer timesharing service, or for conversing with another Apple II.

Entering Terminal Mode, T

Send Control-A T CR to enter terminal mode. This causes the Apple IIe to function as a full-duplex unintelligent terminal. You can use this command together with the Echo command to simulate the half-duplex terminal mode of the old Apple II Communications Card.

| *By the Way:* If you enter terminal mode and don't see what you type echoed on the Apple video screen, probably the modem link has not yet been established, or you need to use the Echo Enable command (Control-A E_E CR).

Transmitting a Break, B

Sending Control-A B CR causes the SSC to transmit a 233-millisecond break signal, recognized by most time-sharing systems as a signoff.

Special Characters, S_E/D

If you send Control-A S_D CR, the SSC will treat the ESCAPE key like any other key.

Quitting Terminal Mode, Q

Send Control-A Q CR to the SSC to exit from terminal mode.

SSC Error Codes

The SSC uses I/O scratchpad address $678+s (*s* is the number of the slot that the SSC is in) to record status after a read operation. The firmware calls this byte STSBYTE. Table H-6 lists the bit definitions of this byte.

Table H-6. STSBYTE Bit Definitions

Bit	"1" Means	"0" Means
0	Parity Error occurred.	No Parity Error occurred.
1	Framing Error occurred.	No Framing Error occurred.
2	Overrun occurred.	No Overrun occurred.
3	Carrier lost.	Carrier present.
5	Error occurred.	No error occurred.

The terms **Parity**, **Framing Error**, and **Overrun** are defined in the glossary.

Bits 0, 1, and 2 are the same as the corresponding three bits of the ACIA Status Register of the SSC. Bit 3 indicates whether or not the Data Carrier Detect (DCD) signal went false at any time during the receive operation.

Bit 5 is set if any of the other bits are set, as an overall error indicator. If bit 5 is the only bit set, an unrecognized command was detected. If all bits are 0, no error occurred.

These error codes begin with the number 32 to avoid conflicting with previously defined and documented system error codes.

In BASIC, you can check this status byte via a PEEK $678+s (*s* is the SSC slot), and reset it with a POKE command at the same location.

In Pascal, the IORESULT function returns the error code value.

> ***By the Way:*** Any character—including the carriage return at the end of a WRITELN statement—will cause posting of a new value in IORESULT.

Table H-7 shows the possible combinations of error bits corresponding to these decimal error codes.

Table H-7. Error Codes and Bits

Error Code*	Carrier Lost	Overrun	Framing Error	Parity Error
0		no error		
32		illegal command		
33	no	no	no	yes
34	no	no	yes	no
35	no	no	yes	yes
36	no	yes	no	no
37	no	yes	no	yes
38	no	yes	yes	no
39	no	yes	yes	yes
40	yes	no	no	no
41	yes	no	no	yes
42	yes	no	yes	no
43	yes	no	yes	yes
44	yes	yes	no	no
45	yes	yes	no	yes
46	yes	yes	yes	no
47	yes	yes	yes	yes

* Result of PEEK $678+s in BASIC or IORESULT in Pascal.

The ACIA

The Asynchronous Communication Interface Adapter (ACIA) chip is the heart of the Super Serial Card. It takes the 1.8432 MHz signal generated by the crystal oscillator on the SSC and divides it down to one of the fifteen baud rates that it supports. The ACIA also handles all incoming and outgoing signals of the RS232-C serial protocol that the ACIA supports.

The ACIA registers control hardware handshaking and select the baud rate, data format, and parity. The ACIA also performs parallel to serial and serial to parallel data conversion, and buffers data transfers.

SSC Firmware Memory Use

Table H-8 is an overall map of the locations that the SSC uses, both in the Apple IIe and in the SSC's own firmware address space.

Table H-8. Memory Use Map

Address	Name of Area	Contents
$0000-$00FF	Page zero	Monitor pointers, I/O hooks, and temporary storage.
$04xx-$07xx (selected locations)	Peripheral slot Scratchpad RAM	Locations (8 per slot) in Apple IIe pages $04 through $07. SSC uses all 8 of them.
$C0(8+s)0-$C0(8+s)F	Peripheral card I/O space	Locations (16 per slot) for general I/O; SSC uses 6 bytes.
$Cs00-$CsFF	Peripheral card ROM space	One 256-byte page reserved for card in slot s; first page of SSC firmware.
$C800-$CFFF	Expansion ROM	Eight 256-byte pages reserved for 2K ROM or PROM; SSC maps its firmware onto $C800-$CEFF.

Zero-Page Locations

The SSC uses the zero-page locations described in Table H-9.

Table H-9. Zero-Page Locations Used by the SSC

Address	Name	Description
$24 *	CH	Monitor pointer to current position of cursor on screen
$26	SLOT16	Usually (slot x 16); that is, $s0
$27	CHARACTER	Input or output character
$28 *	BASL	Monitor pointer to current screen line
$2A	ZPTMP1	Temporary storage (various uses)
$2B	ZPTMP2	Temporary storage (various uses)
$35	ZPTEMP	Temporary storage (various uses)
$36 *	CSWL	BASIC output hook (not for Pascal)
$37 *	CSWH	High byte of CSW
$38 *	KSWL	BASIC input hook (not for Pascal)
$39 *	KSWH	High byte of KSW
$4E *	RNDL	Random number location, updated when looking for a keypress (not used when initialized by Pascal)

* Not used when Pascal initializes SSC.

Peripheral Card I/O Space

There are 16 bytes of I/O space allocated to each slot in the Apple IIe. Each set begins at address $C080 + (slot x 16); for example, if the SSC is in slot 3, its group of bytes extends from $C0B0 to $C0BF. Table H-10 interprets the 6 bytes the SSC uses.

Table H-10. Address Register Bits Interpretation

Address	Register	Bits	Interpretation
$C081+s0	DIPSW1 (SW1-x)	0	SW1-6 is OFF when 1, ON when 0
		1	SW1-5 is OFF when 1, ON when 0
		4-7	Same as above for SW1-4 through SW1-1
$C082+s0	DIPSW2 (SW2-x)	0	Clear To Send (CTS) is true when 0
		1-3	Same as above for SW2-5 through SW2-3
		5, 7	Same as above for SW2-2 and SW2-1
$C088+s0	TDREG	0-7	ACIA transmit register (write)
	RDREG	0-7	ACIA receive register (read)
$C089+s0	STATUS		ACIA status/reset register
		0	Parity error detected when 1
		1	Framing error detected when 1
		2	Overrun detected when 1
		3	ACIA receive register full when 1
		4	ACIA transmit register empty when 1
		5	Data Carrier Detect (DCD) true when 0
		6	Data Set Ready (DSR) true when 0
		7	Interrupt (IRQ) has occurred when 1
$C08A+s0	COMMAND		ACIA command register (read/write)
		0	Data Terminal Ready (DTR): enable (1) or disable (0) receiver and all interrupts
		1	When 1, allow STATUS bit 3 to cause interrupt
		2-3	Control transmit interrupt, Request To Send (RTS) level, and transmitter
		4	When 0, normal mode for receiver; when 1, echo mode (but bits 2 and 3 must be 0)
		5-7	Control parity
$C08B+s0	CONTROL		ACIA control register (read/write)
		0-3	Baud rate: $00 = 16 times external clock; See Table H-1.
		4	When 1, use baud rate generator; when 0, use external clock (not supported)
		5-6	Number of data bits: 8 (bit 5 and 6 = 0) 7 (5 = 1, 6 = 0), 6 (5 = 0, 6 = 1) or 5 (bit 5 and 6 both = 1)
		7	Number of stop bits: 1 if bit 7 = 0; if bit 7 = 1, then 1-1/2 (with 5 data bits, no parity), 1 (8 data plus parity), or 2

Scratchpad RAM Locations

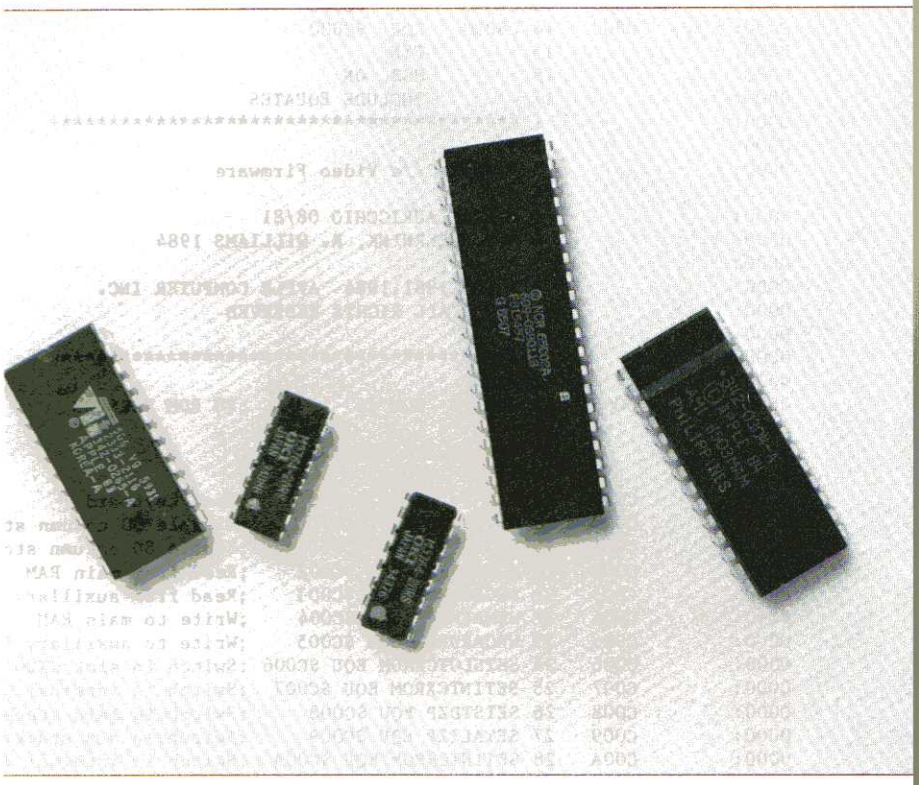
The SSC uses the scratchpad RAM locations listed in Table H-11.

Table H-11. Scratchpad RAM Locations Used by the SSC

Address	Field name	Bit	Interpretation
$0478+s	DELAYFLG	0-1	Form feed delay selection
		2-3	Line feed delay selection
		4-5	Carriage return delay selection
		6-7	Translate option
$04F8+s	PARAMETE	0-7	Accumulator for firmware's command processor
$0578+s	STATEFLG	0-2	Command mode when not 0
		3-5	Slot to chain to (communications mode)
		6	Set to 1 after lowercase input character
		7	Terminal mode when 1 (communications mode)
		7	Enable CR generation when 1 (printer mode)
$05F8+s	CMDBYTE	0-6	Printer mode default is Control-I; communications mode default is Control-A
		7	Set to 1 to Zap control commands
$0678+s	STSBYTE		Status and IORESULT byte
$06F8+s	CHNBYTE	0-2	Current screen slot (communication mode); when slot = 0, chaining is enabled.
		3-7	$Cs00 space entry point (communications mode)
	PWDBYTE	0-7	Current printer width; for listing compensation, auto-CR (printer mode)
$0778+s	BUFBYTE	0-6	One-byte input buffer (communications mode); used in conjunction with XOFF recognition
		7	Set to 1 when buffer full (communications mode)
	COLBYTE	0-7	Current-column counter for tabbing and so forth (printer mode)
$07F8+s	MISCFLG	0	Generate line feed after CR when 1
		1	Printer mode when 0; comminications mode when 1
		2	Keyboard input enabled when 1
		3	Control-S (XOFF), Control-R, and Control-T input checking when 1
		4	Pascal operating system when 1; BASIC when 0
		5	Discard line feed input when 1
		6	Enable lowercase and special character generation when 1 (communications mode)
		6	Tabbing option on when 1 (printer mode)
		7	Echo output to Apple IIe screen when 1

