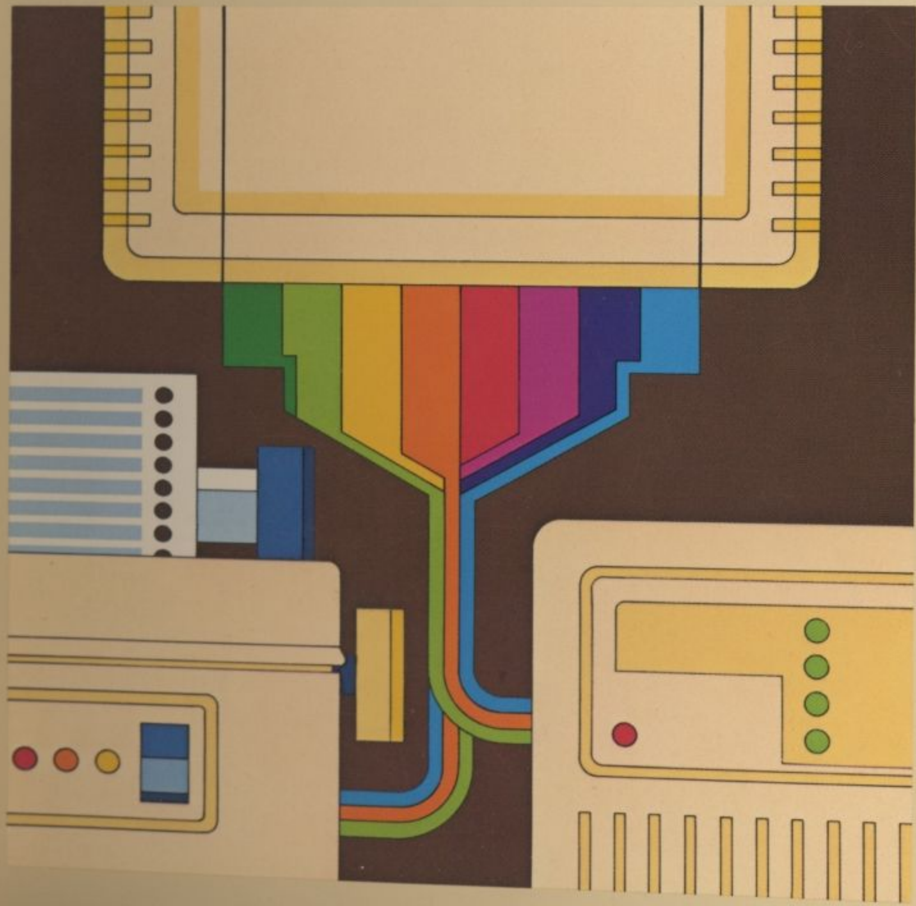
apple computer
10260 Bandley Drive
Cupertino, California 95014
(408) 996-1010

030-0270-A

Apple II

Super Serial Card

Installation and Operating Manual



Notice

Apple Computer Inc. reserves the right to make improvements in the product described in this manual at any time and without notice.

Disclaimer of All Warranties And Liability

Apple Computer Inc. makes no warranties, either express or implied, with respect to this manual or with respect to the software described in this manual, its quality, performance, merchantability, or fitness for any particular purpose. Apple Computer Inc. software is sold or licensed "as is." The entire risk as to its quality and performance is with the buyer. Should the programs prove defective following their purchase, the buyer (and not Apple Computer Inc., its distributor, or its retailer) assumes the entire cost of all necessary servicing, repair, or correction and any incidental or consequential damages. In no event will Apple Computer Inc. be liable for direct, indirect, incidental, or consequential damages resulting from any defect in the software, even if Apple Computer Inc. has been advised of the possibility of such damages. Some states do not allow the exclusion or limitation of implied warranties or liability for incidental or consequential damages, so the above limitation or exclusion may not apply to you.

Reorder Apple Product #A2L0044

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

Apple II

Super Serial Card

Installation and Operating Manual

Please read this manual before attempting to install the Super Serial Card in the Apple Computer. Incorrect installation could cause permanent damage to both the Super Serial Card and the Apple.

RADIO AND TELEVISION INTERFERENCE

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

This equipment has been tested and complies with the limits for a Class B computing device in accordance with the specifications in Subpart J of Part 15 of FCC rules. These rules are designed to provide reasonable protection against such interference in a residential installation. However, there is no guarantee that the interference will not occur in a particular installation.

You can determine whether your computer is causing interference by turning it off. If the interference stops, it was probably caused by the computer. If your computer does cause interference to radio or television reception, you can try to correct the interference by using one or more of the following measures:

- Turn the TV or radio antenna until the interference stops.

- Move the computer to one side or the other of the TV or radio.

- Move the computer farther away from the TV or radio.

- Plug the computer into an outlet that is on a different circuit from the TV or radio. (That is, make certain the computer and the TV or radio are on circuits controlled by different circuit breakers or fuses.)

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

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

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

TABLE OF CONTENTS

Chapter 3
COMMUNICATIONS MODE 21

Chapter 4
HOW THE SSC WORKS 37

Appendix A
FIRMWARE 49

Appendix B
APPLE INTERFACE CARD EMULATION 91

PREFACE

The Super Serial Card (SSC) provides a two-way serial interface to a wide variety of devices, including printers, terminals, plotters, and other computers. All these devices can be connected to the SSC either directly or via modem.

The SSC replaces both the P8 and P8A variety of Apple II Serial Interface Card, although it does not manipulate all specific Apple II memory locations in the same way. The SSC also replaces the Apple II Communications Card, and supports Terminal Mode. Finally, the SSC supports Apple II parallel interface card software commands.

The Super Serial Card conforms to the Electronic Industries Association (EIA) interface definitions A through E. (To obtain a copy of the EIA RS-232-C Standard, write to the EIA Engineering Department, Electronics Industries Association, 2ØØ1 Eye Street, N.W., Washington, D.C. 2ØØØ6.)

The SSC can be configured to the attached external device in three ways: (1) by setting switches on the card itself, (2) by typing in commands at the keyboard under the Monitor, Integer BASIC, Applesoft or DOS, or (3) by issuing commands from assembly language, BASIC or Pascal programs. The SSC can be configured and operated by programs in Integer BASIC, APPLESOFT, Pascal, and assembly language.

How you prepare, install and use the Super Serial Card depends on what you connect to it:

- Read Chapter 1 for unpacking and cable clamp preparation instructions.
- If you are going to connect a printer, terminal or some other device directly to the SSC, then read the first four sections of Chapter 2. (Many commonly used switch settings are listed in Table 2-1 for your convenience.) You only need to read the section Printer Mode Commands of Chapter 2 if you need special commands to change the SSC's characteristics.
- If you are going to connect a device to the SSC via a modem or similar communications equipment, then read the first four sections of Chapter 3. (Switch settings for many Communications Mode applications are listed in Table 3-1.) You only need to read the section Communications Mode Commands of Chapter 3 if you need special commands to change the SSC's characteristics.
- If you want to use the Apple II as an unintelligent terminal connected via a modem, read the section Terminal Mode of Chapter 3.
- Troubleshooting Hints are discussed in Appendix E.

The SSC also emulates ("imitates") the Apple II Serial Interface Card (both the P8 and P8A varieties), and supports many of the software commands used by the Apple II parallel printer interface card and the Apple II Communications Card. These are all discussed in Appendix B.

Chapter 4 explains how the SSC works, both in everyday terms (Serial Data Communication Simply Explained) and from an engineering viewpoint (Theory of Operation). The Theory of Operation section is keyed to the schematic diagram in Appendix C. Chapter 4 also contains a section on SSC modes and configurations.

Appendix A discusses SSC firmware and its entry points in the SSC ROM, as well as the Apple II memory locations the firmware uses.

Appendix C contains SSC specifications and connector pin assignments, and its schematic diagram.

Appendix D lists the ASCII codes and their equivalents. Appendix E has troubleshooting hints. Appendix F explains the SSC error codes.

A glossary explains the meaning of most important terms as they apply to the SSC.

The Reference Card summarizes the switch settings and commands for the SSC Printer Mode and Communications Mode.

There are three symbols that set off information of special importance:

This symbol points to a paragraph that contains especially useful information.

Watch out! This symbol precedes a paragraph that warns you to be careful.

This symbol precedes a warning that you are about to harm hardware or destroy data.

CHAPTER 1
GETTING STARTED

This chapter takes you through the first steps of getting acquainted with your Super Serial Card (SSC). After unpacking the SSC and examining it, you will assemble the short internal cable (if it is not already assembled) that connects the 1Ø-pin cable socket on the SSC to the 25-pin socket at the back of the Apple II case.

UNPACKING

As you unpack your Super Serial Card (Figure 1-1), check the contents against the items described on the packing list.

Fill out the pre-addressed warranty card and mail it in. If any items are missing, contact the dealer you purchased the SSC from.

You will need a shielded external cable (not provided as part of the SSC package) to connect the external device--the printer, modem, terminal, or other computer--to your Apple II. Suitable cables are available through your Apple dealer.

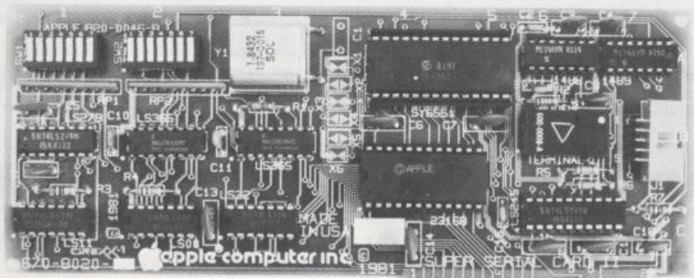


Figure 1-1. Photo of the Super Serial Card

A CLOSE LOOK

Let's examine the Super Serial Card for a moment. Pick up the SSC carefully and put it on a flat surface oriented as shown in Figure 1-1. Now use Figure 1-2 to help identify the chief parts of the SSC. Those that you will have to deal with as you prepare it for installation are:

- The jumper block. This ordinarily points toward the word TERMINAL; if you attach a modem to the SSC, you will turn this around so the arrow points toward the word MODEM (Chapter 3).

- The switches. The left group is numbered from SW1-1 through SW1-7; the right group is numbered from SW2-1 through SW2-7. You can see the characters "SW1" and "SW2" printed on the SSC.

- The edge connector. It is important not to touch the gold fingers on this connector: they must make a clean electrical contact in the Apple II connector slot when you install the SSC (Chapter 2 or Chapter 3).

- The cable socket. The next section of this chapter explains how to install the short internal cable between the SSC and the Apple II case.

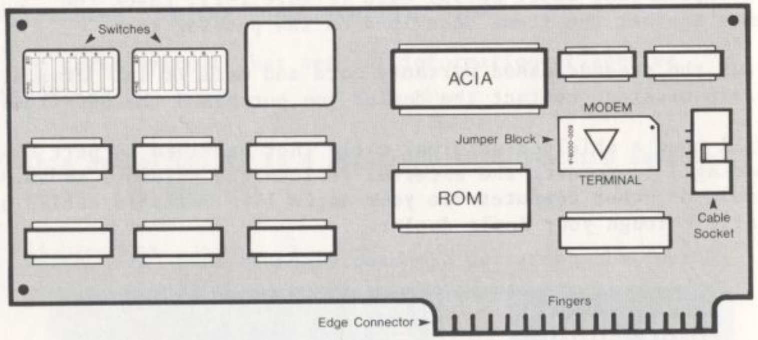


Figure 1-2. Line Drawing of the SSC

PREPARING CABLE AND CLAMP ASSEMBLY

Before preparing and installing the SSC, you may need to prepare the clamp assembly for the internal cable that will go from the SSC to the back of the Apple II's case. The components of this clamp assembly are shown in Figure 1-3. If these components are already assembled, skip to the next section, Attaching the Internal Cable to the SSC.

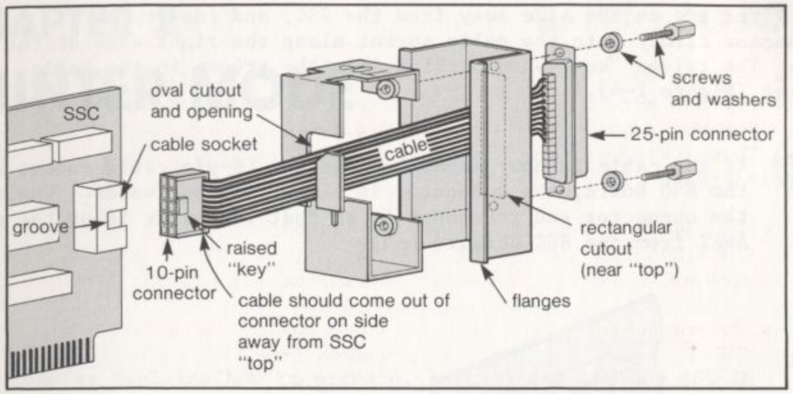


Figure 1-3. Components of Internal Cable and Clamp Assembly

Lay the short cable down as shown in Figure 1-3. Pick up the clamp piece that has the word TOP stamped on one end. Hold this clamp piece with the word TOP facing away from you, and the oval cutout toward the smaller connector on the cable. Bend the cable slightly, and insert it into the oval cutout through the opening; then straighten the cable in the cutout so that it moves easily.

The other clamp piece has flanges (Figure 1-3) and a rectangular opening that is closer to one end (its top end) than to the other. Hold this clamp piece with its top end away from you and its flanges facing the 25-pin connector end of the cable. Then tilt the connector and feed it completely through the rectangular cutout.

Now slide the two clamp pieces all the way down the cable until they are right up against the 25-pin connector, and their screw holes line up with the connector's screw holes. Slide the washers onto the screws and then thread the screws a couple of turns into the lined-up holes. Don't screw them in very far.

ATTACHING INTERNAL CABLE TO SCC

This step in the preparation of your Super Serial Card is simple to do, but you must do it carefully.

It is very important to connect the cable to the SSC correctly. Improper connection of the cable to the SSC may result in damage to the Apple and the SSC; such damage is NOT covered by your warranty.

Lay the SSC down on a flat surface, component-side up and gold fingers at the lower right. Examine the 1Ø-pin end of the cable: the wires come out of the SIDE of the connector--the same side as the raised "key" in the plastic (Figure 1-3). Hold the connector so the wires are on the side away from the SSC, and insert the connector firmly into the cable socket along the right edge of the SSC. The raised "key" should slide into the groove in the cable socket (Figure 1-4).

If the cable is now jammed between the 1Ø-pin cable socket and the SSC board, the connector is plugged in backwards. Unplug the connector and reconnect it so that the cable is on the side AWAY from the SSC (Figure 1-5).

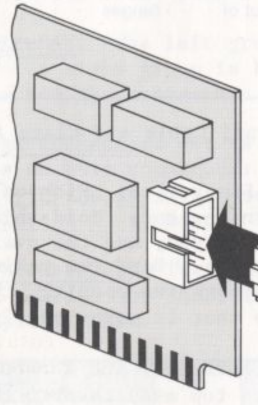


Figure 1-4. Sliding the "Key" into the Groove

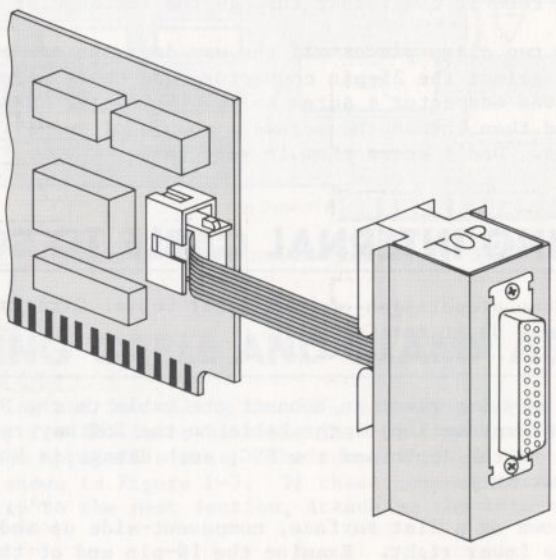


Figure 1-5. Internal Cable Attached Correctly to SSC

CHAPTER 2

PRINTER MODE

This chapter explains how to prepare, install and use the SSC in Printer Mode, and change the SSC's activities via commands.

PREPARING THE SSC FOR PRINTER MODE

The SSC is ready to operate in Printer Mode when the jumper block and switches SW1-5 and SW1-6 are correctly positioned (Figure 2-1).

If the triangle on the jumper block is pointing down toward the word MODEM, remove the block (using an IC Extractor, if necessary) and carefully reinsert it so the triangle is pointing toward TERMINAL.

Using a pointed object, set switch SW1-5 OFF and switch SW1-6 ON as shown in Figure 2-1.

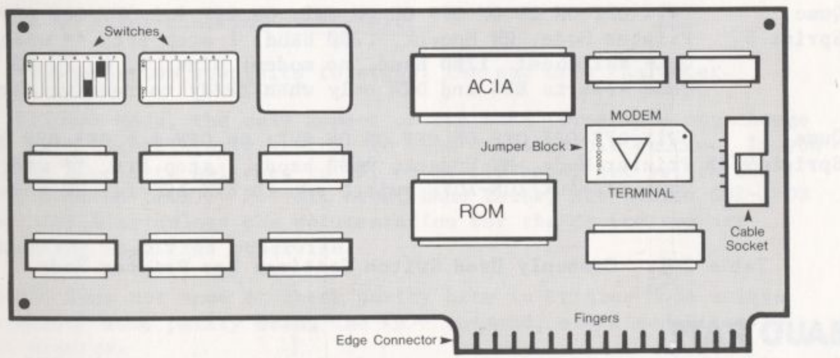


Figure 2-1. SSC Set for Printer Mode

When the jumper block is pointing toward TERMINAL, it is acting as a Modem Eliminator. Therefore, DO NOT connect a separate Modem Eliminator, or it will cancel the effect of the jumper block, and the attached device will not work.

SETTING THE SWITCHES

Use a pointed object, such as the tip of a ballpoint pen, to flip the appropriate tiny switches on the SSC. A switch is ON when the top of the switch rocker is pushed in, and OFF when the bottom is in. The following subsections explain what settings to use.

COMMONLY USED SETTINGS

Table 2-1 lists the switch settings you can use for direct connection, via the SSC, of some commonly used printers. Most printers can use any one of several setups.