Appendix I

Monitor ROM Listing



```
00:         0000     1 TEST    EQU  0          :REAL VERSION

0000:                2          LST  ON         ;DO LISTING AND SYMBOL TABLES
0000:                3          MSB  ON         ;SET THEM HIBITS
0000:          0001  4 IROTEST EQU  1
0000:          0000  5          DO   TEST
 S                   6 F8ORG   EQU  $1800
 S                   7 C1ORG   EQU  $2100
 S                   8 C3ORG   EQU  $2300
 S                   9 C8ORG   EQU  $2800
0000:               10          ELSE
0000:          F800 11 F8ORG   EQU  $F800
0000:          C100 12 C1ORG   EQU  $C100
0000:          C300 13 C3ORG   EQU  $C300
0000:          C800 14 C8ORG   EQU  $C800
0000:               15          FIN
0000:               16          MSB  ON
0000:               17          INCLUDE EQUATES
0000:                1 ****************************************
0000:                2 *
0000:                3 * Apple //e Video Firmware
0000:                4 *
0000:                5 * RICK AURICCHIO 08/81
0000:                6 * E. BEERNINK, R. WILLIAMS 1984
0000:                7 *
0000:                8 * (C) 1981,1984  APPLE COMPUTER INC.
0000:                9 *     ALL RIGHTS RESERVED
0000:               10 *
0000:               11 ****************************************
0000:               12 *
0000:          0006 13 GOODF8  EQU  6          ;F8 ROM VERSION
0000:               14 *
0000:               15 * HARDWARE EQUATES:
0000:               16 *
0000:          C000 17 KBD     EQU  $C000      ;Read keyboard
0000:          C000 18 CLR80COL EQU $C000      ;Disable 80 column store
0000:          C001 19 SET80COL EQU $C001      ;Enable 80 column store
0000:          C002 20 RDMAINRAM EQU $C002     ;Read from main RAM
0000:          C003 21 RDCARDRAM EQU $C003     ;Read from auxiliary RAM
0000:          C004 22 WRMAINRAM EQU $C004     ;Write to main RAM
0000:          C005 23 WRCARDRAM EQU $C005     ;Write to auxiliary RAM
0000:          C006 24 SETSLOTCXROM EQU $C006 ;Switch in slot CX00 ROM
0000:          C007 25 SETINTCXROM EQU $C007  ;Switch in internal CX00 ROM
0000:          C008 26 SETSTDZP EQU $C008      ;Switch in main stack/zp/lang.card
0000:          C009 27 SETALTZP EQU $C009      ;Switch in aux stack/zp/lang.card
0000:          C00A 28 SETINTC3ROM EQU $C00A  ;Switch in internal $C3 ROM
0000:          C00B 29 SETSLOTC3ROM EQU $C00B ;Switch in slot $C3 space
0000:          C00C 30 CLR80VID EQU $C00C      ;Disable 80 column video
0000:          C00D 31 SET80VID EQU $C00D      ;Enable 80 column video
0000:          C00E 32 CLRALTCHAR EQU $C00E   ;Normal Apple II char set
0000:          C00F 33 SETALTCHAR EQU $C00F   ;Norm/inv LC, no flash
0000:          C010 34 KBDSTRB EQU  $C010      ;Clear keyboard strobe
0000:          C011 35 RDLCBNK2 EQU $C011      ;>127 if LC BANK2 in use
0000:          C012 36 RDLCRAM EQU  $C012      ;>127 if LC is read enabled
```

```
0000:          C013   37 RDRAMRD EQU  $C013      ;>127 if main RAM read enabled
0000:          C014   38 RDRAMWRT EQU $C014      ;>127 if main RAM write enabled
0000:          C015   39 RDCXROM EQU  $C015      ;>127 if ROM CX space enabled
0000:          C016   40 RDALTZP EQU  $C016      ;>127 if alt. zp & lc enabled
0000:          C017   41 RDC3ROM EQU  $C017      ;>127 if slot C3 space enabled
0000:          C018   42 RD80COL EQU  $C018      ;>127 if 80 column store enabled
0000:          C019   43 RDVBLBAR EQU $C019      ;>127 if not vertical blanking
0000:          C01A   44 RDTEXT  EQU  $C01A      ;>127 if text mode
0000:          C01C   45 RDPAGE2 EQU  $C01C      ;>127 if page 2
0000:          C01E   46 ALTCHARSET EQU $C01E    ;>127 if alt char set switched in
0000:          C01F   47 RD80VID EQU  $C01F      ;>127 if 80 column video enabled
0000:          C030   48 SPKR    EQU  $C030      ;toggle speaker
0000:          C054   49 TXTPAGE1 EQU $C054      ;switches in text page 1
0000:          C055   50 TXTPAGE2 EQU $C055      ;switches in text page 2
0000:          C05D   51 CLRAN2  EQU  $C05D      ;annunciator 2
0000:          C05F   52 CLRAN3  EQU  $C05F      ;annunciator 3
0000:          C061   53 BUTN0   EQU  $C061      ;open-apple key
0000:          C062   54 BUTN1   EQU  $C062      ;closed-apple key
0000:          C081   55 ROMIN   EQU  $C081      ;swap in D000-FFFF ROM
0000:          C083   56 LCBANK2 EQU  $C083      ;swap in LC bank 2
0000:          C08B   57 LCBANK1 EQU  $C08B      ;swap in LC bank 1
0000:                 58 *
0000:                 59 * MONITOR EQUATES:
0000:                 60 *
0000:          FBB3   61 F8VERSION EQU F80RG+$3B3 ;F8 ROM ID
0000:          FD1B   62 KEYIN   EQU  F80RG+$51B ;normal input
0000:          FDF0   63 COUT1   EQU  F80RG+$5F0 ;normal output
0000:          FF69   64 MONZ    EQU  F80RG+$769 ;monitor entry point
0000:                 65 *
0000:                 66 * ZEROPAGE EQUATES:
0000:                 67 *
0000:          0000   68 LOC0    EQU  0          ;used for doing PR#
0000:          0001   69 LOC1    EQU  1          ;used for doing PR#
0000:                 70         DSECT
0020:          0020   71         ORG  $20
0020:          0001   72 WNDLFT  DS   1          ;scrolling window left
0021:          0001   73 WNDWDTH DS   1          ;scrolling window width
0022:          0001   74 WNDTOP  DS   1          ;scrolling window top
0023:          0001   75 WNDBTM  DS   1          ;scrolling window bottom+1
0024:          0001   76 CH      DS   1          ;cursor horizontal
0025:          0001   77 CV      DS   1          ;cursor vertical
0026:          0002   78         DS   2          ;GBASL,GBASH
0028:          0002   79 BASL    DS   2          ;points to current line of text
002A:          0029   80 BASH    EQU  BASL+1
002A:          0002   81 BAS2L   DS   2          ;pointer used for scroll
002C:          002B   82 BAS2H   EQU  BAS2L+1
002C:                 83 *
002F:          002F   84         ORG  $2F
002F:          0001   85 LENGTH  DS   1          ;length for mnemonics
0030:          0002   86         DS   2
0032:          0001   87 INVFLG  DS   1          ;>127=normal, <127=inverse
0033:          0001   88 PROMPT  DS   1          ;used by monitor upshift
0034:          0001   89 YSAV    DS   1          ;input buffer index for mini
0035:          0001   90 SAVY1   DS   1          ;for restoring Y
```

```
0036:       0002   91 CSWL     DS   2          ;hook for output routine
0038:       0037   92 CSWH     EQU  CSWL+1
0038:       0002   93 KSWL     DS   2          ;hook for input routine
003A:       0039   94 KSWH     EQU  KSWL+1
003C:       003C   95          ORG  $3C
003C:       0002   96 A1L      DS   2          ;Monitor temps for MOVE
003E:       003D   97 A1H      EQU  A1L+1
003E:       0002   98 A2L      DS   2
0040:       003F   99 A2H      EQU  A2L+1
0040:       0002  100          DS   2          ;A3 NOT USED
0042:       0002  101 A4L      DS   2
0044:       0043  102 A4H      EQU  A4L+1
0044:       0001  103 MACSTAT  DS   1          ;machine state on breaks
004E:       004E  104          ORG  $4E
004E:       0002  105 RNDL     DS   2          ;random number seed
0050:       004F  106 RNDH     EQU  RNDL+1
0000:             107          DEND
0000:             108 *
0000:       0200  109 BUF      EQU  $200       ;input buffer
0000:             110 * Permanent data in screenholes
0000:             111 *
0000:             112 * Note: these screenholes are only used by
0000:             113 * the 80 column firmware if an 80 column card
0000:             114 * is detected or if the user explicitly activates
0000:             115 * the firmware.  If the 80 column card is not
0000:             116 * present, only MODE is trashed on RESET.
0000:             117 *
0000:             118 * The success of these routines rely on the
0000:             119 * fact that if 80 column store is on (as it
0000:             120 * normally is during 80 column operation), that
0000:             121 * text page 1 is switched in.  Do not call the
0000:             122 * video firmware if video page 2 is switched in!!
0000:             123 *
0000:       07F8  124 MSLOT    EQU  $7F8       ;=$Cn ;n=slot using $C800
0000:             125 *
0000:       047B  126 OLDCH    EQU  $478+3     ;LAST CH used by video firmware
0000:       04FB  127 MODE     EQU  $4F8+3     ;video firmware operating mode
0000:       057B  128 OURCH    EQU  $578+3     ;80 column CH
0000:       05FB  129 OURCV    EQU  $5F8+3     ;80 column CV
0000:       067B  130 CHAR     EQU  $678+3     ;character to be printed/read
0000:       06FB  131 XCOORD   EQU  $6F8+3     ;GOTOXY X-coord (pascal only)
0000:       077B  132 TEMP1    EQU  $778+3     ;temp
0000:       077B  133 OLDBASL  EQU  $778+3     ;last BASL (pascal only)
0000:       07FB  134 TEMP2    EQU  $7F8+3     ;temp
0000:       07FB  135 OLDBASH  EQU  $7F8+3     ;last BASH (pascal only)
0000:             136 *
0000:             137 * BASIC MODE BITS
0000:             138 *
0000:             139 * 0....... - BASIC active
0000:             140 * 1....... - Pascal active
0000:             141 * .0...... -
0000:             142 * .1...... -
0000:             143 * ..0..... - Print control characters
0000:             144 * ..1..... - Don't print ctrl chars.
```

```
0000:                 145 * ...0.... -
0000:                 146 * ...1.... -
0000:                 147 * ....0... - Print control characters
0000:                 148 * ....1... - Don't print next ctrl char
0000:                 149 * .....0.. -
0000:                 150 * .....1.. -
0000:                 151 * ......0. -
0000:                 152 * ......1. -
0000:                 153 * .......0 - Mouse text inactive
0000:                 154 * .......1 - Mouse text active
0000:                 155 *
0000:          0040   156 M.6      EQU  $40
0000:          0020   157 M.CTL2   EQU  $20        ;Don't print controls
0000:          0010   158 M.4      EQU  $10
0000:          0008   159 M.CTL    EQU  $08        ;Temp ctrl disable
0000:          0004   160 M.2      EQU  $04
0000:          0002   161 M.1      EQU  $02
0000:          0001   162 M.MOUSE  EQU  $01
0000:                 163 *
0000:                 164 * Pascal Mode Bits
0000:                 165 *
0000:                 166 * 0.......  - BASIC active
0000:                 167 * 1.......  - Pascal active
0000:                 168 * .0......  -
0000:                 169 * .1......  -
0000:                 170 * ..0.....  -
0000:                 171 * ..1.....  -
0000:                 172 * ...0....  - Cursor always on
0000:                 173 * ...1....  - Cursor always off
0000:                 174 * ....0...  - GOTOXY n/a
0000:                 175 * ....1...  - GOTOXY in progress
0000:                 176 * .....0..  - Normal Video
0000:                 177 * .....1..  - Inverse Video
0000:                 178 * ......0.  - PASCAL 1.1 F/W ACTIVE
0000:                 179 * ......1.  - PASCAL 1.0 INTERFACE
0000:                 180 * .......0  - Mouse text inactive
0000:                 181 * .......1  - Mouse text active
0000:                 182 *
0000:          0080   183 M.PASCAL EQU $80        ;Pascal active
0000:          0010   184 M.CURSOR EQU $10        ;Don't print cursor
0000:          0008   185 M.GOXY   EQU  $08       ;GOTOXY IN PROGRESS
0000:          0004   186 M.VMODE  EQU  $04       ;PASCAL VIDEO MODE
0000:          0002   187 M.PAS1.0 EQU $02        ;PASCAL 1.0 MODE
0000:                 188 *
0000:                 189 * F8 ROM entries
0000:                 190 *
0000:          FA47   191 NEWBREAK EQU F8ORG+$247
0000:          FC74   192 IRQUSER EQU  F8ORG+$474
0000:          FC7A   193 IRQDONE2 EQU F8ORG+$47A
0000:          F8B7   194 TSTROM  EQU  F8ORG+$B7
0000:                  18          INCLUDE BFUNC
----- NEXT OBJECT FILE NAME IS REFLIST.0
C100:          C100    1          ORG  C1ORG
C100:          C100    2 BFUNCPG EQU  *
```

```
C100:          FEC5     3 FUNCEXIT EQU F8ORG+$6C5 ;RETURN ADDRESS
C100:          FCF0     4 MINI     EQU  F8ORG+$4F0
C100:                   5 *
C100:                   6 * BASIC FUNCTION HOOK:
C100:                   7 *
C100:                   8 * $C100 is called by the patched $F8 ROM.
C100:                   9 * It provides an extension to $F8 routines
C100:                  10 * that do not work in 80 columns.
C100:                  11 *
C100:                  12 * Before jumping here, the $F8 rom disabled
C100:                  13 * slot I/O and enabled ROM I/O.  This makes
C100:                  14 * the entire space from $C100 - $CFFF with the
C100:                  15 * exception of the $C300 page available.
C100:                  16 *
C100:                  17 * On exit slot I/O is restored if necessary.
C100:                  18 *
C100:                  19 * INPUT: Y=FUNCTION AS FOLLOWS:
C100:                  20 *
C100:                  21 *           1 = KEYIN
C100:                  22 *           2 = Fix escape char
C100:                  23 *           3 = BASCALC
C100:                  24 *           4 = VTAB or VTABZ
C100:                  25 *           5 = HOME
C100:                  26 *           6 = SCROLL
C100:                  27 *           7 = CLREOL
C100:                  28 *           8 = CLREOLZ
C100:                  29 *           9 = RESET
C100:                  30 *           A = CLREOP
C100:                  31 *           B = RDKEY
C100:                  32 *           C = SETWND
C100:                  33 *           D = Mini Assembler
C100:                  34 *           E = set 40 columns on PR#0/IN#0
C100:                  35 *           F = Fix pick for monitor
C100:                  36 *
C100:                  37 *  Stack has PHP for status of internal $CN00 ROM
C100:                  38 *
C100:                  39 * Note: If 80 Vid is on and the MODE byte is valid,
C100:                  40 * this call will be dispatched to an 80 column routine
C100:                  41 * by B.FUNC0.  Otherwise it will be dispatched to a
C100:                  42 * 40 column routine by B.OLDFUNC.  In all cases return
C100:                  43 * to the Autostart ROM is done through F.RETURN.
C100:                  44 *
C100:4C 13 C2          45 B.FUNC  JMP  DISPATCH  ;figure out what to do
C103:                  46 *
C103:A4 24             47 F.CLREOP LDY CH        ; ESC F IS CLR TO END OF PAGE
C105:A5 25             48          LDA  CV
C107:48                49 CLEOP1   PHA
C108:20 03 CE          50          JSR  VTABZ
C10B:20 F4 C1          51          JSR  X.CLREOLZ
C10E:A0 00             52          LDY  #$00
C110:68                53          PLA
C111:69 00             54          ADC  #$00       ;(carry set)
C113:C5 23             55          CMP  WNDBTM
C115:90 F0   C107      56          BCC  CLEOP1
```

```
C117:B0 34    C14D    57          BCS  GVTZ       ;=>always to VTABZ
C119:                 58 *
C119:A5 22            59 F.HOME   LDA  WNDTOP
C11B:85 25            60          STA  CV
C11D:A0 00            61          LDY  #$00
C11F:84 24            62          STY  CH
C121:F0 E4    C107    63          BEQ  CLEOP1    ;(ALWAYS TAKEN)
C123:                 64 *
C123:A5 22            65 F.SCROLL LDA WNDTOP
C125:48               66          PHA
C126:20 03 CE         67          JSR  VTABZ
C129:A5 28            68 SCRL1    LDA  BASL
C12B:85 2A            69          STA  BAS2L
C12D:A5 29            70          LDA  BASH
C12F:85 2B            71          STA  BAS2H
C131:A4 21            72          LDY  WNDWDTH
C133:88               73          DEY
C134:68               74          PLA
C135:69 01            75          ADC  #$01
C137:C5 23            76          CMP  WNDBTM
C139:B0 0D    C148    77          BCS  SCRL3
C13B:48               78          PHA
C13C:20 03 CE         79          JSR  VTABZ
C13F:B1 28            80 SCRL2    LDA  (BASL),Y
C141:91 2A            81          STA  (BAS2L),Y
C143:88               82          DEY
C144:10 F9    C13F    83          BPL  SCRL2
C146:30 E1    C129    84          BMI  SCRL1
C148:A0 00            85 SCRL3    LDY  #$00
C14A:20 F4 C1         86          JSR  X.CLREOLZ
C14D:A5 25            87 GVTZ     LDA  CV
C14F:4C 03 CE         88 GVTZ2    JMP  VTABZ     ;set vertical base
C152:                 89 *
C152:         C152    90 F.SETWND EQU  *
C152:A9 28            91          LDA  #40
C154:85 21            92          STA  WNDWDTH
C156:A9 18            93          LDA  #24
C158:85 23            94          STA  WNDBTM
C15A:A9 17            95          LDA  #23
C15C:85 25            96          STA  CV
C15E:D0 EF    C14F    97          BNE  GVTZ2     ;=>go do vtab, exit
C160:                 98 *
C160:                 99 * Load Y from BAS2L and clear line
C160:                100 *
C160:A4 2A           101 F.CLREOLZ LDY BAS2L     ;set up by $F8 ROM
C162:4C F4 C1        102           JMP  X.CLREOLZ ;and clear line
C165:                103 *
C165:                104 * 80 column routines begin here
C165:                105 *
C165:4C EB CB        106 B.SCROLL JMP SCROLLUP  ;DO IT FOR CALLER
C168:                107 *
C168:                108 * Clear to end of line using Y = OURCH
C168:                109 *
C168:4C 9A CC        110 B.CLREOL JMP X.GS       ;clear to end of line
```

```
C16B:                111 *
C16B:                112 * Clear to end of line using Y = BAS2L
C16B:                113 * which was set up by the $F8 ROM
C16B:                114 *
C16B:A4 2A           115 B.CLREOLZ LDY BAS2L     ;get Y
C16D:4C 9D CC        116           JMP  X.GSEOLZ  ;clear to end of line
C170:                117 *
C170:4C 74 CC        118 B.CLREOP JMP X.VT       ;CLEAR TO EOS
C173:4C A0 C2        119 B.SETWND JMP B.SETWNDX
C176:4C B0 C2        120 B.RESET JMP  B.RESETX  ;MUST BE IN BFUNC PAGE
C179:4C F2 C2        121 B.RDKEY JMP  B.RDKEYX
C17C:                122 *
C17C:20 90 CC        123 B.HOME  JSR  X.FF       ;HOME & CLEAR
C17F:AD 7B 05        124           LDA  OURCH
C182:85 24           125           STA  CH       ;COPY CH/CV FOR CALLER
C184:8D 7B 04        126           STA  OLDCH    ;REMEMBER WHAT WE SET
C187:4C FE CD        127           JMP  VTAB     ;calc base & return
C18A:                128 *
C18A:                129 * Complete PR# or IN# call.  Quit video firmware
C18A:                130 * if PR#0 and it was active (B.QUIT).  Complete call
C18A:                131 * if inactive (F.QUIT).
C18A:                132 *
C18A:        C18A    133 B.QUIT  EQU  *
C18A:B4 00           134           LDY  LOC0,X    ;was it PR#0/IN#0?
C18C:F0 0F   C19D    135           BEQ  NOT0      ;=>no, not slot 0
C18E:C0 1B           136           CPY  #KEYIN    ;was it IN#0?
C190:F0 0E   C1A0    137           BEQ  IS0       ;=>yes, update high byte
C192:20 80 CD        138           JSR  QUIT      ;quit the firmware
C195:B4 00           139 F.QUIT  LDY  LOC0,X    ;get low byte into Y
C197:F0 04   C19D    140           BEQ  NOT0      ;not slot 0, firmware inactive
C199:A9 FD           141 F8HOOK  LDA  #<KEYIN   ;set high byte to $FD
C19B:95 01           142           STA  LOC1,X
C19D:B5 01           143 NOT0    LDA  LOC1,X    ;restore accumulator
C19F:60              144           RTS
C1A0:                145 *
C1A0:A5 37           146 IS0     LDA  CSWH      ;is $C3 in output hook?
C1A2:C9 C3           147           CMP  #<BASICIN
C1A4:D0 F3   C199    148           BNE  F8HOOK    ;=>no, set to $FD0C
C1A6:4C 32 C8        149           JMP  C3IN      ;else set to $C305, exit A=$C3
C1A9:                150 *
C1A9:A4 24           151 F.RDKEY LDY  CH        ;else do normal 40 cursor
C1AB:B1 28           152           LDA  (BASL),Y  ;grab the character
C1AD:48              153           PHA
C1AE:29 3F           154           AND  #$3F      ;set screen to flash
C1B0:09 40           155           ORA  #$40
C1B2:91 28           156           STA  (BASL),Y  ;and display it
C1B4:68              157 F.NOCUR PLA
C1B5:60              158           RTS            ;return (A=char)
C1B6:                159 *
C1B6:A8              160 F.BASCALC TAY           ;restore Y
C1B7:A5 28           161           LDA  BASL      ;restore A
C1B9:20 BA CA        162           JSR  BASCALC   ;calculate base address
C1BC:90 4C   C20A    163           BCC  F.RETURN  ;BASCALC always returns BCC!
C1BE:                164 *
```

```
C1BE:          C1BE  165 B.ESCFIX EQU *
C1BE:20 14 CE        166         JSR  UPSHFT    ;upshift lowercase
C1C1:A0 03           167 B.ESCFIX1 LDY #4-1     ;SCAN FOR A MATCH
C1C3:          C1C3  168 B.ESCFIX2 EQU *
C1C3:D9 EE C2        169         CMP  ESCIN,Y   ;IS IT?
C1C6:D0 03   C1CB    170         BNE  B.ESCFIX3 ;=>NAW
C1C8:B9 A4 C9        171         LDA  ESCOUT,Y  ;YES, TRANSLATE IT
C1CB:          C1CB  172 B.ESCFIX3 EQU *
C1CB:88              173         DEY
C1CC:10 F5   C1C3    174         BPL  B.ESCFIX2
C1CE:30 3A   C20A    175         BMI  F.RETURN  ;RETURN:CHAR IN AC
C1D0:                176 *
C1D0:20 70 C8        177 F.BOUT  JSR  BOUT      ;print the character
C1D3:4C 0A C2        178         JMP  F.RETURN  ;AND RETURN
C1D6:                179 *
C1D6:                180 * Do displaced mnemonic stuff
C1D6:                181 *
C1D6:8A              182 MNNDX   TXA            ;get old acc
C1D7:29 03           183         AND  #$03      ;make it a length
C1D9:85 2F           184         STA  LENGTH
C1DB:A5 2A           185         LDA  BAS2L     ;get old Y into A
C1DD:29 8F           186         AND  #$8F
C1DF:4C 71 CA        187         JMP  DOMN      ;and go to open spaces
C1E2:                188 *
C1E2:20 F0 FC        189 GOMINI  JSR  MINI      ;do mini-assembler
C1E5:8A              190         TXA            ;X=0. Set mode to 0, and counter
C1E6:85 34           191         STA  YSAV      ;so not CR on new line
C1E8:60              192         RTS
C1E9:                193 *
C1E9:                194 * Pick an 80 column character for the monitor
C1E9:                195 *
C1E9:AC 7B 05        196 FIXPICK LDY  OURCH     ;get 80 column cursor
C1EC:20 44 CE        197         JSR  PICK      ;pick the character
C1EF:09 80           198         ORA  #$80      ;always pick as normal
C1F1:60              199         RTS            ;and return
C1F2:                200 *
C1F2:                201 * Load CH into Y and clear line
C1F2:                202 *
C1F2:          C1F2  203 F.CLREOL EQU *
C1F2:A4 24           204 X.CLREOL LDY CH        ;get horizontal position
C1F4:A9 A0           205 X.CLREOLZ LDA #$A0     ;store a normal blank
C1F6:2C 1E C0        206         BIT  ALTCHARSET ;unless alternate char set
C1F9:10 06   C201    207         BPL  X.CLREOL2
C1FB:24 32           208         BIT  INVFLG    ;and inverse
C1FD:30 02   C201    209         BMI  X.CLREOL2
C1FF:A9 20           210         LDA  #$20      ;use inverse blank
C201:4C A8 CC        211 X.CLREOL2 JMP CLR40    ;clear to end of line
C204:                212 *
C204:                213 * Call VTAB or VTABZ for 40 or 80 columns.  Acc (CV)
C204:                214 * is saved in BASL.
C204:                215 *
C204:A8              216 F.VTABZ TAY            ;restore Y
C205:A5 28           217         LDA  BASL      ;and A
C207:20 03 CE        218         JSR  VTABZ     ;do VTABZ
```

```
C20A:                 219 *
C20A:                 220 * EXIT. EITHER EXIT WITH OR WITHOUT
C20A:                 221 *  ENABLING I/O SPACE.
C20A:                 222 *
C20A:          C20A   223 F.RETURN EQU *
C20A:28               224         PLP             ;GET PRIOR I/O DISABLE
C20B:30 03    C210    225 F.RET2  BMI  F.RET1     ;=>LEAVE IT DISABLED
C20D:4C C5 FE         226         JMP  FUNCEXIT   ;=>EXIT & ENABLE I/O
C210:4C C8 FE         227 F.RET1  JMP  FUNCEXIT+3 ;EXIT DISABLED
C213:                 228 *
C213:                 229 * Do BOUT, ESCFIX, BASCALC, and KEYIN immediately
C213:                 230 * to avoid destroying Accumulator.
C213:                 231 *
C213:88               232 DISPATCH DEY
C214:30 BA    C1D0    233         BMI  F.BOUT     ;code 0 = 80 column output
C216:88               234         DEY
C217:30 A5    C1BE    235         BMI  B.ESCFIX   ;code 1 = ESCFIX
C219:88               236         DEY
C21A:30 9A    C1B6    237         BMI  F.BASCALC ;code 2 = BASCALC
C21C:88               238         DEY
C21D:30 3D    C25C    239         BMI  B.KEYIN    ;code 3 = KEYIN
C21F:88               240         DEY
C220:30 E2    C204    241         BMI  F.VTABZ    ;code 4 = VTABZ
C222:                 242 *
C222:                 243 * First push address of generic return routine
C222:                 244 *
C222:A9 C2            245         LDA  #<F.RETURN ;return to F.RETURN
C224:48               246         PHA
C225:A9 09            247         LDA  #>F.RETURN-1
C227:48               248         PHA
C228:                 249 *
C228:                 250 * If any of 5 bits in $4FB (MODE) is on, then the mode is not
C228:                 251 * valid for video firmware.  Use old routines.
C228:                 252 *
C228:AD FB 04         253         LDA  MODE       ;no, is mode valid?
C22B:29 D6            254         AND  #M.PASCAL+M.6+M.4+M.2+M.1
C22D:D0 0D    C23C    255         BNE  GETFUNC    ;=>no, use 40 column routines
C22F:98               256         TYA             ;80 column routines in
C230:18               257         CLC             ;2nd half of table
C231:69 0C            258         ADC  #TABLEN
C233:48               259         PHA
C234:20 50 C8         260         JSR  CSETUP     ;set up 80 column cursor
C237:20 FE CD         261         JSR  VTAB       ;calc base
C23A:68               262         PLA
C23B:A8               263         TAY             ;restore Y
C23C:                 264 *
C23C:                 265 * Now push address of routine
C23C:                 266 *
C23C:A9 C1            267 GETFUNC LDA  #<BFUNCPG ;stuff routine address
C23E:48               268         PHA
C23F:B9 44 C2         269         LDA  F.TABLE,Y
C242:48               270         PHA
C243:                 271 *
C243:                 272 * RTS goes to routine on stack.  When the routine
```

```
C243:                273 * does an RTS, it returns to F.RETURN, which restores
C243:                274 * the INTCXROM status and returns.
C243:                275 *
C243:60              276         RTS
C244:                277 *
C244:                278 * Table of routines to call.  All routines are
C244:                279 * in the $C100 page. These are low bytes only.
C244:                280 *
C244:         C244   281 F.TABLE EQU  *
C244:18              282         DFB  #>F.HOME-1 ;(5) 40 column HOME
C245:22              283         DFB  #>F.SCROLL-1 ;(6) 40 column scroll
C246:F1              284         DFB  #>F.CLREOL-1 ;(7) 40 column clear line
C247:5F              285         DFB  #>F.CLREOLZ-1 ;(8) 40 column clear with Y set
C248:75
       286           DFB  #>B.RESET-1 ;(9) 40/80 column reset
C249:02              287         DFB  #>F.CLREOP-1 ;(A) 40 column clear end of page
C24A:A8              288         DFB  #>F.RDKEY-1 ;(B) readkey w/flashing checkerboard
C24B:51              289         DFB  #>F.SETWND-1 ;(C) Set 40 column window
C24C:E1              290         DFB  #>GOMINI-1 ;(D) Mini-assembler
C24D:94              291         DFB  #>F.QUIT-1 ;(E) quit before IN#0,PR#0
C24E:E8              292         DFB  #>FIXPICK-1 ;(F) fix pick for 80 columns
C24F:D5              293         DFB  #>MNNDX-1  ;(10) calc mnemonic index
C250:                294 *
C250:         000C   295 TABLEN  EQU  *-F.TABLE
C250:                296 *
C250:7B              297         DFB  #>B.HOME-1 ;(11) 80 column HOME
C251:64              298         DFB  #>B.SCROLL-1 ;(12) 80 column scroll
C252:67              299         DFB  #>B.CLREOL-1 ;(13) 80 column clear line
C253:6A              300         DFB  #>B.CLREOLZ-1 ;(14) 80 column clear with Y set
C254:75              301         DFB  #>B.RESET-1 ;(15) 40/80 column reset
C255:6F              302         DFB  #>B.CLREOP-1 ;(16) 80 column clear end of page
C256:78              303         DFB  #>B.RDKEY-1 ;(17) readkey w/inverse cursor
C257:72              304         DFB  #>B.SETWND-1 ;(18) 40/80 column VTAB
C258:E1              305         DFB  #>GOMINI-1 ;(19) Mini-Assembler
C259:89              306         DFB  #>B.QUIT-1 ;(1A) quit before IN#0,PR#0
C25A:E8              307         DFB  #>FIXPICK-1 ;(1B) fix pick for 80 columns
C25B:D5              308         DFB  #>MNNDX-1  ;(1C) calc mnemonic index
C25C:                309 *
C25C:         C25C   310 B.KEYIN EQU  *
C25C:2C 1F C0        311         BIT  RD80VID   ;80 columns?
C25F:10 06    C267   312         BPL  B.KEYIN1  ;=>no, flash the cursor
C261:20 74 C8        313         JSR  BIN       ;get a keystroke
C264:4C 0A C2        314 GOF.RET JMP  F.RETURN  ;and return
C267:                315 *
C267:A8              316 B.KEYIN1 TAY            ;preserve A
C268:8A              317         TXA             ;put X on stack
C269:48              318         PHA
C26A:98              319         TYA             ;restore A
C26B:48              320         PHA             ;save char on stack
C26C:48              321         PHA             ;dummy for cursor/char test
C26D:                322 *
C26D:68              323 NEW.CUR PLA             ;get last cursor
C26E:C9 FF           324         CMP  #$FF      ;was it checkerboard?
C270:F0 04    C276   325         BEQ  NEW.CUR1  ;=>yes, get old char
```

```
C272:A9 FF          326          LDA  #$FF       ;no, get checkerboard
C274:D0 02   C278   327          BNE  NEW.CUR2   ;=>always
C276:68             328 NEW.CUR1 PLA             ;get character
C277:48             329          PHA             ;into accumulator
C278:48             330 NEW.CUR2 PHA             ;save for next cursor check
C279:A4 24          331          LDY  CH         ;get cursor horizontal
C27B:91 28          332          STA  (BASL),Y   ;and save char/cursor
C27D:               333 *
C27D:               334 * Now leave char/cursor for awhile or
C27D:               335 * until a key is pressed.
C27D:               336 *
C27D:E6 4E          337 WAITKEY1 INC RNDL        ;bump random seed
C27F:D0 0A   C28B   338          BNE  WAITKEY4   ;=>and check keypress
C281:A5 4F          339          LDA  RNDH       ;is it time to blink yet?
C283:E6 4F          340          INC  RNDH
C285:45 4F          341          EOR  RNDH
C287:29 40          342          AND  #$40
C289:D0 E2   C26D   343          BNE  NEW.CUR    ;=>yes, blink it
C28B:AD 00 C0       344 WAITKEY4 LDA KBD         ;Ivories been tickled?
C28E:10 ED   C27D   345          BPL  WAITKEY1   ;no, keep blinking
C290:               346 *
C290:68             347          PLA             ;pop char/cursor
C291:68             348          PLA             ;pop  character
C292:A4 24          349          LDY  CH         ;and display it
C294:91 28          350          STA  (BASL),Y   ;(erase cursor)
C296:68             351          PLA             ;restore X
C297:AA             352          TAX
C298:AD 00 C0       353          LDA  KBD        ;now retrieve the key
C29B:8D 10 C0       354          STA  KBDSTRB    ;clear the strobe
C29E:30 C4   C264   355          BMI  GOF.RET    ;=>exit always
C2A0:               356 *
C2A0:        C2A0   357 B.SETWNDX EQU *
C2A0:20 52 C1       358          JSR  F.SETWND   ;set 40 column width
C2A3:2C 1F C0       359          BIT  RD80VID    ;80 columns?
C2A6:10 02   C2AA   360          BPL  SKPSHFT    ;=>no, width ok
C2A8:06 21          361          ASL  WNDWDTH    ;make it 80
C2AA:A5 25          362 SKPSHFT LDA  CV
C2AC:8D FB 05       363          STA  OURCV      ;update OURCV
C2AF:60             364          RTS
C2B0:               365 *
C2B0:               366 * HANDLE RESET FOR MONITOR:
C2B0:               367 *
C2B0:        C2B0   368 B.RESETX EQU *
C2B0:A9 FF          369          LDA  #$FF       ;DESTROY MODE BYTE
C2B2:8D FB 04       370          STA  MODE
C2B5:AD 5D C0       371          LDA  CLRAN2     ;SETUP
C2B8:AD 5F C0       372          LDA  CLRAN3     ; ANNUNCIATORS
C2BB:               373 *
C2BB:               374 * IF THE OPEN APPLE KEY
C2BB:               375 *  (ALIAS PADDLE BUTTONS 0) IS
C2BB:               376 *  DEPRESSED, COLDSTART THE SYSTEM
C2BB:               377 *  AFTER DESTROYING MEMORY:
C2BB:               378 *
C2BB:AD 62 C0       379          LDA  BUTN1      ;GET BUTTON 1 (SOLID)
```

```
C2BE:10 03   C2C3  380          BPL  NODIAGS    ;=>Up, no diags
C2C0:4C 00 C6      381          JMP  DIAGS      ;=>else go do diagnostics
C2C3:AD 61 C0      382 NODIAGS LDA  BUTN0      ;GET BUTTON 0 (OPEN)
C2C6:10 1A   C2E2  383          BPL  RESETRET   ;=>NOT JIVE OR DIAGS
C2C8:              384 *
C2C8:              385 * BLAST 2 BYTES OF EACH PAGE,
C2C8:              386 *  INCLUDING THE RESET VECTOR:
C2C8:              387 *
C2C8:A0 B0         388          LDY  #$B0      ;LET IT PRECESS DOWN
C2CA:A9 00         389          LDA  #0
C2CC:85 3C         390          STA  A1L
C2CE:A9 BF         391          LDA  #$BF      ;START FROM BFXX DOWN
C2D0:38            392          SEC            ;FOR SUBTRACT
C2D1:        C2D1  393 BLAST   EQU  *
C2D1:85 3D         394          STA  A1H
C2D3:48            395          PHA            ;save acc to store
C2D4:A9 A0         396          LDA  #$A0      ;blanks
C2D6:91 3C         397          STA  (A1L),Y
C2D8:88            398          DEY
C2D9:91 3C         399          STA  (A1L),Y
C2DB:68            400          PLA            ;restore acc for counter
C2DC:E9 01         401          SBC  #1        ;BACK DOWN TO NEXT PAGE
C2DE:C9 01         402          CMP  #1        ;STAY AWAY FROM STACK!
C2E0:D0 EF   C2D1  403          BNE  BLAST
C2E2:              404 *
C2E2:              405 * If there is a ROM card plugged into slot 3,
C2E2:              406 * don't switch in the internal ROM C3 space.  If not,
C2E2:              407 * only switch them in if there is a RAM card
C2E2:              408 * in the video slot.
C2E2:              409 *
C2E2:              410 * NOTE: The //e powers up with internal $C3 ROM switched
C2E2:              411 * in.  TSTROMCARD switches it out, RESETRET may or may
C2E2:              412 * not switch it back in.
C2E2:              413 *
C2E2:        C2E2  414 RESETRET EQU *
C2E2:8D 0B C0      415          STA  SETSLOTC3ROM ;swap in slot 3