Printer	Switch Settings, Cable Connections, Other Information
IDS 56Ø Paper Tiger	SW1: OFF OFF OFF ON OFF ON ON SW2: ON ON * * OFF OFF OFF Printer Mode, HW Hndshk, 96ØØ baud, 1 stop bit, ** width IDS SW1: - - - ON ON OFF OFF SW2: - - - - OFF - - SSC/IDS pins: 3/3, 7/7, 2Ø/2Ø; all IDS jumpers removed
NEC 551Ø Spinwriter	SW1: OFF ON ON ON OFF OFF OFF SW2: ON ON * * OFF OFF ON P8A Mode, ETX/ACK, 12ØØ baud, 1 stop bit, ** line width NEC switches: OFF ON OFF OFF OFF OFF ON ON SSC/NEC pins: 2/2, 3/3, 7/7, 2Ø/6&8; 4&5 tied on NEC end May need keystroke to force first ETX after power-up.
NEC 551Ø Spinwriter	SW1: OFF ON ON ON OFF ON OFF SW2: ON ON * * OFF OFF ON Printer Mode, hardware handshake, rest same as above NEC switches: OFF ON OFF OFF OFF OFF ON ON SSC/NEC pins: 3/3, 6/6&8, 7/7, 2Ø/2Ø; 4&5 NOT tied
Qume Sprint 5	SW1: OFF ON ON ON OFF ON ON SW2: ON OFF * * OFF OFF OFF Printer Mode, HW Hndshk, 12ØØ baud, 1 stop bit, ** width Qume switches: 12ØØ baud, no modem; pins: 3, 4, 7, 2Ø Qume asserts RTS and DTR only when ready to receive data
Qume Sprint 9/35	SW1: OFF OFF OFF ON OFF ON ON SW2: ON OFF * * OFF OFF OFF Printer Mode, HW Hndshk, 96ØØ baud, 1 stop bit, ** width Qume ETX-ACK/XON-XOFF switch set to ETX-ACK for HW Hndshk

Table 2-1. Commonly Used Switch Settings for Printer Mode

BAUD RATE

No matter what type of printer or terminal you connect to the SSC, the SSC is going to pass information between the Apple II and the device at a certain prearranged speed, called the baud rate. Since the Apple II can usually send and receive information faster than what is connected to it, the simplest way to determine the baud rate is to consult the user manual for the device you will connect. Find out what rate is the fastest the device can handle (up to 19,2ØØ baud). Once you know this, you are ready to set the baud rate switches on the SSC.

Baud	SW1-1	SW1-2	SW1-3	SW1-4	Baud	SW1-1	SW1-2	SW1-3	SW1-4
5Ø	ON	ON	ON	OFF	12ØØ	OFF	ON	ON	ON
75	ON	ON	OFF	ON	18ØØ	OFF	ON	ON	OFF
11Ø*	ON	ON	OFF	OFF	24ØØ	OFF	ON	OFF	ON
135**	ON	OFF	ON	ON	36ØØ	OFF	ON	OFF	OFF
15Ø	ON	OFF	ON	OFF	48ØØ	OFF	OFF	ON	ON
3ØØ	ON	OFF	OFF	ON	72ØØ	OFF	OFF	ON	OFF
6ØØ	ON	OFF	OFF	OFF	96ØØ	OFF	OFF	OFF	ON
(* 1Ø9.92)		(** 134.58)			192ØØ	OFF	OFF	OFF	OFF

Table 2-2. Baud Rate Switch Settings

Make sure the printer or terminal you connect is set (with its own switches, dials or thumb wheels) to the SAME baud rate! If you don't, the SSC will send and receive unrecognizable garbage.

DATA FORMAT AND PARITY

The SSC sends each character (such as a "3" or an "F" or a Carriage Return) as a string of zeroes and ones (bits). The way it can send a character in Printer Mode, using switch settings, is this:

- first a single start bit to signal to the printer or terminal that a character is coming;
- then a string of 8 data bits representing the character;
- no error-checking parity bit;
- one or two stop bits to signal the end of a character.

For Printer Mode, the only aspect of the data format you can change with switch settings is whether to send one stop bit or two. If you set the baud rate switches to 5Ø, 75 or 11Ø baud, set switch SW2-1 OFF (two stop bits). For all other baud rates, set switch SW2-1 ON (one stop bit) unless the documentation for the device you are connecting specifies otherwise.

The SSC does not send or check parity bits in Printer Mode unless you select some parity using the <n>P command, explained later in this chapter.

CARRIAGE RETURN DELAY

If you connect a slow printer to the SSC, and it has no handshaking capability, you may need to set switch SW2-2 ON to cause the Apple II to wait 1/4 second after a Carriage Return (<CR>). This gives the print head assembly time to reposition to the beginning of the next line. Otherwise, set switch SW2-2 OFF (no delay).

Additional delay values (32 ms and 2 s) are available via the <n>C command described later in this chapter.

LINE WIDTH AND VIDEO ON/OFF

Switches SW2-3 and SW2-4 determine the printer or terminal line width and also turn the Apple II video screen on or off.

If you are connecting a printer to the SSC, select the appropriate switch settings for the number of characters the printer can fit on a line. If you set the line width to 4Ø, the Apple II video screen is turned on, since it too can display 4Ø characters per line, and so can display an exact replica of what is being printed.

If you plan to connect a terminal to the SSC, set the switches for the number of characters the terminal screen can display on a line--usually 72 or 8Ø. For these line widths, the Apple II video screen is off.

Line Width	Video Screen	SW2-3	SW2-4
4Ø char/line	on	ON	ON
72 char/line	off	ON	OFF
8Ø char/line	off	OFF	ON
132 char/line	off	OFF	OFF

Table 2-3. Line Width and Video Switch Settings

The switch settings that turn off the Apple II video screen take effect only after PR# under BASIC or DOS. <CTRL-I> commands are still recognized, and cause the message APPLE SSC: to appear on the Apple II video screen.

GENERATE ⟨LF⟩ OUT

If you are connecting a printer to the SSC, check the printer's user manual to see if it automatically generates a linefeed (<LF>) after a carriage return (<CR>). If it does not, set switch SW2-5 ON.

If your printer does automatically generate a linefeed after a carriage return, or if you are connecting some other device that does not need automatic linefeed generation, set switch SW2-5 OFF.

SPECIAL SWITCHES

Switch SW2-6 controls forwarding of interrupts to the Apple II. Since the Apple II and II+ do not handle interrupts, set SW2-6 OFF.

Normally, switch SW1-7 is ON and switch SW2-7 is OFF. In the rare cases where the device uses pin 19, Secondary Clear To Send, in place of pin 4 or 2Ø, Clear To Send, set SW1-7 OFF and SW2-7 ON.

Your Super Serial Card is now ready to install and use in Printer Mode.

INSTALLATION PROCEDURE

This section explains how to install the SSC and its internal cable in the Apple II. If the cable clamp is not already assembled, do so now, following the instructions given in Chapter 1.

Before connecting or disconnecting anything on the Apple, turn off the power with the switch at the back left corner of the Apple case. THIS IS ABSOLUTELY NECESSARY. If you try to connect or disconnect anything from the inside of your Apple when the power is on, you are likely to damage the circuits.

Do not unplug the Apple, just turn it off. If you unplug the Apple, you will isolate it from earth ground and leave it vulnerable to static discharges.

Remove the Apple cover by pulling up on the two back corners of the cover until the two corner fasteners pop apart. Slide the cover back until it is free of the case and then lift the cover off.

Look inside the Apple and locate the power supply case--the rectangular metal box along the left inside the Apple II. To avoid damaging the SSC, touch the power supply case with one hand; this discharges any static charge that may be on your clothes or body.

Along the back inside edge of the Apple you will see eight long narrow slots called <u>connector slots</u>. The connector slots are numbered from Ø at the left to 7 at the right. The numbers are printed along the back edge behind the connector slots. For use with Pascal, install the SSC in slot #1 for a printer, or slot #3 for a terminal. For use with BASIC, install the SSC in any slot from #1 through #7.

Handle the Super Serial Card as you would handle an expensive phonograph record. Grasp it only by the corners or edges, and do not touch the components or pins, especially the gold fingers on the edge connector.

There are three deep notches along the back of the Apple II case. Temporarily set the SSC down near the desired slot. Then take the clamp assembly and slide it down into the notch closest to the slot that the SSC will be in. Tighten the screws until the connector assembly can no longer be moved in the opening.

Grasp the upper corners of the SSC and insert the gold fingers of the edge connector into the slot in the back of the Apple, rear edge first. Gently push the front edge of the card down until it is level and firmly seated.

Note that the outer ends of the screws in the clamp assembly can act as nuts. They are threaded and can receive screws from the printer or terminal connector, to ensure a good connection with the Apple.

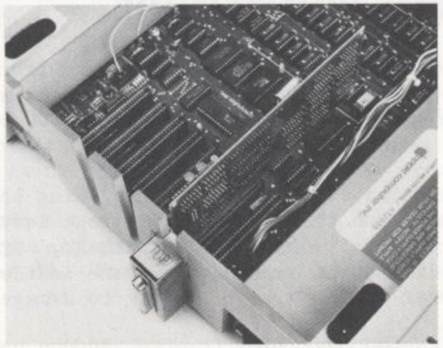


Figure 2-2. SSC in Slot #1 and Clamp Assembly in Notch

Slide the Apple case top plate in place and press down on the rear corners until the corner fasteners pop into place. The Super Serial Card is now installed.

EXTERNAL CABLE AND CONNECTOR

The SSC cable connector you installed in the notch is a standard DB-25 connector with 25 pins. Ten pins of the connector are connected internally to the SSC. Connector pin assignments are listed in Appendix C.

You will need a cable to connect your external device to the SSC connector on the Apple II. Shielded cables with 25-pin connectors on one end are available from your Apple dealer.

The cable must have internal shielding, with the shielding properly terminated at both ends, to prevent electromagnetic interference to nearby radios, television sets, and communication equipment. This shielding is necessary for the system to comply with Class B Federal Communications Commission limits as defined by Subpart J of Part 15 of the FCC rules. Unshielded cables are not recommended.