C2E5:20 89 CA      416          JSR  TSTROMCRD ;ROM or no card plugged in?
C2E8:D0 03   C2ED  417          BNE  GORETN1   ;=>ROM or no card, leave $C3 slot
C2EA:8D 0A C0      418          STA  SETINTC3ROM ;card, enable internal ROM
C2ED:60            419 GORETN1 RTS
C2EE:              420 *
C2EE:88 95 8A 8B   421 ESCIN   DFB  $88,$95,$8A,$8B
C2F2:              422 *
C2F2:A4 24         423 B.RDKEYX LDY CH         ;get cursor position
C2F4:B1 28         424          LDA  (BASL),Y  ;and character
C2F6:2C 1F C0      425          BIT  RD80VID   ;80 columns?
C2F9:30 F2   C2ED  426          BMI  GORETN1   ;=>don't display cursor
C2FB:4C 26 CE      427          JMP  INVERT    ;else display cursor, exit
C2FE:              428 *
C2FE:        0002  429 ZSPAREC2 EQU C3ORG-*
C2FE:        0002  430          DS   C3ORG-*,0
C300:        0000  431          IFNE *-C3ORG
 S                 432          FAIL 2,'C300   overflow'
C300:              433          FIN
```

```
C300:                19         INCLUDE C3SPACE
C300:                 1 ***************************************
C300:                 2 *
C300:                 3 * THIS IS THE $C3XX ROM SPACE:
C300:                 4 * Note:  This page must not be used by any routines
C300:                 5 * called by the F8 ROM.  When it is referenced, it claims
C300:                 6 * the C800 space (kicking out anyone who was using it).
C300:                 7 * This also means that peripheral cards cannot use the AUXMOVE
C300:                 8 * and XFER routines from their C800 space.
C300:                 9 *
C300:                10 ***************************************
C300:         C300   11 CN00    EQU  *
C300:         C300   12 BASICINT EQU *
C300:2C 43 CE        13         BIT  SEV       ;set vflag (init)
C303:70 12    C317   14         BVS  BASICENT  ;(ALWAYS TAKEN)
C305:                15 *
C305:                16 * BASIC input entry point.  After a PR#3, this is the
C305:                17 * address that is called to input each character.
C305:                18 *
C305:         C305   19 BASICIN EQU  *
C305:38              20         SEC
C306:90              21         DFB  $90       ;BCC OPCODE (NEVER TAKEN)
C307:                22 *
C307:                23 * BASIC output entry point.  After a PR#3, this is the
C307:                24 * address that is called to output each character.
C307:                25 *
C307:         C307   26 BASICOUT EQU *
C307:18              27         CLC
C308:B8              28         CLV            ;CLEAR VFLAG (NOT INIT)
C309:50 0C    C317   29         BVC  BASICENT  ;(ALWAYS TAKEN)
C30B:                30 *
C30B:                31 * Pascal 1.1 Firmware Protocol table:
C30B:                32 *
C30B:                33 * This tables identifies this as an Apple //e 80 column
C30B:                34 * card.  It points to the four routines available to
C30B:                35 * programs doing I/O using the Pascal 1.1 Firmware
C30B:                36 * Protocol.
C30B:                37 *
C30B:01              38         DFB  $01       ;GENERIC SIGNATURE BYTE
C30C:88              39         DFB  $88       ;DEVICE SIGNATURE BYTE
C30D:                40 *
C30D:4A              41         DFB  #>JPINIT   ;PASCAL INIT
C30E:50              42         DFB  #>JPREAD   ;PASCAL READ
C30F:56              43         DFB  #>JPWRITE  ;PASCAL WRITE
C310:5C              44         DFB  #>JPSTAT   ;PASCAL STATUS
C311:                45 ***************************************
C311:                46 *
C311:                47 * 128K SUPPORT ROUTINE ENTRIES:
C311:                48 *
C311:4C 76 C3        49         JMP  MOVE      ;MEMORY MOVE ACROSS BANKS
C314:4C C3 C3        50         JMP  XFER      ;TRANSFER ACROSS BANKS
C317:                51 ***************************************
C317:                52 *
C317:8D 7B 06        53 BASICENT STA CHAR
```

```
C31A:98              54          TYA             ; AND Y
C31B:48              55          PHA
C31C:8A              56          TXA             ; AND X
C31D:48              57          PHA
C31E:08              58          PHP             ;SAVE CARRY & VFLAG
C31F:                59 *
C31F:                60 * If escape mode is allowed, the high bit of MSLOT is
C31F:                61 * clear.  Set M.CTL to flag that 1) escapes are allowed, and
C31F:                62 * 2) that control characters should not be echoed.
C31F:                63 * M.CTL is cleared by BPRINT.
C31F:                64 *
C31F:AD FB 04        65          LDA  MODE       ;else esc enable, ctl disable
C322:2C F8 07        66          BIT  MSLOT      ;get MSLOT
C325:30 05   C32C    67          BMI  NOGETLN    ;=>Esc disable, ctl char enable
C327:09 08           68          ORA  #M.CTL
C329:8D FB 04        69          STA  MODE
C32C:                70 *
C32C:        C32C    71 NOGETLN EQU  *
C32C:20 6D C3        72          JSR  SETC8      ;SETUP C8 INDICATOR
C32F:28              73          PLP             ;GET VFLAG (INIT)
C330:70 15   C347    74          BVS  JBASINIT   ;=>DO THE INIT
C332:                75 *
C332:                76 * If a PR#0 has been done, input should be transferred
C332:                77 * from the video firmware to KEYIN.  This is detected
C332:                78 * if the high bit of the mode byte is set.
C332:                79 *
C332:90 10   C344    80          BCC  JC8        ;=>output, no problem
C334:AA              81          TAX             ;test mode
C335:10 0D   C344    82          BPL  JC8        ;video firmware is on
C337:20 5B CD        83          JSR  SETKEYIN   ;else set FD1B as input
C33A:68              84          PLA             ;restore registers
C33B:AA              85          TAX
C33C:68              86          PLA
C33D:A8              87          TAY
C33E:AD 7B 06        88          LDA  CHAR
C341:6C 38 00        89          JMP  (KSWL)     ;go input the character
C344:                90 *
C344:4C 7C C8        91 JC8      JMP  C8BASIC    ;GET OUT OF CN SPACE
C347:4C 03 C8        92 JBASINIT JMP BASICINIT ;=>GOTO C8 SPACE
C34A:                93 *
C34A:        C34A    94 JPINIT  EQU  *
C34A:20 6D C3        95          JSR  SETC8      ;SETUP C8 INDICATOR
C34D:4C B4 C9        96          JMP  PINIT      ;XFER TO PASCAL INIT
C350:        C350    97 JPREAD  EQU  *
C350:20 6D C3        98          JSR  SETC8      ;SETUP C8 INDICATOR
C353:4C D6 C9        99          JMP  PREAD      ;XFER TO PASCAL READ
C356:        C356   100 JPWRITE EQU  *
C356:20 6D C3       101          JSR  SETC8      ;SETUP C8 INDICATOR
C359:4C F0 C9       102          JMP  PWRITE     ;XFER TO PASCAL WRITE
C35C:               103 *
C35C:AA             104 JPSTAT  TAX              ;is request code = 0?
C35D:F0 08   C367   105          BEQ  PIORDY     ;=>yes, ready for output
C35F:CA             106          DEX             ;check for any input
C360:D0 07   C369   107          BNE  PSTERR     ;=>bad request, return error
```

```
C362:2C 00 C0      108          BIT  KBD        ;look for a key
C365:10 04   C36B  109          BPL  PNOTRDY    ;=>no keystroked
C367:38            110 PIORDY   SEC
C368:60            111          RTS
C369:              112 *
C369:A2 03         113 PSTERR   LDX  #3         ;else flag error
C36B:18            114 PNOTRDY  CLC
C36C:60            115          RTS
C36D:              116 ****************************************
C36D:              117 * NAME     : SETC8
C36D:              118 * FUNCTION: SETUP IRQ $C800 PROTOCOL
C36D:              119 * INPUT    : NONE
C36D:              120 * OUTPUT   : NONE
C36D:              121 * VOLATILE: NOTHING
C36D:              122 * CALLS    : NOTHING
C36D:              123 ****************************************
C36D:              124 *
C36D:       C36D   125 SETC8    EQU  *
C36D:A2 C3         126          LDX  #<CN00     ;SLOT NUMBER
C36F:8E F8 07      127          STX  MSLOT      ;STUFF IT
C372:AE FF CF      128          LDX  $CFFF      ;kick out other $C8 ROMs
C375:60            129          RTS
C376:              130 ****************************************
C376:              131 * NAME     : MOVE
C376:              132 * FUNCTION: PERFORM CROSSBANK MEMORY MOVE
C376:              133 * INPUT    : A1=SOURCE ADDRESS
C376:              134 *          : A2=SOURCE END
C376:              135 *          : A4=DESTINATION START
C376:              136 *          : CARRY SET=MAIN-->CARD
C376:              137 *                CLR=CARD-->MAIN
C376:              138 * OUTPUT   : NONE
C376:              139 * VOLATILE: NOTHING
C376:              140 * CALLS    : NOTHING
C376:              141 ****************************************
C376:              142 *
C376:       C376   143 MOVE     EQU  *
C376:48            144          PHA             ;SAVE AC
C377:98            145          TYA             ; AND Y
C378:48            146          PHA
C379:AD 13 C0      147          LDA  RDRAMRD    ;SAVE STATE OF
C37C:48            148          PHA             ; MEMORY FLAGS
C37D:AD 14 C0      149          LDA  RDRAMWRT
C380:48            150          PHA
C381:              151 *
C381:              152 * SET FLAGS FOR CROSSBANK MOVE:
C381:              153 *
C381:90 08   C38B  154          BCC  MOVEC2M    ;=>CARD-->MAIN
C383:8D 02 C0      155          STA  RDMAINRAM ;SET FOR MAIN
C386:8D 05 C0      156          STA  WRCARDRAM ; TO CARD
C389:B0 06   C391  157          BCS  MOVESTRT   ;=>(ALWAYS TAKEN)
C38B:              158 *
C38B:       C38B   159 MOVEC2M EQU  *
C38B:8D 04 C0      160          STA  WRMAINRAM ;SET FOR CARD
C38E:8D 03 C0      161          STA  RDCARDRAM ; TO MAIN
```

```
C391:                 162 *
C391:          C391   163 MOVESTRT EQU *
C391:A0 00            164          LDY  #0          ;DUMMY INDEX
C393:                 165 *
C393:          C393   166 MOVELOOP EQU *
C393:B1 3C            167          LDA  (A1L),Y     ;GET A BYTE
C395:91 42            168          STA  (A4L),Y     ;MOVE IT
C397:E6 42            169          INC  A4L
C399:D0 02     C39D   170          BNE  NXTA1
C39B:E6 43            171          INC  A4H
C39D:A5 3C            172 NXTA1    LDA  A1L
C39F:C5 3E            173          CMP  A2L
C3A1:A5 3D            174          LDA  A1H
C3A3:E5 3F            175          SBC  A2H
C3A5:E6 3C            176          INC  A1L
C3A7:D0 02     C3AB   177          BNE  C01
C3A9:E6 3D            178          INC  A1H
C3AB:90 E6     C393   179 C01      BCC  MOVELOOP    ;=>MORE TO MOVE
C3AD:                 180 *
C3AD:                 181 * RESTORE ORIGINAL FLAGS:
C3AD:                 182 *
C3AD:8D 04 C0         183          STA  WRMAINRAM ;CLEAR FLAG2
C3B0:68               184          PLA            ;GET ORIGINAL STATE
C3B1:10 03     C3B6   185          BPL  C03       ;=>IT WAS OFF
C3B3:8D 05 C0         186          STA  WRCARDRAM
C3B6:          C3B6   187 C03      EQU  *
C3B6:8D 02 C0         188          STA  RDMAINRAM ;CLEAR FLAG1
C3B9:68               189          PLA            ;GET ORIGINAL STATE
C3BA:10 03     C3BF   190          BPL  MOVERET   ;=>IT WAS OFF
C3BC:8D 03 C0         191          STA  RDCARDRAM
C3BF:          C3BF   192 MOVERET  EQU  *
C3BF:68               193          PLA            ;RESTORE Y
C3C0:A8               194          TAY
C3C1:68               195          PLA            ; AND AC
C3C2:60               196          RTS
C3C3:                 197 ****************************************
C3C3:                 198 * NAME     : XFER
C3C3:                 199 * FUNCTION: TRANSFER CONTROL CROSSBANK
C3C3:                 200 * INPUT    : $03ED=TRANSFER ADDR
C3C3:                 201 *          : CARRY SET=XFER TO CARD
C3C3:                 202 *                  CLR=XFER TO MAIN
C3C3:                 203 *          : VFLAG CLR=USE STD ZP/STK
C3C3:                 204 *          :       SET=USE ALT ZP/STK
C3C3:                 205 * OUTPUT   : NONE
C3C3:                 206 * VOLATILE: $03ED/03EE IN DEST BANK
C3C3:                 207 * CALLS    : NOTHING
C3C3:                 208 * NOTE     : ENTERED VIA JMP, NOT JSR
C3C3:                 209 ****************************************
C3C3:                 210 *
C3C3:          C3C3   211 XFER     EQU  *
C3C3:48               212          PHA            ;SAVE AC ON CURRENT STACK
C3C4:                 213 *
C3C4:                 214 * COPY DESTINATION ADDRESS TO THE
C3C4:                 215 *  OTHER BANK SO THAT WE HAVE IT
```

```
C3C4:                216 *  IN CASE WE DO A SWAP:
C3C4:                217 *
C3C4:AD ED 03        218          LDA  $03ED      ;GET XFERADDR LO
C3C7:48              219          PHA             ;SAVE ON CURRENT STACK
C3C8:AD EE 03        220          LDA  $03EE      ;GET XFERADDR HI
C3CB:48              221          PHA             ;SAVE IT TOO
C3CC:                222 *
C3CC:                223 * SWITCH TO APPROPRIATE BANK:
C3CC:                224 *
C3CC:90 08   C3D6    225          BCC  XFERC2M    ;=>CARD-->MAIN
C3CE:8D 03 C0        226          STA  RDCARDRAM ;SET FOR RUNNING
C3D1:8D 05 C0        227          STA  WRCARDRAM ; IN CARD RAM
C3D4:B0 06   C3DC    228          BCS  XFERZP     ;=> always taken
C3D6:        C3D6    229 XFERC2M  EQU  *
C3D6:8D 02 C0        230          STA  RDMAINRAM ;SET FOR RUNNING
C3D9:8D 04 C0        231          STA  WRMAINRAM ; IN MAIN RAM
C3DC:                232 *
C3DC:        C3DC    233 XFERZP   EQU  *          ;SWITCH TO ALT ZP/STK
C3DC:68              234          PLA             ;STUFF XFERADDR
C3DD:8D EE 03        235          STA  $03EE      ; HI AND
C3E0:68              236          PLA
C3E1:8D ED 03        237          STA  $03ED      ;  LO
C3E4:68              238          PLA             ;RESTORE AC
C3E5:70 05   C3EC    239          BVS  XFERAZP    ;=>switch in alternate zp
C3E7:8D 08 C0        240          STA  SETSTDZP   ;else force standard zp
C3EA:50 03   C3EF    241          BVC  JMPDEST    ;=>always perform transfer
C3EC:8D 09 C0        242 XFERAZP  STA  SETALTZP   ;switch in alternate zp
C3EF:6C ED 03        243 JMPDEST  JMP  ($03ED)    ;=>off we go
C3F2:                244 *
C3F2:        0002    245          DS   C3ORG+$F4-*,0 ;pad to interrupt stuff
C3F4:                246 *
C3F4:                247 * This is where the interrupt routine returns to.
C3F4:                248 * At this point the ROM is not necessarily switched in so...
C3F4:                249 *
C3F4:8D 81 C0        250 IRQDONE  STA  $C081      ;read ROM, write RAM
C3F7:4C 7A FC        251          JMP  IRQDONE2   ;and jump to ROM
C3FA:                252 *
C3FA:                253 * This is the main entry point for the interrupt
C3FA:                254 * handler.  This switches in the internal ROM and
C3FA:                255 * jumps to the main part of the interrupt handler
C3FA:                256 * at $C400.
C3FA:                257 *
C3FA:2C 15 C0        258 irq      bit  rdcxrom    ;Test internal or external rom
C3FD:8D 07 C0        259          sta  setintcxrom ;Force in ROM to get to interrupt handler
C400:                260 *
C400:                261 * Fall into $C400 which is now switched in!!
C400:                262 *
C400:                 20          INCLUDE IRQ
C400:                  1 *
C400:                  2 * Here is the main interrupt handler
C400:                  3 *
C400:                  4 ****************************************
C400:        C400      5 newirq   equ  *
C400:D8                6          cld             ;make no assumptions!!
```

```
C401:38              7          sec             ;C=1 if internal slot space
C402:30 01   C405    8          bmi  irqintcx
C404:18              9          clc
C405:48             10 irqintcx pha             ;Save A on stack instead of $45
C406:48             11          pha             ;Make room for rts if needed
C407:48             12          pha
C408:8A             13          txa             ;Save X
C409:BA             14          tsx             ;Get stack pointer for BRK bit
C40A:E8             15          inx             ;Can't do add cause we need C
C40B:E8             16          inx
C40C:E8             17          inx
C40D:E8             18          inx
C40E:48             19          pha
C40F:98             20          tya             ;and Y
C410:48             21          pha
C411:BD 00 01       22          lda  $100,x     ;Get status for break test
C414:29 10          23          and  #$10       ;A = $10 if break
C416:A8             24          tay              ;Save it for later
C417:               25 * Now test & set the state of the machine. Don't alter Y
C417:AD 18 C0       26          lda  rd80col    ;Test for 80 store and page 2
C41A:2D 1C C0       27          and  rdpage2
C41D:29 80          28          and  #$80       ;Make it 0 or $80
C41F:F0 05   C426   29          beq  irq2       ;Branch if no change needed
C421:A9 20          30          lda  #$20       ;Set shifted page 2 reset bit
C423:8D 54 C0       31          sta  txtpage1   ;Set page 1
C426:2A             32 irq2     rol  A          ;Align bit & shift in slotcx bit
C427:2C 13 C0       33          bit  rdramrd    ;Are we reading from aux ram?
C42A:10 05   C431   34          bpl  irq3       ;Branch if main ram read
C42C:8D 02 C0       35          sta  rdmainram ;Else, switch main in
C42F:09 20          36          ora  #$20       ;and record the event
C431:2C 14 C0       37 irq3     bit  rdramwrt   ;Do the same for ram write
C434:10 05   C43B   38          bpl  irq4
C436:8D 04 C0       39          sta  wrmainram
C439:09 10          40          ora  #$10
C43B:        C43B   41 irq4     equ  *
C43B:2C 12 C0       42 irq5     bit  rdlcram    ;Determine if language card active
C43E:10 0C   C44C   43          bpl  irq7
C440:09 0C          44          ora  #$0C       ;Sets two bits. Second is redundant
C442:2C 11 C0       45          bit  rdlcbnk2   ;if INC used to restore.
C445:10 02   C449   46          bpl  irq6       ;Branch if not page 2 of $D000
C447:49 06          47          eor  #$06       ;Set bits for page 2
C449:8D 81 C0       48 irq6     sta  romin      ;Enable ROM STA leaves write enable alone
C44C:2C 16 C0       49 irq7     bit  rdaltzp    ;Last...and very important
C44F:10 0D   C45E   50          bpl  irq8       ;If alternate stack
C451:BA             51          tsx              ;store current stack pointer at $101
C452:8E 01 01       52          stx  $101
C455:AE 00 01       53          ldx  $100       ;Retreve main stack pointer from $100
C458:9A             54          txs
C459:8D 08 C0       55          sta  setstdzp
C45C:09 80          56          ora  #$80       ;Mark stack switched
C45E:88             57 irq8     dey              ;Was it a break?
C45F:30 0C   C46D   58          bmi  irq9
C461:85 44          59          sta  macstat    ;Save state of machine
C463:68             60          pla              ;Restore registers
```

```
C464:A8             61          tay
C465:68             62          pla
C466:AA             63          tax
C467:68             64          pla
C468:68             65          pla             ;A stored where RTS address would go
C469:68
  66          pla
C46A:4C 47 FA       67          jmp  newbreak   ;Go to normal break routine stuff
C46D:48             68 irq9     pha             ;Save state of machine on stack
C46E:AD F8 07       69          lda  mslot      ;Save mslot
C471:48             70          pha
C472:A9 C3          71          lda  #<irqdone  ;Save return irq address
C474:48             72          pha
C475:A9 F4          73          lda  #>irqdone  ;so when interrupt does RTI
C477:48             74          pha             ;It returns to irqdone
C478:08             75          php             ;Status for user's RTI
C479:4C 74 FC       76          jmp  irquser    ;Off to the user
C47C:               77 * The user's RTI returns here
C47C:               78 * BEWARE
C47C:               79 *  The rom must be reenabled with a LDA romin
C47C:               80 *  This way if the LC was write protected, it still is
C47C:               81 *   if it was write enabled, it still is
C47C:               82 *   if it was being write enabled ( 2 ldas), it still will be
C47C:               83 *  The restore loop uses an INC because some of the switches are read
C47C:               84 *  and some are write.  It must be an INC abs,x since both the 6502 and
C47C:               85 *  the 65C02 do two reads before the write.
C47C:AD 81 C0       86 irqfix   lda  romin      ;Must be lda!
C47F:68             87          pla             ;Recover machine state
C480:10 07   C489   88          bpl  irqdnl     ;Branch if main ZP
C482:8D 09 C0       89          sta  setaltzp
C485:AE 01 01       90          ldx  $101       ;Get alt stack pointer
C488:9A             91          txs
C489:A0 06          92 irqdnl   ldy  #$06       ;Y = index into table of switch addresses
C48B:10 06   C493   93 irqdn2   bpl  irqdn3     ;Branch if no change
C48D:BE C1 C4       94          ldx  irqtble,y  ;Get soft switch address
C490:FE 00 C0       95          inc  $C000,x    ;Hit the switch. NO PAGE CROSS!
C493:88             96 irqdn3   dey
C494:30 03   C499   97          bmi  irqdn4
C496:0A             98          asl  A          ;Get next bit to check
C497:D0 F2   C48B   99          bne  irqdn2
C499:0A            100 irqdn4   asl  A          ;C = 1 if internal slot space
C49A:0A            101          asl  A
C49B:68            102          pla             ;Restore the registers
C49C:A8            103          tay
C49D:BA            104          tsx             ;Save the stack pointer
C49E:A9 40         105          lda  #$40       ;RTI opcode
C4A0:48            106          pha
C4A1:A9 C0         107          lda  #<setslotcxrom
C4A3:48            108          pha
C4A4:A9 06         109          lda  #>setslotcxrom
C4A6:69 00         110          adc  #0         ;Add 1 if internal slot space
C4A8:48            111          pha
C4A9:A9 8D         112          lda  #$8D       ;STA setslotcxrom
C4AB:48            113          pha
```

```
C4AC:9A            114         txs             ;Restore stack pointer
C4AD:8A            115         txa             ;Make return address on stack point to code on stack
C4AE:69 03         116         adc  #3        ;C = 0 from earlier adc
C4B0:AA            117         tax
C4B1:38            118         sec
C4B2:E9 07         119         sbc  #7        ;Point to where code starts
C4B4:9D 00 01      120         sta  $100,x
C4B7:E8            121         inx
C4B8:A9 01         122         lda  #$1
C4BA:9D 00 01      123         sta  $100,x
C4BD:68            124         pla
C4BE:AA            125         tax
C4BF:68            126         pla
C4C0:60            127         rts             ;Go to code on stack


C4C1:83 8B 8B      129 irqtble dfb  >lcbank2,>lcbank1,>lcbank1
C4C4:05 03 55      130         dfb  >wrcardram,>rdcardram,>txtpage2
C4C7:               21         INCLUDE DIAGS
----- NEXT OBJECT FILE NAME IS REFLIST.1
C600:        C600    1         ORG  C30RG+$300
C600:                2 * These routines test all 64K RAM, as well as the 64K on an Auxiliary
C600:                3 * memory card (when present).  With the exception of the INTCXROM switch
C600:                4 * of the IOU, all combinations of the IOU switches are tested and ver-
C600:                5 * ified.  All configurations of the MMU switches are also tested.
C600:                6 *
C600:                7 * In the event of any failure, the diagnostic is halted. A message
C600:                8 * is written to screen memory indicating the source of the failure.
C600:                9 * When RAM fails the message is composed of "RAM ZP" (indicating failure
C600:               10 * detected in the first page of RAM) or "RAM" (meaning the other 63.75K),
C600:               11 * followed by a binary representation of the failing bits set to "1".
C600:               12 * For example, "RAM   0 1 1 0 0 0 0 0" indicates that bits 5 and 6 were
C600:               13 * detected as failing.  To represent auxiliary memory, a "*" symbol is
C600:               14 * printed preceeding the message.
C600:               15 *
C600:               16 * When the MMU or IOU fail, the message is simply "MMU" or "IOU".
C600:               17 *
C600:               18 * The test will run continuously for as long as the Open and Closed
C600:               19 * Apple keys remain depressed (or no keyboard is connected) and no
C600:               20 * failures are encountered.  The message "System OK" will appear in
C600:               21 * the middle of the screen when a successful cycle has been run and
C600:               22 * either of the Apple keys are no longer depressed. Another cycle
C600:               23 * may be initiated by pressing both Apple keys again while this message
C600:               24 * is on the screen.  To exit diagnostics, Control-Reset must be pressed
C600:               25 * without the Apple keys depressed.
C600:               26 *
C600:        C051   27 TEXT    equ  $C051
C600:        0009   28 IOUIDX  equ  $09
C600:        0001   29 MMUIDX  equ  $01
C600:        05B8   30 SCREEN  equ  $5B8
C600:        C000   31 IOSPACE equ  $C000
C600:               32 *
C600:        C600   33 DIAGS   equ  *
```

```
C600:8D 50 C0          34          sta  $C050
C603:                  35 * Test Zero-Page, then all of memory.  Report errors when encountered.
C603:                  36 * Accumulator can be anything on entry. All registers used, but no stack.
C603:                  37 * Addresses between $C000 and $CFFF are mapped to main $D000 bank.
C603:                  38 * Auxillary 64K is also tested if present.

C603:A0 04             40 TSTZPG  ldy  #$4
C605:A2 00             41         ldx  #0
C607:18                42 zp1     clc             ;fill zero page with a pattern
C608:79 B4 C7          43         adc  ntbl,y
C60B:95 00             44         sta  $00,x
C60D:E8                45         inx
C60E:D0 F7    C607     46         bne  zp1        ;after all bytes filled,
C610:18                47 zp2     clc             ; ACC has original value again.
C611:79 B4 C7          48         adc  ntbl,y     ;so values can be tested
C614:D5 00             49         cmp  $00,x
C616:D0 10    C628     50         bne  ZPERROR    ;branch if memory failed
C618:E8                51         inx
C619:D0 F5    C610     52         bne  zp2        ;loop until all 256 bytes tested
C61B:6A                53         ror  a          ;change ACC so location $FF will change
C61C:2C 19 C0          54         bit  RDVBLBAR   ; use RDVBLBAR for a little randomness...
C61F:10 02    C623     55         bpl  zp3
C621:49 A5             56         eor  #$A5
C623:88                57 zp3     dey             ;use a different pattern now
C624:10 E1    C607     58         bpl  zp1        ;branch to retest with other value
C626:30 06    C62E     59         bmi  TSTMEM     ;branch always


C628:55 00             61 ZPERROR eor  $00,x      ;which bits are bad?
C62A:18                62         clc             ;indicate zero page failure
C62B:4C CD C6          63         jmp  BADBITS
C62E:         C62E     64 TSTMEM  equ  *
C62E:86 01             65         stx  $01
C630:86 02             66         stx  $02
C632:86 03             67         stx  $03
C634:A2 04             68         ldx  #4         ;do RAM $100-$FFFF five times
C636:86 04             69         stx  $04
C638:E6 01             70 mem1    inc  $01        ;point to page 1 first
C63A:A8                71 mem2    tay             ;save ACC in Y for now
C63B:8D 83 C0          72         sta  $C083      ;anticipate not $C000 range...
C63E:8D 83 C0          73         sta  $C083
C641:A5 01             74         lda  $01        ;get page address
C643:29 F0             75         and  #$F0       ;test for $C0-$CF range
C645:C9 C0             76         cmp  #$C0
C647:D0 0C    C655     77         bne  mem3       ;branch if not...
C649:AD 8B C0          78         lda  $C08B
C64C:AD 8B C0          79         lda  $C08B      ;select primary $D000 space
C64F:A5 01             80         lda  $01
C651:69 0F             81         adc  #$F        ;Plus carry =+$10
C653:D0 02    C657     82         bne  mem4       ;branch always taken
C655:A5 01             83 mem3    lda  $01
C657:85 03             84 mem4    sta  $03
C659:98                85         tya             ;restore pattern to ACC
C65A:A0 00             86         ldy  #$00       ;fill this page with the pattern
```

```
C65C:18              87 mem5    clc
C65D:7D B4 C7        88         adc  ntbl,x
C660:91 02           89         sta  ($02),y
C662:CA              90         dex              ;keep x in the range 0-4
C663:10 02   C667    91         bpl  mem6
C665:A2 04           92         ldx  #4
C667:C8              93 mem6    iny              ;all 256 filled yet?
C668:D0 F2   C65C    94         bne  mem5        ;branch if not
C66A:E6 01           95         inc  1           ;bump page #
C66C:D0 CC   C63A    96         bne  mem2        ;loop through $0100 to $FF00

C66E:E6 01           98         inc  $01         ;point to page 1 again
C670:A8              99 mem7    tay              ;save ACC in Y for now
C671:AD 83 C0       100         lda  $C083       ;anticipate not $C000 range...
C674:AD 83 C0       101         lda  $C083
C677:A5 01          102         lda  $01         ;get page address
C679:29 F0          103         and  #$F0        ;test for $C0-$CF range
C67B:C9 C0          104         cmp  #$C0
C67D:D0 09   C688   105         bne  mem8        ;branch if not...
C67F:AD 8B C0       106         lda  $C08B       ;select primary $D000 space
C682:A5 01          107         lda  $01
C684:69 0F          108         adc  #$F         ;Plus carry =+$10
C686:D0 02   C68A   109         bne  mem9        ;branch always taken
C688:A5 01          110 mem8    lda  $01
C68A:85 03          111 mem9    sta  $03
C68C:98             112         tya              ;restore pattern to ACC
C68D:A0 00          113         ldy  #$00        ;fill this page with the pattern
C68F:18             114 memA    clc
C690:7D B4 C7       115         adc  ntbl,x
C693:51 02          116         eor  ($02),y
C695:D0 35   C6CC   117         bne  MEMERROR    ;if any bits are different, give up!!!
C697:B1 02          118         lda  ($02),y     ;restore correct pattern
C699:CA             119         dex              ;keep x in the range 0-4
C69A:10 02   C69E   120         bpl  memB
C69C:A2 04          121         ldx  #4
C69E:C8             122 memB    iny              ;all 256 filled yet?
C69F:D0 EE   C68F   123         bne  memA        ;branch if not
C6A1:E6 01          124         inc  1           ;bump page #
C6A3:D0 CB   C670   125         bne  mem7        ;loop through $0100 to $FF00
C6A5:6A             126         ror  a           ;change ACC for next pass
C6A6:2C 19 C0       127         bit  RDVBLBAR    ; use RDVBLBAR for a little randomness...
C6A9:10 02   C6AD   128         bpl  memC
C6AB:49 A5          129         eor  #$A5
C6AD:C6 04          130 memC    dec  $04         ;have 5 passes been done yet?
C6AF:10 87   C638   131         bpl  mem1        ;branch if not...

C6B1:AA             133         TAX              ;save acc
C6B2:20 8D C9       134         JSR  STAUX       ;set aux memory & write $EE to $C00,$800
C6B5:D0 07   C6BE   135         BNE  SWCHTST1    ;=>not 128K
C6B7:0E 00 0C       136         ASL  $C00        ;shift test byte
C6BA:0A             137         ASL  A
C6BB:CD 00 0C       138         CMP  $C00        ;check memory
```

```
C6BE:D0 76   C736  139 SWCHTST1 BNE SWCHTST   ;=>not 128K
C6C0:CD 00 08      140          CMP  $800      ;look for shadowing
C6C3:F0 71   C736  141          BEQ  SWCHTST   ;=>not 128K
C6C5:8A            142          txa
C6C6:8D 09 C0      143          STA  SETALTZP  ;swap in alt zero page
C6C9:4C 03 C6      144          jmp  TSTZPG    ; and test it!
C6CC:38            145 MEMERROR sec            ;indicate main ram failure
C6CD:AA            146 BADBITS  tax            ;save bit pattern in x for now
C6CE:AD 13 C0      147          lda  RDRAMRD   ;determine if primary or auxillary RAM
C6D1:B8            148          clv            ;with V-FLG
C6D2:10 03   C6D7  149          bpl  bbitsl    ;branch if primary bank
C6D4:2C B4 C7      150          bit  setv
C6D7:A9 A0         151 bbitsl   lda  #$A0      ;try to clear video screen
C6D9:A0 06         152          ldy  #6
C6DB:99 FE BF      153 clrsts   sta  IOSPACE-2,y
C6DE:99 06 C0      154          sta  IOSPACE+6,y
C6E1:88            155          dey
C6E2:88            156          dey
C6E3:D0 F6   C6DB  157          bne  clrsts
C6E5:8D 51 C0      158          sta  TEXT
C6E8:8D 54 C0      159          sta  TXTPAGE1
C6EB:99 00 04      160 clrs     sta  $400,y
C6EE:99 00 05      161          sta  $500,y
C6F1:99 00 06      162          sta  $600,y
C6F4:99 00 07      163          sta  $700,y
C6F7:C8            164          iny
C6F8:D0 F1   C6EB  165          bne  clrs
C6FA:8A            166          txa            ;test for switch test failure
C6FB:F0 27   C724  167          beq  BADSWTCH  ;branch if it was a switch
C6FD:A0 03         168          ldy  #3
C6FF:B0 02   C703  169          bcs  badmain   ;branch if ZP ok
C701:A0 05         170          ldy  #5
C703:A9 AA         171 badmain  lda  #$AA      ;mark aux report with an asterisks
C705:50 03   C70A  172          bvc  badprim
C707:8D B0 05      173          sta  screen-8
C70A:B9 EA C7      174 badprim  lda  rmess,y
C70D:99 B1 05      175          sta  screen-7,y
C710:88            176          dey
C711:10 F7   C70A  177          bpl  badprim   ;message is either "RAM" or "RAM ZP"
C713:A0 10         178          ldy  #$10      ;print bits
C715:8A            179 bbits2   txa
C716:4A            180          lsr  a
C717:AA            181          tax
C718:A9 58         182          lda  #$58      ;bits are printed as ascii 0 or 1
C71A:2A            183          rol  a
C71B:99 B6 05      184          sta  screen-2,y
C71E:88            185          dey
C71F:88            186          dey
C720:D0 F3   C715  187          bne  bbits2
C722:F0 FE   C722  188 hangx    beq  hangx     ;hang forever and ever
C724:A0 02         189 BADSWTCH ldy  #2
C726:B9 F0 C7      190 bswtchl  lda  smess,y
C729:90 03   C72E  191          bcc  bswtch2   ;branch if MMU in error
C72B:B9 F3 C7      192          lda  smess+3,y ;else indicate IOU error
```

```