Make sure that all devices are connected to the same grounded AC power circuit (three-wire wall outlet) as the Apple II. Connecting ungrounded equipment to your Apple II can cause severe electrical damage.

USING THE SSC IN PRINTER MODE

Printer Mode allows you to use the SSC with a local (that is, directly connected) printer or terminal, as well as other local serial devices. After installing the SSC, you can control its operation from a BASIC, Pascal or assembly-language program, or even directly from the keyboard. The two parts of this section explain the easiest way to get the SSC up and running from the keyboard with a printer or terminal.

WITH A PRINTER

To use the SSC with a printer, do the following:

- Make sure the jumper block points toward TERMINAL.
- Under BASIC or DOS, boot the Apple II and then type in PR#s to send output to the printer (with the SSC in slot s).
- Under Pascal, boot the Apple II and then use the F(iler T(ransfer command to send output data to #6: or PRINTER: (with the SSC in slot #1).
- If the printer doesn't work, refer to Appendix E for troubleshooting hints, or consult your Apple dealer.

WITH A TERMINAL

To use the SSC with a terminal, do the following:

- Make sure the jumper block points toward TERMINAL.
- Under BASIC or DOS, boot the Apple II and then type in PR#s and IN#s to route both input and output through the terminal (with the SSC in slot #s).
- Under Pascal, boot the Apple II and then use the terminal as the input/output console (with the SSC in slot #3).
- If the terminal doesn't work, refer to Appendix E for troubleshooting hints, or consult your Apple dealer.

PRINTER MODE COMMANDS

You can issue any of the commands described in this section by embedding them in a computer program. Under BASIC, DOS or the Apple Monitor, you can also enter them directly at the Apple (or terminal) keyboard.

In a BASIC program, put the control character and command in a PRINT statement. In a Pascal program, issue the command in a WRITE or WRITELN statement.

When you enter the command character (usually <CTRL-I>; see below), the prompting message APPLE SSC: appears on the display screen. Subsequent characters, up to <RETURN>, will be interpreted as an SSC command. Pressing the left arrow key before pressing <RETURN> cancels the command and causes the APPLE SSC: prompt to reappear.

Many of these commands override the physical switch settings on the SSC. This makes it unnecessary to open the Apple II case and manually flip the SSC switches. To change the values back to the physical switch settings, reboot or reset the Apple II, or type in the Reset command described below.

COMMAND FORMATS

All commands are preceded by the Printer Mode command character (usually <CTRL-I>, see below) and followed by <RETURN>. The notation <CTRL-I> means "hold down the CTRL key while pressing I." There are three types of command formats:

- a number <n> followed by an uppercase letter (for example, 4D to set Data Format 4)
- simply an uppercase letter (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 generation of linefeed characters)

The allowable range of <n> is given in each command description (next section). The choice of Enable or Disable is indicated as <E/D>.

The underscore character (_) before the <E/D> in Enable/Disable commands is merely a reminder 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, X(OFF E(nable is the same as X E. The SSC ignores invalid commands.

THE COMMAND CHARACTER

The normal command character in Printer Mode is <CTRL-I> (decimal 9; Appendix D). You can send the command character itself through the SSC by typing it twice in a row: <CTRL-I><CTRL-I>; no <RETURN> 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.

If you want to change the command character from <CTRL-I> to <CTRL-something else>, type <CTRL-I><CTRL-something else>. For example, to change the command character to <CTRL-W>, type <CTRL-I><CTRL-W>. To change back, type <CTRL-W><CTRL-I>. No <RETURN> is required after either of these commands.

The command character <CTRL-I> is ASCII code 9. Here is how to generate this character in BASIC and Pascal:

```
Integer BASIC:      PRINT "*command"  *embedded <CTRL-I>
Applesoft BASIC:    PRINT CHR$(9): "command"
Pascal:             WRITELN (CHR(9),'command');
```

PRINTER MODE COMMAND SUMMARY

Table 2-4 is a summary of the commands available in Printer Mode. Some details, explained fully in the remainder of this chapter, have been omitted from the table for the sake of brevity. Commands marked with an asterisk are not supported by Pascal.

Format	Command Name	Values	Interpretation
<n>B	Baud Rate	Ø - 15	see Table 2-5
<n>C	<CR> Delay	Ø	no delay
		1	32 milliseconds
		2	25Ø milliseconds (1/4 s)
		3	2 seconds
<n>D	Data Format	Ø	8 data bits, 1 stop bit
		1	7 data bits, 1 stop bit
		2	6 data bits, 1 stop bit
		3	5 data bits, 1 stop bit
		4	8 data bits, 2 stop bits
		5	7 data bits, 2 stop bits
		6	6 data bits, 2 stop bits
		7	5 data bits, 2 stop bits
<n>F	<FF> Delay	Ø	no delay (default)
		1	32 milliseconds
		2	25Ø milliseconds (1/4 s)
		3	2 seconds
<n>L	<LF> Delay	Ø	no delay (default)
		1	32 milliseconds
		2	25Ø milliseconds (1/4 s)
		3	2 seconds
<n>P	Parity	Ø,2,4,6	no parity (default = ØP)
		1	odd parity
		3	even parity
		5	MARK (parity bit always 1)
		7	SPACE (parity bit always Ø)
* <n>T	Translate Lowercase (LC)	Ø	change LC to UC (default)
		1	leave LC (possible garbage)
		2	LC to UC inverse; leave UC
		3	LC to UC; UC to inverse
* C	Column Overflow		auto-<CR> at column's end
* R	Reset the SSC		reset SSC + PR#Ø and IN#Ø
Z	Zap <CTRL>		ignore all <CTRL> commands
F_<E/D>	Find Keyboard	E or D	accept keyboard entries
L_<E/D>	Generate <LF> Out	E or D	send <LF> out after <CR>
M_<E/D>	Mask <LF> In	E or D	drop <LF> in after <CR>
* T_<E/D>	Tab in BASIC	E or D	recognize BASIC tabs
X_<E/D>	XOFF Recognition	E or D	detect XOFF; await XON

* Not supported by Pascal.

Table 2-4. Printer Mode Commands

COMMANDS THAT CHANGE SWITCH SETTINGS

The group of commands discussed in this section either directly override the SSC switch settings, or affect related behavior of the SSC. The Reset command restores the switch selections.

Baud Rate–<n>B

This command overrides the physical settings of switches SW1-1 through SW1-4 on the SSC. For example, to change the baud rate to 135 baud, type in <CTRL-I>4B<RETURN>.

<n>=	SSC Baud Rate	<n>=	SSC Baud Rate
Ø	use SW1-1 to SW1-4	8	12ØØ
1	5Ø	9	18ØØ
2	75	1Ø	24ØØ
3	1Ø9.92 (11Ø)	11	36ØØ
4	134.58 (135)	12	48ØØ
5	15Ø	13	72ØØ
6	3ØØ	14	96ØØ
7	6ØØ	15	192ØØ

Table 2-5. Baud Rate Selections

Data Format–<n>D

With this command you can override the settings of switch SW2-1. The table below shows how many data and stop bits correspond to each value of <n>. For example, <CTRL-I>2D<RETURN> causes the SSC to transmit each character in the form: one start bit (always transmitted), six data bits, and one stop bit.

<n>=	Data Bits	Stop Bits
Ø	8	1
1	7	1
2	6	1
3	5	1
4	8	2 (1 with Parity options 4 through 7)
5	7	2
6	6	2
7	5	2 (1-1/2 with Parity options Ø through 3)

Table 2-6. Data Format Selections

Parity–<n>P

You can use this command to determine the kind of parity the SSC is to generate when sending data and check for when receiving data. In general, parity checking is not needed in Printer Mode. However, there are five parity options available (Table 2-4).

<n>=	Parity to Use
Ø, 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 Ø)

Table 2-7. Parity Selections

For example, type <CTRL-I>1P<RETURN> to cause the SSC to transmit and check for odd parity. Odd parity means that the high bit of every character is Ø if there is already an odd number of 1 bits in that character, or 1 if there is otherwise an even number of 1 bits in the character, making the total always odd. This is an easy (but not foolproof) way to check data for transmission errors. Parity errors are recorded in a status byte (Appendix F).

Set Time Delay–⟨n⟩C, ⟨n⟩L, ⟨n⟩F

Some printers are slow and do not provide a "printer busy" or handshake signal to the Apple II. The <n>C command causes the Apple II to delay a specified amount of time, after sending a carriage return character, before sending another group (usually another line) to it. This gives the print head enough time to return to the left side of the page so it is ready to continue printing.

The <n>C command overrides the setting of switch SW2-2 on the SSC. That switch provides only two choices: no delay or a 25Ø millisecond delay.

The <n>L command allows time after a linefeed character for a printer platen to turn so the paper is vertically positioned to receive the next line.

The <n>F 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 linefeed).

<n>=	Time Delay
Ø	none
1	32 milliseconds
2	25Ø milliseconds (1/4 second)
3	2 seconds

Table 2-8. Time Delay Selections

Consult the user manual for the printer to find out how much time it takes to move its print head and platen, and so to 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. A typical set for a VERY slow printer would be <CTRL-I>2C<RETURN>, <CTRL-I>2L<RETURN>,<CTRL-I>3F<RETURN>; that is, the SSC waits 25Ø milliseconds after transmitting carriage returns, 25Ø milliseconds after transmitting linefeeds, and 2 seconds after transmitting form feed characters.