C72E:99 B8 05      193 bswtch2 sta  screen,y
C731:88            194         dey
C732:10 F2   C726  195         bpl  bswtch1   ;print "MMU" or "IOU"
C734:30 FE   C734  196 hangy   bmi  hangy     ;branch forever


C736:A0 01         198 SWCHTST ldy  #MMUIDX
C738:A9 7F         199 swtst1  lda  #$7F
C73A:6A            200 swtst2  ror  a         ;set switches of the IOU/MMU to match Accumulator
C73B:BE B9 C7      201         ldx  SWTBL0,y
C73E:F0 0F   C74F  202         beq  swtst4    ;branch if done setting switches
C740:90 03   C745  203         bcc  swtst3    ;branch if setting switch to 0-state
C742:BE C9 C7      204         ldx  SWTBL1,y  ;else get index to set switch to 1
C745:9D FF BF      205 swtst3  sta  IOSPACE-1,x ;set switch
C748:C8            206         iny
C749:D0 EF   C73A  207         bne  swtst2    ;branch always taken...
C74B:              208 *
C74B:AE 30 C0      209 click   ldx  $C030
C74E:2A            210         rol  a
C74F:88            211 swtst4  dey
C750:BE D9 C7      212         ldx  RSWTBL,y  ;now verify the settings just made
C753:F0 13   C768  213         beq  swtst6    ;branch if done this pass
C755:30 F4   C74B  214         bmi  click     ;branch if this switch no to be verified.
C757:2A            215         rol  a
C758:90 07   C761  216         bcc  swtst5
C75A:1E 00 C0      217         asl  IOSPACE,x
C75D:90 17   C776  218         bcc  swerr
C75F:B0 EE   C74F  219         bcs  swtst4    ;branch always
C761:1E 00 C0      220 swtst5  asl  IOSPACE,x
C764:B0 10   C776  221         bcs  swerr
C766:90 E7   C74F  222         bcc  swtst4    ;branch always
C768:              223 *
C768:2A            224 swtst6  rol  a         ;restore original value
C769:C8            225         iny            ; and IOU/MMU index
C76A:38            226         sec
C76B:E9 01         227         sbc  #1        ;try next pattern
C76D:B0 CB   C73A  228         bcs  swtst2
C76F:88            229         dey            ;was MMU just tested?
C770:D0 0B   C77D  230         bne  BIGLOOP   ;branch if IOU was just tested
C772:A0 09         231         ldy  #IOUIDX   ;else, go test IOU.
C774:D0 C2   C738  232         bne  swtst1    ;branch always taken...
C776:              233 *
C776:A2 00         234 swerr   ldx  #0        ;indicate switch error
C778:C0 0A         235         cpy  #IOUIDX+1 ;set carry if IOU was cause
C77A:4C D7 C6      236         jmp  bbits1
C77D:46 80         237 BIGLOOP lsr  $80
C77F:D0 B5   C736  238         bne  SWCHTST
C781:A9 A0         239 blp2    lda  #$A0
C783:A0 00         240         ldy  #0
C785:99 00 04      241 blp3    sta  $400,y    ;clear screen for success message
C788:99 00 05      242         sta  $500,y
C78B:99 00 06      243         sta  $600,y
C78E:99 00 07      244         sta  $700,y
C791:C8            245         iny
```

```
C792:D0 F1   C785  246          bne   blp3
C794:AD 61 C0      247 blp4     LDA   $C061      ;test for both Open and Closed Apple
C797:2D 62 C0      248          AND   $C062      ; pressed
C79A:0A            249          asl   a          ;put result in carry
C79B:E6 FF         250          INC   $FF
C79D:A5 FF         251          LDA   $FF
C79F:90 03   C7A4  252          bcc   dquit
C7A1:4C 00 C6      253          jmp   DIAGS
C7A4:              254 *
C7A4:AD 51 C0      255 dquit    lda   TEXT       ;put success message on the screen
C7A7:A0 08         256          ldy   #8
C7A9:B9 F6 C7      257 suc2     lda   success,y
C7AC:99 B8 05      258          sta   SCREEN,y
C7AF:88            259          dey
C7B0:10 F7   C7A9  260          bpl   suc2
C7B2:30 E0   C794  261          bmi   blp4       ;loop forever
C7B4:              262 *
C7B4:        C7B4  263 setv     equ   *
C7B4:53 43 2B 29   264 ntbl     dfb   83,67,43,41,7
C7B9:00 89 31 03   265 swtbl0   dfb $00,$89,$31,$03,$05,$09,$0b,$01,$00,$83,$51,$53,$55,$57,$0F, $0D
C7C9:00 81 31 04   266 swtbl1   dfb $00,$81,$31,$04,$06,$0A,$0C,$02,$00,$84,$52,$54,$56,$58,$10, $0E
C7D9:00 11 FF 13   267 rswtbl   dfb $00,$11,$FF,$13,$14,$16,$17,$18,$00,$12,$1A,$1B,$1C,$1D,$1E, $1F,$00
C7EA:              268          MSB   ON
C7EA:D2 C1 CD A0   269 rmess    asc   "RAM       ZP"
C7F0:CD CD D5 C9   270 smess    asc   "MMUIOU"


C7F6:D3 F9 F3 F4   272 success asc   "System   OK"
C7FF:        C7FF  273 zzzend   equ   *
C7FF:               22          INCLUDE C8SPACE
C7FF:        0001    1          DS    C80RG-*,0 ;pad to C800
C800:                2 *
C800:                3 * This entry point is only used by Pascal 1.0
C800:                4 *
C800:4C B0 C9        5          JMP   PINIT1.0  ;PASCAL 1.0 INIT
C803:                6 *
C803:                7 * BASIC initialization:
C803:                8 *
C803:                9 * This is called by the $C3 space only after a PR#3 or
C803:               10 * the equivalent (a JSR $C300).
C803:               11 *
C803:               12 * It causes a copy of the $F8 ROM to be placed in the
C803:               13 * language card if the language card is switched in and
C803:               14 * the ID byte doesn't match.  It sets up all the
C803:               15 * screenhole variables to support its operation.  If the
C803:               16 * 80 column card is detected, it sets things up for 80 column
C803:               17 * operation, else 40 column operation.  Then it clears the
C803:               18 * screen and prints the character that was in the accumulator
C803:               19 * upon entry.
C803:               20 *
C803:        C803   21 BASICINIT EQU *
C803:20 F4 CE       22          JSR  COPYROM   ;If LC in, copy F8 to it
C806:20 2A C8       23          JSR  C3HOOKS   ;out=$C307, in=$C305
C809:20 2E CD       24          JSR  D040      ;set full 40-col window
```

```
C80C:A9 01          25          LDA  #M.MOUSE  ;init with mouse text off
C80E:8D FB 04       26          STA  MODE      ;Set BASIC video mode
C811:               27 *
C811:               28 * IS THERE A CARD?
C811:               29 *
C811:20 90 CA       30          JSR  TESTCARD  ;SEE IF CARD PLUGGED IN
C814:D0 08   C81E   31          BNE  CLEARIT   ;=>IT'S 40
C816:06 21          32          ASL  WNDWDTH   ;SET 80-COL WINDOW
C818:8D 01 C0       33          STA  SET80COL  ;ENABLE 80 STORE
C81B:8D 0D C0       34          STA  SET80VID  ; AND 80 VIDEO
C81E:               35 *
C81E:               36 * HOME & CLEAR:
C81E:               37 *
C81E:        C81E   38 CLEARIT EQU  *
C81E:8D 0F C0       39          STA  SETALTCHAR ;SET NORM/INV LCASE
C821:20 90 CC       40          JSR  X.FF      ;CLEAR IT
C824:AC 7B 05       41          LDY  OURCH     ;set up cursor for store
C827:4C 7E C8       42          JMP  BPRINT    ;always print a character
C82A:               43 *
C82A:A9 07          44 C3HOOKS LDA  #>BASICOUT ;set output hook first
C82C:85 36          45          STA  CSWL
C82E:A9 C3          46          LDA  #<CN00
C830:85 37          47          STA  CSWH
C832:               48 *
C832:               49 * C3IN is called by IN#0 if CSWH = #$C3
C832:               50 *
C832:A9 05          51 C3IN    LDA  #>BASICIN ;set input hook
C834:85 38          52          STA  KSWL
C836:A9 C3          53          LDA  #<CN00
C838:85 39          54          STA  KSWH
C83A:60             55          RTS            ;exit with A=$C3 for IN#0 stuff
C83B:               56 *
C83B:E6 4E          57 GETKEY  INC  RNDL      ;BUMP RANDOM SEED
C83D:D0 02   C841   58          BNE  GETK2
C83F:E6 4F          59          INC  RNDH
C841:AD 00 C0       60 GETK2   LDA  KBD       ;KEYPRESS?
C844:10 F5   C83B   61          BPL  GETKEY    ;=>NOPE
C846:8D 10 C0       62          STA  KBDSTRB   ;CLEAR STROBE
C849:60             63          RTS
C84A:               64 *
C84A:               65 ****************************************
C84A:               66 *
C84A:               67 * PASCAL 1.0 INPUT HOOK:
C84A:               68 *
C84A:        0003   69          DS   C80RG+$4D-*,0 ;pad to 1.0 hooks
C84D:        0000   70          IFNE *-C80RG-$4D ;ERR IF WRONG ADDR
 S                  71          FAIL 2,'C84D   HOOK ALIGNMENT'
C84D:               72          FIN
C84D:4C 50 C3       73          JMP  JPREAD    ;=>GO TO STANDARD READ
C850:               74 ****************************************
C850:               75 *
C850:               76 * CSETUP compensates for everything that the user
C850:               77 * can do to change the cursor status: poke CV, CH,
C850:               78 * OURCH, WNDWDTH.  It updates the video firmware's
```

```
C850:                 79 * versions of these values for its own use.
C850:                 80 * COPY USER'S CURSOR IF IT DIFFERS FROM
C850:                 81 *  WHAT WE LAST PUT THERE:
C850:                 82 *
C850:A5 25            83 CSETUP  LDA  CV         ;set up OURCV
C852:8D FB 05         84         STA  OURCV
C855:A4 24            85         LDY  CH         ;GET IT
C857:CC 7B 04         86         CPY  OLDCH      ;IS IT THE SAME?
C85A:F0 03   C85F     87         BEQ  CS2        ;=>YES, USE OUR OWN
C85C:8C 7B 05         88         STY  OURCH      ;update our cursor
C85F:A5 21            89 CS2     LDA  WNDWDTH    ;cursor horizontal must not
C861:18               90         CLC             ;be greater than window width
C862:ED 7B 05         91         SBC  OURCH      ;if it is, then put cursor
C865:B0 05   C86C     92         BCS  CS3        ;at left edge of window
C867:A0 00            93         LDY  #0
C869:8C 7B 05         94         STY  OURCH
C86C:AC 7B 05         95 CS3     LDY  OURCH      ;exit with Y = CH
C86F:60               96         RTS
C870:                 97 *
C870:                 98 * BIN and BOUT are used when characters are
C870:                 99 * input and output by the $F8 ROM while 80VID
C870:                100 * is on.  They cannot use the $C3 entry points
C870:                101 * because that switches in the $C8 space, causing
C870:                102 * possible conflict with other $C8 users.
C870:                103 * These routines are only called by the $C100-$C2FF space.
C870:                104 *
C870:                105 * These entry points will only work if the card was
C870:                106 * first initialized using a PR#3.  80 columns will not
C870:                107 * work simply by turning on the 80VID flag.
C870:                108 *
C870:A4 35           109 BOUT   LDY  SAVY1      ;load Y stuffed by $F8 ROM
C872:18              110        CLC             ;signal an output
C873:B0 FE   C873    111        BCS  *          ;skip SEC
C874:        C874    112        ORG  *-1
C874:38              113 BIN    SEC             ;signal an input
C875:8D 7B 06        114        STA  CHAR       ;save the char
C878:98              115        TYA             ;save Y
C879:48              116        PHA
C87A:8A              117        TXA             ;save X
C87B:48              118        PHA
C87C:        C87C    119 C8BASIC EQU *          ;BASIC IN/OUT
C87C:B0 5E   C8DC    120        BCS  BINPUT     ;=>input a character
0000:        0000      1 TEST   EQU  0          ;REAL VERSION
C87E:                 23        LST  ON,A,V
C87E:                 24        INCLUDE BPRINT
C87E:                  1 *
C87E:                  2 * This is the place where characters printed using the
C87E:                  3 * CSW hook are actually printed (or executed if they are
C87E:                  4 * control characters).
C87E:                  5 *
C87E:20 50 C8          6 BPRINT  JSR  CSETUP    ;setup user cursor
C881:AD 7B 06          7        LDA  CHAR       ;GET CHARACTER
C884:C9 8D             8        CMP  #$8D       ;IS IT C/R?
C886:D0 18   C8A0      9        BNE  NOWAIT     ;=>don't wait, OURCH ok
```

```
C888:AE 00 C0       10          LDX  KBD        ;IS KEY PRESSED?
C88B:10 13   C8A0   11          BPL  NOWAIT     ;NO
C88D:E0 93          12          CPX  #$93       ;IS IT CTL-S?
C88F:D0 0F   C8A0   13          BNE  NOWAIT     ;NO, IGNORE IT
C891:2C 10 C0       14          BIT  KBDSTRB    ;CLEAR STROBE
C894:AE 00 C0       15 KBDWAIT  LDX  KBD        ;WAIT FOR NEXT KEYPRESS
C897:10 FB   C894   16          BPL  KBDWAIT
C899:E0 83          17          CPX  #$83       ;IF CTL-C, LEAVE IT
C89B:F0 03   C8A0   18          BEQ  NOWAIT     ; IN THE KBD BUFFER
C89D:2C 10 C0       19          BIT  KBDSTRB    ;CLEAR OTHER CHARACTER
C8A0:29 7F          20 NOWAIT   AND  #$7F       ;drop possible hi bit
C8A2:C9 20          21          CMP  #$20       ;IS IT CONTROL CHAR?
C8A4:B0 06   C8AC   22          BCS  BPNCTL     ;=>NOPE
C8A6:20 D2 CA       23          JSR  CTLCHAR0   ;execute CTL if M.CTL ok
C8A9:4C BD C8       24          JMP  CTLON      ;=>enable ctl chrs
C8AC:               25 *
C8AC:               26 * NOT A CTL CHAR. PRINT IT.
C8AC:               27 *
C8AC:        C8AC   28 BPNCTL   EQU  *
C8AC:AD 7B 06       29          LDA  CHAR       ;get char (all 8 bits)
C8AF:20 38 CE       30          JSR  STORCHAR   ;and display it
C8B2:               31 *
C8B2:               32 * BUMP THE CURSOR HORIZONTAL:
C8B2:               33 *
C8B2:C8             34          INY             ;bump it
C8B3:8C 7B 05       35          STY  OURCH      ;are we past the
C8B6:C4 21          36          CPY  WNDWDTH    ; end of the line?
C8B8:90 03   C8BD   37          BCC  CTLON      ;=>NO, NO PROBLEM
C8BA:20 51 CB       38          JSR  X.CR       ;YES, DO C/R
C8BD:               39 *
C8BD:               40 * M.CTL is set by RDCHAR and cleared here, after each
C8BD:               41 * character is displayed.
C8BD:               42 *
C8BD:AD FB 04       43 CTLON    LDA  MODE       ;enable printing of control chars
C8C0:29 F7          44          AND  #255-M.CTL
C8C2:8D FB 04       45          STA  MODE
C8C5:AD 7B 05       46 BIORET   LDA  OURCH      ;get newest cursor position
C8C8:2C 1F C0       47          BIT  RD80VID    ;IN 80-MODE?
C8CB:10 02   C8CF   48          BPL  SETALL     ;=>no, set other cursors
C8CD:A9 00          49          LDA  #0         ;pin CH to 0 for 80 columns
C8CF:85 24          50 SETALL   STA  CH
C8D1:8D 7B 04       51          STA  OLDCH      ;REMEMBER THE SETTING
C8D4:68             52 GETREGS  PLA              ;RESTORE
C8D5:AA             53          TAX
C8D6:68             54          PLA              ;X AND Y
C8D7:A8             55          TAY
C8D8:AD 7B 06       56          LDA  CHAR
C8DB:60             57          RTS              ;RETURN TO BASIC
C8DC:               25          INCLUDE BINPUT
C8DC:                1 *
C8DC:                2 * BASIC input entry point called by entry point in the
C8DC:                3 * $C3 space.  This is the way things normally happen.
C8DC:                4 *
C8DC:A4 24           5 BINPUT   LDY  CH
```

```
C8DE:AD 7B 06        6          LDA  CHAR
C8E1:91 28           7          STA  (BASL),Y
C8E3:20 50 C8        8          JSR  CSETUP     ;get newest cursor
C8E6:20 26 CE        9 B.INPUT  JSR  INVERT     ;invert that char
C8E9:20 3B C8       10          JSR  GETKEY     ;GET A KEY
C8EC:8D 7B 06       11          STA  CHAR       ;SAVE IT
C8EF:20 26 CE       12          JSR  INVERT     ;REMOVE CURSOR
C8F2:A8             13          TAY             ;preserve acc.
C8F3:               14 *
C8F3:               15 * On pure input, an uninterpreted character code should
C8F3:               16 * be returned.  If M.CTL is set, however, escape functions
C8F3:               17 * are enabled, and CTL-U causes the character under the
C8F3:               18 * cursor to be picked up from the screen.
C8F3:               19 * M.CTL is set whenever a character is requested using
C8F3:               20 * RDCHAR in the $F8 ROM.
C8F3:               21 *
C8F3:AD FB 04       22          LDA  MODE       ;is escape mode enabled?
C8F6:29 08          23          AND  #M.CTL
C8F8:F0 CB   C8C5   24          BEQ  BIORET     ;=>no,return
C8FA:C0 8D          25          CPY  #$8D       ;was it a CR
C8FC:D0 08   C906   26          BNE  NOTACR     ;=>nope, not a CR
C8FE:AD FB 04       27          LDA  MODE
C901:29 F7          28          AND  #255-M.CTL ;else end of line...
C903:8D FB 04       29          STA  MODE       ; disable escape
C906:        C906   30 NOTACR   EQU  *
C906:C0 9B          31          CPY  #$9B       ;ESCAPE KEY?
C908:F0 11   C91B   32          BEQ  ESCAPING   ;=>YES IT IS
C90A:               33 *
C90A:               34 * Not an escape sequence.  Check for control-u.
C90A:               35 *
C90A:C0 95          36          CPY  #$95       ;is it control-U?
C90C:D0 B7   C8C5   37          BNE  BIORET     ;no, return to caller
C90E:AC 7B 05       38          LDY  OURCH      ;get horizontal position
C911:20 44 CE       39          JSR  PICK       ;and pick up the char
C914:09 80          40          ORA  #$80       ;always pick as normal
C916:8D 7B 06       41          STA  CHAR       ;save keystroke
C919:D0 AA   C8C5   42          BNE  BIORET     ;=>(always) return to caller
C91B:               43 *
C91B:               44 * Start an escape sequence.  If the next character
C91B:               45 * pressed is one of the following, it is executed.
C91B:               46 * Otherwise it is ignored.
C91B:               47 *
C91B:               48 *    @ - home & clear
C91B:               49 *    E - clear to end of line
C91B:               50 *    F - clear to end of screen
C91B:               51 *    I - move cursor up
C91B:               52 *    J - move cursor left
C91B:               53 *    K - move cursor right
C91B:               54 *    M - move cursor down
C91B:               57 *    4 - enter 40 column mode
C91B:               58 *    8 - enter 80 column mode
C91B:               59 * CTL-D- disable the printing of control characters
C91B:               60 * CTL-E- enable the printing of control characters
C91B:               61 * CTL-Q- quit (PR#0/IN#0)
```

```
C91B:                   62 *   The four arrow keys (as IJKM)
C91B:                   63 *
C91B:                   64          MSB  OFF
C91B:           C91B    65 ESCAPING EQU *
C91B:20 B1 CE           66          JSR  ESCON      ;ESCAPE CURSOR ON
C91E:20 3B C8           67          JSR  GETKEY     ;GET ESCAPE FUNCTION
C921:20 C4 CE           68          JSR  ESCOFF     ;REPLACE ORIGINAL CHARACTER
C924:20 14 CE           69          JSR  UPSHFT     ;upshift the char
C927:29 7F              70          AND  #$7F       ;DROP HI BIT
C929:A0 10              71          LDY  #ESCNUM-1  ;COUNT/INDEX
C92B:D9 7C C9           72 ESC2     CMP  ESCTAB,Y   ;IS IT A VALID ESCAPE?
C92E:F0 05   C935       73          BEQ  ESC3       ;=>YES
C930:88                 74          DEY
C931:10 F8   C92B       75          BPL  ESC2       ;TRY 'EM ALL...
C933:30 0F   C944       76          BMI  ESCSPEC    ;=>MAYBE IT'S A SPECIAL ONE
C935:                   77 *
C935:           C935    78 ESC3     EQU  *
C935:B9 6B C9           79          LDA  ESCCHAR,Y  ;GET CHAR TO "PRINT"
C938:29 7F              80          AND  #$7F       ;DROP HI BIT (FLAG)
C93A:20 D6 CA           81          JSR  CTLCHAR    ;EXECUTE IT
C93D:B9 6B C9           82          LDA  ESCCHAR,Y  ;GET FLAG
C940:30 D9   C91B       83          BMI  ESCAPING   ;=>STAY IN ESCAPE MODE
C942:10 A2   C8E6       84          BPL  B.INPUT    ;=>QUIT ESCAPE MODE
C944:                   85 *
C944:           C944    86 ESCSPEC EQU  *
C944:A8                 87          TAY             ;put char here
C945:AD FB 04           88          LDA  MODE       ;so we can put this here
C948:C0 11              89          CPY  #$11       ;was it Quit?
C94A:D0 0B   C957       90          BNE  ESCSP1     ;=>no
C94C:20 4D CD           91          JSR  X.NAK      ;do the quitting stuff
C94F:A9 98              92          LDA  #$98       ;make it look like
C951:8D 7B 06           93          STA  CHAR       ;CTL-X was pressed
C954:4C C5 C8           94          JMP  BIORET     ;=>quit the card forever
C957:                   95 *
C957:C0 05              96 ESCSP1   CPY  #$05       ;was it CTL-E for enable
C959:D0 08   C963       97          BNE  ESCSP4     ;=>no
C95B:29 DF              98          AND  #255-M.CTL2 ;yes, enable ctl chars
C95D:8D FB 04           99 ESCSP2   STA  MODE       ;save new mode
C960:4C E6 C8          100 ESCSP3   JMP  B.INPUT    ;=> exit escape mode
C963:                  101 *
C963:C0 04             102 ESCSP4   CPY  #$04       ;was it CTL-D for disable
C965:D0 F9   C960      103          BNE  ESCSP3     ;=>no, exit escape mode
C967:09 20             104          ORA  #M.CTL2    ;disable ctl chars
C969:D0 F2   C95D      105          BNE  ESCSP2     ;=> exit escape mode
C96B:                  106 *
C96B:                  107 * This table contains the control characters which,
C96B:                  108 * when executed, carry out the escape functions. If
C96B:                  109 * the high bit of the character is set, it means that
C96B:                  110 * escape mode should not be exited after execution of
C96B:                  111 * the character.
C96B:                  112 *
C96B:           C96B   113 ESCCHAR EQU  *
C96B:0C                114          DFB  $0C        ;@: FORMFEED
C96C:1C                115          DFB  $1C        ;A: FS
```

```
C96D:08             116          DFB  $08        ;B: BS
C96E:0A             117          DFB  $0A        ;C: LF
C96F:1F             118          DFB  $1F        ;D: US
C970:1D             119          DFB  $1D        ;E: GS
C971:0B             120          DFB  $0B        ;F: VT
C972:9F             121          DFB  $1F+$80    ;I: US (STAY ESC)
C973:88             122          DFB  $08+$80    ;J: BS (STAY ESC)
C974:9C             123          DFB  $1C+$80    ;K: FS (STAY ESC)
C975:8A             124          DFB  $0A+$80    ;M: LF (STAY ESC)
C976:11             125          DFB  $11        ;4 :DC1
C977:12             126          DFB  $12        ;8 :DC2
C978:88             127          DFB  $08+$80    ;<-:BS (STAY ESC)
C979:8A             128          DFB  $0A+$80    ;DN:LF (STAY ESC)
C97A:9F             129          DFB  $1F+$80    ;UP:US (STAY ESC)
C97B:9C             130          DFB  $1C+$80    ;->:FS (STAY ESC)
C97C:               131 *
C97C:               132          MSB  OFF        ;high bit already masked
C97C:        C97C   133 ESCTAB   EQU  *
C97C:40             134          ASC  '@'
C97D:41             135          ASC  'A'        ;HANDLE OLD ESCAPES
C97E:42             136          ASC  'B'
C97F:43             137          ASC  'C'
C980:44             138          ASC  'D'
C981:45             139          ASC  'E'
C982:46             140          ASC  'F'
C983:49             141          ASC  'I'
C984:4A             142          ASC  'J'
C985:4B             143          ASC  'K'
C986:4D             144          ASC  'M'
C987:34             145          ASC  '4'
C988:38             146          ASC  '8'
C989:08             147          DFB  $08        ;LEFT ARROW
C98A:0A             148          DFB  $0A        ;DOWN ARROW
C98B:0B             149          DFB  $0B        ;UP   ARROW
C98C:15             150          DFB  $15        ;RITE ARROW
C98D:        0011   151 ESCNUM   EQU  *-ESCTAB
C98D:               152          MSB  ON
C98D:               153 *
C98D:               154 * Tack on diag 128K test here
C98D:               155 *
C98D:2C 13 C0       156 STAUX    BIT  RDRAMRD    ;aux done yet?
C990:30 11   C9A3   157          BMI  XSTAUX     ;=>yes, exit
C992:A9 EE          158          LDA  #$EE       ;get test pattern
C994:8D 05 C0       159          STA  WRCARDRAM  ;write AUX RAM
C997:8D 03 C0       160          STA  RDCARDRAM  ;read AUX RAM
C99A:8D 00 0C       161          STA  $C00       ;test this byte
C99D:8D 00 08       162          STA  $800       ;and this is 1K off
C9A0:CD 00 0C       163          CMP  $C00       ;has $C00 been updated?
C9A3:60             164 XSTAUX   RTS             ;check in main diags.
C9A4:               165 *
C9A4:               166 * ESCOUT used by ESCFIX in $C1 page
C9A4:               167 *
C9A4:               168          MSB  ON
C9A4:CA CB CD C9    169 ESCOUT   ASC  'JKMI'     ;The arrows
```

```
C9A8:                170         MSB  OFF
C9A8:                 26         INCLUDE PASCAL
C9A8:                  1 ***************************************
C9A8:                  2 * PASCAL 1.0 OUTPUT HOOK:
C9A8:                  3 ***************************************
C9A8:          0002    4         DS   C80RG+$1AA-*,0
C9AA:          0000    5         IFNE *-C80RG-$1AA
 S                     6         FAIL 2,'C9AA   HOOK ALIGNMENT'
C9AA:                  7         FIN
C9AA:AD 7B 06          8         LDA  CHAR      ;GET OUTPUT CHARACTER
C9AD:4C 56 C3          9         JMP  JPWRITE   ;=>USE STANDARD WRITE
C9B0:                 10 ***************************************
C9B0:                 11 *
C9B0:                 12 ***************************************
C9B0:                 13 * PASCAL INITIALIZATION:
C9B0:                 14 * Disable printing of mouse text
C9B0:                 15 ***************************************
C9B0:          C9B0   16 PINIT1.0 EQU *
C9B0:A9 83            17         LDA  #M.PASCAL+M.PAS1.0+M.MOUSE
C9B2:D0 02    C9B6    18         BNE  PINIT2    ;=>always
C9B4:          C9B4   19 PINIT   EQU  *
C9B4:A9 81            20         LDA  #M.PASCAL+M.MOUSE ;SAY WE'RE
C9B6:                 21 *
C9B6:          C9B6   22 PINIT2  EQU  *
C9B6:48               23         PHA            ;save version ID
C9B7:                 24 *
C9B7:                 25 * SEE IF THE CARD'S PLUGGED IN:
C9B7:                 26 *
C9B7:20 90 CA         27         JSR  TESTCARD  ;IS IT THERE?
C9BA:F0 04    C9C0    28         BEQ  PIGOOD    ;=>YES
C9BC:68               29         PLA            ;discard ID byte
C9BD:A2 09            30         LDX  #9        ;IORESULT='NO DEVICE'
C9BF:60               31         RTS
C9C0:                 32 *
C9C0:          C9C0   33 PIGOOD  EQU  *
C9C0:68               34         PLA            ;get version ID
C9C1:8D FB 04         35         STA  MODE      ; and save it
C9C4:8D 01 C0         36         STA  SET80COL  ;ENABLE 80 STORE
C9C7:8D 0D C0         37         STA  SET80VID  ; AND 80 VIDEO
C9CA:8D 0F C0         38         STA  SETALTCHAR ;NORM+INV LCASE
C9CD:20 D4 CE         39         JSR  PSETUP    ;set window and cursor
C9D0:20 90 CC         40         JSR  X.FF      ;HOME & CLEAR IT
C9D3:4C 1F CA         41         JMP  DOBASL    ;fix OLDBASL/H, display cursor, exit
C9D6:                 42 ***************************************
C9D6:                 43 * PASCAL INPUT:
C9D6:                 44 *
C9D6:                 45 * Character always returned with high bit clear.
C9D6:                 46 *
C9D6:                 47 ***************************************
C9D6:          C9D6   48 PREAD   EQU  *
C9D6:20 D4 CE         49         JSR  PSETUP    ;SETUP ZP STUFF
C9D9:20 3B C8         50         JSR  GETKEY    ;GET A KEYSTROKE
C9DC:29 7F            51         AND  #$7F      ;DROP HI BIT
C9DE:8D 7B 06         52         STA  CHAR      ;SAVE THE CHAR
```

```
C9E1:A2 00          53          LDX  #0         ;IORESULT='GOOD'
C9E3:AD FB 04       54          LDA  MODE       ;ARE WE IN 1.0-MODE?
C9E6:29 02          55          AND  #M.PAS1.0
C9E8:F0 02   C9EC   56          BEQ  PREADRET2  ;=>NOPE
C9EA:A2 C3          57          LDX  #<CN00     ;YES, RETURN CN IN X
C9EC:               58 *
C9EC:        C9EC   59 PREADRET2 EQU *
C9EC:AD 7B 06       60          LDA  CHAR       ;RESTORE CHAR
C9EF:60             61          RTS
C9F0:               62 *
C9F0:               63 * PASCAL OUTPUT:
C9F0:               64 * Note:  to be executed, control characters must have
C9F0:               65 * their high bits cleared.  All other characters are
C9F0:               66 * displayed regardless of their high bits.
C9F0:               67 *
C9F0:        C9F0   68 PWRITE   EQU  *
C9F0:29 7F          69          AND  #$7F       ;clear high bits
C9F2:AA             70          TAX             ;save character
C9F3:20 D4 CE       71          JSR  PSETUP     ;SETUP ZP STUFF, don't set ROM
C9F6:A9 08          72          LDA  #M.GOXY    ;ARE WE DOING GOTOXY?
C9F8:2C FB 04       73          BIT  MODE
C9FB:D0 32   CA2F   74          BNE  GETX       ;=>Doing X or Y?
C9FD:8A             75          TXA             ;now check for control char
C9FE:2C 2E CA       76          BIT  PRTS       ;is it control?
CA01:F0 50   CA53   77          BEQ  PCTL       ;=>yes, do control
CA03:AC 7B 05       78          LDY  OURCH      ;get horizontal position
CA06:24 32          79          BIT  INVFLG     ;check for inverse
CA08:10 02   CA0C   80          BPL  PWR1       ;inverse, go store it
CA0A:09 80          81          ORA  #$80
CA0C:20 70 CE       82 PWR1     JSR  STORIT     ;now store it (erasing cursor)
CA0F:C8             83          INY             ;INC CH
CA10:8C 7B 05       84          STY  OURCH
CA13:C4 21          85          CPY  WNDWDTH
CA15:90 08   CA1F   86          BCC  DOBASL
CA17:A9 00          87          LDA  #0         ;do carriage return
CA19:8D 7B 05       88          STA  OURCH
CA1C:20 D8 CB       89          JSR  X.LF       ;and linefeed
CA1F:A5 28          90 DOBASL   LDA  BASL       ;save BASL for pascal
CA21:8D 7B 07       91          STA  OLDBASL
CA24:A5 29          92          LDA  BASH
CA26:8D FB 07       93          STA  OLDBASH
CA29:20 1F CE       94 PWRITERET JSR PASINV     ;display new cursor
CA2C:A2 00          95 PRET     LDX  #$0        ;return with no error
CA2E:60             96 PRTS     RTS
CA2F:               97 *
CA2F:               98 *  HANDLE GOTOXY STUFF:
CA2F:               99 *
CA2F:20 1F CE      100 GETX     JSR  PASINV     ;turn off cursor
CA32:8A            101          TXA             ;get character
CA33:38            102          SEC
CA34:E9 20         103          SBC  #32        ;MAKE BINARY
CA36:2C FB 06      104          BIT  XCOORD     ;doing X?
CA39:30 30   CA6B  105          BMI  PSETX      ;=>yes, set it
CA3B:              106 *
```

```
CA3B:                  107 * Set Y and do the GOTOXY
CA3B:                  108 *
CA3B:8D FB 05          109 GETY    STA  OURCV
CA3E:85 25             110         STA  CV
CA40:20 BA CA          111         JSR  BASCALC   ;calc base addr
CA43:AD FB 06          112         LDA  XCOORD
CA46:8D 7B 05          113         STA  OURCH     ;set cursor horizontal
CA49:A9 F7             114         LDA  #255-M.GOXY ;turn off gotoxy
CA4B:2D FB 04          115         AND  MODE
CA4E:8D FB 04          116         STA  MODE
CA51:D0 CC   CA1F      117         BNE  DOBASL    ;=>DONE (ALWAYS TAKEN)
CA53:                  118 *
CA53:20 1F CE          119 PCTL    JSR  PASINV    ;turn off cursor
CA56:8A                120         TXA            ;get char
CA57:C9 1E             121         CMP  #$1E      ;is it gotoXY?
CA59:F0 06   CA61      122         BEQ  STARTXY   ;=>yes, start it up
CA5B:20 D6 CA          123         JSR  CTLCHAR   ;EXECUTE IT IF POSSIBLE
CA5E:4C 1F CA          124         JMP  DOBASL    ;=>update BASL/H, cursor, exit
CA61:                  125 *
CA61:                  126 * START THE GOTOXY SEQUENCE:
CA61:                  127 *
CA61:        CA61      128 STARTXY EQU  *
CA61:A9 08             129         LDA  #M.GOXY
CA63:0D FB 04          130         ORA  MODE      ;turn on gotoxy
CA66:8D FB 04          131         STA  MODE
CA69:A9 FF             132         LDA  #$FF      ;set XCOORD to -1
CA6B:8D FB 06          133 PSETX   STA  XCOORD    ;set X
CA6E:4C 29 CA          134         JMP  PWRITERET ;=>display cursor and exit
CA71:                   27         INCLUDE SUBS1
CA71:        CA71       1 DOMN    EQU  *
CA71:AA                 2         TAX            ;SAVE IT
CA72:A5 2A              3         LDA  BAS2L     ;GET OPCODE AGAIN
CA74:A0 03              4         LDY  #$03
CA76:E0 8A              5         CPX  #$8A
CA78:F0 0B   CA85       6         BEQ  MNNDX3
CA7A:4A                 7 MNNDX1  LSR  A
CA7B:90 08   CA85       8         BCC  MNNDX3    ;FORM INDEX INTO MNEMONIC TABLE
CA7D:4A                 9         LSR  A
CA7E:4A                10 MNNDX2  LSR  A          ;  1) 1XXX1010 => 00101XXX
CA7F:09 20             11         ORA  #$20       ;  2) XXXYYY01 => 00111XXX
CA81:88                12         DEY             ;  3) XXXYYY10 => 00110XXX
CA82:D0 FA   CA7E      13         BNE  MNNDX2     ;  4) XXXYY100 => 00100XXX
CA84:C8                14         INY             ;  5) XXXXX000 => 000XXXXX
CA85:88                15 MNNDX3  DEY
CA86:D0 F2   CA7A      16         BNE  MNNDX1
CA88:60                17         RTS
CA89:                  18 *
CA89:                  19 * Switch in slot 3, then test for a ROM card.
CA89:                  20 * If none found, test for 80 column card,
CA89:                  21 * else return with BNE.
CA89:                  22 *
CA89:        CA89      23 TSTROMCRD EQU *
CA89:20 B7 F8          24         JSR  TSTROM    ;test for ROM card
CA8C:D0 02   CA90      25         BNE  TESTCARD  ;=>no ROM, check for 80 column card
```

```
CA8E:C8              26          INY             ;make BNE for return
CA8F:60              27          RTS
CA90:                28 *
CA90:                29 ****************************************
CA90:                30 * NAME     : TESTCARD
CA90:                31 * FUNCTION: SEE IF 80COL CARD PLUGGED IN
CA90:                32 * INPUT    : NONE
CA90:                33 * OUTPUT   : 'BEQ' IF CARD AVAILABLE
CA90:                34 *          : 'BNE' IF NOT
CA90:                35 * VOLATILE: AC,Y
CA90:                36 ****************************************
CA90:                37 *
CA90:         CA90   38 TESTCARD EQU *
CA90:AD 1C C0        39          LDA  RDPAGE2    ;REMEMBER CURRENT VIDEO DISPLAY
CA93:0A              40          ASL  A          ; IN THE CARRY
CA94:A9 88           41          LDA  #$88       ;USEFUL CHAR FOR TESTING
CA96:2C 18 C0        42          BIT  RD80COL    ;REMEMBER VIDEO MODE IN 'N'
CA99:8D 01 C0        43          STA  SET80COL   ;ENABLE 80COL STORE
CA9C:08              44          PHP             ;SAVE 'N' AND 'C' FLAGS
CA9D:8D 55 C0        45          STA  TXTPAGE2   ;SET PAGE2
CAA0:AC 00 04        46          LDY  $0400      ;GET FIRST CHAR
CAA3:8D 00 04        47          STA  $0400      ;SET TO A '*'
CAA6:AD 00 04        48          LDA  $0400      ;GET IT BACK FROM RAM
CAA9:8C 00 04        49          STY  $0400      ;RESTORE ORIG CHAR
CAAC:28              50          PLP             ;RESTORE 'N' AND 'C' FLAGS
CAAD:B0 03    CAB2   51          BCS  STAY2      ;STAY IN PAGE2
CAAF:8D 54 C0        52          STA  TXTPAGE1   ;RESTORE PAGE1
CAB2:         CAB2   53 STAY2    EQU  *
CAB2:30 03    CAB7   54          BMI  STAY80     ;=>STAY IN 80COL MODE
CAB4:8D 00 C0        55          STA  CLR80COL   ;TURN OFF 80COL STORE
CAB7:         CAB7   56 STAY80   EQU  *
CAB7:C9 88           57          CMP  #$88       ;WAS CHAR VALID?
CAB9:60              58          RTS             ;RETURN RESULT AS BEQ/BNE
CABA:                59 *
CABA:                60 * Do the
 normal monitor ROM BASCALC
CABA:                61 *
CABA:         CABA   62 BASCALC EQU  *
CABA:48              63          PHA
CABB:4A              64          LSR  A
CABC:29 03           65          AND  #$03
CABE:09 04           66          ORA  #$04
CAC0:85 29           67          STA  BASH
CAC2:68              68          PLA
CAC3:29 18           69          AND  #$18
CAC5:90 02    CAC9   70          BCC  BSCLC2
CAC7:69 7F           71          ADC  #$7F
CAC9:85 28           72 BSCLC2  STA  BASL
CACB:0A              73          ASL  A
CACC:0A              74          ASL  A
CACD:05 28           75          ORA  BASL
CACF:85 28           76          STA  BASL
CAD1:60              77          RTS
CAD2:                78 *
```

```
CAD2:                 79 ****************************************
CAD2:                 80 * NAME     : CTLCHAR0
CAD2:                 81 * FUNCTION: Execute CTL char if M.CTL=0
CAD2:                 82 * INPUT    : AC=CHAR
CAD2:                 83 * OUTPUT   : 'BCS' if not executed
CAD2:                 84 *          : 'BCC' if executed
CAD2:                 85 * VOLATILE: NOTHING
CAD2:                 86 * CALLS    : MANY THINGS
CAD2:                 87 ****************************************
CAD2:                 88 *
CAD2:2C 06 CB         89 CTLCHAR0 BIT SEV1       ;set V (use M.CTL)
CAD5:50 FE    CAD5    90          BVC  *         ;skip CLC
CAD6:         CAD6    91          ORG  *-1
CAD6:                 92 *
CAD6:                 93 ****************************************
CAD6:                 94 * NAME     : CTLCHAR
CAD6:                 95 * FUNCTION: Always execute CTL char
CAD6:                 96 * INPUT    : AC=CHAR
CAD6:                 97 * OUTPUT   : 'BCS' if not executed
CAD6:                 98 *          : 'BCC' if ctl executed
CAD6:                 99 * VOLATILE: NOTHING
CAD6:                100 * CALLS    : MANY THINGS
CAD6:                101 ****************************************
CAD6:                102 *
CAD6:B8              103 CTLCHAR CLV            ;clear V (ignore M.CTL)
CAD7:8D 7B 07        104         STA  TEMP1     ;TEMP SAVE OF CHAR
CADA:48              105         PHA            ;SAVE AC
CADB:98              106         TYA            ;SAVE Y
CADC:48              107         PHA
CADD:                108 *
CADD:AC 7B 07        109         LDY  TEMP1     ;GET CHAR IN QUESTION
CAE0:C0 05           110         CPY  #$05      ;IS IT NUL..EOT?
CAE2:90 13    CAF7   111         BCC  CTLCHARX  ;=>YES, NOT USED
CAE4:B9 B4 CB        112         LDA  CTLADH-5,Y ;Get high byte of address
CAE7:F0 0E    CAF7   113         BEQ  CTLCHARX  ;=>ctl not implemented
CAE9:50 12    CAFD   114         BVC  CTLGOO    ;=> CLTCHAR: always execute
CAEB:                115 *
CAEB:         0000   116         DO   TEST
 S                   117         BPL  CTLGOO    ;=>CR,BEL,LF,BS always done
CAEB:                118         ELSE
CAEB:30 10    CAFD   119         BMI  CTLGOO    ;=>CR,BEL,LF,BS always done
CAED:                120         FIN
CAED:                121 *
CAED:8D 7B 07        122         STA  TEMP1     ;save high byte of address
CAF0:AD FB 04        123         LDA  MODE      ;if control chars
CAF3:29 28           124         AND  #M.CTL+M.CTL2 ;are enabled
CAF5:F0 03    CAFA   125         BEQ  CTLGO     ;=>then go do them
CAF7:                126 *
CAF7:         CAF7   127 CTLCHARX EQU *
CAF7:38              128         SEC            ;SAY 'NOT CTL'
CAF8:B0 09    CB03   129         BCS  CTLRET    ;=>DONE
CAFA:                130 *
CAFA:AD 7B 07        131 CTLGO   LDA  TEMP1     ;get address back
CAFD:         CAFD   132 CTLGOO  EQU  *
```

```
CAFD:         0000  133          DO   TEST
 S                  134          AND  #$7F       ;for test, hi bit clear
CAFD:               135          ELSE
CAFD:09 80          136          ORA  #$80       ;hi bit always set
CAFF:               137          FIN
CAFF:20 07 CB       138          JSR  CTLXFER    ;EXECUTE SUBROUTINE
CB02:               139 *
CB02:18             140          CLC             ;SAY 'CTL CHAR EXECUTED'
CB03:         CB03  141 CTLRET   EQU  *
CB03:68             142          PLA             ;RESTORE
CB04:A8             143          TAY             ; Y
CB05:68             144          PLA             ;  AND AC
CB06:60             145 SEV1     RTS
CB07:               146 *
CB07:         CB07  147 CTLXFER  EQU  *
CB07:48             148          PHA             ;PUSH ONTO STACK FOR
CB08:B9 99 CB       149          LDA  CTLADL-5,Y ; TRANSFER TRICK
CB0B:48             150          PHA
CB0C:60             151          RTS             ;XFER TO ROUTINE
CB0D:               152 *
CB0D:               153 * Turn cursor on for Pascal only
CB0D:               154 *
CB0D:AD FB 04       155 X.CUR.ON LDA MODE        ;get mode byte
CB10:10 05    CB17  156          BPL  CURON.X    ;=>not pascal, don't do it
CB12:29 EF          157          AND  #255-M.CURSOR ;clear cursor bit
CB14:8D FB 04       158 SAVCUR   STA  MODE       ;save it
CB17:60             159 CURON.X  RTS             ;and exit
CB18:               160 *
CB18:               161 * Turn cursor off for Pascal only.
CB18:               162 * Cursor is not displayed during call.
CB18:               163 *
CB18:AD FB 04       164 X.CUR.OFF LDA MODE       ;get mode byte
CB1B:10 FA    CB17  165          BPL  CURON.X    ;=>not pascal, don't do it
CB1D:09 10          166          ORA  #M.CURSOR  ;turn on cursor bit
CB1F:D0 F3    CB14  167          BNE  SAVCUR     ;save and exit
CB21:               168 *
CB21:               169 * EXECUTE BELL:
CB21:               170 *
CB21:         CB21  171 X.BELL   EQU  *
CB21:A9 40          172          LDA  #$40       ;RIPPED OFF FROM MONITOR
CB23:20 34 CB       173          JSR  WAIT
CB26:A0 C0          174          LDY  #$C0
CB28:A9 0C          175 BELL2    LDA  #$0C
CB2A:20 34 CB       176          JSR  WAIT
CB2D:AD 30 C0       177          LDA  SPKR
CB30:88             178          DEY
CB31:D0 F5    CB28  179          BNE  BELL2
CB33:60             180          RTS
CB34:               181 *
CB34:         CB34  182 WAIT     EQU  *          ;RIPPED OFF FROM MONITOR ROM
CB34:38             183          SEC
CB35:48             184 WAIT2    PHA
CB36:E9 01          185 WAIT3    SBC  #1
CB38:D0 FC    CB36  186          BNE  WAIT3
```

```
CB3A:68             187          PLA
CB3B:E9 01          188          SBC  #1
CB3D:D0 F6   CB35   189          BNE  WAIT2
CB3F:60             190          RTS
CB40:               191 *
CB40:               192 * EXECUTE BACKSPACE:
CB40:               193 *
CB40:        CB40   194 X.BS      EQU  *
CB40:CE 7B 05       195          DEC  OURCH      ;BACK UP CH
CB43:10 0B   CB50   196          BPL  BSDONE     ;=>DONE
CB45:A5 21          197          LDA  WNDWDTH    ;BACK UP TO PRIOR LINE
CB47:8D 7B 05       198          STA  OURCH      ;SET CH
CB4A:CE 7B 05       199          DEC  OURCH
CB4D:20 79 CB       200          JSR  X.US       ;NOW DO REV LINEFEED
CB50:        CB50   201 BSDONE    EQU  *
CB50:60             202          RTS
CB51:               203 *
CB51:               204 * EXECUTE CARRIAGE RETURN:
CB51:               205 *
CB51:        CB51   206 X.CR      EQU  *
CB51:A9 00          207          LDA  #0         ;BACK UP CH TO
CB53:8D 7B 05       208          STA  OURCH      ; BEGINNING OF LINE
CB56:AD FB 04       209          LDA  MODE       ;ARE WE IN BASIC?
CB59:30 03   CB5E   210          BMI  X.CRRET    ;=> Pascal, avoid auto LF
CB5B:20 D8 CB       211          JSR  X.LF       ;EXECUTE AUTO LF FOR BASIC
CB5E:        CB5E   212 X.CRRET   EQU  *
CB5E:60             213          RTS
CB5F:               214 *
CB5F:               215 * EXECUTE HOME:
CB5F:               216 *
CB5F:        CB5F   217 X.EM      EQU  *
CB5F:A5 22          218          LDA  WNDTOP
CB61:85 25          219          STA  CV
CB63:A9 00          220          LDA  #0
CB65:8D 7B 05       221          STA  OURCH      ;STUFF CH
CB68:4C FE CD       222          JMP  VTAB       ;set base for OURCV
CB6B:               223 *
CB6B:               224 * EXECUTE FORWARD SPACE:
CB6B:               225 *
CB6B:        CB6B   226 X.FS      EQU  *
CB6B:EE 7B 05       227          INC  OURCH      ;BUMP CH
CB6E:AD 7B 05       228          LDA  OURCH      ;GET THE POSITION
CB71:C5 21          229          CMP  WNDWDTH    ;OFF THE RIGHT SIDE?
CB73:90 03   CB78   230          BCC  X.FSRET    ;=>NO, GOOD
CB75:20 51 CB       231          JSR  X.CR       ;=>YES, WRAP AROUND
CB78:               232 *
CB78:        CB78   233 X.FSRET   EQU  *
CB78:60             234          RTS
CB79:               235 *
CB79:               236 * EXECUTE REVERSE LINEFEED:
CB79:               237 *
CB79:A5 22          238 X.US      LDA  WNDTOP     ;are we at top?
CB7B:C5 25          239          CMP  CV
CB7D:B0 1E   CB9D   240          BCS  X.USRET    ;=>yes, stay there
```

```
CB7F:C6 25          241          DEC  CV         ;else go up a line
CB81:4C FE CD       242          JMP  VTAB       ;exit thru VTAB (update OURCV)
CB84:               243 *
CB84:               244 * EXECUTE "NORMAL VIDEO"
CB84:               245 *
CB84:         CB84  246 X.SO     EQU  *
CB84:AD FB 04       247          LDA  MODE       ;SET MODE BIT
CB87:10 02    CB8B  248          BPL  X.SO1      ;don't set mode for BASIC
CB89:29 FB          249          AND  #255-M.VMODE ;SET 'NORMAL'
CB8B:A0 FF          250 X.SO1    LDY  #255
CB8D:D0 09    CB98  251          BNE  STUFFINV   ;(ALWAYS)
CB8F:               252 *
CB8F:               253 * EXECUTE "INVERSE VIDEO"
CB8F:               254 *
CB8F:         CB8F  255 X.SI     EQU  *
CB8F:AD FB 04       256          LDA  MODE       ;SET MODE BIT
CB92:10 02    CB96  257          BPL  X.SI1      ;don't set mode for BASIC
CB94:09 04          258          ORA  #M.VMODE   ;SET 'INVERSE'
CB96:A0 7F          259 X.SI1    LDY  #127
CB98:8D FB 04       260 STUFFINV STA  MODE       ;SET MODE
CB9B:84 32          261          STY  INVFLG     ;STUFF FLAG TOO
CB9D:60             262 X.USRET RTS
CB9E:               263 *
CB9E:         CB9E  264 CTLADL   EQU  *
CB9E:0C             265          DFB  #>X.CUR.ON-1 ;ENQ
CB9F:17             266          DFB  #>X.CUR.OFF-1 ;ACK
CBA0:20             267          DFB  #>X.BELL-1 ;BEL
CBA1:3F             268          DFB  #>X.BS-1   ;BS
CBA2:00             269          DFB  0          ;HT
CBA3:D7             270          DFB  #>X.LF-1   ;LF
CBA4:73             271          DFB  #>X.VT-1   ;VT
CBA5:8F             272          DFB  #>X.FF-1   ;FF
CBA6:50             273          DFB  #>X.CR-1   ;CR
CBA7:83             274          DFB  #>X.SO-1   ;SO
CBA8:8E             275          DFB  #>X.SI-1   ;SI
CBA9:00             276          DFB  0          ;DLE
CBAA:E9             277          DFB  #>X.DC1-1  ;DC1
CBAB:FB             278          DFB  #>X.DC2-1  ;DC2
CBAC:00             279          DFB  0          ;DC3
CBAD:00             280          DFB  0          ;DC4
CBAE:4C             281          DFB  #>X.NAK-1  ;NAK
CBAF:D3             282          DFB  #>SCROLLDN-1 ;SYN
CBB0:EA             283          DFB  #>SCROLLUP-1 ;ETB
CBB1:3C             284          DFB  #>MOUSEOFF-1
CBB2:5E             285          DFB  #>X.EM-1   ;EM
CBB3:95             286          DFB  #>X.SUB-1  ;SUB
CBB4:43             287          DFB  #>MOUSEON-1
CBB5:6A             288          DFB  #>X.FS-1   ;FS
CBB6:99             289          DFB  #>X.GS-1   ;GS
CBB7:00             290          DFB  0          ;RS
CBB8:78             291          DFB  #>X.US-1   ;US
CBB9:               292 *
CBB9:         CBB9  293 CTLADH   EQU  *
CBB9:4B             294          DFB  #<X.CUR.ON-$8001 ;ENQ
```

```
CBBA:4B             295          DFB  #<X.CUR.OFF-$8001 ;ACK
CBBB:CB             296          DFB  #<X.BELL-1 ;BEL
CBBC:CB             297          DFB  #<X.BS-1   ;BS
CBBD:00             298          DFB  0          ;HT
CBBE:CB             299          DFB  #<X.LF-1   ;LF
CBBF:4C             300          DFB  #<X.VT-$8001 ;VT
CBC0:4C             301          DFB  #<X.FF-$8001 ;FF
CBC1:CB             302          DFB  #<X.CR-1   ;CR
CBC2:4B             303          DFB  #<X.SO-$8001 ;SO
CBC3:4B             304          DFB  #<X.SI-$8001 ;SI
CBC4:00             305          DFB  0          ;DLE
CBC5:4C             306          DFB  #<X.DC1-$8001 ;DC1
CBC6:4C             307          DFB  #<X.DC2-$8001 ;DC2
CBC7:00             308          DFB  0          ;DC3
CBC8:00             309          DFB  0          ;DC4
CBC9:4D             310          DFB  #<X.NAK-$8001 ;NAK
CBCA:4B             311          DFB  #<SCROLLDN-$8001 ;SYN
CBCB:4B             312          DFB  #<SCROLLUP-$8001 ;ETB
CBCC:4D             313          DFB  #<MOUSEOFF-$8001
CBCD:4B             314          DFB  #<X.EM-$8001 ;EM
CBCE:4C             315          DFB  #<X.SUB-$8001 ;SUB
CBCF:4D             316          DFB  #<MOUSEON-$8001
CBD0:4B             317          DFB  #<X.FS-$8001 ;FS
CBD1:4C             318          DFB  #<X.GS-$8001 ;GS
CBD2:00             319          DFB  0          ;RS
CBD3:4B             320          DFB  #<X.US-$8001 ;US
CBD4:                28          INCLUDE SUBS2
CBD4:                1 *
CBD4:                2 * SCROLLIT scrolls the screen either up or down, depending
CBD4:                3 * on the value of X.  It scrolls within windows with even
CBD4:                4 * or odd edges for both 40 and 80 columns.  It can scroll
CBD4:                5 * windows down to 1 characters wide.
CBD4:                6 *
CBD4:A0 00           7 SCROLLDN LDY #0        ;direction = down
CBD6:F0 15   CBED    8          BEQ  SCROLLIT ;=>go do scroll
CBD8:                9 *
CBD8:               10 * EXECUTE LINEFEED:
CBD8:               11 *
CBD8:        CBD8   12 X.LF     EQU  *
CBD8:E6 25          13          INC  CV
CBDA:A5 25          14          LDA  CV        ;SEE IF OFF BOTTOM
CBDC:8D FB 05       15          STA  OURCV
CBDF:C5 23          16          CMP  WNDBTM    ;OFF THE END?
CBE1:B0 03   CBE6   17          BCS  X.LF2     ;=>yes, scroll screen
CBE3:4C 03 CE       18          JMP  VTABZ     ;exit thru VTABZ
CBE6:               19 *
CBE6:        CBE6   20 X.LF2    EQU  *
CBE6:CE FB 05       21          DEC  OURCV     ;back up to bottom
CBE9:C6 25          22          DEC  CV        ;and fall into scroll
CBEB:               23 *
CBEB:A0 01          24 SCROLLUP LDY #1         ;direction = up
CBED:8A             25 SCROLLIT TXA            ;save X
CBEE:48             26          PHA
CBEF:8C 7B 07       27          STY  TEMP1     ;save direction
```

```
CBF2:A5 21          28          LDA  WNDWDTH   ;get width of screen window
CBF4:48             29          PHA            ;save original width
CBF5:2C 1F C0       30          BIT  RD80VID   ;in 40 or 80 columns?
CBF8:10 1C   CC16   31          BPL  GETST1    ;=>40, determine starting line
CBFA:8D 01 C0       32          STA  SET80COL  ;make sure this is enabled
CBFD:4A             33          LSR  A         ;divide by 2 for 80 column index
CBFE:AA             34          TAX            ;and save
CBFF:A5 20          35          LDA  WNDLFT    ;test oddity of right edge
CC01:4A             36          LSR  A         ;by rotating low bit into carry
CC02:B8             37          CLV            ;V=0 if left edge even
CC03:90 03   CC08   38          BCC  CHKRT     ;=>check right edge
CC05:2C 06 CB       39          BIT  SEV1      ;V=1 if left edge odd
CC08:2A             40 CHKRT    ROL  A         ;restore WNDLFT
CC09:45 21          41          EOR  WNDWDTH   ;get oddity of right edge
CC0B:4A             42          LSR  A         ;C=1 if right edge even
CC0C:70 03   CC11   43          BVS  GETST     ;if odd left, don't DEY
CC0E:B0 01   CC11   44          BCS  GETST     ;if even right, don't DEY
CC10:CA             45          DEX            ;if right edge odd, need one less
CC11:86 21          46 GETST    STX  WNDWDTH   ;save window width
CC13:AD 1F C0       47          LDA  RD80VID   ;N=1 if 80 columns
CC16:08             48 GETST1   PHP            ;save N,Z,V
CC17:A6 22          49          LDX  WNDTOP    ;assume scroll from top
CC19:98             50          TYA            ;up or down?
CC1A:D0 03   CC1F   51          BNE  SETDBAS   ;=>up
CC1C:A6 23          52          LDX  WNDBTM    ;down, start scrolling at bottom
CC1E:CA             53          DEX            ;really need one less
CC1F:               54 *
CC1F:8A             55 SETDBAS  TXA            ;get current line
CC20:20 03 CE       56          JSR  VTABZ     ;calculate base with window width
CC23:               57 *
CC23:A5 28          58 SCRLIN   LDA  BASL      ;current line is destination
CC25:85 2A          59          STA  BAS2L
CC27:A5 29          60          LDA  BASH
CC29:85 2B          61          STA  BAS2H
CC2B:               62 *
CC2B:AD 7B 07       63          LDA  TEMP1     ;test direction
CC2E:F0 32   CC62   64          BEQ  SCRLDN    ;=>do the downer
CC30:E8             65          INX            ;do next line
CC31:E4 23          66          CPX  WNDBTM    ;done yet?
CC33:B0 32   CC67   67          BCS  SCRLL3    ;=>yup, all done
CC35:8A             68 SETSRC   TXA            ;set new line
CC36:20 03 CE       69          JSR  VTABZ     ;get base for new current line
CC39:A4 21          70          LDY  WNDWDTH   ;get width for scroll
CC3B:28             71          PLP            ;get status for scroll
CC3C:08             72          PHP            ;N=1 if 80 columns
CC3D:10 1E   CC5D   73          BPL  SKPRT     ;=>only do 40 columns
CC3F:AD 55 C0       74          LDA  TXTPAGE2  ;scroll aux page first (even bytes)
CC42:98             75          TYA            ;test Y
CC43:F0 07   CC4C   76          BEQ  SCRLFT    ;if Y=0, only scroll one byte
CC45:B1 28          77 SCRLEVEN LDA  (BASL),Y
CC47:91 2A          78          STA  (BAS2L),Y
CC49:88             79          DEY
CC4A:D0 F9   CC45   80          BNE  SCRLEVEN  ;do all but last even byte
CC4C:70 04   CC52   81 SCRLFT   BVS  SKPLFT    ;odd left edge, skip this byte
```

```
CC4E:B1 28          82          LDA  (BASL),Y
CC50:91 2A          83          STA  (BAS2L),Y
CC52:AD 54 C0       84 SKPLFT   LDA  TXTPAGE1   ;now do main page (odd bytes)
CC55:A4 21          85          LDY  WNDWDTH    ;restore width
CC57:B0 04   CC5D   86          BCS  SKPRT      ;even right edge, skip this byte
CC59:B1 28          87 SCRLODD  LDA  (BASL),Y
CC5B:91 2A          88          STA  (BAS2L),Y
CC5D:88             89 SKPRT    DEY
CC5E:10 F9   CC59   90          BPL  SCRLODD
CC60:30 C1   CC23   91          BMI  SCRLIN     ;=> always scroll next line
CC62:               92 *
CC62:CA             93 SCRLDN   DEX             ;do next line
CC63:E4 22          94          CPX  WNDTOP     ;done yet
CC65:10 CE   CC35   95          BPL  SETSRC     ;=>nope, not yet
CC67:               96 *
CC67:28             97 SCRLL3   PLP             ;pull status off stack
CC68:68             98          PLA             ;restore window width
CC69:85 21          99          STA  WNDWDTH
CC6B:20 96 CC      100          JSR  X.SUB      ;clear current line
CC6E:20 FE CD      101          JSR  VTAB       ;restore original cursor line
CC71:68            102          PLA             ;and X
CC72:AA            103          TAX
CC73:60            104          RTS             ;done!!!
CC74:              105 *
CC74:              106 * EXECUTE CLR TO EOS:
CC74:              107 *
CC74:20 9A CC      108 X.VT     JSR  X.GS       ;CLEAR TO EOL
CC77:A5 25         109          LDA  CV         ;SAVE CV
CC79:48            110          PHA
CC7A:10 06   CC82  111          BPL  X.VTNEXT   ;DO NEXT LINE (ALWAYS TAKEN)
CC7C:20 03 CE      112 X.VTLOOP JSR VTABZ      ;set base address
CC7F:20 96 CC      113          JSR  X.SUB      ;CLEAR LINE
CC82:E6 25         114 X.VTNEXT INC CV
CC84:A5 25         115          LDA  CV
CC86:C5 23         116          CMP  WNDBTM     ;OFF SCREEN?
CC88:90 F2   CC7C  117          BCC  X.VTLOOP   ;=>NO, KEEP GOING
CC8A:68            118          PLA             ;RESTORE
CC8B:85 25         119          STA  CV         ; CV
CC8D:4C FE CD      120          JMP  VTAB       ;return via VTAB (blech)
CC90:              121 *
CC90:              122 * EXECUTE CLEAR:
CC90:              123 *
CC90:        CC90  124 X.FF     EQU  *
CC90:20 5F CB      125          JSR  X.EM       ;HOME THE CURSOR
CC93:4C 74 CC      126          JMP  X.VT       ;RETURN VIA CLREOS (UGH!)
CC96:              127 *
CC96:              128 * EXECUTE CLEAR LINE
CC96:              129 *
CC96:A0 00         130 X.SUB    LDY  #0         ;start at left
CC98:F0 03   CC9D  131          BEQ  X.GSEOLZ   ;and clear to end of line
CC9A:              132 *
CC9A:              133 * EXECUTE CLEAR TO EOL:
CC9A:              134 *
CC9A:AC 7B 05      135 X.GS     LDY  OURCH      ;get CH
```

```
CC9D:A5 32          136 X.GSEOLZ LDA INVFLG     ;mask blank
CC9F:29 80          137          AND #$80       ;with high bit of invflg
CCA1:09 20          138          ORA #$20       ;make it a blank
CCA3:2C 1F C0       139          BIT RD80VID    ;is it 80 columns?
CCA6:30 15   CCBD   140          BMI CLR80      ;=>yes do quick clear
CCA8:91 28          141 CLR40    STA (BASL),Y
CCAA:C8             142          INY
CCAB:C4 21          143          CPY WNDWDTH
CCAD:90 F9   CCA8   144          BCC CLR40
CCAF:60             145          RTS
CCB0:               146 *
CCB0:               147 * Clear right half of screen for 40 to 80
CCB0:               148 * screen conversion
CCB0:               149 *
CCB0:86 2A          150 CLRHALF STX  BAS2L      ;save X
CCB2:A2 D8          151          LDX #$D8       ;set horizontal counter
CCB4:A0 14          152          LDY #20
CCB6:A5 32          153          LDA INVFLG     ;set (inverse) blank
CCB8:29 A0          154          AND #$A0
CCBA:4C D5 CC       155          JMP CLR2
CCBD:               156 *
CCBD:               157 * Clear to end of line for 80 columns
CCBD:               158 *
CCBD:86 2A          159 CLR80    STX  BAS2L     ;save X
CCBF:48             160          PHA            ;and blank
CCC0:98             161          TYA            ;get count for CH
CCC1:48             162          PHA            ;save for left edge check
CCC2:38             163          SEC            ;count=WNDWDTH-Y-1
CCC3:E5 21          164          SBC WNDWDTH
CCC5:AA             165          TAX            ;save CH counter
CCC6:98             166          TYA            ;div CH by 2 for half pages
CCC7:4A             167          LSR A
CCC8:A8             168          TAY
CCC9:68             169          PLA            ;restore original ch
CCCA:45 20          170          EOR WNDLFT     ;get starting page
CCCC:6A             171          ROR A
CCCD:B0 03   CCD2   172          BCS CLR0
CCCF:10 01   CCD2   173          BPL CLR0
CCD1:C8             174          INY            ;iff WNDLFT odd, starting byte odd
CCD2:68             175 CLR0     PLA            ;get blank
CCD3:B0 0B   CCE0   176          BCS CLR1       ;starting page is 1 (default)
CCD5:2C 55 C0       177 CLR2     BIT TXTPAGE2   ;else do page 2
CCD8:91 28          178          STA (BASL),Y
CCDA:2C 54 C0       179          BIT TXTPAGE1   ;now do page 1
CCDD:E8             180          INX
CCDE:F0 06   CCE6   181          BEQ CLR3       ;all done
CCE0:91 28          182 CLR1     STA (BASL),Y
CCE2:C8             183          INY            ;forward 2 columns
CCE3:E8             184          INX            ;next ch
CCE4:D0 EF   CCD5   185          BNE CLR2       ;not done yet
CCE6:A6 2A          186 CLR3     LDX BAS2L      ;restore X
CCE8:38             187          SEC            ;good exit condition
CCE9:60             188          RTS            ;and return
CCEA:               189 *
```

```
CCEA:                 190 * EXECUTE '40COL MODE':
CCEA:                 191 *
CCEA:          CCEA   192 X.DC1    EQU  *
CCEA:AD FB 04         193           LDA  MODE       ;don't convert if Pascal
CCED:30 4D
  CD3C  194            BMI  X.DC1RTS  ;=>it's Pascal
CCEF:20 31 CD         195 X.DC1A    JSR  SETTOP     ;set top of window (0 or 20)
CCF2:2C 1F C0         196           BIT  RD80VID    ;are we in 80 columns?
CCF5:10 12    CD09    197           BPL  X.DC1B     ;=>no, no convert needed
CCF7:20 91 CD         198           JSR  SCRN84     ;else convert 80 to 40
CCFA:90 0D    CD09    199           BCC  X.DC1B     ;=>always set new window
CCFC:                 200 *
CCFC:                 201 * Set 80 column mode
CCFC:                 202 *
CCFC:          CCFC   203 X.DC2    EQU  *
CCFC:20 90 CA         204           JSR  TESTCARD   ;is there an 80 column card?
CCFF:D0 3B    CD3C    205           BNE  X.DC1RTS   ;=>no, can't do this
CD01:2C 1F C0         206           BIT  RD80VID    ;are we in 40 columns?
CD04:30 03    CD09    207           BMI  X.DC1B     ;=>no, no convert needed
CD06:20 C4 CD         208           JSR  SCRN48     ;else convert 40 to 80
CD09:                 209 *
CD09:AD 7B 05         210 X.DC1B    LDA  OURCH      ;get cursor
CD0C:18               211           CLC             ;since new window left = 0
CD0D:65 20            212           ADC  WNDLFT     ;NEWCH=OLDCH+OLDWNDLFT
CD0F:2C 1F C0         213           BIT  RD80VID    ;in 80 columns?
CD12:30 06    CD1A    214           BMI  X.DC1C     ;=>yes, CH is ok
CD14:C9 28            215           CMP  #40        ;else if CH is too big,
CD16:90 02    CD1A    216           BCC  X.DC1C     ;set it to 39
CD18:A9 27            217           LDA  #39
CD1A:8D 7B 05         218 X.DC1C    STA  OURCH      ;save new CH
CD1D:85 24            219           STA  CH
CD1F:A5 25            220           LDA  CV         ;base
CD21:20 BA CA         221           JSR  BASCALC
CD24:2C 1F C0         222           BIT  RD80VID    ;in 80 columns?
CD27:10 05    CD2E    223           BPL  D040       ;=>no, set forty column window
CD29:                 224 *
CD29:20 71 CD         225 D080      JSR  FULL80     ;set 80 column window
CD2C:F0 03    CD31    226           BEQ  SETTOP     ;=>always branch
CD2E:                 227 *
CD2E:20 6D CD         228 D040      JSR  FULL40     ;set 40 column window
CD31:A9 00            229 SETTOP    LDA  #0         ;assume normal window
CD33:2C 1A C0         230           BIT  RDTEXT     ;text or mixed?
CD36:30 02    CD3A    231           BMI  D040A      ;=>text, all ok
CD38:A9 14            232           LDA  #20
CD3A:85 22            233 D040A     STA  WNDTOP     ;set new top
CD3C:60               234 X.DC1RTS  RTS
CD3D:                 235 *
CD3D:                 236 * EXECUTE MOUSE TEXT OFF
CD3D:                 237 *
CD3D:AD FB 04         238 MOUSEOFF  LDA  MODE
CD40:09 01            239           ORA  #M.MOUSE   ;set mouse bit
CD42:D0 05    CD49    240           BNE  SMOUSE     ;to disable mouse chars
CD44:                 241 *
CD44:                 242 * EXECUTE MOUSE TEXT ON
```

```
CD44:                 243 *
CD44:AD FB 04         244 MOUSEON LDA  MODE
CD47:29 FE            245         AND  #255-M.MOUSE ;clear mouse bit
CD49:8D FB 04         246 SMOUSE  STA  MODE      ;to enable mouse chars
CD4C:60               247         RTS
CD4D:                 248 *
CD4D:                 249 * EXECUTE 'QUIT':
CD4D:                 250 *
CD4D:          CD4D   251 X.NAK   EQU  *
CD4D:AD FB 04         252         LDA  MODE      ;ONLY VALID IN BASIC
CD50:30 1A    CD6C    253         BMI  SKRTS     ;ignore if pascal
CD52:20 2E CD         254         JSR  D040      ;force 40 column window
CD55:20 80 CD         255         JSR  QUIT      ;do stuff used by PR#0
CD58:20 64 CD         256         JSR  SETCOUT1  ;set output hook
CD5B:                 257 *
CD5B:A9 FD            258 SETKEYIN LDA #<KEYIN   ;set input hook
CD5D:85 39            259         STA  KSWH
CD5F:A9 1B            260         LDA  #>KEYIN
CD61:85 38            261         STA  KSWL
CD63:60               262         RTS
CD64:                 263 *
CD64:A9 FD            264 SETCOUT1 LDA #<COUT1   ;set output hook
CD66:85 37            265         STA  CSWH
CD68:A9 F0            266         LDA  #>COUT1
CD6A:85 36            267         STA  CSWL
CD6C:60               268 SKRTS   RTS
CD6D:                 269 *
CD6D:                 270 ****************************************
CD6D:                 271 * NAME    : FULL40
CD6D:                 272 * FUNCTION: SET FULL 40COL WINDOW
CD6D:                 273 * INPUT   : NONE
CD6D:                 274 * OUTPUT  : WINDOW PARAMETERS, A=0
CD6D:                 275 * VOLATILE: AC
CD6D:                 276 ****************************************
CD6D:                 277 *
CD6D:          CD6D   278 FULL40  EQU  *
CD6D:A9 28            279         LDA  #40       ;set window width to 40
CD6F:D0 02    CD73    280         BNE  SAVWDTH   ;=>(always taken)
CD71:                 281 *
CD71:                 282 ****************************************
CD71:                 283 * NAME    : FULL80
CD71:                 284 * FUNCTION: SET FULL 80COL WINDOW
CD71:                 285 * INPUT   : NONE
CD71:                 286 * OUTPUT  : WINDOW PARAMETERS, A=0
CD71:                 287 * VOLATILE: AC
CD71:                 288 ****************************************
CD71:                 289 *
CD71:A9 50            290 FULL80  LDA  #80       ;set full 80 column window
CD73:85 21            291 SAVWDTH STA  WNDWDTH
CD75:A9 18            292         LDA  #24
CD77:85 23            293         STA  WNDBTM
CD79:A9 00            294         LDA  #0
CD7B:85 22            295         STA  WNDTOP
CD7D:85 20            296         STA  WNDLFT
```

```
CD7F:60             297          RTS
CD80:               298 *
CD80:               299 * QUIT is used by PR#0 to turn off everything
CD80:               300 *
CD80:         CD80  301 QUIT     EQU  *
CD80:2C 1F C0       302          BIT  RD80VID    ;were we in 80 columns?
CD83:10 03    CD88  303          BPL  QUIT2      ;=> not a chance
CD85:20 EF CC       304          JSR  X.DC1A     ;switch to 40 columns
CD88:8D 0E C0       305 QUIT2    STA  CLRALTCHAR ;don't use lower case
CD8B:A9 FF          306          LDA  #$FF       ;DESTROY THE
CD8D:8D FB 04       307          STA  MODE       ; MODE BYTE
CD90:60             308          RTS
CD91:               309 *
CD91:               310 * SCRN84 and SCRN48 convert screens between 40 & 80 cols.
CD91:               311 * WNDTOP must be set up to indicate the last line to
CD91:               312 * be done.  All registers are trashed.
CD91:               313 *
CD91:8A             314 SCRN84   TXA              ;save X
CD92:48             315          PHA
CD93:A2 17          316          LDX  #23         ;start at bottom of screen
CD95:8D 01 C0       317          STA  SET80COL    ;allow page 2 access
CD98:8A             318 SCR1     TXA              ;calc base for line
CD99:20 BA CA       319          JSR  BASCALC
CD9C:A0 27          320          LDY  #39         ;start at right of screen
CD9E:84 2A          321 SCR2     STY  BAS2L       ;save 40 index
CDA0:98             322          TYA              ;div by 2 for 80 column index
CDA1:4A             323          LSR  A
CDA2:B0 03    CDA7  324          BCS  SCR3
CDA4:2C 55 C0       325          BIT  TXTPAGE2    ;even column, do page 2
CDA7:A8             326 SCR3     TAY              ;get 80 index
CDA8:B1 28          327          LDA  (BASL),Y    ;get 80 char
CDAA:2C 54 C0       328          BIT  TXTPAGE1    ;restore page1
CDAD:A4 2A          329          LDY  BAS2L       ;get 40 index
CDAF:91 28          330          STA  (BASL),Y
CDB1:88             331          DEY
CDB2:10 EA    CD9E  332          BPL  SCR2        ;do next 40 byte
CDB4:CA             333          DEX              ;do next line
CDB5:30 04    CDBB  334          BMI  SCR4        ;=>done with setup
CDB7:E4 22          335          CPX  WNDTOP      ;at top yet?
CDB9:B0 DD    CD98  336          BCS  SCR1
CDBB:8D 00 C0       337 SCR4     STA  CLR80COL    ;clear 80STORE for 40 columns
CDBE:8D 0C C0       338          STA  CLR80VID    ;clear 80VID for 40 columns
CDC1:4C F8 CD       339          JMP  SCRNRET     ;calc base, restore X, exit
CDC4:               340 *
CDC4:8A             341 SCRN48   TXA              ;save X
CDC5:48             342          PHA
CDC6:A2 17          343          LDX  #23         ;start at bottom of screen
CDC8:8A             344 SCR5     TXA              ;set base for current line
CDC9:20 BA CA       345          JSR  BASCALC
CDCC:A0 00          346          LDY  #0          ;start at left of screen
CDCE:8D 01 C0       347          STA  SET80COL    ;enable page2 store
CDD1:B1 28          348 SCR6     LDA  (BASL),Y    ;get 40 column char
CDD3:84 2A          349 SCR8     STY  BAS2L       ;save 40 column index
CDD5:48             350          PHA              ;save char
```

```
CDD6:98             351          TYA             ;div 2 for 80 column index
CDD7:4A             352          LSR  A
CDD8:B0 03   CDDD   353          BCS  SCR7       ;save on page1
CDDA:8D 55 C0       354          STA  TXTPAGE2
CDDD:A8             355 SCR7     TAY             ;get 80 column index
CDDE:68             356          PLA             ;now save character
CDDF:91 28          357          STA  (BASL),Y
CDE1:8D 54 C0       358          STA  TXTPAGE1   ;flip page1
CDE4:A4 2A          359          LDY  BAS2L      ;restore 40 column index
CDE6:C8             360          INY             ;move to the right
CDE7:C0 28          361          CPY  #40        ;at right yet?
CDE9:90 E6   CDD1   362          BCC  SCR6       ;=>no, do next column
CDEB:20 B0 CC       363          JSR  CLRHALF    ;clear half of screen
CDEE:CA             364          DEX             ;else do next line of screen
CDEF:30 04   CDF5   365          BMI  SCR9       ;=>done with top line
CDF1:E4 22          366          CPX  WNDTOP     ;at top yet?