Generate ⟨CR⟩ On Column Overflow–C

Typing <CTRL-I>C<RETURN> causes the SSC to generate a carriage return character automatically any time the column count exceeds the printer line width.

Once this 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.

Generate ⟨LF⟩ Out–L_⟨E/D⟩

You can use this command to have the SSC automatically generate and transmit a linefeed character after each carriage return character. This overides the setting of switch SW2-5. For example, you can type <CTRL-I>L E<RETURN> to cause your printer to print listings or double-spaced manuscripts for editing.

Mask (Suppress) ⟨LF⟩ In–M_⟨E/D⟩

If you type <CTRL-I>M E<RETURN>, the SSC will suppress any incoming linefeed character that immediately follows a carriage return character.

Reset the SSC–R

Typing <CTRL-I>R<RETURN> has the same effect as sending a PR#Ø and an IN#Ø to a BASIC program and then resetting the SSC. This keyboard command cancels all previous commands to the SSC and puts the physical switch settings back into force.

OTHER COMMANDS

The commands described here affect the handling of characters and tabs. The Translate command determines how characters will appear on the video screen. The Z and F commands prevent the SSC from responding to control characters or ALL characters coming from the keyboard, respectively. The X command causes the SSC to respond to the XON/XOFF software protocol. Finally, the T command implements the tabbing feature of BASIC.

Translate Lowercase Characters–<n>T

The Apple II Monitor "translates" all incoming lowercase characters into uppercase ones before sending them to the video screen or to a BASIC program. The SSC offers four translation options:

<n>=	What to Do with Lowercase Characters
Ø	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 II 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).

Table 2-9. Lowercase Character Displays

Zap (Suppress) Control Characters–Z

Typing <CTRL-I>Z<RETURN> prevents the SSC from recognizing any further control characters (and hence commands) whether coming from the keyboard or contained in a stream of characters moving through the SSC.

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

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

Find Keyboard–F_<E/D>

You can protect incoming data from disruption by keystrokes with this command. For example, you can include an F D command in a program, followed by a routine that retrieves data coming in through the SSC, followed by F E later in the program. Default is F E.

XOFF Recognition–X_<E/D>

Typing <CTRL-I>X E<RETURN> causes the SSC to look for any XOFF (decimal 19; Appendix D) 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 (decimal 17; Appendix D) from the device, signalling the SSC to continue transmission. In Printer Mode, the default value of this command is X D.

In Printer Mode, full duplex communication may not work with XOFF recognition turned on, so be careful.

Tab in BASIC–T_<E/D>

If you type in <CTRL-I>T E<RETURN>, the BASIC horizontal position counter is left equal to the column count. All TABs work, including back-tabs. TABs beyond column 4Ø require a POKE to location 36, as usual. Commas only work as far as column 4Ø, and BASIC programs will be listed in 4Ø-column format.

CHAPTER 3

COMMUNICATIONS MODE

This chapter explains how to prepare, install and use the SSC in Communications Mode, and change the SSC's activities via commands.

PREPARING THE SSC FOR COMMUNICATIONS MODE

The SSC is ready to operate in Communications Mode when the jumper block and switches SW1-5 and SW1-6 are correctly positioned.

If the triangle on the jumper block is pointing up toward the word MODEM, remove the block (using an IC Extractor, if necessary) and reinsert it with the triangle pointing toward MODEM (Figure 3-1).

Using a pointed object, set switches SW1-5 and SW1-6 both ON as shown in Figure 3-1. This puts the SSC in Communications Mode.

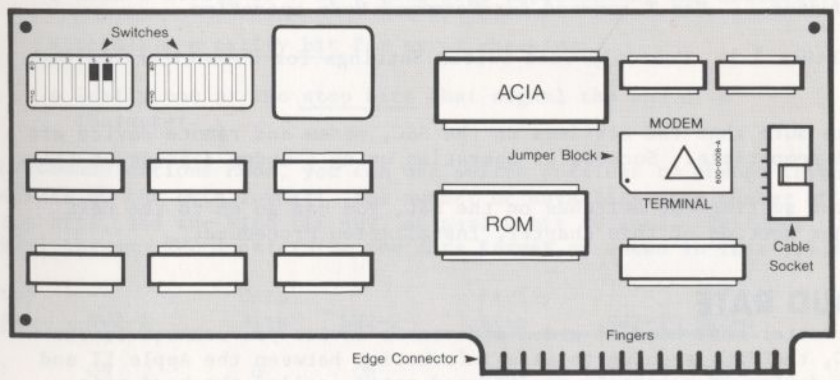


Figure 3-1. SSC Set for Communications Mode

SETTING THE SWITCHES

Use the tip of a ballpoint pen or some other sharp object to flip the appropriate tiny switches on the SSC. A switch is ON when the top of the switch rocker is pushed in. The following subsections explain what settings to use.

COMMONLY USED SETTINGS

Table 3-1 lists the switch settings you can use for connection to various devices and services via the SSC and a modem.

Application	Switch Settings, Cable Connections, Other Information
Apple II via modem	SW1: ON OFF OFF ON ON ON ON SW2: ON ON * * OFF OFF OFF Comm Mode, 3ØØ baud, 8 data, 1 stop, * * parity If using SSC in each Apple, set both the same; for local connection, second jumper block points toward TERMINAL.
Apple III via modem	SW1: ON OFF OFF ON ON ON ON SW2: ON ON * * OFF OFF OFF Comm Mode, 3ØØ baud, 8 data, 1 stop, * * parity Set Apple III RS-232-C Device Control Block to same values (See Apple III Standard Device Drivers manual).
Printer via modem	SW1: ON OFF OFF ON ON ON ON SW2: ON OFF * * OFF OFF OFF Comm Mode, 3ØØ baud, 7 data, 1 stop, * * parity Baud rate is limited by modem and transmission lines; some modems can also use 12ØØ baud; SW1-7 is always ON, and SW2-7 is always OFF; SCTS hookup is at remote modem.
Dow Jones News and Quotes Reporter	SW1: ON OFF OFF ON ON ON ON SW2: ON OFF - ON OFF OFF OFF Comm Mode, 3ØØ baud, 7 data, 1 stop, no parity Sample program at end of this chapter sets same traits. Use T command for Terminal Mode operation.

Table 3-1. Commonly Used Switch Settings for Communications Mode

Make sure that the settings on the SSC, modem and remote device are all compatible. Successful operation using a modem depends on this.

After setting the switches on the SSC, you can go on to the next major section of this chapter, Installation Procedure.

BAUD RATE

No matter what kind of modem and remote device you connect to the SSC, the SSC is going to pass information between the Apple II and the device at a certain prearranged speed, called the baud rate. Since the Apple II can usually send and receive information faster than what is connected to it, the simplest way to determine the maximum baud rate you can use is to consult the user manual for the modem and remote device you will connect. Find out what rate is the fastest they both can handle. Once you know this, you are ready to

set the baud rate switches on the SSC. The following table shows the correct switch positions.

Baud	SW1-1	SW1-2	SW1-3	SW1-4	Baud	SW1-1	SW1-2	SW1-3	SW1-4
5Ø	ON	ON	ON	OFF	12ØØ	OFF	ON	ON	ON
75	ON	ON	OFF	ON	18ØØ	OFF	ON	ON	OFF
11Ø*	ON	ON	OFF	OFF	24ØØ	OFF	ON	OFF	ON
135**	ON	OFF	ON	ON	36ØØ	OFF	ON	OFF	OFF
15Ø	ON	OFF	ON	OFF	48ØØ	OFF	OFF	ON	ON
3ØØ	ON	OFF	OFF	ON	72ØØ	OFF	OFF	ON	OFF
6ØØ	ON	OFF	OFF	OFF	96ØØ	OFF	OFF	OFF	ON
(* 1Ø9.92)		(** 134.58)			192ØØ	OFF	OFF	OFF	OFF

Table 3-2. Baud Rate Switch Settings

If you are connecting a printer or terminal at the other end of the modem, make sure that it is set (with its own switches, dials or thumb wheels) to the SAME baud rate! If you don't, the SSC will send and receive unrecognizable garbage.

DATA FORMAT AND PARITY

The SSC sends each character (such as a "7" or an "H" or a "?") as a string of zeroes and ones (bits). The way it can send a character in Communications Mode, using switch settings, is this:

- first a single start bit to signal to the printer or terminal that a character is coming;
- then a string of 7 or 8 data bits representing the character;
- possibly a parity bit for error checking;
- lastly one or two stop bits that signal the end of a character.

For Communications Mode, you can use switch settings to change three aspects of the data format: the number of data bits, the number of stop bits, and the kind (if any) of parity bit to send. Switches SW2-1 through SW2-4 determine the data format as shown in this table.

Stop Bits	SW2-1	Data Bits	SW2-2	Parity Bits	SW2-3	SW2-4
1	ON	8	ON	none	--	ON
2	OFF	7	OFF	odd	ON	OFF
				even	OFF	OFF

Table 3-3. Data Format Selections

If SW2-1 is OFF, the number of stop bits will be 1 instead of 2 if both 8 data bits (SW2-2 ON) and a parity bit (SW2-4 OFF) have been selected.

To determine the correct combination of switch settings, consult the literature describing the device or timesharing service you plan to connect to the SSC in this mode.

The most commonly used format for ASCII data is: 7 data bits, 1 stop bit, and no parity bit (SW2-1 and SW2-4 ON; SW2-2 OFF).

If you set the data rate switches to 5Ø, 75 or 11Ø baud, choose a switch combination that specifies 2 stop bits; for all data rates 135 baud or higher, use 1 stop bit (switch SW2-1 ON), unless device or timesharing service literature specifies otherwise.