CDF3:B0 D3   CDC8   367          BCS  SCR5
CDF5:8D 0D C0       368 SCR9     STA  SET80VID   ;convert to 80 columns
CDF8:20 FE CD       369 SCRNRET  JSR  VTAB       ;update base
CDFB:68             370          PLA             ;restore X
CDFC:AA             371          TAX
CDFD:60             372          RTS
CDFE:               373 *
CDFE:A5 25          374 VTAB     LDA  CV         ;get 80 column CV
CE00:8D FB 05       375          STA  OURCV      ;copy to OURCV
CE03:20 BA CA       376 VTABZ    JSR  BASCALC    ;calc base address
CE06:A5 20          377          LDA  WNDLFT     ;and add window left to it
CE08:2C 1F C0       378          BIT  RD80VID    ;is it 80 columns?
CE0B:10 01   CE0E   379          BPL  VTAB40     ;window width ok
CE0D:4A             380          LSR  A          ;else divide width by 2
CE0E:18             381 VTAB40   CLC             ;prepare to add
CE0F:65 28          382          ADC  BASL       ;add in window left
CE11:85 28          383          STA  BASL       ;and update base
CE13:60             384 VTABX    RTS             ;and exit
CE14:                29          INCLUDE SUBS3
CE14:C9 E1            1 UPSHFT   CMP  #$E1       ;is it lowercase?
CE16:90 06   CE1E     2          BCC  UPSHFT2    ;=>nope
CE18:C9 FB            3          CMP  #$FB       ;lowercase?
CE1A:B0 02   CE1E     4          BCS  UPSHFT2    ;=>nope
CE1C:29 DF            5          AND  #$DF       ;else upshift
CE1E:60               6 UPSHFT2  RTS
CE1F:                 7 *
CE1F:                 8 ****************************************
CE1F:                 9 * NAME     : INVERT
CE1F:                10 * FUNCTION: INVERT CHAR AT CH/CV
CE1F:                11 *          : Unless Pascal and M.CURSOR=1
CE1F:                12 * INPUT    : NOTHING
CE1F:                13 * OUTPUT   : CHAR AT CH/CV INVERTED
CE1F:                14 * VOLATILE: NOTHING
CE1F:                15 * CALLS    : PICK, STORCHAR
CE1F:                16 ****************************************
CE1F:                17 *
CE1F:AD FB 04        18 PASINV   LDA  MODE       ;check pascal cursor flag
CE22:29 10           19          AND  #M.CURSOR  ;before displaying cursor
```

```
CE24:D0 11    CE37   20          BNE  INVX       ;=>cursor off, don't invert
CE26:48              21 INVERT    PHA             ;save AC
CE27:98              22           TYA             ; AND Y
CE28:48              23           PHA
CE29:AC 7B 05        24           LDY  OURCH      ;GET CH
CE2C:20 44 CE        25           JSR  PICK       ;GET CHARACTER
CE2F:49 80           26           EOR  #$80       ;FLIP INVERSE/NORMAL
CE31:20 70 CE        27           JSR  STORIT     ; ONTO SCREEN
CE34:68              28           PLA             ;RESTORE Y
CE35:A8              29           TAY             ; AND AC
CE36:68              30           PLA
CE37:60              31 INVX      RTS
CE38:                32 ****************************************
CE38:                33 * NAME     : STORCHAR
CE38:                34 * FUNCTION: STORE A CHAR ON SCREEN
CE38:                35 * INPUT    : AC=CHAR
CE38:                36 *          :  Y=CH POSITION
CE38:                37 * OUTPUT   : CHAR ON SCREEN
CE38:                38 * VOLATILE: NOTHING
CE38:                39 * CALLS    : SCREENIT
CE38:                40 ****************************************
CE38:                41 *
CE38:         CE38   42 STORCHAR EQU *
CE38:48              43           PHA             ;SAVE AC
CE39:24 32           44           BIT  INVFLG     ;NORMAL OR INVERSE?
CE3B:30 02    CE3F   45           BMI  STOR2      ;=>NORMAL
CE3D:29 7F           46           AND  #$7F       ;inverse it
CE3F:         CE3F   47 STOR2    EQU  *
CE3F:20 70 CE        48           JSR  STORIT     ;=>do it!!
CE42:68              49           PLA             ;RESTORE AC
CE43:60              50 SEV       RTS
CE44:                51 ****************************************
CE44:                52 * NAME     : PICK
CE44:                53 * FUNCTION: GET A CHAR FROM SCREEN
CE44:                54 * INPUT    : Y=CH POSITION
CE44:                55 * OUTPUT   : AC=CHARACTER
CE44:                56 * VOLATILE: NOTHING
CE44:                57 * CALLS    : SCREENIT
CE44:                58 ****************************************
CE44:                59 *
CE44:B1 28           60 PICK      LDA  (BASL),Y   ;get 40 column character
CE46:2C 1F C0        61           BIT  RD80VID    ;80 columns?
CE49:10 19    CE64   62           BPL  PICK3      ;=>no, do text shift
CE4B:8D 01 C0        63           STA  SET80COL   ;force 80STORE for 80 columns
CE4E:84 2A           64           STY  BAS2L      ;temp store for position
CE50:98              65           TYA             ;divide CH by two
CE51:45 20           66           EOR  WNDLFT     ;C=1 if char in main RAM
CE53:6A              67           ROR  A          ;get low bit into carry
CE54:B0 04    CE5A   68           BCS  PICK1      ;=>store in main memory
CE56:AD 55 C0        69           LDA  TXTPAGE2   ;else switch in page 2
CE59:C8              70           INY             ;for odd left, aux bytes
CE5A:98              71 PICK1     TYA             ;divide position by 2
CE5B:4A              72           LSR  A          ;and use carry as
CE5C:A8              73           TAY             ;page indicator
```

```
CE5D:B1 28          74 PICK2   LDA  (BASL),Y  ;get that char
CE5F:2C 54 C0       75         BIT  TXTPAGE1  ;flip to page 1
CE62:A4 2A          76         LDY  BAS2L
CE64:2C 1E C0       77 PICK3   BIT  ALTCHARSET ;only allow mouse text
CE67:10 06   CE6F   78         BPL  PICK4     ;if alternate character set
CE69:C9 20          79         CMP  #$20
CE6B:B0 02   CE6F   80         BCS  PICK4
CE6D:09 40          81         ORA  #$40
CE6F:60             82 PICK4   RTS
CE70:               83 *
CE70:               84 ****************************************
CE70:               85 * NAME    : STORIT
CE70:               86 * FUNCTION: STORE CHAR
CE70:               87 * INPUT   : AC=char for store
CE70:               88 *         :  Z=high bit of char
CE70:               89 *         :  Y=CH POSITION
CE70:               90 * OUTPUT  : AC=CHAR (PICK)
CE70:               91 * VOLATILE: NOTHING
CE70:               92 * CALLS   : NOTHING
CE70:               93 ****************************************
CE70:               94 *
CE70:48             95 STORIT  PHA            ;save char
CE71:29 FF          96         AND  #$FF      ;if high bit set...
CE73:30 16   CE8B   97         BMI  STORE1    ;=>not mouse text
CE75:AD FB 04       98         LDA  MODE      ;is mouse text enabled?
CE78:6A             99         ROR  A         ;use carry as flag
CE79:68            100         PLA             ;and restore char
CE7A:48            101         PHA            ;need to save it too
CE7B:90 0E   CE8B  102         BCC  STORE1
CE7D:2C 1E C0      103         BIT  ALTCHARSET ;only do mouse text if
CE80:10 09   CE8B  104         BPL  STORE1    ;alt char set switched in
CE82:49 40         105         EOR  #$40      ;do mouse shift
CE84:2C AC CE      106         BIT  HEX60     ;is it in proper range?
CE87:F0 02   CE8B  107         BEQ  STORE1    ;=>yes, leave it
CE89:49 40         108         EOR  #$40      ;else shift it back
CE8B:              109 *
CE8B:2C 1F C0      110 STORE1 BIT  RD80VID   ;80 columns?
CE8E:10 1D   CEAD  111         BPL  STOR40    ;=>no, 40 columns
CE90:8D 01 C0      112         STA  SET80COL  ;force 80STORE for 80 columns
CE93:48            113         PHA            ;save shifted character
CE94:84 2A         114         STY  BAS2L     ;temp storage
CE96:98            115         TYA            ;get position
CE97:45 20         116         EOR  WNDLFT    ;C=1 if char in main RAM
CE99:4A            117         LSR  A
CE9A:B0 04   CEA0  118         BCS  STORE2    ;=>yes, main RAM
CE9C:AD 55 C0      119         LDA  TXTPAGE2  ;else flip in main RAM
CE9F:C8            120         INY            ;do this for odd left bytes
CEA0:98            121 STORE2 TYA            ;get position
CEA1:4A            122         LSR  A         ;and divide it by 2
CEA2:A8            123         TAY
CEA3:68            124 STORIT2 PLA            ;restore acc
CEA4:91 28         125         STA  (BASL),Y  ;save to screen
CEA6:AD 54 C0      126         LDA  TXTPAGE1  ;flip to page 1
CEA9:A4 2A         127         LDY  BAS2L
```

```
CEAB:68            128         PLA            ;restore true Acc
CEAC:60            129 HEX60   RTS            ;and exit
CEAD:              130 *
CEAD:91 28         131 STOR40  STA  (BASL),Y  ;quick 40 column store
CEAF:68            132         PLA            ;restore real char
CEB0:60            133         RTS
CEB1:              134 ****************************************
CEB1:              135 * NAME    : ESCON
CEB1:              136 * FUNCTION: TURN ON 'ESCAPE' CURSOR
CEB1:              137 * INPUT   : NONE
CEB1:              138 * OUTPUT  : 'CHAR'=ORIGINAL CHAR
CEB1:              139 * VOLATELE: NOTHING
CEB1:              140 * CALLS   : PICK,STORCHAR
CEB1:              141 ****************************************
CEB1:              142 *
CEB1:       CEB1   143 ESCON   EQU  *
CEB1:48            144         PHA            ;SAVE AC
CEB2:98            145         TYA            ; AND Y
CEB3:48            146         PHA
CEB4:AC 7B 05      147         LDY  OURCH     ;GET CH
CEB7:20 44 CE      148         JSR  PICK      ;GET ORIGINAL CHARACTER
CEBA:8D 7B 06      149         STA  CHAR      ; AND REMEMBER FOR ESCOFF
CEBD:29 80         150         AND  #$80      ;SAVE NORMAL/INVERSE BIT
CEBF:49 AB         151         EOR  #$AB      ;MAKE IT AN INVERSE '+'
CEC1:4C CD CE      152         JMP  ESCRET    ;RETURN VIA SIMILAR CODE
CEC4:              153 ****************************************
CEC4:              154 * NAME    : ESCOFF
CEC4:              155 * FUNCTION: TURN OFF 'ESCAPE' CURSOR
CEC4:              156 * INPUT   : 'CHAR'=ORIGINAL CHAR
CEC4:              157 * OUTPUT  : NONE
CEC4:              158 * VOLATILE: NOTHING
CEC4:              159 * CALLS   : STORCHAR
CEC4:              160 ****************************************
CEC4:              161 *
CEC4:       CEC4   162 ESCOFF  EQU  *
CEC4:48            163         PHA            ;SAVE AC
CEC5:98            164         TYA            ; AND Y
CEC6:48            165         PHA
CEC7:AC 7B 05      166         LDY  OURCH     ;GET CH
CECA:AD 7B 06      167         LDA  CHAR      ;GET ORIGINAL CHARACTER
CECD:       CECD   168 ESCRET  EQU  *         ;USED BY ESCON
CECD:20 70 CE      169         JSR  STORIT    ;  EXACTLY AS IT WAS
CED0:68            170         PLA            ;RESTORE Y
CED1:A8            171         TAY
CED2:68            172         PLA            ; AND AC
CED3:60            173         RTS
CED4:              174 ****************************************
CED4:              175 * NAME    : PSETUP
CED4:              176 * FUNCTION: SETUP ZP FOR PASCAL
CED4:              177 * INPUT   : NONE
CED4:              178 * OUTPUT  : NONE
CED4:              179 * VOLATILE: AC
CED4:              180 * CALLS   : NOTHING
CED4:              181 ****************************************
```

```
CED4:                 182 *
CED4:           CED4  183 PSETUP  EQU  *
CED4:20 71 CD         184         JSR  FULL80     ;SET FULL 80COL WINDOW
CED7:A9 FF            185 IS80    LDA  #255
CED9:85 32            186         STA  INVFLG     ;ASSUME NORMAL MODE
CEDB:                 187 *
CEDB:AD FB 04         188         LDA  MODE
CEDE:29 04            189         AND  #M.VMODE
CEE0:F0 02    CEE4    190         BEQ  PSETUPRET ;=>IT'S NORMAL
CEE2:46 32            191         LSR  INVFLG     ;MAKE IT INVERSE
CEE4:                 192 *
CEE4:           CEE4  193 PSETUPRET EQU *
CEE4:AD 7B 07         194         LDA  OLDBASL    ;SET UP BASE ADDRESS
CEE7:85 28            195         STA  BASL
CEE9:AD FB 07         196         LDA  OLDBASH
CEEC:85 29            197         STA  BASH
CEEE:AD FB 05         198         LDA  OURCV      ;get user's cursor vertical
CEF1:85 25            199         STA  CV         ;and set it up
CEF3:60               200         RTS
CEF4:                 201 ****************************************
CEF4:                 202 *
CEF4:                 203 * COPYROM is called when the video firmware is
CEF4:                 204 * initialized.  If the language card is switched
CEF4:                 205 * in for reading, it copies the F8 ROM to the
CEF4:                 206 * language card and restores the state of the
CEF4:                 207 * language card.
CEF4:                 208 *
CEF4:2C 12 C0         209 COPYROM BIT  RDLCRAM    ;is the LC switched in?
CEF7:10 3D    CF36    210         BPL  ROMOK      ;=>no, do nothing
CEF9:A9 06            211         LDA  #GOODF8    ;yes, check $F8 RAM
CEFB:CD B3 FB         212         CMP  F8VERSION ;does it match?
CEFE:F0 36    CF36    213         BEQ  ROMOK      ;=> assum ROM is there
CF00:A2 03            214         LDX  #3         ;indicate bank 2, RAM write enabled
CF02:2C 11 C0         215         BIT  RDLCBNK2   ;is it bank 2?
CF05:30 02    CF09    216         BMI  BANK2      ;=>yes, we were right
CF07:A2 0B            217         LDX  #$B        ;no, bank 1, RAM write enabled
CF09:8D B3 FB         218 BANK2   STA  F8VERSION ;write to see if LC is
CF0C:2C 80 C0         219         BIT  $C080      ;write protected (read RAM)
CF0F:AD B3 FB         220         LDA  F8VERSION ;did it change?
CF12:C9 06            221         CMP  #GOODF8
CF14:F0 01    CF17    222         BEQ  WRTENBL    ;=>yes, write enabled
CF16:E8               223         INX             ;else indicate write protect
CF17:2C 81 C0         224 WRTENBL BIT  $C081      ;read ROM, write RAM
CF1A:2C 81 C0         225         BIT  $C081      ;twice is nice
CF1D:A0 00            226         LDY  #$0        ;now copy ROM to RAM
CF1F:A9 F8            227         LDA  #$F8
CF21:85 37            228         STA  CSWH       ;hooks set later
CF23:84 36            229         STY  CSWL
CF25:B1 36            230 COPYROM2 LDA (CSWL),Y  ;get a byte
CF27:91 36            231         STA  (CSWL),Y   ;and move it
CF29:C8               232         INY
CF2A:D0 F9    CF25    233         BNE  COPYROM2
CF2C:E6 37            234         INC  CSWH       ;next page
CF2E:D0 F5    CF25    235         BNE  COPYROM2   ;finish copy
CF30:BD 80 C0         236         LDA  $C080,x    ;read RAM
CF33:BD 80 C0         237         LDA  $C080,x
CF36:60               238 ROMOK   RTS             ;done with ROM copy
```

```
0000:          0000     1 TEST    EQU  0

0000:                   2         LST  On,A,V
0000:          0001     3 IRQTEST EQU  1
0000:                   4         MSB  ON         ;SET THEM HIBITS
0000:          0000     5         DO   TEST
 S                      6 F8ORG   EQU  $1800
 S                      7 IOADR   EQU  $2000      ;For setting PR# hooks
 S                      8 C1ORG   EQU  $2100
 S                      9 C3ORG   EQU  $2300
 S                     10 C8ORG   EQU  $2800
0000:                  11         ELSE
0000:          F800    12 F8ORG   EQU  $F800
0000:          C100    13 C1ORG   EQU  $C100
0000:          C300    14 C3ORG   EQU  $C300
0000:          C800    15 C8ORG   EQU  $C800
0000:                  16         FIN

0000:                   2 ********************************
0000:                   3 *
0000:                   4 * APPLE II
0000:                   5 * MONITOR II
0000:                   6 *
0000:                   7 * COPYRIGHT 1978, 1981, 1984 BY
0000:                   8 * APPLE COMPUTER, INC.
0000:                   9 *
0000:                  10 * ALL RIGHTS RESERVED
0000:                  11 *
0000:                  12 * S. WOZNIAK              1977
0000:                  13 * A. BAUM                 1977
0000:                  14 * JOHN A            NOV  1978
0000:                  15 * R. AURICCHIO      SEP  1981
0000:                  16 * E. BEERNINK            1984
0000:                  17 *
0000:          0001    18 APPLE2E EQU  1          ;COND ASSM/RRA0981
0000:                  19 *
0000:                  20 ********************************
F800:          F800    21         ORG  F8ORG
F800:          2000    22         OBJ  $2000
F800:                  23 ********************************
F800:                  24 *
F800:                  25 * Zero Page Equates
F800:                  26 *
F800:          0000    27 LOC0    EQU  $00        ;vector for autost from disk
F800:          0001    28 LOC1    EQU  $01
F800:          0020    29 WNDLFT  EQU  $20        ;left edge of text window
F800:          0021    30 WNDWDTH EQU  $21        ;width of text window
F800:          0022    31 WNDTOP  EQU  $22        ;top of text window
F800:          0023    32 WNDBTM  EQU  $23        ;bottom+1 of text window
F800:          0024    33 CH      EQU  $24        ;cursor horizontal position
F800:          0025    34 CV      EQU  $25        ;cursor vertical position
F800:          0026    35 GBASL   EQU  $26        ;lo-res graphics base addr.
F800:          0027    36 GBASH   EQU  $27
F800:          0028    37 BASL    EQU  $28        ;text base address
```

```
F800:          0029    38 BASH     EQU  $29
F800:          002A    39 BAS2L    EQU  $2A       ;temp base for scrolling
F800:          002B    40 BAS2H    EQU  $2B
F800:          002C    41 H2       EQU  $2C       ;temp for lo-res graphics
F800:          002C    42 LMNEM    EQU  $2C       ;temp for mnemonic decoding
F800:          002D    43 V2       EQU  $2D       ;temp for lo-res graphics
F800:          002D    44 RMNEM    EQU  $2D       ;temp for mnemonic decoding
F800:          002E    45 MASK     EQU  $2E       ;color mask for lo-res gr.
F800:          002E    46 CHKSUM   EQU  $2E       ;temp for opcode decode
F800:          002E    47 FORMAT   EQU  $2E       ;temp for opcode decode
F800:          002F    48 LASTIN   EQU  $2F       ;temp for tape read csum
F800:          002F    49 LENGTH   EQU  $2F       ;temp for opcode decode
F800:          0030    50 COLOR    EQU  $30       ;color for lo-res graphics
F800:          0031    51 MODE     EQU  $31       ;Monitor mode
F800:          0032    52 INVFLG   EQU  $32       ;normal/inverse(/flash)
F800:          0033    53 PROMPT   EQU  $33       ;prompt character
F800:          0034    54 YSAV     EQU  $34       ;position in Monitor command
F800:          0035    55 YSAV1    EQU  $35       ;temp for Y register
F800:          0036    56 CSWL     EQU  $36       ;character output hook
F800:          0037    57 CSWH     EQU  $37
F800:          0038    58 KSWL     EQU  $38       ;character input hook
F800:          0039    59 KSWH     EQU  $39
F800:          003A    60 PCL      EQU  $3A       ;temp for program counter
F800:          003B    61 PCH      EQU  $3B
F800:          003C    62 A1L      EQU  $3C       ;A1-A5 are Monitor temps
F800:          003D    63 A1H      EQU  $3D
F800:          003E    64 A2L      EQU  $3E
F800:          003F    65 A2H      EQU  $3F
F800:          0040    66 A3L      EQU  $40
F800:          0041    67 A3H      EQU  $41
F800:          0042    68 A4L      EQU  $42
F800:          0043    69 A4H      EQU  $43
F800:          0044    70 A5L      EQU  $44
F800:          0044    71 MACSTAT  EQU  $44       ;machine state for break
F800:          0045    72 A5H      EQU  $45
F800:          0045    73 ACC      EQU  $45        ;Acc after break (destroys A5H)
F800:          0046    74 XREG     EQU  $46        ;X reg after break
F800:          0047    75 YREG     EQU  $47        ;Y reg after break
F800:          0048    76 STATUS   EQU  $48        ;P reg after break
F800:          0049    77 SPNT     EQU  $49        ;SP after break
F800:          004E    78 RNDL     EQU  $4E        ;random counter low
F800:          004F    79 RNDH     EQU  $4F        ;random counter high
F800:                  80 *
F800:          0095    81 PICK     EQU  $95       ;CONTROL-U character
F800:                  82 *
F800:          0200    83 IN       EQU  $0200     ;input buffer for GETLN
F800:                  84 *
F800:                  85 * Page 3 vectors
F800:                  86 *
F800:          03F0    87 BRKV     EQU  $03F0     ;vectors here after break
F800:          03F2    88 SOFTEV   EQU  $03F2     ;vector for warm start
F800:          03F4    89 PWREDUP  EQU  $03F4     ;THIS MUST = EOR #$A5 OF SOFTEV+1
F800:          03F5    90 AMPERV   EQU  $03F5     ;APPLESOFT & EXIT VECTOR
F800:          03F8    91 USRADR   EQU  $03F8     ;Applesoft USR function vector
```

```
F800:          03FB    92 NMI      EQU  $03FB      ;NMI vector
F800:          03FE    93 IRQLOC   EQU  $03FE      ;Maskable interrupt vector
F800:                  94 *
F800:          0400    95 LINE1    EQU  $0400      ;first line of text screen
F800:          07F8    96 MSLOT    EQU  $07F8      ;current user of $C8 space
F800:                  97 *
F800:          0000    98          DO   TEST
F800:                  99          ELSE
F800:          C000   100 IOADR    EQU  $C000
F800:                 101          FIN
F800:                 102 *
F800:          C000   103 KBD      EQU  $C000
F800:          C006   104 SLOTCXROM EQU $C006     ;enable slots 1-7
F800:          C007   105 INTCXROM EQU $C007      ;swap out slots for firmware
F800:          C010   106 KBDSTRB EQU  $C010
F800:          C01F   107 RD80VID EQU  $C01F
F800:          C020   108 TAPEOUT EQU  $C020
F800:          C030   109 SPKR     EQU  $C030
F800:          C050   110 TXTCLR   EQU  $C050
F800:          C051   111 TXTSET   EQU  $C051
F800:          C052   112 MIXCLR   EQU  $C052
F800:          C053   113 MIXSET   EQU  $C053
F800:          C054   114 LOWSCR   EQU  $C054
F800:          C055   115 HISCR    EQU  $C055
F800:          C056   116 LORES    EQU  $C056
F800:          C057   117 HIRES    EQU  $C057
F800:          C058   118 SETAN0   EQU  $C058
F800:          C059   119 CLRAN0   EQU  $C059
F800:          C05A   120 SETAN1   EQU  $C05A
F800:          C05B   121 CLRAN1   EQU  $C05B
F800:          C05C   122 SETAN2   EQU  $C05C
F800:          C05D   123 CLRAN2   EQU  $C05D
F800:          C05E   124 SETAN3   EQU  $C05E
F800:          C05F   125 CLRAN3   EQU  $C05F
F800:          C060   126 TAPEIN   EQU  $C060
F800:          C064   127 PADDL0   EQU  $C064
F800:          C070   128 PTRIG    EQU  $C070
F800:                 129 *
F800:          C3FA   130 IRQ      EQU  C3ORG+$FA ;IRQ entry in $C3 page
F800:          C47C   131 IRQFIX   EQU  C3ORG+$17C ;Restore state at IRQ
F800:                 132 *
F800:          C567   133 XHEADER  EQU  C3ORG+$267
F800:          C5D1   134 XREAD    EQU  C3ORG+$2D1
F800:          C5AA   135 WRITE2   EQU  C3ORG+$2AA
F800:                 136 *
F800:          CFFF   137 CLRROM   EQU  $CFFF
F800:          E000   138 BASIC    EQU  $E000
F800:          E003   139 BASIC2   EQU  $E003
F800:                 140 *
F800:4A               141 PLOT     LSR  A          ;Y-COORD/2
F801:08               142          PHP             ;SAVE LSB IN CARRY
F802:20 47 F8         143          JSR  GBASCALC   ;CALC BASE ADR IN GBASL,H
F805:28               144          PLP             ;RESTORE LSB FROM CARRY
F806:A9 0F            145          LDA  #$0F       ;MASK $0F IF EVEN
```

```
F808:90 02    F80C  146          BCC   RTMASK
F80A:69 E0          147          ADC   #$E0        ;MASK $F0 IF ODD
F80C:85 2E          148 RTMASK   STA   MASK
F80E:B1 26          149 PLOT1    LDA   (GBASL),Y   ;DATA
F810:45 30          150          EOR   COLOR       ; XOR COLOR
F812:25 2E          151          AND   MASK        ;  AND MASK
F814:51 26          152          EOR   (GBASL),Y   ;   XOR DATA
F816:91 26          153          STA   (GBASL),Y   ;    TO DATA
F818:60             154          RTS
F819:               155 *
F819:20 00 F8       156 HLINE    JSR   PLOT        ;PLOT SQUARE
F81C:C4 2C          157 HLINE1   CPY   H2          ;DONE?
F81E:B0 11    F831  158          BCS   RTS1        ; YES, RETURN
F820:C8             159          INY               ; NO, INCR INDEX (X-COORD)
F821:20 0E F8       160          JSR   PLOT1       ;PLOT NEXT SQUARE
F824:90 F6    F81C  161          BCC   HLINE1      ;ALWAYS TAKEN
F826:69 01          162 VLINEZ   ADC   #$01        ;NEXT Y-COORD
F828:48             163 VLINE    PHA               ; SAVE ON STACK
F829:20 00 F8       164          JSR   PLOT        ; PLOT SQUARE
F82C:68             165          PLA
F82D:C5 2D          166          CMP   V2          ;DONE?
F82F:90 F5    F826  167          BCC   VLINEZ      ; NO, LOOP.
F831:60             168 RTS1     RTS
F832:               169 *
F832:A0 2F          170 CLRSCR   LDY   #$2F        ;MAX Y, FULL SCRN CLR
F834:D0 02    F838  171          BNE   CLRSC2      ;ALWAYS TAKEN
F836:A0 27          172 CLRTOP   LDY   #$27        ;MAX Y, TOP SCRN CLR
F838:84 2D          173 CLRSC2   STY   V2          ;STORE AS BOTTOM COORD
F83A:               174 ;                               FOR VLINE CALLS
F83A:A0 27          175          LDY   #$27        ;RIGHTMOST X-COORD (COLUMN)
F83C:A9 00          176 CLRSC3   LDA   #$00        ;TOP COORD FOR VLINE CALLS
F83E:85 30          177          STA   COLOR       ;CLEAR COLOR (BLACK)
F840:20 28 F8       178          JSR   VLINE       ;DRAW VLINE
F843:88             179          DEY               ;NEXT LEFTMOST X-COORD
F844:10 F6    F83C  180          BPL   CLRSC3      ;LOOP UNTIL DONE.
F846:60             181          RTS
F847:               182 *
F847:48             183 GBASCALC PHA               ;FOR INPUT  00DEFGH
F848:4A             184          LSR   A
F849:29 03          185          AND   #$03
F84B:09 04          186          ORA   #$04        ;GENERATE GBASH=000001FG
F84D:85 27          187          STA   GBASH
F84F:68             188          PLA               ;AND GBASL=HDEDE000
F850:29 18          189          AND   #$18
F852:90 02    F856  190          BCC   GBCALC
F854:69 7F          191          ADC   #$7F
F856:85 26          192 GBCALC   STA   GBASL
F858:0A             193          ASL   A
F859:0A             194          ASL   A
F85A:05 26          195          ORA   GBASL
F85C:85 26          196          STA   GBASL
F85E:60             197          RTS
F85F:               198 *
F85F:A5 30          199 NXTCOL   LDA   COLOR       ;INCREMENT COLOR BY 3
```

```
F861:18             200          CLC
F862:69 03          201          ADC  #$03
F864:29 0F          202 SETCOL   AND  #$0F       ;SETS COLOR=17*A MOD 16
F866:85 30          203          STA  COLOR
F868:0A             204          ASL  A          ;BOTH HALF BYTES OF COLOR EQUAL
F869:0A             205          ASL  A
F86A:0A             206          ASL  A
F86B:0A             207          ASL  A
F86C:05 30          208          ORA  COLOR
F86E:85 30          209          STA  COLOR
F870:60             210          RTS
F871:               211 *
F871:4A             212 SCRN     LSR  A          ;READ SCREEN Y-COORD/2
F872:08             213          PHP             ;SAVE LSB (CARRY)
F873:20 47 F8       214          JSR  GBASCALC   ;CALC BASE ADDRESS
F876:B1 26          215          LDA  (GBASL),Y  ;GET BYTE
F878:28             216          PLP             ;RESTORE LSB FROM CARRY
F879:90 04   F87F   217 SCRN2    BCC  RTMSKZ     ;IF EVEN, USE LO H
F87B:4A             218          LSR  A
F87C:4A             219          LSR  A
F87D:4A             220          LSR  A          ;SHIFT HIGH HALF BYTE DOWN
F87E:4A             221          LSR  A
F87F:29 0F          222 RTMSKZ   AND  #$0F       ;MASK 4-BITS
F881:60             223          RTS
F882:               224 *
F882:A6 3A          225 INSDS1   LDX  PCL        ;PRINT PCL,H
F884:A4 3B          226          LDY  PCH
F886:20 96 FD       227          JSR  PRYX2
F889:20 48 F9       228          JSR  PRBLNK     ;FOLLOWED BY A BLANK
F88C:A1 3A          229          LDA  (PCL,X)    ;GET OPCODE
F88E:A8             230 INSDS2   TAY
F88F:4A             231          LSR  A          ;EVEN/ODD TEST
F890:90 09   F89B   232          BCC  IEVEN
F892:6A             233          ROR  A          ;BIT 1 TEST
F893:B0 10   F8A5   234          BCS  ERR        ;XXXXXX11 INVALID OP
F895:C9 A2          235          CMP  #$A2
F897:F0 0C   F8A5   236          BEQ  ERR        ;OPCODE $89 INVALID
F899:29 87          237          AND  #$87       ;MASK BITS
F89B:4A             238 IEVEN    LSR  A          ;LSB INTO CARRY FOR L/R TEST
F89C:AA             239          TAX
F89D:BD 62 F9       240          LDA  FMT1,X     ;GET FORMAT INDEX BYTE
F8A0:20 79 F8       241          JSR  SCRN2      ;R/L H-BYTE ON CARRY
F8A3:D0 04   F8A9   242          BNE  GETFMT
F8A5:A0 80          243 ERR      LDY  #$80       ;SUBSTITUTE $80 FOR INVALID OPS
F8A7:A9 00          244          LDA  #$00       ;SET PRINT FORMAT INDEX TO 0
F8A9:AA             245 GETFMT   TAX
F8AA:BD A6 F9       246          LDA  FMT2,X     ;INDEX INTO PRINT FORMAT TABLE
F8AD:85 2E          247          STA  FORMAT     ;SAVE FOR ADR FIELD FORMATTING
F8AF:               248 ; (0=1 BYTE, 1=2 BYTE, 2=3 BYTE)
F8AF:               249 *
F8AF:               250 * Move code to C1-C2 because the code
F8AF:               251 * that tests for ROM in slot 3 must be in
F8AF:               252 * the F8 ROM.
F8AF:               253 *
```

```
F8AF:AA            254          TAX             ;save ACC in X
F8B0:84 2A         255          STY  BAS2L      ;and Y in scrolling temp
F8B2:A0 10         256          LDY  #$10       ;call = finish mnemonics
F8B4:4C B4 FB      257          JMP  GOTOCX     ;off to C100
F8B7:              258 *
F8B7:              259 * Test slot 3 for a card containing ROM.
F8B7:              260 * If there is one, we'll not switch in our internal
F8B7:              261 * slot 3 firmware (for 80 columns).
F8B7:              262 * On entry Y has a high value like $F2, so the
F8B7:              263 * ROM/bus is read a bunch of times
F8B7:              264 *
F8B7:8D 06 C0      265 TSTROM   STA  SLOTCXROM ;swap in slots
F8BA:A2 02         266 TSTROM0  LDX  #2        ;check 2 ID bytes
F8BC:BD 05 C3      267 TSTROM1  LDA  $C305,X   ;at C305 and $C307
F8BF:DD 9C FC      268          CMP  CLREOL,X  ;with two bytes that are same
F8C2:D0 07   F8CB  269          BNE  XTST
F8C4:CA            270          DEX            ;check next ID byte
F8C5:CA            271          DEX
F8C6:10 F4   F8BC  272          BPL  TSTROM1
F8C8:88            273          DEY
F8C9:D0 EF   F8BA  274          BNE  TSTROM0   ;if ROM ok, exit with BEQ
F8CB:8D 07 C0      275 XTST     STA  INTCXROM  ;swap internal ROM
F8CE:60            276          RTS            ;and return there
F8CF:              277 *
F8CF:EA            278          NOP            ;line things up
F8D0:              279 *
F8D0:20 82 F8      280 INSTDSP  JSR  INSDS1     ;GEN FMT, LEN BYTES
F8D3:48            281          PHA            ;SAVE MNEMONIC TABLE INDEX
F8D4:B1 3A         282 PRNTOP   LDA  (PCL),Y
F8D6:20 DA FD      283          JSR  PRBYTE
F8D9:A2 01         284          LDX  #$01      ;PRINT 2 BLANKS
F8DB:20 4A F9      285 PRNTBL   JSR  PRBL2
F8DE:C4 2F         286          CPY  LENGTH    ;PRINT INST (1-3 BYTES)
F8E0:C8            287          INY            ;IN A 12 CHR FIELD
F8E1:90 F1   F8D4  288          BCC  PRNTOP
F8E3:A2 03         289          LDX  #$03      ;CHAR COUNT FOR MNEMONIC INDEX
F8E5:C0 04         290          CPY  #$04
F8E7:90 F2   F8DB  291          BCC  PRNTBL
F8E9:68            292          PLA            ;RECOVER MNEMONIC INDEX
F8EA:A8            293          TAY
F8EB:B9 C0 F9      294          LDA  MNEML,Y
F8EE:85 2C         295          STA  LMNEM     ;FETCH 3-CHAR MNEMONIC
F8F0:B9 00 FA      296          LDA  MNEMR,Y   ;  (PACKED INTO 2-BYTES)
F8F3:85 2D         297          STA  RMNEM
F8F5:A9 00         298 PRMN1    LDA  #$00
F8F7:A0 05         299          LDY  #$05
F8F9:06 2D         300 PRMN2    ASL  RMNEM     ;SHIFT 5 BITS OF CHARACTER INTO A
F8FB:26 2C         301          ROL  LMNEM
F8FD:2A            302          ROL  A         ;  (CLEARS CARRY)
F8FE:88            303          DEY
F8FF:D0 F8   F8F9  304          BNE  PRMN2
F901:69 BF         305          ADC  #$BF      ;ADD "?" OFFSET
F903:20 ED FD      306          JSR  COUT      ;OUTPUT A CHAR OF MNEM
F906:CA            307          DEX
```

```
F907:D0 EC   F8F5  308          BNE  PRMN1
F909:20 48 F9     309          JSR  PRBLNK     ;OUTPUT 3 BLANKS
F90C:A4 2F        310          LDY  LENGTH
F90E:A2 06        311          LDX  #$06       ;CNT FOR 6 FORMAT BITS
F910:E0 03        312 PRADR1   CPX  #$03
F912:F0 1C   F930 313          BEQ  PRADR5     ;IF X=3 THEN ADDR.
F914:06 2E        314 PRADR2   ASL  FORMAT
F916:90 0E   F926 315          BCC  PRADR3
F918:BD B3 F9     316          LDA  CHAR1-1,X
F91B:20 ED FD     317          JSR  COUT
F91E:BD B9 F9     318          LDA  CHAR2-1,X
F921:F0 03   F926 319          BEQ  PRADR3
F923:20 ED FD     320          JSR  COUT
F926:CA           321 PRADR3   DEX
F927:D0 E7   F910 322          BNE  PRADR1
F929:60           323          RTS
F92A:88           324 PRADR4   DEY
F92B:30 E7   F914 325          BMI  PRADR2
F92D:20 DA FD     326          JSR  PRBYTE
F930:A5 2E        327 PRADR5   LDA  FORMAT
F932:C9 E8        328          CMP  #$E8       ;HANDLE REL ADR MODE
F934:B1 3A        329          LDA  (PCL),Y    ;SPECIAL (PRINT TARGET,
F936:90 F2   F92A 330          BCC  PRADR4     ;  NOT OFFSET)
F938:20 56 F9     331 RELADR   JSR  PCADJ3
F93B:AA           332          TAX             ;PCL,PCH+OFFSET+1 TO A,Y
F93C:E8           333          INX
F93D:D0 01   F940 334          BNE  PRNTYX     ;+1 TO Y,X
F93F:C8           335          INY
F940:98           336 PRNTYX   TYA
F941:20 DA FD     337 PRNTAX   JSR  PRBYTE     ;OUTPUT TARGET ADR
F944:8A           338 PRNTX    TXA             ;  OF BRANCH AND RETURN
F945:4C DA FD     339          JMP  PRBYTE
F948:             340 *
F948:A2 03        341 PRBLNK   LDX  #$03       ;BLANK COUNT
F94A:A9 A0        342 PRBL2    LDA  #$A0       ;LOAD A SPACE
F94C:20 ED FD     343 PRBL3    JSR  COUT       ;OUTPUT A BLANK
F94F:CA           344          DEX
F950:D0 F8   F94A 345          BNE  PRBL2      ;LOOP UNTIL COUNT=0
F952:60           346          RTS
F953:             347 *
F953:38           348 PCADJ    SEC             ;0=1 BYTE, 1=2 BYTE,
F954:A5 2F        349 PCADJ2   LDA  LENGTH     ;  2=3 BYTE
F956:A4 3B        350 PCADJ3   LDY  PCH
F958:AA           351          TAX             ;TEST DISPLACEMENT SIGN
F959:10 01   F95C 352          BPL  PCADJ4     ;  (FOR REL BRANCH)
F95B:88           353          DEY             ;EXTEND NEG BY DECR PCH
F95C:65 3A        354 PCADJ4   ADC  PCL
F95E:90 01   F961 355          BCC  RTS2       ;PCL+LENGTH(OR DISPL)+1 TO A
F960:C8           356          INY             ;  CARRY INTO Y (PCH)
F961:60           357 RTS2     RTS
F962:             358 ;
F962:             359 ; FMT1 BYTES:    XXXXXXY0 INSTRS
F962:             360 ; IF Y=0         THEN LEFT HALF BYTE
F962:             361 ; IF Y=1         THEN RIGHT HALF BYTE
```

```
F962:              362 ;                   (X=INDEX)
F962:              363 ;
F962:04            364 FMT1    DFB  $04
F963:20            365         DFB  $20
F964:54            366         DFB  $54
F965:30            367         DFB  $30
F966:0D            368         DFB  $0D
F967:80            369         DFB  $80
F968:04            370         DFB  $04
F969:90            371         DFB  $90
F96A:03            372         DFB  $03
F96B:22            373         DFB  $22
F96C:54            374         DFB  $54
F96D:33            375         DFB  $33
F96E:0D            376         DFB  $0D
F96F:80            377         DFB  $80
F970:04            378         DFB  $04
F971:90            379         DFB  $90
F972:04            380         DFB  $04
F973:20            381         DFB  $20
F974:54            382         DFB  $54
F975:33            383         DFB  $33
F976:0D            384         DFB  $0D
F977:80            385         DFB  $80
F978:04            386         DFB  $04
F979:90            387         DFB  $90
F97A:04            388         DFB  $04
F97B:20            389         DFB  $20
F97C:54            390         DFB  $54
F97D:3B            391         DFB  $3B
F97E:0D            392         DFB  $0D
F97F:80            393         DFB  $80
F980:04            394         DFB  $04
F981:90            395         DFB  $90
F982:00            396         DFB  $00
F983:22            397         DFB  $22
F984:44            398         DFB  $44
F985:33            399         DFB  $33
F986:0D            400         DFB  $0D
F987:C8            401         DFB  $C8
F988:44            402         DFB  $44
F989:00            403         DFB  $00
F98A:11            404         DFB  $11
F98B:22            405         DFB  $22
F98C:44            406         DFB  $44
F98D:33            407         DFB  $33
F98E:0D            408         DFB  $0D
F98F:C8            409         DFB  $C8
F990:44            410         DFB  $44
F991:A9            411         DFB  $A9
F992:01            412         DFB  $01
F993:22            413         DFB  $22
F994:44            414         DFB  $44
F995:33            415         DFB  $33
```

```
F996:0D          416          DFB  $0D
F997:80          417          DFB  $80
F998:04          418          DFB  $04
F999:90          419          DFB  $90
F99A:01          420          DFB  $01
F99B:22          421          DFB  $22
F99C:44          422          DFB  $44
F99D:33          423          DFB  $33
F99E:0D          424          DFB  $0D
F99F:80          425          DFB  $80
F9A0:04          426          DFB  $04
F9A1:90          427          DFB  $90
F9A2:26          428          DFB  $26
F9A3:31          429          DFB  $31
F9A4:87          430          DFB  $87
F9A5:9A          431          DFB  $9A
F9A6:            432 ;
F9A6:            433 ; ZZXXXY01 INSTR'S
F9A6:            434 ;
F9A6:00          435 FMT2     DFB  $00        ;ERR
F9A7:21          436          DFB  $21        ;IMM
F9A8:81          437          DFB  $81        ;Z-PAGE
F9A9:82          438          DFB  $82        ;ABS
F9AA:00          439          DFB  $00        ;IMPLIED
F9AB:00          440          DFB  $00        ;ACCUMULATOR
F9AC:59          441          DFB  $59        ;(ZPAG,X)
F9AD:4D          442          DFB  $4D        ;(ZPAG),Y
F9AE:91          443          DFB  $91        ;ZPAG,X
F9AF:92          444          DFB  $92        ;ABS,X
F9B0:86          445          DFB  $86        ;ABS,Y
F9B1:4A          446          DFB  $4A        ;(ABS)
F9B2:85          447          DFB  $85        ;ZPAG,Y
F9B3:9D          448          DFB  $9D        ;RELATIVE
F9B4:AC          449 CHAR1    DFB  $AC        ;','
F9B5:A9          450          DFB  $A9        ;')'
F9B6:AC          451          DFB  $AC        ;','
F9B7:A3          452          DFB  $A3        ;'#'
F9B8:A8          453          DFB  $A8        ;'('
F9B9:A4          454          DFB  $A4        ;'$'
F9BA:D9          455 CHAR2    DFB  $D9        ;'Y'
F9BB:00          456          DFB  $00
F9BC:D8          457          DFB  $D8        ;'Y'
F9BD:A4          458          DFB  $A4        ;'$'
F9BE:A4          459          DFB  $A4        ;'$'
F9BF:00          460          DFB  $00
F9C0:1C          461 MNEML    DFB  $1C
F9C1:8A          462          DFB  $8A
F9C2:1C          463          DFB  $1C
F9C3:23          464          DFB  $23
F9C4:5D          465          DFB  $5D
F9C5:8B          466          DFB  $8B
F9C6:1B          467          DFB  $1B
F9C7:A1          468          DFB  $A1
F9C8:9D          469          DFB  $9D
```

```
F9C9:8A          470          DFB  $8A
F9CA:1D          471          DFB  $1D
F9CB:23          472          DFB  $23
F9CC:9D          473          DFB  $9D
F9CD:8B          474          DFB  $8B
F9CE:1D          475          DFB  $1D
F9CF:A1          476          DFB  $A1
F9D0:00          477          DFB  $00
F9D1:29          478          DFB  $29
F9D2:19          479          DFB  $19
F9D3:AE          480          DFB  $AE
F9D4:69          481          DFB  $69
F9D5:A8          482          DFB  $A8
F9D6:19          483          DFB  $19
F9D7:23          484          DFB  $23
F9D8:24          485          DFB  $24
F9D9:53          486          DFB  $53
F9DA:1B          487          DFB  $1B
F9DB:23          488          DFB  $23
F9DC:24          489          DFB  $24
F9DD:53          490          DFB  $53
F9DE:19          491          DFB  $19        ; (A) FORMAT ABOVE
F9DF:A1          492          DFB  $A1
F9E0:00          493          DFB  $00
F9E1:1A          494          DFB  $1A
F9E2:5B          495          DFB  $5B
F9E3:5B          496          DFB  $5B
F9E4:A5          497          DFB  $A5
F9E5:69          498          DFB  $69
F9E6:24          499          DFB  $24        ; (B) FORMAT
F9E7:24          500          DFB  $24
F9E8:AE          501          DFB  $AE
F9E9:AE          502          DFB  $AE
F9EA:A8          503          DFB  $A8
F9EB:AD          504          DFB  $AD
F9EC:29          505          DFB  $29
F9ED:00          506          DFB  $00
F9EE:7C          507          DFB  $7C        ; (C) FORMAT
F9EF:00          508          DFB  $00
F9F0:15          509          DFB  $15
F9F1:9C          510          DFB  $9C
F9F2:6D          511          DFB  $6D
F9F3:9C          512          DFB  $9C
F9F4:A5          513          DFB  $A5
F9F5:69          514          DFB  $69
F9F6:29          515          DFB  $29        ; (D) FORMAT
F9F7:53          516          DFB  $53
F9F8:84          517          DFB  $84
F9F9:13          518          DFB  $13
F9FA:34          519          DFB  $34
F9FB:11          520          DFB  $11
F9FC:A5          521          DFB  $A5
F9FD:69          522          DFB  $69
F9FE:23          523          DFB  $23        ; (E) FORMAT
```

```
F9FF:A0           524          DFB  $A0
FA00:D8           525 MNEMR    DFB  $D8
FA01:62           526          DFB  $62
FA02:5A           527          DFB  $5A
FA03:48           528          DFB  $48
FA04:26           529          DFB  $26
FA05:62           530          DFB  $62
FA06:94           531          DFB  $94
FA07:88           532          DFB  $88
FA08:54           533          DFB  $54
FA09:44           534          DFB  $44
FA0A:C8           535          DFB  $C8
FA0B:54           536          DFB  $54
FA0C:68           537          DFB  $68
FA0D:44           538          DFB  $44
FA0E:E8           539          DFB  $E8
FA0F:94           540          DFB  $94
FA10:00           541          DFB  $00
FA11:B4           542          DFB  $B4
FA12:08           543          DFB  $08
FA13:84           544          DFB  $84
FA14:74           545          DFB  $74
FA15:B4           546          DFB  $B4
FA16:28           547          DFB  $28
FA17:6E           548          DFB  $6E
FA18:74           549          DFB  $74
FA19:F4           550          DFB  $F4
FA1A:CC           551          DFB  $CC
FA1B:4A           552          DFB  $4A
FA1C:72           553          DFB  $72
FA1D:F2           554          DFB  $F2
FA1E:A4           555          DFB  $A4        ; (A) FORMAT
FA1F:8A           556          DFB  $8A
FA20:00           557          DFB  $00
FA21:AA           558          DFB  $AA
FA22:A2           559          DFB  $A2
FA23:A2           560          DFB  $A2
FA24:74           561          DFB  $74
FA25:74           562          DFB  $74
FA26:74           563          DFB  $74        ; (B) FORMAT
FA27:72           564          DFB  $72
FA28:44           565          DFB  $44
FA29:68           566          DFB  $68
FA2A:B2
    567          DFB  $B2
FA2B:32           568          DFB  $32
FA2C:B2           569          DFB  $B2
FA2D:00           570          DFB  $00
FA2E:22           571          DFB  $22        ; (C) FORMAT
FA2F:00           572          DFB  $00
FA30:1A           573          DFB  $1A
FA31:1A           574          DFB  $1A
FA32:26           575          DFB  $26
FA33:26           576          DFB  $26
```

```
FA34:72              577          DFB  $72
FA35:72              578          DFB  $72
FA36:88              579          DFB  $88         ; (D) FORMAT
FA37:C8              580          DFB  $C8
FA38:C4              581          DFB  $C4
FA39:CA              582          DFB  $CA
FA3A:26              583          DFB  $26
FA3B:48              584          DFB  $48
FA3C:44              585          DFB  $44
FA3D:44              586          DFB  $44
FA3E:A2              587          DFB  $A2         ; (E) FORMAT
FA3F:C8              588          DFB  $C8
FA40:                589 *
FA40:        C3FA    590 NEWIRQ   EQU  $C3FA      ;new IRQ entry
FA40:                591 *
FA40:85 45           592 OLDIRQ   STA  $45        ;(should never be used)
FA42:A5 45           593          LDA  $45        ;for those who save A to $45
FA44:4C FA C3        594          JMP  NEWIRQ     ;go to interrupt handler
FA47:                595 *
FA47:8D 06 C0        596 NEWBREAK STA SETSLOTCXROM ;force in slots
FA4A:85 45           597          STA  ACC        ;save accumulator
FA4C:                598 *
FA4C:28              599 BREAK    PLP
FA4D:20 4C FF        600          JSR  SAV1        ;SAVE REG'S ON BREAK
FA50:68              601          PLA              ; INCLUDING PC
FA51:85 3A           602          STA  PCL
FA53:68              603          PLA
FA54:85 3B           604          STA  PCH
FA56:6C F0 03        605          JMP  (BRKV)      ;BRKV WRITTEN OVER BY DISK BOOT
FA59:                606 *
FA59:20 82 F8        607 OLDBRK   JSR  INSDS1      ;PRINT USER PC
FA5C:20 DA FA        608          JSR  RGDSP1      ;  AND REGS
FA5F:4C 65 FF        609          JMP  MON         ;GO TO MONITOR (NO PASS GO, NO $200!)
FA62:D8              610 RESET    CLD              ;DO THIS FIRST THIS TIME
FA63:20 84 FE        611          JSR  SETNORM
FA66:20 2F FB        612          JSR  INIT
FA69:20 93 FE        613          JSR  SETVID
FA6C:20 89 FE        614          JSR  SETKBD
FA6F:AD 58 C0        615 INITAN   LDA  SETAN0      ; AN0 = TTL LO
FA72:AD 5A C0        616          LDA  SETAN1      ; AN1 = TTL LO
FA75:A0 09           617          LDY  #9          ;CODE=INIT/RRA0981
FA77:20 B4 FB        618          JSR  GOTOCX      ;DO APPLE2E INIT/RRA0981
FA7A:EA              619          NOP              ;/RRA0981
FA7B:AD FF CF        620          LDA  CLRROM      ; TURN OFF EXTNSN ROM
FA7E:2C 10 C0        621          BIT  KBDSTRB     ; CLEAR KEYBOARD
FA81:D8              622 NEWMON   CLD
FA82:20 3A FF        623          JSR  BELL        ; CAUSES DELAY IF KEY BOUNCES
FA85:AD F3 03        624          LDA  SOFTEV+1    ;IS RESET HI
FA88:49 A5           625          EOR  #$A5        ;A FUNNY COMPLEMENT OF THE
FA8A:CD F4 03        626          CMP  PWREDUP     ; PWR UP BYTE ???
FA8D:D0 17    FAA6   627          BNE  PWRUP       ; NO SO PWRUP
FA8F:AD F2 03        628          LDA  SOFTEV      ; YES SEE IF COLD START
FA92:D0 0F    FAA3   629          BNE  NOFIX       ; HAS BEEN DONE YET?
FA94:A9 E0           630          LDA  #$E0        ; DOES SOFT ENTRY VECTOR POINT AT BASIC?
```

```
FA96:CD F3 03      631          CMP  SOFTEV+1
FA99:D0 08   FAA3  632          BNE  NOFIX       ; YES SO REENTER SYSTEM
FA9B:A0 03         633 FIXSEV   LDY  #3          ; NO SO POINT AT WARM START
FA9D:8C F2 03      634          STY  SOFTEV      ; FOR NEXT RESET
FAA0:4C 00 E0      635          JMP  BASIC       ; AND DO THE COLD START
FAA3:6C F2 03      636 NOFIX    JMP  (SOFTEV)    ; SOFT ENTRY VECTOR
FAA6:              637 ******************
FAA6:20 60 FB      638 PWRUP    JSR  APPLEII
FAA9:        FAA9  639 SETPG3   EQU  *           ; SET PAGE 3 VECTORS
FAA9:A2 05         640          LDX  #5
FAAB:BD FC FA      641 SETPLP   LDA  PWRCON-1,X  ; WITH CNTRL B ADRS
FAAE:9D EF 03      642          STA  BRKV-1,X    ; OF CURRENT BASIC
FAB1:CA            643          DEX
FAB2:D0 F7   FAAB  644          BNE  SETPLP
FAB4:A9 C8         645          LDA  #$C8        ; LOAD HI SLOT +1
FAB6:86 00         646          STX  LOC0        ; SETPG3 MUST RETURN X=0
FAB8:85 01         647          STA  LOC1        ; SET PTR H
FABA:              648 *
FABA:              649 * Check 3 ID bytes instead of 4.  Allows devices
FABA:              650 * other than Disk II's to be bootable.
FABA:              651 *
FABA:A0 05         652 SLOOP    LDY  #5         ;Y is byte ptr
FABC:C6 01         653          DEC  LOC1
FABE:A5 01         654          LDA  LOC1
FAC0:C9 C0         655          CMP  #$C0        ; AT LAST SLOT YET?
FAC2:F0 D7   FA9B  656          BEQ  FIXSEV      ; YES AND IT CAN'T BE A DISK
FAC4:8D F8 07      657          STA  MSLOT
FAC7:B1 00         658 NXTBYT   LDA  (LOC0),Y    ; FETCH A SLOT BYTE
FAC9:D9 01 FB      659          CMP  DISKID-1,Y   ; IS IT A DISK ??
FACC:D0 EC   FABA  660          BNE  SLOOP       ; NO, SO NEXT SLOT DOWN
FACE:88            661          DEY
FACF:88            662          DEY              ; YES, SO CHECK NEXT BYTE
FAD0:10 F5   FAC7  663          BPL  NXTBYT      ; UNTIL 3 BYTES CHECKED
FAD2:6C 00 00      664          JMP  (LOC0)      ; GO BOOT...
FAD5:              665 *
FAD5:EA            666          NOP
FAD6:EA            667          NOP
FAD7:              668 *
FAD7:20 8E FD      669 REGDSP   JSR  CROUT       ;DISPLAY USER REG CONTENTS
FADA:A9 45         670 RGDSP1   LDA  #$45        ;WITH LABELS
FADC:85 40         671          STA  A3L
FADE:A9 00         672          LDA  #$00
FAE0:85 41         673          STA  A3H
FAE2:A2 FB         674          LDX  #$FB
FAE4:A9 A0         675 RDSP1    LDA  #$A0
FAE6:20 ED FD      676          JSR  COUT
FAE9:BD 1E FA      677          LDA  RTBL-251,X
FAEC:20 ED FD      678          JSR  COUT
FAEF:A9 BD         679          LDA  #$BD
FAF1:20 ED FD      680          JSR  COUT
FAF4:B5 4A         681          LDA  ACC+5,X
FAF6:20 DA FD      682          JSR  PRBYTE
FAF9:E8            683          INX
FAFA:30 E8   FAE4  684          BMI  RDSP1
```

```
FAFC:60             685          RTS
FAFD:               686 *
FAFD:59 FA          687 PWRCON   DW   OLDBRK
FAFF:00 E0 45       688          DFB  $00,$E0,$45
FB02:20 FF 00 FF    689 DISKID   DFB  $20,$FF,$00,$FF
FB06:03 FF 3C       690          DFB  $03,$FF,$3C
FB09:C1 F0 F0 EC    691          ASC  'Apple    ]['
FB11:         FB11  692 XLTBL    EQU  *
FB11:C4 C2 C1       693          DFB  $C4,$C2,$C1
FB14:FF C3          694          DFB  $FF,$C3
FB16:FF FF FF       695          DFB  $FF,$FF,$FF
FB19:               696 *
FB19:C1 D8 D9       697 RTBL     DFB  $C1,$D8,$D9  ;REGISTER NAMES FOR REGDSP:
FB1C:D0 D3          698          DFB  $D0,$D3     ;'AXYPS'
FB1E:AD 70 C0       699 PREAD    LDA  PTRIG       ;TRIGGER PADDLES
FB21:A0 00          700          LDY  #$00        ;INIT COUNT
FB23:EA             701          NOP              ;COMPENSATE FOR 1ST COUNT
FB24:EA             702          NOP
FB25:BD 64 C0       703 PREAD2   LDA  PADDL0,X  ;COUNT Y-REG EVERY 12 USEC.
FB28:10 04    FB2E  704          BPL  RTS2D
FB2A:C8             705          INY
FB2B:D0 F8    FB25  706          BNE  PREAD2     ;EXIT AT 255 MAX
FB2D:88             707          DEY
FB2E:60             708 RTS2D    RTS
FB2F:                 1 *
FB2F:A9 00            2 INIT     LDA  #$00        ;CLR STATUS FOR DEBUG SOFTWARE
FB31:85 48            3          STA  STATUS
FB33:AD 56 C0         4          LDA  LORES
FB36:AD 54 C0         5          LDA  LOWSCR      ;INIT VIDEO MODE
FB39:AD 51 C0         6 SETTXT   LDA  TXTSET      ;SET FOR TEXT MODE
FB3C:A9 00            7          LDA  #$00        ;FULL SCREEN WINDOW
FB3E:F0 0B    FB4B    8          BEQ  SETWND
FB40:AD 50 C0         9 SETGR    LDA  TXTCLR      ;SET FOR GRAPHICS MODE
FB43:AD 53 C0        10          LDA  MIXSET      ;LOWER 4 LINES AS TEXT WINDOW
FB46:20 36 F8        11          JSR  CLRTOP
FB49:A9 14           12          LDA  #$14
FB4B:85 22           13 SETWND  STA  WNDTOP      ;SET FOR 40 COL WINDOW
FB4D:A9 00           14          LDA  #$00        ;TOP IN A-REG,
FB4F:85 20           15          STA  WNDLFT      ; BOTTOM AT LINE $24
FB51:A0 0C           16          LDY  #$C         ;CODE=SETWND /RRA0981
FB53:D0 5F    FBB4   17          BNE  GOTOCX
FB55:A9 18           18          LDA  #$18
FB57:85 23           19          STA  WNDBTM
FB59:A9 17           20          LDA  #$17        ;VTAB TO ROW 23
FB5B:85 25           21 TABV    STA  CV          ;VTABS TO ROW IN A-REG
FB5D:4C 22 FC        22          JMP  VTAB
FB60:                23 *
FB60:20 58 FC        24 APPLEII JSR  HOME        ;CLEAR THE SCRN
FB63:A0 09           25          LDY  #9
FB65:B9 09 FF        26 STITLE  LDA  TITLE-1,Y  ;GET A CHAR
FB68:99 0E 04        27          STA  LINE1+14,Y  ;PUT IT AT TOP CENTER OF SCREEN
FB6B:88              28          DEY
FB6C:D0 F7    FB65   29          BNE  STITLE
FB6E:60              30          RTS
```

```
FB6F:                   31 *
FB6F:AD F3 03           32 SETPWRC LDA  SOFTEV+1   ;ROUTINE TO CALCULATE THE 'FUNNY
FB72:49 A5              33         EOR  #$A5       ;COMPLEMENT' FOR THE RESET VECTOR
FB74:8D F4 03           34         STA  PWREDUP
FB77:60                 35         RTS
FB78:                   36 *
FB78:           FB78    37 VIDWAIT EQU  *          ;CHECK FOR A PAUSE (CONTROL-S).
FB78:C9 8D              38         CMP  #$8D       ;ONLY WHEN I HAVE A CR
FB7A:D0 18      FB94    39         BNE  NOWAIT     ;NOT SO, DO REGULAR
FB7C:AC 00 C0           40         LDY  KBD        ;IS KEY PRESSED?
FB7F:10 13      FB94    41         BPL  NOWAIT     ;NO.
FB81:C0 93              42         CPY  #$93       ;YES -- IS IT CTRL-S?
FB83:D0 0F      FB94    43         BNE  NOWAIT     ;NOPE - IGNORE
FB85:2C 10 C0           44         BIT  KBDSTRB    ;CLEAR STROBE
FB88:AC 00 C0           45 KBDWAIT LDY  KBD        ;WAIT TILL NEXT KEY TO RESUME
FB8B:10 FB      FB88    46         BPL  KBDWAIT    ;WAIT FOR KEYPRESS
FB8D:C0 83              47         CPY  #$83       ;IS IT CONTROL-C?
FB8F:F0 03      FB94    48         BEQ  NOWAIT     ;YES, SO LEAVE IT
FB91:2C 10 C0           49         BIT  KBDSTRB    ;CLR STROBE
FB94:4C FD FB           50 NOWAIT  JMP  VIDOUT     ;DO AS BEFORE
FB97:                   51 *
FB97:38                 52 ESCOLD  SEC             ;INSURE CARRY SET
FB98:4C 2C FC           53         JMP  ESC1
FB9B:A8                 54 ESCNOW  TAY             ;USE CHAR AS INDEX
FB9C:B9 48 FA           55         LDA  XLTBL-$C9,Y  ;TRANSLATE IJKM TO CBAD
FB9F:20 97 FB           56         JSR  ESCOLD     ;DO THE CURSOR MOTION
FBA2:20 21 FD           57         JSR  RDESC      ;GET IJKM, ijkm, ARROWS/RRA0981
FBA5:C9 CE              58 ESCNEW  CMP  #$CE       ;IS THIS AN 'N'?
FBA7:B0 EE      FB97    59         BCS  ESCOLD     ;'N' OR GREATER - DO IT!
FBA9:C9 C9              60         CMP  #$C9       ;LESS THAN 'I'?
FBAB:90 EA      FB97    61         BCC  ESCOLD     ;YES, SO DO OLD WAY
FBAD:C9 CC              62         CMP  #$CC       ;IS IT AN 'L'?
FBAF:F0 E6      FB97    63         BEQ  ESCOLD     ;DO NORMAL
FBB1:D0 E8      FB9B    64         BNE  ESCNOW     ;GO DO IT
FBB3:                   65 *
FBB3:           C006    66 SETSLOTCXROM EQU $C006  ;/RRA0981
FBB3:           C007    67 SETINTCXROM EQU $C007   ;/RRA0981
FBB3:           C015    68 RDCXROM EQU  $C015      ;/RRA0981
FBB3:                   69 *                              /RRA0981
FBB3:06                 70 VERSION DFB  $06        ;FOR IDCHECK/RRA0981
FBB4:                   71 *
FBB4:           FBB4    72 GOTOCX  EQU  *          ;/RRA0981
FBB4:2C 15 C0           73         BIT  RDCXROM    ;GET CURRENT STATE/RRA0981
FBB7:08                 74         PHP             ;SAVE ROMBANK STATE/RRA0981
FBB8:8D 07 C0           75         STA  SETINTCXROM  ;SET ROMS ON/RRA0981
FBBB:4C 00 C1           76         JMP  C1ORG      ;=>OFF TO CXSPACE/RRA0981
FBBE:                   77 *
FBBE:00                 78         DFB  0
FBBF:00                 79         DFB  0
FBC0:                   80 *
FBC0:E0                 81 ZIDBYTE DFB  $E0       ;//e ROM rev ID byte
FBC1:                   82 *
FBC1:48                 83 BASCALC PHA             ;CALC BASE ADDR IN BASL,H
FBC2:4A                 84         LSR  A         ;FOR GIVEN LINE NO.
```

```
FBC3:29 03          85          AND  #$03       ; 0<=LINE NO.<=$17
FBC5:09 04          86          ORA  #$04       ;ARG = 000ABCDE, GENERATE
FBC7:85 29          87          STA  BASH       ; BASH = 000001CD
FBC9:68             88          PLA             ; AND
FBCA:29 18          89          AND  #$18       ; BASL = EABAB000
FBCC:90 02   FBD0   90          BCC  BASCLC2
FBCE:69 7F          91          ADC  #$7F
FBD0:85 28          92 BASCLC2  STA  BASL
FBD2:0A             93          ASL  A
FBD3:0A             94          ASL  A
FBD4:05 28          95          ORA  BASL
FBD6:85 28          96          STA  BASL
FBD8:60             97          RTS
FBD9:               98 *
FBD9:C9 87          99 BELL1    CMP  #$87       ;BELL CHAR? (CONTROL-G)
FBDB:D0 12   FBEF  100          BNE  RTS2B      ; NO, RETURN.
FBDD:A9 40         101          LDA  #$40       ; YES...
FBDF:20 A8 FC      102          JSR  WAIT       ;DELAY .01 SECONDS
FBE2:A0 C0         103          LDY  #$C0
FBE4:A9 0C         104 BELL2    LDA  #$0C       ;TOGGLE SPEAKER AT 1 KHZ
FBE6:20 A8 FC      105          JSR  WAIT       ; FOR .1 SEC.
FBE9:AD 30 C0      106          LDA  SPKR
FBEC:88            107          DEY
FBED:D0 F5   FBE4  108          BNE  BELL2
FBEF:60            109 RTS2B    RTS
FBF0:              110 *
FBF0:A4 24         111 STORADV  LDY  CH         ;CURSOR H INDEX TO Y-REG
FBF2:91 28         112          STA  (BASL),Y   ;STORE CHAR IN LINE
FBF4:E6 24         113 ADVANCE  INC  CH         ;INCREMENT CURSOR H INDEX
FBF6:A5 24         114          LDA  CH         ; (MOVE RIGHT)
FBF8:C5 21         115          CMP  WNDWDTH    ;BEYOND WINDOW WIDTH?
FBFA:B0 66   FC62  116          BCS  CR         ; YES, CR TO NEXT LINE.
FBFC:60            117 RTS3     RTS             ; NO, RETURN.
FBFD:              118 *
FBFD:C9 A0         119 VIDOUT   CMP  #$A0       ;CONTROL CHAR?
FBFF:B0 EF   FBF0  120          BCS  STORADV    ; NO, OUTPUT IT.
FC01:A8            121          TAY             ;INVERSE VIDEO?
FC02:10 EC   FBF0  122          BPL  STORADV    ; YES, OUTPUT IT.
FC04:C9 8D         123          CMP  #$8D       ;CR?
FC06:F0 5A   FC62  124          BEQ  CR         ; YES.
FC08:C9 8A         125          CMP  #$8A       ;LINE FEED?
FC0A:F0 5A   FC66  126          BEQ  LF         ; IF SO, DO IT.
FC0C:C9 88         127          CMP  #$88       ;BACK SPACE? (CONTROL-H)
FC0E:D0 C9   FBD9  128          BNE  BELL1      ; NO, CHECK FOR BELL.
FC10:C6 24         129 BS       DEC  CH         ;DECREMENT CURSOR H INDEX
FC12:10 E8   FBFC  130          BPL  RTS3       ;IF POSITIVE, OK; ELSE MOVE UP.
FC14:A5 21         131          LDA  WNDWDTH    ;SET CH TO WINDOW WIDTH - 1.
FC16:85 24         132          STA  CH
FC18:C6 24         133          DEC  CH         ;(RIGHTMOST SCREEN POS)
FC1A:A5 22         134 UP       LDA  WNDTOP     ;CURSOR V INDEX
FC1C:C5 25         135          CMP  CV
FC1E:B0 DC   FBFC  136          BCS  RTS3       ;IF TOP LINE THEN RETURN
FC20:C6 25         137          DEC  CV         ;DECR CURSOR V INDEX
FC22:              138 *
```

```
FC22:A5 25          139 VTAB    LDA  CV          ;GET CURSOR V INDEX
FC24:85 28          140 VTABZ   STA  BASL       ;temporarily save Acc
FC26:98             141         TYA             ;and Y
FC27:A0 04          142         LDY  #$4        ;this is VTABZ call
FC29:D0 89   FBB4   143 GOTOCX1 BNE  GOTOCX     ;=> always perform call
FC2B:               144 *
FC2B:EA             145         NOP
FC2C:               146 *
FC2C:49 C0          147 ESC1    EOR  #$C0       ;ESC '@'?
FC2E:F0 28   FC58   148         BEQ  HOME       ; IF SO DO HOME AND CLEAR
FC30:69 FD          149         ADC  #$FD       ;ESC-A OR B CHECK
FC32:90 C0   FBF4   150         BCC  ADVANCE    ; A, ADVANCE
FC34:F0 DA   FC10   151         BEQ  BS         ; B, BACKSPACE
FC36:69 FD          152         ADC  #$FD       ;ESC-C OR D CHECK
FC38:90 2C   FC66   153         BCC  LF         ; C, DOWN
FC3A:F0 DE   FC1A   154         BEQ  UP         ; D, GO UP
FC3C:69 FD          155         ADC  #$FD       ;ESC-E OR F CKECK
FC3E:90 5C   FC9C   156         BCC  CLREOL     ; E, CLEAR TO END OF LINE
FC40:D0 BA   FBFC   157         BNE  RTS3       ; ELSE NOT F,RETURN
FC42:               158 *
FC42:       FC42    159 CLREOP  EQU  *          ;/RRA0981
FC42:A0 0A          160         LDY  #$A        ;CODE=CLREOP/RRA0981
FC44:D0 E3   FC29   161         BNE  GOTOCX1    ;DO 40/80 /RRA0981
FC46:               162 *
FC46:2C 1F C0       163 NEWVW   BIT  RD80VID    ;in 80 columns?
FC49:10 04   FC4F   164         BPL  NEWVW1     ;=>not 80 columns
FC4B:A0 00          165         LDY  #$0        ;Print a character
FC4D:F0 0B   FC5A   166         BEQ  GOTOCX3    ;through video firmware
FC4F:98             167 NEWVW1  TYA             ;get masked character
FC50:48             168         PHA             ;and set up for vidwait
FC51:20 78 FB       169         JSR  VIDWAIT    ;print the character
FC54:68             170         PLA             ;restore Acc
FC55:A4 35          171         LDY  YSAV1      ;and Y
FC57:60             172         RTS
FC58:               173 *
FC58:       FC58    174 HOME    EQU  *          ;/RRA0981
FC58:A0 05          175         LDY  #5         ;CODE=HOME/RRA0981
FC5A:4C B4 FB       176 GOTOCX3 JMP  GOTOCX     ;do 40/80
FC5D:               177 *
FC5D:EA             178         NOP
FC5E:EA             179         NOP
FC5F:EA             180         NOP
FC60:EA             181         NOP
FC61:EA             182         NOP
FC62:               183 *
FC62:A9 00          184 CR      LDA  #$00       ;CURSOR TO LEFT OF INDEX
FC64:85 24          185         STA  CH         ;(RET CURSOR H=0)
FC66:E6 25          186 LF      INC  CV         ;INCR CURSOR V. (DOWN 1 LINE)
FC68:A5 25          187         LDA  CV
FC6A:C5 23          188         CMP  WNDBTM     ;OFF SCREEN?
FC6C:90 B6   FC24   189         BCC  VTABZ      ;  NO, SET BASE ADDR
FC6E:C6 25          190         DEC  CV         ;DECR CURSOR V. (BACK TO BOTTOM)
FC70:               191 *
FC70:       FC70    192 SCROLL  EQU  *          ;/RRA0981
```

```
FC70:A0 06          193          LDY  #6         ;CODE=SCROLL/RRA0981
FC72:D0 B5   FC29   194          BNE  GOTOCX1     ;DO 40/80 /RRA0981
FC74:               195 *
FC74:               196 * Jump here to swap out ROMs
FC74:               197 * for interrupt handlers in peripheral cards
FC74:               198 *
FC74:8D 06 C0       199 IRQUSER STA  SETSLOTCXROM ;switch in slots
FC77:6C FE 03       200          JMP  ($3FE)     ;and jump to user
FC7A:               201 *
FC7A:               202 * IRQDONE ($C3F4) jumps here after interrupt
FC7A:               203 * because this cannot be done from $Cn00 space
FC7A:               204 *
FC7A:68             205 IRQDONE2 PLA              ;Fix $C800 space
FC7B:8D F8 07       206          STA  MSLOT      ;restore MSLOT
FC7E:C9 C1          207          CMP  #$C1       ;valid Cn?
FC80:90 0D   FC8F   208          BCC  IRQNOSLT
FC82:8D FF CF       209          STA  $CFFF      ;Deselect all $C800
FC85:A0 00          210          LDY  #0
FC87:A6 01          211          LDX  $1
FC89:85 01          212          STA  $1
FC8B:B1 00          213          LDA  ($0),Y     ;do $Cn00 reference
FC8D:86 01          214          STX  $1         ;fix zp location
FC8F:8D 07 C0       215 IRQNOSLT STA SETINTCXROM
FC92:4C 7C C4       216          JMP  IRQFIX     ;and restore the machine state
FC95:               217 *
FC95:90 02   FC99   218 DOCOUT1 BCC  DOCOUT2    ;don't mask controls
FC97:25 32          219          AND  INVFLG     ;apply inverse mask
FC99:4C F7 FD       220 DOCOUT2 JMP  COUTZ1     ;go back to COUT1
FC9C:               221 *
FC9C:       0000    222          DS   F8ORG+$49C-*,0 ;pad to clreol
FC9C:               223 *
FC9C:               224 * Note: bytes CLREOL and CLREOLZ ($38 and $18)
FC9C:               225 * are used by slot test at $FBB7.
FC9C:               226 *
FC9C:38             227 CLREOL  SEC              ;say it is EOL
FC9D:90             228          DFB  $90        ;'BCC' opcode
FC9E:18             229 CLREOLZ CLC              ;say it is EOLZ
FC9F:84 2A          230          STY  BAS2L      ;save Y in temp
FCA1:A0 07          231          LDY  #7         ;code=CLREOL
FCA3:B0 78   FD1D   232          BCS  GOTOCX2    ;do it
FCA5:C8             233          INY             ;code 8=CLREOLZ
FCA6:D0 75   FD1D   234          BNE  GOTOCX2
FCA8:               235 *
FCA8:38             236 WAIT     SEC             ;enter with count in A
FCA9:48             237 WAIT2    PHA             ;delay is:
FCAA:E9 01          238 WAIT3    SBC  #$01
FCAC:D0 FC   FCAA   239          BNE  WAIT3      ;13+11*A+5*A*A cycles
FCAE:68             240          PLA             ;@ 1.023 usec per cycle
FCAF:E9 01          241          SBC  #$01
FCB1:D0 F6   FCA9   242          BNE  WAIT2
FCB3:60             243          RTS
FCB4:               244 *
FCB4:E6 42          245 NXTA4    INC  A4L        ;INCR 2-BYTE A4
FCB6:D0 02   FCBA   246          BNE  NXTA1      ; AND A1
```

```
FCB8:E6 43          247          INC  A4H
FCBA:A5 3C          248 NXTA1    LDA  A1L        ;1NCR 2-BYTE A1.
FCBC:C5 3E          249          CMP  A2L        ; AND COMPARE TO A2
FCBE:A5 3D          250          LDA  A1H        ; (CARRY SET IF >=)
FCC0:E5 3F          251          SBC  A2H
FCC2:E6 3C          252          INC  A1L
FCC4:D0 02   FCC8   253          BNE  RTS4B
FCC6:E6 3D          254          INC  A1H
FCC8:60             255 RTS4B    RTS
FCC9:               256 *
FCC9:8D 07 C0       257 HEADR    STA  SETINTCXROM ;force internal ROM
FCCC:20 67 C5       258          JSR  XHEADER   ;write header
FCCF:4C C5 FE       259          JMP  RETCX1    ;force slots and return
FCD2:               260 *
FCD2:               261 * For the disassembler to be able to do I/O to slots,
FCD2:               262 * it cannot make calls to the I/O routines with the
FCD2:               263 * internal ROM switched in.  This stuff switches the
FCD2:               264 * ROM out for such instances.
FCD2:               265 *
FCD2:8D 06 C0       266 ERR3     STA  SETSLOTCXROM ;force slot ROM
FCD5:20 4A F9       267          JSR  PRBL2     ;tab to the error
FCD8:A9 DE          268          LDA  #$DE      ;to print a caret "^"
FCDA:20 ED FD       269          JSR  COUT      ;print it
FCDD:20 3A FF       270          JSR  BELL      ;and beep
FCE0:4C F0 FC       271          JMP  GETINST1  ;and go get next instruction
FCE3:               272 *
FCE3:8D 06 C0       273 DISLIN   STA  SETSLOTCXROM ;force slot ROM
FCE6:20 D0 F8       274          JSR  INSTDSP   ;disassemble the instruction
FCE9:20 53 F9       275          JSR  PCADJ     ;calculate new PC
FCEC:84 3B          276          STY  PCH       ;and update PC
FCEE:85 3A          277          STA  PCL
FCF0:               278 *
FCF0:               279 * NOTE: The entry point GETINST1 is hard-coded in
FCF0:               280 * BFUNC of the Video firmware.
FCF0:               281 *
FCF0:A9 A1          282 GETINST1 LDA #$A1       ;get mini-prompt "!"
FCF2:85 33          283          STA  PROMPT
FCF4:20 67 FD       284          JSR  GETLNZ    ;go get a line of input
FCF7:8D 07 C0       285          STA  SETINTCXROM ;force internal ROM
FCFA:4C 9C CF       286          JMP  DOINST    ;and return to CX space
FCFD:               287 *
FCFD:B9 00 02       288 UPMON    LDA  IN,Y      ;get character
FD00:C8             289          INY              ;point to next char
FD01:C9 E1          290          CMP  #$E1      ;is it lowercase?
FD03:90 06   FD0B   291          BCC  UPMON2    ;=>nope
FD05:C9 FB          292          CMP  #$FB      ;lowercase?
FD07:B0 02   FD0B   293          BCS  UPMON2    ;=>nope
FD09:29 DF          294          AND  #$DF      ;else upshift
FD0B:60             295 UPMON2   RTS
FD0C:               296 *
FD0C:A0 0B          297 RDKEY    LDY  #$B       ;code=RDKEY
FD0E:D0 03   FD13   298          BNE  RDKEY0    ;allow $FD10 entry
FD10:4C 18 FD       299 FD10     JMP  RDKEY1    ;if enter here, do nothing
FD13:20 B4 FB       300 RDKEY0   JSR  GOTOCX    ;display cursor
```

```
FD16:EA             301          NOP
FD17:EA             302          NOP
FD18:6C 38 00       303 RDKEY1   JMP  (KSWL)      ;GO TO USER KEY-IN
FD1B:               304 *
FD1B:         FD1B  305 KEYIN    EQU  *
FD1B:A0 03          306          LDY  #3           ;RDKEY/RRA0981
FD1D:4C B4 FB       307 GOTOCX2  JMP  GOTOCX       ;/RRA0981
FD20:EA             308          NOP               ;/RRA0981
FD21:               309 *
FD21:         FD21  310 RDESC    EQU  *
FD21:20 0C FD       311          JSR  RDKEY        ;GET A KEY