To set the SSC for a data format different from those shown in this table, or to change the data format temporarily, use the SSC commands described later in this chapter.

GENERATE ⟨LF⟩ OUT

If the remote device (for example, a faraway printer) does not automatically generate linefeeds after carriage returns, and it desperately needs them, then set switch SW2-5 ON. Otherwise set SW2-5 OFF.

In Communications Mode, the SSC automatically discards incoming linefeeds that immediately follow carriage returns, unless you use the M D command as described later in this chapter.

SPECIAL SWITCHES

Switch SW2-6 controls forwarding of interrupts to the Apple II. Since the Apple II and II+ do not handle interrupts, set SW2-6 OFF.

For Communications Mode, set SW1-7 ON and SW2-7 OFF.

Your Super Serial Card is now ready to install and use in Communications Mode.

INSTALLATION PROCEDURE

This section explains how to install the SSC and its internal cable in the Apple II. If the cable clamp is not already assembled, do so now, following the instructions given in Chapter 1.

Before connecting or disconnecting anything on the Apple, turn off the power with the switch at the back left corner of the Apple case. THIS IS ABSOLUTELY NECESSARY. If you try to connect or disconnect anything from the inside of your Apple when the power is on, you are likely to damage the circuits.

Do not unplug the Apple, just turn it off. If you unplug the Apple, you will isolate it from earth ground and leave it vulnerable to static discharges.

Remove the Apple cover by pulling up on the two back corners of the cover until the two corner fasteners pop apart. Slide the cover back until it is free of the case and then lift the cover off.

Look inside the Apple and locate the power supply case--the rectangular metal box along the left inside the Apple II. To avoid damaging the SSC, touch the power supply case with one hand; this discharges any static charge that may be on your clothes or body.

Along the back inside edge of the Apple you will see eight long narrow slots called <u>connector slots</u>. The connector slots are numbered from Ø at the left to 7 at the right. The numbers are printed along the back edge behind the connector slots. For use with Pascal and a modem, install the SSC in slot #2. For use with BASIC, install the SSC in any slot from #1 through #7.

Handle the Super Serial Card as you would handle an expensive phonograph record. Grasp it only by the corners or edges, and do not touch the components or pins, especially the gold fingers on the edge connector.

There are three deep notches along the back of the Apple II case. Temporarily set the SSC down near the desired slot. Then take the clamp assembly and slide it down into the notch closest to the slot that the SSC will be in. Tighten the screws until the connector assembly can no longer be moved in the opening.

Grasp the upper corners of the SSC and insert the gold fingers of the edge connector into the slot in the back of the Apple, rear edge first. Gently push the front edge of the card down until it is level and firmly seated. Figure 3-2 shows how the SSC looks when installed in slot #2.

Note that the outer ends of the screws in the clamp assembly can act as nuts. They are threaded and can receive screws from the printer or terminal connector, to ensure a good connection with the Apple.

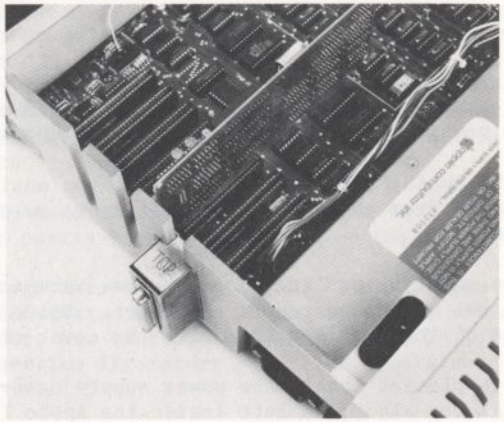


Figure 3-2. SSC in Slot #2 and Clamp Assembly in Notch

Slide the Apple case top plate in place and press down on the rear corners until the corner fasteners pop into place. The Super Serial Card is now installed.

EXTERNAL CABLE AND CONNECTOR

The SSC cable connector you installed in the notch is a standard DB-25 connector with 25 pins. Ten pins of the connector are connected internally to the SSC.

You will need a cable to connect the modem or other device to the SSC connector on the Apple II. Cables with 25-pin connectors on one end are available from your Apple dealer.

The cable must have internal shielding, with the shielding properly terminated at both ends, to prevent electromagnetic interference to nearby radios, television sets, and communication equipment. This shielding is necessary for the system to comply with Class B Federal Communications Commission limits as defined by Subpart J of Part 15 of the FCC rules. Unshielded cables are not recommended.

Make sure that all devices are connected to the same grounded AC power circuit (three-wire wall outlet) as the Apple II. Connecting ungrounded equipment to your Apple II can cause severe electrical damage.

USING SSC IN COMMUNICATIONS MODE

Communications Mode allows you to use the SSC with a modem, connected to a remote device (such as a remote printer, terminal, or other computer). After installing the SSC, you can control its operation from a BASIC, Pascal or assembly-language program, or even directly from the keyboard. To use the SSC in Communications Mode, do the following:

- Make sure the jumper block points toward MODEM.
- Under BASIC or DOS, boot the Apple II, and then type in PR#s and IN#s to route input and output, respectively, to and from the remote device. (The SSC is in slot s.)
- Under Pascal, boot the Apple II and then use #7: or REMIN: for input, and #8: or REMOUT: for output. (The SSC is in slot #2.)
- If the modem and remote device don't work, refer to Appendix E for troubleshooting hints, or consult your Apple dealer.

COMMUNICATIONS MODE COMMANDS

You can issue any of the commands described in this section by embedding them in a computer program. Under BASIC or DOS, you can also enter them directly at the Apple (or remote terminal) keyboard.

In a BASIC program, put the control character and command in a PRINT statement. In a Pascal program, embed the command in a WRITE or WRITELN statement.

Before keyboard entry of these commands has any effect on the SSC, you must first issue an IN#s command (with the SSC in slot s). When you then enter the command character (usually <CTRL-A>, see below), the prompt APPLE SSC: appears on the display screen. Subsequent characters up to <RETURN> will be interpreted as an SSC command. Pressing the left arrow key before pressing <RETURN> cancels the command and causes the APPLE SSC: prompt to reappear.

Many of these commands override the physical switch settings on the SSC. This makes it unnecessary to open the Apple II case and manually change the SSC switch settings. To change the values back to the physical switch settings, reboot or reset the Apple II, or type in the Reset command described below.

COMMAND FORMATS

All commands are preceded by the Communications Mode command character (usually <CTRL-A>, see below) and followed by <RETURN>. The notation <CTRL-A> means "hold down the CTRL key while pressing A." There are three types of command formats:

- a number <n> followed by an uppercase letter (for example, 4D to set Data Format 4)

- simply an uppercase letter (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 generation of linefeed characters)

The allowable range of <n> is given in each command description below. The choice of Enable or Disable is written as <E/D>.

The underscore character (_) before the <E/D> in Enable/Disable commands is merely a reminder 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, E(cho E(nable is the same as E E. The SSC ignores invalid commands.

THE COMMAND CHARACTER

The normal command character in Communications Mode is <CTRL-A>. You can send the command character itself through the SSC by typing it twice in a row: <CTRL-A><CTRL-A> (no <RETURN> necessary). 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.

If you want to change the command character from <CTRL-A> to <CTRL-something else>--for example, <CTRL-W>--type <CTRL-A><CTRL-W>. To change back, type <CTRL-W><CTRL-A>. No <RETURN> is required after either of these commands.

Do not change the control character to <CTRL-S>, <CTRL-T> or <CTRL-R>, since in Communications Mode the SSC interprets these as special control commands from a remote device.

The command character <CTRL-A> is ASCII code 1. Here is how to generate this character in BASIC and Pascal:

```
Integer BASIC:     PRINT "*command"  *embedded <CTRL-A>
Applesoft BASIC:   PRINT CHR$(2): "command"
Pascal:            WRITELN (CHR(2),'command');
```

COMMUNICATIONS MODE COMMAND SUMMARY

Table 3-4 is a summary of the commands available in Communications Mode. Some details, explained fully in the remainder of this chapter, have been omitted from the table for the sake of brevity. Commands marked with an asterisk are not supported by Pascal.

Format	Command Name	Values	Interpretation
<n>B	Baud Rate	Ø - 15	see Table 3-5
<n>C	<CR> Delay	Ø 1 2 3	no delay 32 milliseconds 25Ø milliseconds (1/4 s) 2 seconds
<n>D	Data Format	Ø 1 2 3 4 5 6 7	8 data bits, 1 stop bit 7 data bits, 1 stop bit 6 data bits, 1 stop bit 5 data bits, 1 stop bit 8 data bits, 2 stop bits 7 data bits, 2 stop bits 6 data bits, 2 stop bits 5 data bits, 2 stop bits
<n>F	<FF> Delay	Ø 1 2 3	no delay (default) 32 milliseconds 25Ø milliseconds (1/4 s) 2 seconds
<n>L	<LF> Delay	Ø 1 2 3	no delay (default) 32 milliseconds 25Ø milliseconds (1/4 s) 2 seconds
<n>P	Parity	Ø,2,4,6 1 3 5 7	no parity (default = ØP) odd parity even parity MARK (parity bit always 1) SPACE (parity bit always Ø)
* <n>S	Screen Slot	Ø-7	chain SSC output to slot n
* <n>T	Translate Lowercase (LC)	Ø 1 2 3	change all LC to UC leave LC (possible garbage) LC to UC inverse; leave UC LC to UC; UC to inverse
B * R * T Z	Break Reset the SSC Terminal Mode Zap <CTRL>		transmit 233 ms BREAK SW reset + PR#Ø and IN#Ø (see Terminal Mode section) ignore all <CTRL> commands
* E_<E/D> F_<E/D> L_<E/D> M_<E/D> X_<E/D>	Echo Find Keyboard Generate <LF> Out Mask <LF> In XOFF Recognition	E or D E or D E or D E or D E or D	echo input on the screen accept keyboard entries send <LF> out after <CR> drop <LF> in after <CR> detect XOFF; await XON

* Not supported by Pascal.

Table 3-4. Summary of Communications Mode Commands

COMMANDS THAT CHANGE SWITCH SETTINGS

The commands discussed in this section either override the SSC switch settings, or affect related behavior of the SSC. The Reset command restores the switch selections.

Baud Rate–⟨n⟩B

This command overrides the physical settings of switches SW1-1 to SW1-4 on the SSC. For example, to change the rate to 96ØØ baud, type <CTRL-A>14B<RETURN>.

<n>=	SSC Baud Rate	<n>=	SSC Baud Rate
Ø	use SW1-1 to SW1-4	8	12ØØ
1	5Ø	9	18ØØ
2	75	1Ø	24ØØ
3	1Ø9.92 (11Ø)	11	36ØØ
4	134.58 (135)	12	48ØØ
5	15Ø	13	72ØØ
6	3ØØ	14	96ØØ
7	6ØØ	15	192ØØ

Table 3-5. Baud Rate Selections

Data Format–⟨n⟩D

With this command you can override the settings of switches SW2-1 and SW2-2. The table below shows how many data and stop bits correspond to each value of <n>. For example, typing <CTRL-A>3D <RETURN> causes the SSC to transmit each character in the form: one start bit (always transmitted), five data bits, and one stop bit.

<n>=	Data Bits	Stop Bits
Ø	8	1
1	7	1
2	6	1
3	5	1
4	8	2 (1 with <n>P options 4 through 7)
5	7	2
6	6	2
7	5	2 (1-1/2 with <n>P options Ø through 3)

Table 3-6. Data Format Selections

Parity–⟨n⟩P

You can use this command to determine the kind of parity the SSC is to generate when sending data and check for when receiving data. There are five parity options available:

<n>=	Parity to Use
Ø, 2, 4 or 6	none
1	odd parity (odd number of 1's)
3	even parity (even number of 1's)
5	MARK parity (parity bit always 1)
7	SPACE parity (parity bit always Ø)

Table 3-7. Parity Selections

For example, type <CTRL-A>1P<RETURN> to cause the SSC to transmit and check for odd parity. Odd parity means that the high bit of every character is Ø if there is already an odd number of 1 bits in that character, or 1 if there is otherwise an even number of 1 bits, making the total always odd. This is an easy (but not foolproof) way to check data for transmission errors. (See Appendix F.)

Generate ⟨LF⟩ Out–L_⟨E/D⟩

You can use this command to have the SSC automatically generate and transmit a linefeed (<LF>) character after each carriage return (<CR>) character. This overides the setting of switch SW2-5. For example, you can type <CTRL-A>L E<RETURN> to cause your printer to produce double-spaced listings or manuscripts for editing.

Mask (Suppress) ⟨LF⟩ In–M_⟨E/D⟩

If you type <CTRL-A>M D<RETURN>, the SSC will not remove incoming linefeed (<LF>) characters that immediately follow carriage return (<CR>) characters.

Reset the SSC–R

Typing <CTRL-A>R<RETURN> has the same effect as sending a PR#Ø and an IN#Ø to a BASIC program and then resetting the SSC. This keyboard command cancels all previous commands to the SSC and puts the physical switch settings back into force.

OTHER COMMANDS

The commands described in this subsection control the handling of characters and of the video screen. Three commands control timed delays following transmission of <CR>, <LF> and <FF> characters. The Translate command controls the display of lowercase and uppercase characters. The Z and F commands suppress control characters and characters entered at the keyboard, respectively. The X command causes the SSC to check the character stream for XOFF, as part of the XON/XOFF protocol. Finally, the <n>S command routes video output to a selected slot, and the E command suppresses display (echo) of characters on the screen.

Set Time Delays–⟨n⟩C, ⟨n⟩L, ⟨n⟩F

Some printers are slow and do not provide a "printer busy" or handshake signal to the Apple II. If such a printer is connected to the SSC via a modem, you may want to use these three delay commands.

The <n>C command causes the Apple II to delay a specified amount of time, after sending a carriage return character, before sending another group (usually another line) to it. This gives the print head enough time to return to the left side of the page so it is ready to continue printing.

The <n>L command allows time after a linefeed character for a printer platen to turn so the paper is vertically positioned to receive the next line.

The <n>F 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 Linefeed).

<n>=	Time Delay
Ø	none
1	32 milliseconds
2	25Ø milliseconds (1/4 second)
3	2 seconds

Table 3-8. Time Delay Selections

Consult the user manual for the printer to find out how much time it takes to move its print head and platen, and so to determine an appropriate set of values for these three delays if a printer is used as the remote device. 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.

Translate Lowercase Characters–⟨n⟩T

The Apple II monitor "translates" all incoming lowercase characters into uppercase ones before sending them to the video screen or to a BASIC program. With the <n>T command, four options are available:

<n>=	What to Do with Lowercase Characters
Ø	Change all lowercase characters to uppercase before passing them to a BASIC program or to the video screen. This is what the Apple II monitor does to 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).

Table 3-9. Lowercase Character Displays

Zap (Suppress) Control Characters–Z

Typing <CTRL-A>Z<RETURN> prevents the SSC from recognizing any further control characters (and hence commands) in the stream of characters moving through the SSC.

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

The only way to reinstate command recognition after invoking the Z command is to reset the SSC, or clear the high-order bit at location $5F8+s (with the SSC in slot s).

Find Keyboard–F_⟨E/D⟩

You can protect incoming data from disruption by keystrokes with this command. For example, you can include <CTRL-A>F D in a program, followed by a routine that retrieves data coming in through the SSC, followed by <CTRL-A>F E later in the program.

XOFF Recognition–X_⟨E/D⟩

In Communications Mode, the SSC automatically recognizes any XOFF (decimal 19; Appendix D) character coming from a device attached to it, and responds to it by halting transmission of characters. The SSC resumes transmission as soon as it receives an XON character (decimal 17; Appendix D) from the device. To disable XOFF recognition, use <CTRL-A>X D<RETURN>.

Specify Screen Slot–<n>S

With this command you can specify the slot number of the device where you want text or listings displayed. (Normally this is slot #Ø, the Apple II video screen.) This allows "chaining" of the SSC to another card slot, such as an 8Ø-column-display peripheral 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 CsØØ space; this is the case for all currently available 8Ø-column-display cards for the Apple II.

For example, let's say you have the SSC in slot #2 with a remote terminal connected to it, and an 8Ø-column-display card in slot #3. Type <CTRL-A>3S<RETURN> 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 II in 8Ø-column format. (Not available in Pascal.)

Echo Characters on the Screen–E_<E/D>

For the Apple II, as for most computers, displaying (echoing) a character on the video screen 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 type in <CTRL-A>E D<RETURN>, the SSC does not forward incoming characters to the Apple II screen. This can be used to hide someone's password entered at a terminal, or to avoid double-display of characters.

TERMINAL MODE

Under Communication Mode, the SSC can enter Terminal Mode and make the Apple II act like an unintelligent terminal. This is useful for connecting the Apple II to a computer timesharing service, or for conversing with another Apple II.

Terminal Mode makes it possible to generate lowercase characters, plus the ten ASCII characters not provided on the Apple II keyboard (plus ESC, since <ESC> is used for this feature).

To generate lowercase characters, press <ESC> (the "ESCAPE" key near the upper left corner of the Apple II keyboard) once, and then type alphabetic characters as you would normally do. After that, to capitalize a single letter, press <ESC> again before typing the letter. To lock the keyboard in uppercase, press <ESC> twice in succession. To get back to lowercase, press <ESC> once, as before.

To generate one of the special ASCII characters listed in Table 3-1Ø, first press <ESC> once (if necessary) to place the keyboard in lowercase mode. Then press <ESC> a second time, followed by one of the top-row keys as shown in Table 3-1Ø. For example, to send a tilde, make sure the keyboard is in lowercase mode, then type <ESC> followed by 9.

<ESC> followed by:	1	2	3	4	5	6	7	8	9	Ø	:
generates:	FS	US	[	\	_	{	|	}	~	ESC	RUB
or in hexadecimal:	9C	9F	DB	DC	DF	FB	FC	FD	FE	9B	FF

Table 3-1Ø. Special ASCII Character Generation

TERMINAL MODE COMMANDS

The commands that specifically affect Terminal Mode are listed in Table 3-11. The Translate, Echo and XOFF commands are described earlier in this chapter.

Format	Command Name	Interpretation
T	Enter Terminal Mode	Go into Terminal Mode.
B	Transmit a Break Signal	Send a 233-millisecond BREAK (signoff) signal.
* E_<E/D>	Echo Enable/Disable	Default E D (full-duplex); use E E for half-duplex.
S_<E/D>	Special Characters Enable/Disable	Default S E; allows/defeats generation of lowercase and special characters (Table 3-1Ø).
* <n>T	Translate Lowercase Characters	Determine treatment of incoming lowercase characters.
* X_<E/D>	XOFF Recognition Enable/Disable	Default X E; in Terminal Mode, X E makes SSC detect <CTRL-R> and <CTRL-T> (remote-control OFF & ON, respectively), but not <CTRL-S>.
Q	Quit (Exit from) Terminal Mode	Return to normal Communications Mode operation.