FD24:A0 01          312          LDY  #1           ;CODE=FIXIT
FD26:D0 F5    FD1D  313          BNE  GOTOCX2      ;=>always
FD28:               314 *
FD28:               315 * Flag to the video firmware that escapes are allowed.
FD28:               316 * This routine is called by RDCHAR which is called by
FD28:               317 * GETLN.  The high bit of MSLOT is set by all cards
FD28:               318 * that use the C800 space.
FD28:               319 *
FD28:4E F8 07       320 NEWRDKEY LSR MSLOT      ;<128 means escape allowed
FD2B:4C 0C FD       321          JMP  RDKEY       ;now read the key
FD2E:EA             322          NOP
FD2F:               323 *
FD2F:20 21 FD       324 ESC      JSR  RDESC       ;/RRA0981
FD32:20 A5 FB       325          JSR  ESCNEW      ;HANDLE ESC FUNCTION.
FD35:20 28 FD       326 RDCHAR   JSR  NEWRDKEY    ;Flag RDCHAR and read key
FD38:C9 9B          327          CMP  #$9B        ;'ESC'?
FD3A:F0 F3    FD2F  328          BEQ  ESC         ;  YES, DON'T RETURN.
FD3C:60             329          RTS
FD3D:               330 *
FD3D:A0 0F          331 PICKFIX  LDY  #$F         ;code = fixpick
FD3F:20 B4 FB       332          JSR  GOTOCX      ;do 80 column pick
FD42:A4 24          333          LDY  CH          ;restore Y
FD44:9D 00 02       334          STA  IN,X        ;and save new character
FD47:               335 *#03 AUTOST2          Auto-Start Monitor ROM 27-AUG-84        PAGE 20

FD47:20 ED FD       336 NOTCR    JSR  COUT        ;echo typed char
FD4A:EA             337          NOP
FD4B:EA             338          NOP
FD4C:EA             339          NOP
FD4D:BD 00 02       340          LDA  IN,X
FD50:C9 88          341          CMP  #$88        ;CHECK FOR EDIT KEYS
FD52:F0 1D    FD71  342          BEQ  BCKSPC      ;  - BACKSPACE
FD54:C9 98          343          CMP  #$98
FD56:F0 0A    FD62  344          BEQ  CANCEL      ;  - CONTROL-X
FD58:E0 F8          345          CPX  #$F8
FD5A:90 03    FD5F  346          BCC  NOTCR1      ;MARGIN?
FD5C:20 3A FF       347          JSR  BELL        ; YES, SOUND BELL
FD5F:E8             348 NOTCR1   INX              ;ADVANCE INPUT INDEX
FD60:D0 13    FD75  349          BNE  NXTCHAR
FD62:               350 *
FD62:A9 DC          351 CANCEL   LDA  #$DC        ;BACKSLASH AFTER CANCELLED LINE
FD64:20 ED FD       352          JSR  COUT
FD67:20 8E FD       353 GETLNZ   JSR  CROUT       ;OUTPUT 'CR'
```

```
FD6A:A5 33          354 GETLN    LDA  PROMPT     ;OUTPUT PROMPT CHAR
FD6C:20 ED FD       355          JSR  COUT
FD6F:A2 01          356          LDX  #$01       ;INIT INPUT INDEX
FD71:8A             357 BCKSPC   TXA
FD72:F0 F3   FD67   358          BEQ  GETLNZ     ;WILL BACKSPACE TO 0
FD74:CA             359          DEX
FD75:20 35 FD       360 NXTCHAR  JSR  RDCHAR
FD78:C9 95          361          CMP  #$95      ;USE SCREEN CHAR
FD7A:D0 08   FD84   362          BNE  ADDINP    ; FOR CONTROL-U
FD7C:B1 28          363          LDA  (BASL),Y  ;do 40 column pick
FD7E:2C 1F C0       364          BIT  RD80VID   ;80 columns?
FD81:30 BA   FD3D   365          BMI  PICKFIX   ;=>yes, fix it
FD83:EA             366          NOP
FD84:9D 00 02       367 ADDINP   STA  IN,X       ;ADD TO INPUT BUFFER
FD87:C9 8D          368          CMP  #$8D
FD89:D0 BC   FD47   369          BNE  NOTCR
FD8B:20 9C FC       370          JSR  CLREOL     ;CLR TO EOL IF CR
FD8E:A9 8D          371 CROUT    LDA  #$8D
FD90:D0 5B   FDED   372          BNE  COUT       ;(ALWAYS)
FD92:               373 *
FD92:A4 3D          374 PRA1     LDY  A1H        ;PRINT CR,A1 IN HEX
FD94:A6 3C          375          LDX  A1L
FD96:20 8E FD       376 PRYX2    JSR  CROUT
FD99:20 40 F9       377          JSR  PRNTYX
FD9C:A0 00          378          LDY  #$00
FD9E:A9 AD          379          LDA  #$AD       ;PRINT '-'
FDA0:4C ED FD       380          JMP  COUT
FDA3:               381 *
FDA3:A5 3C          382 XAM8     LDA  A1L
FDA5:09 07          383          ORA  #$07       ;SET TO FINISH AT
FDA7:85 3E          384          STA  A2L        ; MOD 8=7
FDA9:A5 3D          385          LDA  A1H
FDAB:85 3F          386          STA  A2H
FDAD:A5 3C          387 MO
D8CHK LDA  A1L
FDAF:29 07          388          AND  #$07
FDB1:D0 03   FDB6   389          BNE  DATAOUT
FDB3:20 92 FD       390 XAM      JSR  PRA1
FDB6:A9 A0          391 DATAOUT  LDA  #$A0
FDB8:20 ED FD       392          JSR  COUT       ;OUTPUT BLANK
FDBB:B1 3C          393          LDA  (A1L),Y
FDBD:20 DA FD       394          JSR  PRBYTE     ;OUTPUT BYTE IN HEX
FDC0:20 BA FC       395          JSR  NXTA1
FDC3:90 E8   FDAD   396          BCC  MOD8CHK    ;NOT DONE YET. GO CHECK MOD 8
FDC5:60             397 RTS4C    RTS             ;DONE.
FDC6:               398 *
FDC6:4A             399 XAMPM    LSR  A          ;DETERMINE IF MONITOR MODE IS
FDC7:90 EA   FDB3   400          BCC  XAM        ; EXAMINE, ADD OR SUBTRACT
FDC9:4A             401          LSR  A
FDCA:4A             402          LSR  A
FDCB:A5 3E          403          LDA  A2L
FDCD:90 02   FDD1   404          BCC  ADD
FDCF:49 FF          405          EOR  #$FF       ;FORM 2'S COMPLEMENT FOR SUBTRACT.
FDD1:65 3C          406 ADD      ADC  A1L
```

```
FDD3:48              407          PHA
FDD4:A9 BD           408          LDA  #$BD       ;PRINT '=', THEN RESULT
FDD6:20 ED FD        409          JSR  COUT
FDD9:68              410          PLA
FDDA:48              411 PRBYTE   PHA              ;PRINT BYTE AS 2 HEX DIGITS
FDDB:4A              412          LSR  A          ; (DESTROYS A-REG)
FDDC:4A              413          LSR  A
FDDD:4A              414          LSR  A
FDDE:4A              415          LSR  A
FDDF:20 E5 FD        416          JSR  PRHEXZ
FDE2:68              417          PLA
FDE3:29 0F           418 PRHEX    AND  #$0F       ;PRINT HEX DIGIT IN A-REG
FDE5:09 B0           419 PRHEXZ   ORA  #$B0       ;LSBITS ONLY.
FDE7:C9 BA           420          CMP  #$BA
FDE9:90 02   FDED    421          BCC  COUT
FDEB:69 06           422          ADC  #$06
FDED:                423 *
FDED:6C 36 00        424 COUT     JMP  (CSWL)     ;VECTOR TO USER OUTPUT ROUTINE
FDF0:                425 *
FDF0:48              426 COUT1    PHA             ;save original character
FDF1:C9 A0           427          CMP  #$A0       ;is it a control?
FDF3:4C 95 FC        428          JMP  DOCOUT1    ;=>mask if not; return to COUTZ1
FDF6:                429 *
FDF6:48              430 COUTZ    PHA             ;save original character
FDF7:84 35           431 COUTZ1   STY  YSAV1      ;save Y
FDF9:A8              432          TAY              ;save masked character
FDFA:68              433          PLA             ;get original char
FDFB:4C 46 FC        434          JMP  NEWVW      ;new entry to vidwait
FDFE:EA              435          NOP
FDFF:EA              436          NOP
FE00:                437 *
FE00:C6 34           438 BL1      DEC  YSAV
FE02:F0 9F   FDA3    439          BEQ  XAM8
FE04:CA              440 BLANK    DEX             ;BLANK TO MON
FE05:D0 16   FE1D    441          BNE  SETMDZ     ;AFTER BLANK
FE07:C9 BA           442          CMP  #$BA       ;DATA STORE MODE?
FE09:D0 BB   FDC6    443          BNE  XAMPM      ; NO; XAM, ADD, OR SUBTRACT.
FE0B:85 31           444 STOR     STA  MODE       ;KEEP IN STORE MODE
FE0D:A5 3E           445          LDA  A2L
FE0F:91 40           446          STA  (A3L),Y    ;STORE AS LOW BYTE AT (A3)
FE11:E6 40           447          INC  A3L
FE13:D0 02   FE17    448          BNE  RTS5       ;INCR A3, RETURN.
FE15:E6 41           449          INC  A3H
FE17:60              450 RTS5     RTS
FE18:                451 *
FE18:A4 34           452 SETMODE  LDY  YSAV       ;SAVE CONVERTED ':', '+',
FE1A:B9 FF 01        453          LDA  IN-1,Y     ; '-', '.' AS MODE
FE1D:85 31           454 SETMDZ   STA  MODE
FE1F:60              455          RTS
FE20:                456 *
FE20:A2 01           457 LT       LDX  #$01
FE22:B5 3E           458 LT2      LDA  A2L,X      ;COPY A2 (2 BYTES) TO
FE24:95 42           459          STA  A4L,X      ; A4 AND A5
FE26:95 44           460          STA  A5L,X
```

```
FE28:CA             461          DEX
FE29:10 F7   FE22  462          BPL  LT2
FE2B:60            463          RTS
FE2C:              464 *
FE2C:B1 3C         465 MOVE     LDA  (A1L),Y    ;MOVE (A1) THRU (A2) TO (A4)
FE2E:91 42         466          STA  (A4L),Y
FE30:20 B4 FC      467          JSR  NXTA4
FE33:90 F7   FE2C  468          BCC  MOVE
FE35:60            469          RTS
FE36:              470 *
FE36:B1 3C         471 VFY      LDA  (A1L),Y    ;VERIFY (A1) THRU (A2)
FE38:D1 42         472          CMP  (A4L),Y    ; WITH (A4)
FE3A:F0 1C   FE58  473          BEQ  VFYOK
FE3C:20 92 FD      474          JSR  PRA1
FE3F:B1 3C         475          LDA  (A1L),Y
FE41:20 DA FD      476          JSR  PRBYTE
FE44:A9 A0         477          LDA  #$A0
FE46:20 ED FD      478          JSR  COUT
FE49:A9 A8         479          LDA  #$A8
FE4B:20 ED FD      480          JSR  COUT
FE4E:B1 42         481          LDA  (A4L),Y
FE50:20 DA FD      482          JSR  PRBYTE
FE53:A9 A9         483          LDA  #$A9
FE55:20 ED FD      484          JSR  COUT
FE58:20 B4 FC      485 VFYOK    JSR  NXTA4
FE5B:90 D9   FE36  486          BCC  VFY
FE5D:60            487          RTS
FE5E:              488 *
FE5E:20 75 FE      489 LIST     JSR  A1PC       ;MOVE A1 (2 BYTES) TO
FE61:A9 14         490          LDA  #$14       ; PC IF SPEC'D AND
FE63:48            491 LIST2    PHA             ; DISASSEMBLE 20 INSTRUCTIONS.
FE64:20 D0 F8      492          JSR  INSTDSP
FE67:20 53 F9      493          JSR  PCADJ      ;ADJUST PC AFTER EACH INSTRUCTION.
FE6A:85 3A         494          STA  PCL
FE6C:84 3B         495          STY  PCH
FE6E:68            496          PLA
FE6F:38            497          SEC
FE70:E9 01         498          SBC  #$01       ;NEXT OF 20 INSTRUCTIONS
FE72:D0 EF   FE63  499          BNE  LIST2
FE74:60            500          RTS
FE75:              501 *
FE75:8A            502 A1PC     TXA             ;IF USER SPECIFIED AN ADDRESS,
FE76:F0 07   FE7F  503          BEQ  A1PCRTS    ; COPY IT FROM A1 TO PC.
FE78:B5 3C         504 A1PCLP   LDA  A1L,X      ;YEP, SO COPY IT.
FE7A:95 3A         505          STA  PCL,X
FE7C:CA            506          DEX
FE7D:10 F9   FE78  507          BPL  A1PCLP
FE7F:60            508 A1PCRTS RTS
FE80:              509 *
FE80:A0 3F         510 SETINV   LDY  #$3F       ;SET FOR INVERSE VID
FE82:D0 02   FE86  511          BNE  SETIFLG    ; VIA COUT1
FE84:A0 FF         512 SETNORM  LDY  #$FF       ;SET FOR NORMAL VID
FE86:84 32         513 SETIFLG  STY  INVFLG
FE88:60            514          RTS
```

```
FE89:                  515 *
FE89:A9 00             516 SETKBD  LDA  #$00        ;DO 'IN#0'
FE8B:85 3E             517 INPORT  STA  A2L         ;DO 'IN#AREG'
FE8D:A2 38             518 INPRT   LDX  #KSWL
FE8F:A0 1B             519         LDY  #KEYIN
FE91:D0 08    FE9B     520         BNE  IOPRT
FE93:                  521 *
FE93:A9 00             522 SETVID  LDA  #$00        ;DO 'PR#0'
FE95:85 3E             523 OUTPORT STA  A2L         ;DO 'PR#AREG'
FE97:A2 36             524 OUTPRT  LDX  #CSWL
FE99:A0 F0             525         LDY  #COUT1
FE9B:A5 3E             526 IOPRT   LDA  A2L         ;SET INPUT/OUTPUT VECTORS
FE9D:29 0F             527         AND  #$0F
FE9F:F0 04    FEA5     528         BEQ  IOPRT1
FEA1:09 C0             529         ORA  #<IOADR
FEA3:A0 00             530         LDY  #$00
FEA5:94 00             531 IOPRT1  STY  LOC0,X     ;save low byte of hook
FEA7:95 01             532         STA  LOC1,X     ;save acc
FEA9:A0 0E             533         LDY  #$E        ;code=PR#/IN#
FEAB:4C B4 FB          534 GOTOCX4 JMP  GOTOCX     ;perform call
FEAE:                  535 *
FEAE:EA                536         NOP
FEAF:00                537 CKSUMFIX DFB 0          ;/RRA0981
FEB0:                  538 *     ;-->CORRECT CKSUM AT CREATE TIME.
FEB0:4C 00 E0          539 XBASIC  JMP  BASIC      ;TO BASIC, COLD START
FEB3:4C 03 E0          540 BASCONT JMP  BASIC2     ;TO BASIC, WARM START
FEB6:20 75 FE          541 GO      JSR  A1PC       ;ADDR TO PC IF SPECIFIED
FEB9:20 3F FF          542         JSR  RESTORE    ;RESTORE FAKE REGISTERS
FEBC:6C 3A 00          543         JMP  (PCL)      ;AND GO!
FEBF:4C D7 FA          544 REGZ    JMP  REGDSP     ;GO DISPLAY REGISTERS
FEC2:60                545 TRACE   RTS             ;TRACE IS GONE
FEC3:EA                546         NOP
FEC4:60                547 STEPZ   RTS             ;STEP IS GONE
FEC5:                  548 *
FEC5:                  549 * Return here from GOTOCX
FEC5:                  550 *
FEC5:                  551 * NOTE: This address is hard-coded in BFUNC of the
FEC5:                  552 * video firmware
FEC5:                  553 *
FEC5:8D 06 C0          554 RETCX1  STA  SETSLOTCXROM ;restore bank
FEC8:60                555 RETCX2  RTS            ;simply return
FEC9:EA                556         NOP
FECA:                  557 *
FECA:4C F8 03          558 USR     JMP  USRADR     ;JUMP TO CONTROL-Y VECTOR IN RAM
FECD:                  559 *
FECD:A9 40             560 WRITE   LDA  #$40
FECF:8D 07 C0          561 WRT2    STA  SETINTCXROM ;set internal ROM
FED2:20 AA C5          562         JSR  WRITE2     ;write to tape
FED5:F0 2C    FF03     563         BEQ  RD2        ;=>always set slots, beep
FED7:                  564 *
FED7:                  565 * SEARCH is called with a Monitor command of the form
FED7:                  566 * HHLL<ADR1.ADR2  in which ADR1 < ADR2 and LL precedes HH
FED7:                  567 * in memory.  If HH is 0, or omitted (LL<ADR1.ADR2), then
FED7:                  568 * the single byte LL is searched for.  You cannot search for
```

```
FED7:                 569 * a two byte pair with a high byte of 0.  A list of all
FED7:                 570 * adresses containing the specified pattern is displayed.
FED7:                 571 *
FED7:A0 01            572 SEARCH  LDY  #1         ;set Y to 1
FED9:A5 43            573         LDA  A4H        ;is high byte 0?
FEDB:F0 04    FEE1    574         BEQ  SRCH1      ;=>yes, only look for low byte
FEDD:D1 3C            575         CMP  (A1L),Y    ;check high byte first
FEDF:D0 0A    FEEB    576         BNE  SRCH2      ;=>no match, try next byte
FEE1:88               577 SRCH1   DEY             ;match, now check low byte
FEE2:A5 42            578         LDA  A4L        ;get low byte
FEE4:D1 3C            579         CMP  (A1L),Y    ;does it match?
FEE6:D0 03    FEEB    580         BNE  SRCH2      ;=>no match, try next byte
FEE8:20 92 FD         581         JSR  PRA1       ;bytes match, print address
FEEB:20 BA FC         582 SRCH2   JSR  NXTA1      ;increment address
FEEE:90 E7    FED7    583         BCC  SEARCH     ;set Y back to 1
FEF0:60               584         RTS
FEF1:                 585 *
FEF1:A0 0D            586 MINI    LDY  #$D        ;dispatch mini-assembler call to
FEF3:20 B4 FB         587         JSR  GOTOCX     ;get internal ROM switched in
FEF6:                 588 *
FEF6:20 00 FE         589 CRMON   JSR  BL1         ;HANDLE CR AS BLANK
FEF9:68               590         PLA             ; THEN POP STACK
FEFA:68               591         PLA             ; AND RETURN TO MON
FEFB:D0 6C    FF69    592         BNE  MONZ       ;(ALWAYS)
FEFD:                 593 *
FEFD:8D 07 C0         594 READ    STA  SETINTCXROM ;set internal ROM
FF00:20 D1 C5         595         JSR  XREAD      ;do tape read
FF03:8D 06 C0         596 RD2     STA  SETSLOTCXROM ;restore slot CX
FF06:F0 32    FF3A    597         BEQ  BELL       ;read (write) ok, beep
FF08:D0 23    FF2D    598         BNE  PRERR      ;error, print message
FF0A:                 599 *
FF0A:C1 F0 F0 EC      600 TITLE   ASC  "Apple    //e"
FF13:                 601 *
FF13:                 602 * NNBL gets the next non-blank for the mini-assembler
FF13:                 603 *
FF13:20 FD FC         604 NNBL    JSR  UPMON      ;get char, upshift, INY
FF16:C9 A0            605         CMP  #$A0       ;is it blank?
FF18:F0 F9    FF13    606         BEQ  NNBL       ;yes, keep looking
FF1A:60               607         RTS
FF1B:                 608 *
FF1B:B0 6D    FF8A    609 LOOKASC BCS  DIG        ;it was a digit
FF1D:C9 A0            610         CMP  #$A0       ;check for quote (')
FF1F:D0 28    FF49    611         BNE  RTS6       ;nope, return char
FF21:B9 00 02         612         LDA  $200,Y     ;else get next char
FF24:A2 07            613         LDX  #7         ;for shifting asc into A2L and A2H
FF26:C9 8D            614         CMP  #$8D       ;was it CR?
FF28:F0 7D    FFA7    615         BEQ  GETNUM     ;yes, go handle CR
FF2A:C8               616         INY             ;advance index
FF2B:D0 63    FF90    617         BNE  NXTBIT     ;=>(always) into A2L and A2H
FF2D:                 618 *
FF2D:A9 C5            619 PRERR   LDA  #$C5        ;PRINT 'ERR', THEN FALL INTO
FF2F:20 ED FD         620         JSR  COUT        ; FWEEPER.
FF32:A9 D2            621         LDA  #$D2
FF34:20 ED FD         622         JSR  COUT
```

```
FF37:20 ED FD      623          JSR  COUT
FF3A:              624 *
FF3A:A9 87         625 BELL     LDA  #$87        ;MAKE A JOYFUL NOISE, THEN RETURN.
FF3C:4C ED FD      626          JMP  COUT
FF3F:              627 *
FF3F:A5 48         628 RESTORE  LDA  STATUS      ;RESTORE 6502 REGISTER CONTENTS
FF41:48            629          PHA              ; USED BY DEBUG SOFTWARE
FF42:A5 45         630          LDA  A5H
FF44:A6 46         631 RESTR1   LDX  XREG
FF46:A4 47         632          LDY  YREG
FF48:28            633          PLP
FF49:60            634 RTS6     RTS
FF4A:              635 *
FF4A:85 45         636 SAVE     STA  A5H         ;SAVE 6502 REGISTER CONTENTS
FF4C:86 46         637 SAV1     STX  XREG        ; FOR DEBUG SOFTWARE
FF4E:84 47         638          STY  YREG
FF50:08            639          PHP
FF51:68            640          PLA
FF52:85 48         641          STA  STATUS
FF54:BA            642          TSX
FF55:86 49         643          STX  SPNT
FF57:D8            644          CLD
FF58:60            645          RTS
FF59:              646 *
FF59:20 84 FE      647 OLDRST   JSR  SETNORM     ;SET SCREEN MODE
FF5C:20 2F FB      648          JSR  INIT        ; AND INIT KBD/SCREEN
FF5F:20 93 FE      649          JSR  SETVID      ; AS I/O DEVS.
FF62:20 89 FE      650          JSR  SETKBD
FF65:              651 *
FF65:D8            652 MON      CLD              ;MUST SET HEX MODE!
FF66:20 3A FF      653          JSR  BELL        ;FWEEPER.
FF69:A9 AA         654 MONZ     LDA  #$AA        ;'*' PROMPT FOR MONITOR
FF6B:85 33         655          STA  PROMPT
FF6D:20 67 FD      656          JSR  GETLNZ      ;READ A LINE OF INPUT
FF70:20 C7 FF      657          JSR  ZMODE       ;CLEAR MONITOR MODE, SCAN IDX
FF73:20 A7 FF      658 NXTITM   JSR  GETNUM      ;GET ITEM, NON-HEX
FF76:84 34         659          STY  YSAV        ; CHAR IN A-REG.
FF78:A0 17         660          LDY  #$17        ; X-REG=0 IF NO HEX INPUT
FF7A:88            661 CHRSRCH  DEY
FF7B:30 E8   FF65  662          BMI  MON         ;COMMAND NOT FOUND, BEEP & TRY AGAIN.
FF7D:D9 CC FF      663          CMP  CHRTBL,Y    ;FIND COMMAND CHAR IN TABLE
FF80:D0 F8   FF7A  664          BNE  CHRSRCH     ;NOT THIS TIME
FF82:20 BE FF      665          JSR  TOSUB       ;GOT IT! CALL CORRESPONDING SUBROUTINE
FF85:A4 34         666          LDY  YSAV        ;PROCESS NEXT ENTRY ON HIS LINE
FF87:4C 73 FF      667          JMP  NXTITM
FF8A:              668 *
FF8A:A2 03         669 DIG      LDX  #$03
FF8C:0A            670          ASL  A
FF8D:0A            671          ASL  A           ;GOT HEX DIGIT,
FF8E:0A            672          ASL  A           ; SHIFT INTO A2
FF8F:0A            673          ASL  A
FF90:0A            674 NXTBIT   ASL  A
FF91:26 3E         675          ROL  A2L
FF93:26 3F         676          ROL  A2H
```

```
FF95:CA             677           DEX             ;LEAVE X=$FF IF DIG
FF96:10 F8   FF90   678           BPL   NXTBIT
FF98:A5 31          679 NXTBAS    LDA   MODE
FF9A:D0 06   FFA2   680           BNE   NXTBS2    ;IF MODE IS ZERO,
FF9C:B5 3F          681           LDA   A2H,X     ; THEN COPY A2 TO A1 AND A3
FF9E:95 3D          682           STA   A1H,X
FFA0:95 41          683           STA   A3H,X
FFA2:E8             684 NXTBS2    INX
FFA3:F0 F3   FF98   685           BEQ   NXTBAS
FFA5:D0 06   FFAD   686           BNE   NXTCHR
FFA7:               687 *
FFA7:A2 00          688 GETNUM    LDX   #$00      ;CLEAR A2
FFA9:86 3E          689           STX   A2L
FFAB:86 3F          690           STX   A2H
FFAD:20 FD FC       691 NXTCHR    JSR   UPMON     ;get char, upshift, INY
FFB0:EA             692           NOP             ;INY now done in UPMON
FFB1:49 B0          693           EOR   #$B0
FFB3:C9 0A          694           CMP   #$0A
FFB5:90 D3   FF8A   695           BCC   DIG       ;BR IF HEX DIGIT
FFB7:69 88          696           ADC   #$88
FFB9:C9 FA          697           CMP   #$FA
FFBB:4C 1B FF       698           JMP   LOOKASC   ;check for ASCII input
FFBE:               699 *
FFBE:A9 FE          700 TOSUB     LDA   #<GO      ;DISPATCH TO SUBROUTINE, BY
FFC0:48             701           PHA             ; PUSHING THE HI-ORDER SUBR ADDR,
FFC1:B9 E3 FF       702           LDA   SUBTBL,Y  ; THEN THE LO-ORDER SUBR ADDR
FFC4:48             703           PHA             ; ONTO THE STACK,
FFC5:A5 31          704           LDA   MODE      ; (CLEARING THE MODE, SAVE THE OLD
FFC7:A0 00          705 ZMODE     LDY   #$00      ; MODE IN A-REG),
FFC9:84 31          706           STY   MODE
FFCB:60             707           RTS             ; AND 'RTS' TO THE SUBROUTINE!
FFCC:               708 *
FFCC:BC             709 CHRTBL    DFB   $BC       ;^C  (BASIC WARM START)
FFCD:B2             710           DFB   $B2       ;^Y  (USER VECTOR)
FFCE:BE             711           DFB   $BE       ;^E  (OPEN AND DISPLAY REGISTERS)
FFCF:9A             712           DFB   $9A       ;!  (enter mini-assembler)
FFD0:EF             713           DFB   $EF       ;V   (MEMORY VERIFY)
FFD1:C4             714           DFB   $C4       ;^K  (IN#SLOT)
FFD2:EC             715           DFB   $EC       ;S   (search for 2 bytes)
FFD3:A9             716           DFB   $A9       ;^P  (PR#SLOT)
FFD4:BB             717           DFB   $BB       ;^B  (BASIC COLD START)
FFD5:A6             718           DFB   $A6       ;'-' (SUBTRACTION)
FFD6:A4             719           DFB   $A4       ;'+' (ADDITION)
FFD7:06             720           DFB   $06       ;M   (MEMORY MOVE)
FFD8:95             721           DFB   $95       ;'<' (DELIMITER FOR MOVE, VFY)
FFD9:07             722           DFB   $07       ;N   (SET NORMAL VIDEO)
FFDA:02             723           DFB   $02       ;I   (SET INVERSE VIDEO)
FFDB:05             724           DFB   $05       ;L   (DISASSEMBLE 20 INSTRS)
FFDC:F0             725           DFB   $F0       ;W   (WRITE TO TAPE)
FFDD:00             726           DFB   $00       ;G   (EXECUTE PROGRAM)
FFDE:EB             727           DFB   $EB       ;R   (READ FROM TAPE)
FFDF:93             728           DFB   $93       ;':' (MEMORY FILL)
FFE0:A7             729           DFB   $A7       ;'.' (ADDRESS DELIMITER)
FFE1:C6             730           DFB   $C6       ;'CR' (END OF INPUT)
```

```
FFE2:99             731          DFB  $99         ;BLANK
FFE3:               732 *
FFE3:               733 * Table of low order monitor routine dispatch
FFE3:               734 * addresses.  High byte always $FE
FFE3:               735 *
FFE3:B2             736 SUBTBL   DFB  >BASCONT-1 ;^C   (BASIC warm start)
FFE4:C9             737          DFB  >USR-1     ;^Y   (not used)
FFE5:BE             738          DFB  >REGZ-1    ;^E   (open and display registers)
FFE6:F0             739          DFB  >MINI-1    ;mini assembler
FFE7:35             740          DFB  >VFY-1     ;V    (memory verify)
FFE8:8C             741          DFB  >INPRT-1   ;^K   (IN#SLOT)
FFE9:D6             742          DFB  >SEARCH-1  ;search for pattern
FFEA:96             743          DFB  >OUTPRT-1  ;^P   (PR#SLOT)
FFEB:AF             744          DFB  >XBASIC-1  ;^B   (BASIC cold start)
FFEC:17             745          DFB  >SETMODE-1 ;'-'  (subtraction)
FFED:17             746          DFB  >SETMODE-1 ;'+'  (addition)
FFEE:2B             747          DFB  >MOVE-1    ;M    (memory move)
FFEF:1F             748          DFB  >LT-1      ;'<'  (delim for move,vfy)
FFF0:83             749          DFB  >SETNORM-1 ;N    (set normal video)
FFF1:7F             750          DFB  >SETINV-1  ;I    (set inverse video)
FFF2:5D             751          DFB  >LIST-1    ;L    (disassemble 20 instrs)
FFF3:CC             752          DFB  >WRITE-1   ;W    (write to tape)
FFF4:B5             753          DFB  >GO-1      ;G    (execute program)
FFF5:FC             754          DFB  >READ-1    ;R    (read from tape)
FFF6:17             755          DFB  >SETMODE-1 ;':'  (memory fill)
FFF7:17             756          DFB  >SETMODE-1 ;'.'  (address delimiter)
FFF8:F5             757          DFB  >CRMON-1   ;'CR' (end of input)
FFF9:03             758          DFB  >BLANK-1   ;BLANK
FFFA:               759 *
FFFA:FB 03          760          DW   NMI        ;NON-MASKABLE INTERRUPT VECTOR
FFFC:62 FA          761          DW   RESET      ;RESET VECTOR
FFFE:FA C3          762          DW   IRQ        ;INTERRUPT REQUEST VECTOR
0000:                19          INCLUDE MINI
0000:                 1 *
0000:                 2 * Apple //e Mini Assembler
0000:                 3 *
0000:                 4 * Got mnemonic, check address mode
0000:                 5 *
C4C8:        C4C8     6          ORG  C3ORG+$1C8
C4C8:                 7 *
C4C8:20 13 FF         8 AMOD1    JSR  NNBL       ;get next non-blank
C4CB:84 34            9          STY  YSAV       ;save Y
C4CD:DD B4 F9        10          CMP  CHAR1,X
C4D0:D0 13   C4E5    11          BNE  AMOD2
C4D2:20 13 FF        12          JSR  NNBL       ;get next non-blank
C4D5:DD BA F9        13          CMP  CHAR2,X
C4D8:F0 0D   C4E7    14          BEQ  AMOD3
C4DA:BD BA F9        15          LDA  CHAR2,X    ;done yet?
C4DD:F0 07   C4E6    16          BEQ  AMOD4
C4DF:C9 A4           17          CMP  #$A4       ;if "$" then done
C4E1:F0 03   C4E6    18          BEQ  AMOD4
C4E3:A4 34           19          LDY  YSAV       ;restore Y
C4E5:18              20 AMOD2    CLC
C4E6:88              21 AMOD4    DEY
```

```
C4E7:26 44          22 AMOD3   ROL  A5L       ;shift bit into format
C4E9:E0 03          23         CPX  #$03
C4EB:D0 0D   C4FA   24         BNE  AMOD6
C4ED:20 A7 FF       25         JSR  GETNUM
C4F0:A5 3F          26         LDA  A2H       ;get high byte of address
C4F2:F0 01   C4F5   27         BEQ  AMOD5     ;=>
C4F4:E8             28         INX
C4F5:86 35          29 AMOD5   STX  YSAV1
C4F7:A2 03          30         LDX  #$03
C4F9:88             31         DEY
C4FA:86 3D          32 AMOD6   STX  A1H
C4FC:CA             33         DEX
C4FD:10 C9   C4C8   34         BPL  AMOD1
C4FF:60             35         RTS
C500:               36 *
CF3A:        CF3A   37         ORG  C8ORG+$73A
CF3A:               38 *
CF3A:               39 * Calculate offset byte for relative addresses
CF3A:               40 *
CF3A:E9 81          41 REL     SBC  #$81      ;calc relative address
CF3C:4A             42         LSR  A
CF3D:D0 14   CF53   43         BNE  GOERR     ;bad branch
CF3F:A4 3F          44         LDY  A2H
CF41:A6 3E          45         LDX  A2L
CF43:D0 01   CF46   46         BNE  REL1
CF45:88             47         DEY            ;point to offset
CF46:CA             48 REL1    DEX             ;displacement - 1
CF47:8A             49         TXA
CF48:18             50         CLC
CF49:E5 3A          51         SBC  PCL       ;subtract current PCL
CF4B:85 3E          52         STA  A2L       ;and save as displacement
CF4D:10 01   CF50   53         BPL  REL2      ;check page
CF4F:C8             54         INY
CF50:98             55 REL2    TYA             ;get page
CF51:E5 3B          56         SBC  PCH       ;check page
CF53:D0 40   CF95   57 GOERR   BNE  MINIERR   ;display error
CF55:               58 *
CF55:               59 * Move instruction to memory
CF55:               60 *
CF55:A4 2F          61 MOVINST LDY  LENGTH    ;get instruction length
CF57:B9 3D 00       62 MOV1    LDA  A1H,Y     ;get a byte
CF5A:91 3A          63         STA  (PCL),Y   ;and move it
CF5C:88             64         DEY
CF5D:10 F8   CF57   65         BPL  MOV1
CF5F:               66 *
CF5F:               67 * Display instruction
CF5F:               68 *
CF5F:20 48 F9       69         JSR  PRBLNK    ;print blanks to make ProDOS work
CF62:20 1A FC       70         JSR  UP        ;move up 2 lines
CF65:20 1A FC       71         JSR  UP
CF68:4C E3 FC       72         JMP  DISLIN    ;disassemble it, =>DOINST
CF6B:               73 *
CF6B:               74 * Compare disassembly of all known opcodes with
CF6B:               75 * the one typed in until a match is found
```

```
CF6B:                  76 *
CF6B:A5 3D             77 GETOP   LDA  A1H        ;get opcode
CF6D:20 8E F8          78         JSR  INSDS2     ;determine mnemonic index
CF70:AA                79         TAX             ;X = index
CF71:BD 00 FA          80         LDA  MNEMR,X    ;get right half of index
CF74:C5 42             81         CMP  A4L        ;does it match entry?
CF76:D0 13   CF8B      82         BNE  NXTOP      ;=>try next opcode
CF78:BD C0 F9          83         LDA  MNEML,X    ;get left half of index
CF7B:C5 43             84         CMP  A4H        ;does it match entry?
CF7D:D0 0C   CF8B      85         BNE  NXTOP      ;=>no, try next opcode
CF7F:A5 44             86         LDA  A5L        ;found opcode, check address mode
CF81:A4 2E             87         LDY  FORMAT     ;get addr. mode format for that opcode
CF83:C0 9D             88         CPY  #$9D       ;is it relative?
CF85:F0 B3   CF3A      89         BEQ  REL        ;=>yes, calc relative address
CF87:C5 2E             90         CMP  FORMAT     ;does mode match?
CF89:F0 CA   CF55      91         BEQ  MOVINST    ;=>yes, move instruction to memory
CF8B:C6 3D             92 NXTOP   DEC  A1H        ;else try next opcode
CF8D:D0 DC   CF6B      93         BNE  GETOP      ;=>go try it
CF8F:E6 44             94         INC  A5L        ;else try next format
CF91:C6 35             95         DEC  YSAV1
CF93:F0 D6   CF6B      96         BEQ  GETOP      ;=>go try next format
CF95:                  97 *
CF95:                  98 * Point to the error with a caret, beep, and fall
CF95:                  99 * into the mini-assembler.
CF95:                 100 *
CF95:A4 34            101 MINIERR LDY  YSAV       ;get position
CF97:98               102 ERR2    TYA
CF98:AA               103         TAX
CF99:4C D2 FC         104         JMP  ERR3       ;display error, =>DOINST
CF9C:                 105 *
CF9C:                 106 * Read a line of input.  If prefaced with " ", decode
CF9C:                 107 * mnemonic. If "$" do monitor command.  Otherwise parse
CF9C:                 108 * hex address before decoding mnemonic.
CF9C:                 109 *
CF9C:20 C7 FF         110 DOINST  JSR  ZMODE      ;clear mode
CF9F:AD 00 02         111         LDA  $200       ;get first char in line
CFA2:C9 A0            112         CMP  #$A0       ;if blank,
CFA4:F0 12   CFB8     113         BEQ  DOLIN      ;=>go attempt disassembly
CFA6:C9 8D            114         CMP  #$8D       ;is it return?
CFA8:D0 01   CFAB     115         BNE  GETIl      ;=>no, continue
CFAA:60               116         RTS             ;else return to Monitor
CFAB:                 117 *
CFAB:20 A7 FF         118 GETIl   JSR  GETNUM     ;parse hexadecimal input
CFAE:C9 93            119         CMP  #$93       ;look for "ADDR:"
CFB0:D0 E5   CF97     120 GOERR2  BNE  ERR2       ;no ":", display error
CFB2:8A               121         TXA             ;X nonzero if address entered
CFB3:F0 E2   CF97     122         BEQ  ERR2       ;no "ADDR", display error
CFB5:                 123 *
CFB5:20 78 FE         124         JSR  A1PCLP     ;move address to PC
CFB8:A9 03            125 DOLIN   LDA  #$03       ;get starting opcode
CFBA:85 3D            126         STA  A1H        ;and save
CFBC:20 13 FF         127 NXTCH   JSR  NNBL       ;get next non-blank
CFBF:0A               128         ASL  A          ;validate entry
CFC0:E9 BE            129         SBC  #$BE
```

```
CFC2:C9 C2          130          CMP  #$C2
CFC4:90 D1   CF97   131          BCC  ERR2       ;=>flag bad mnemonic
CFC6:               132 *
CFC6:               133 * Form mnemonic for later comparison
CFC6:               134 *
CFC6:0A             135          ASL  A
CFC7:0A             136          ASL  A
CFC8:A2 04          137          LDX  #$04
CFCA:0A             138 NXTMN    ASL  A
CFCB:26 42          139          ROL  A4L
CFCD:26 43          140          ROL  A4H
CFCF:CA             141          DEX
CFD0:10 F8   CFCA   142          BPL  NXTMN
CFD2:C6 3D          143          DEC  A1H        ;decrement mnemonic count
CFD4:F0 F4   CFCA   144          BEQ  NXTMN
CFD6:10 E4   CFBC   145          BPL  NXTCH
CFD8:A2 05          146          LDX  #$5        ;index into address mode tables
CFDA:20 C8 C4       147          JSR  AMOD1      ;do this elsewhere
CFDD:A5 44          148          LDA  A5L        ;get format
CFDF:0A             149          ASL  A
CFE0:0A             150          ASL  A
CFE1:05 35          151          ORA  YSAV1
CFE3:C9 20          152          CMP  #$20
CFE5:B0 06   CFED   153          BCS  AMOD7
CFE7:A6 35          154          LDX  YSAV1      ;get our format
CFE9:F0 02   CFED   155          BEQ  AMOD7
CFEB:09 80          156          ORA  #$80
CFED:85 44          157 AMOD7    STA  A5L        ;update format
CFEF:84 34          158          STY  YSAV       ;update position
CFF1:B9 00 02       159          LDA  $0200,Y    ;get next character
CFF4:C9 BB          160          CMP  #$BB       ;is it a ";"?
CFF6:F0 04   CFFC   161          BEQ  AMOD8      ;=>yes, skip comment
CFF8:C9 8D          162          CMP  #$8D       ;is it carriage return
CFFA:D0 B4   CFB0   163          BNE  GOERR2
CFFC:4C 6B CF       164 AMOD8    JMP  GETOP      ;get next opcode
CFFF:               165 *
CFFF:00             166          DFB  $00        ;byte for making CTOD checksum ok
```