* Fully described earlier in this chapter.

Table 3-11. Terminal Mode Commands

Enter Terminal Mode–T

This causes the Apple II to function as a full-duplex unintelligent terminal. You can use this command in conjunction with the ECHO command to simulate the half-duplex terminal mode of the old Apple II Communications Card. Type <CTRL-A>T<RETURN> to enter this mode.

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 E(cho E(nable command.

Transmit a Break Signal-B

Typing <CTRL-A>B<RETURN> causes the SSC to transmit a 233-millisecond break signal, recognized by most time-sharing systems as a signoff.

Special Characters-S_<E/D>

Typing <CTRL-A>S E<RETURN> causes the SSC to interpret <ESC><n> pairs as special characters, allowing a keyboard in this way to generate all possible ASCII characters. If you type <CTRL-A>S D<RETURN>, the SSC will treat the <ESC> key like any other key.

Quit (Exit from) Terminal Mode-Q

Type <CTRL-A>Q<RETURN> to exit from terminal mode.

A TERMINAL MODE EXAMPLE

You can use the sample program below to change the SSC temporarily from the characteristics you ordinarily use, to the characteristics needed to make the Apple II into a dumb terminal connected to the Dow Jones News & Quotes Reporter. This program assumes that the SSC is set for Communications Mode and that the jumper block is pointing toward MODEM. Neither of these conditions can be changed by software. This program also assumes that the SSC is in slot #1 and that you want to chain I/O to an 80-column card in slot #3; these conditions you can change via software. To change this Integer BASIC program to an Applesoft program, substitute CHR$(5) for D$ and CHR$(2) for A$, and leave out program lines 40 and 42.

```
 10 REM ***************************************************
 20 REM *    THIS PROGRAM SETS UP THE SSC FOR DOW JONES   *
 30 REM ***************************************************
 40 D$="": REM  TYPE <CTRL-D> ESCAPE CHARACTER BETWEEN QUOTES
 42 A$="": REM  TYPE <CTRL-A> COMMAND CHARACTER BETWEEN QUOTES
 50 PRINT D$;"PR#1":       REM  SSC IS IN SLOT #1;
 52 PRINT A$;"6 BAUD":     REM  SET BAUD RATE TO 300;
 54 PRINT A$;"1 DATA":     REM  DATA FORMAT OF 7 DATA, 1 STOP
 56 PRINT A$;"0 PARITY":   REM  AND NO PARITY;
 58 PRINT A$;"LF DISABLE": REM  NO <LF> GENERATION AFTER <CR>.
 60 PRINT A$;"3 SLOTCHN":  REM  CHAIN TO CARD IN SLOT #3
 62 PRINT A$;"TERM MODE":  REM  AND ENTER TERMINAL MODE.
 70 REM ***************************************************
 72 REM * NOW YOU SHOULD BE IN TERMINAL MODE, GETTING THE *
 74 REM * INFO YOU NEED FROM THE DOW JONES SERVICE.  WHEN *
 76 REM * FINISHED, EXIT WITH THE <CTRL-A>Q(UIT COMMAND.  *
 78 REM ***************************************************
100 REM  Q(UIT COMMAND SENDS CONTROL BACK TO THIS PROGRAM:
110 PRINT A$;"RESET":      REM  RESET SWITCH-SELECTED OPTIONS
120 END
```


CHAPTER 4 HOW THE SCC WORKS

This chapter is divided into three major sections. The first explains what the SSC does, using everyday terms wherever possible. Those of you already familiar with serial data communication can skip this section.

The second section is for anyone who wants an overview of the SSC's operating modes and configuration possibilities.

The third section is a dyed-in-the-wool hardware theory of operation for both the expert and the adventuresome layperson.

SERIAL DATA COMMUNICATION

The SSC is a device that performs serial data communication. Let's consider communication first, then data, and then serial data and data transfer.

Communication is easy enough: getting information from here to there or from there to here. In this discussion, the Apple II is "here." "There" can be nearby (local) or far enough away (remote) that some intermediate device, like a telephone, is needed. Information moving from here to there (out of the Apple) is called output; information moving from there to here (into the Apple) is called input.

Data denotes information in its many forms. For successful data communication, it is essential that both the sender and receiver agree on their interpretation of the data transferred.

Inside the Apple II, data can be numbers and letters and symbols, or program instructions for the computer to carry out, or pointers to storage locations, or error message numbers, or codes for generating pictures or sounds (or lots of other things).

In the Apple II, as in all other computers, data is represented in codes made up of ones and zeros, the only two digits allowed in the binary (two-element) system. Each one or zero is called a BInary digiT or bit. In the binary system, as in our ordinary decimal system, you can count to as high a number as you want--it just takes more digits to get there than in the decimal system--and use each number as a code to represent that number of different items. Table 4-1 gives some examples of how many items you can represent with various quantities of digits.

System	Digits	Using	You can represent
decimal	Ø - 9	1	ten items (Ø through 9)
		2	one hundred (Ø through 99)
		3	one thousand (Ø through 999)
binary	Ø and 1	1	two items (Ø or 1)
		2	four (Ø, 1, 1Ø or 11)
		3	eight (Ø through 111)
		4	sixteen (Ø through 1111)
		5	thirty-two (Ø through 11111)
		6	sixty-four (Ø through 111111)
		7	one hundred twenty-eight
		8	two hundred fifty-six, etc.

Table 4-1. Binary and Decimal Digits and Quantities

For printers, plotters, terminals, and many other devices, 128 codes are enough to distinguish all the necessary characters: 52 for the upper and lowercase alphabet, 1Ø for the decimal digits, and dozens of others for punctuation marks and special symbols. As a result, the 128-character American Standard Code for Information Interchange (ASCII) is widely used. (This 7-bit code is listed in Appendix D.)

Throughout the world, post, telegraph, telex and wire services use 5-bit and 6-bit code sets, even though so few bits cannot represent a very large selection of items. Meanwhile, computers have a penchant for sending each other streams of 8-bit codes with obscure meanings. As long as sender and receiver agree on interpretation, any set of codes will do. The SSC can send all of them.

PARALLEL DATA IN THE APPLE II

The Apple II is called an eight-bit processor because the basic unit of data it uses and moves around internally is an eight-bit byte. The Apple II has sets of eight lines interconnecting its various internal parts, so it can move around all eight bits at the same time. Since the bits travel together like eight cars side by side on an eight-lane highway, data in the Apple II is called parallel data, and data movements within the Apple II are called parallel data transfers (Figure 4-1).

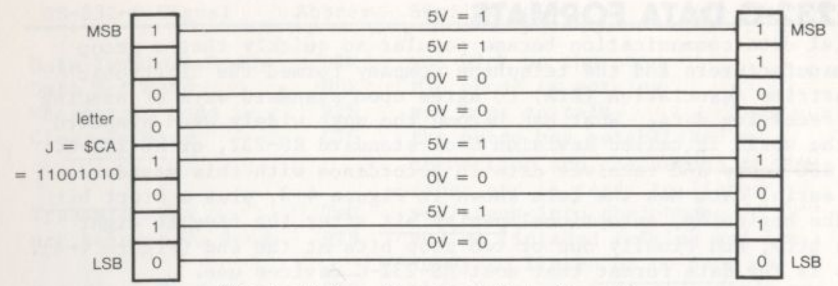


Figure 4-1. Parallel Data Transfer

SERIAL DATA FOR LONG DISTANCES

Just as it would be extremely costly to build highways with eight lanes in each direction over great distances, so it is costly to connect two widely separated pieces of equipment using eight lines in each direction. So, many manufacturers produce computers, printers, plotters, terminals and so forth that send and receive information along one line in each direction, one bit after another. Such a setup, with bits moving from one place to another like a string of cars in a single lane, is called a serial data transfer (Figure 4-2).

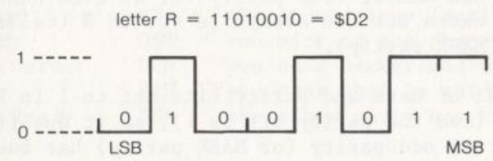


Figure 4-2. Serial Data Transfer

DATA CONVERSION

Changing parallel data to serial data or vice versa is called data conversion (Figure 4-3). By convention (see the later subsection describing RS-232-C), whenever parallel data is converted to serial data, the right-hand bit is sent first. It is as though there were a traffic law that when a multi-lane highway narrows to a single lane, the car in the right lane goes first, then the car from the next lane to the left, etc.

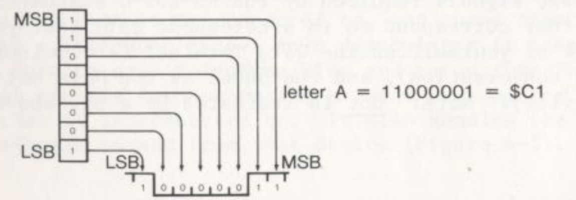


Figure 4-3. Parallel-to-Serial Data Conversion

RS-232-C DATA FORMATS

Serial data communication became popular so quickly that a group of manufacturers and the telephone company formed the Electronic Industries Association (EIA) to agree upon standard ways of sending and receiving data. What has become the most widely used standard in the world is called Revision C of standard RS-232, or RS-232-C. The SSC sends and receives data in accordance with this standard. The serial data has the form shown in Figure 4-3, plus a start bit at the beginning, an optional parity bit after the five to eight data bits, and finally one or two stop bits at the end (Figure 4-4). This is the data format that most RS-232-C devices use.

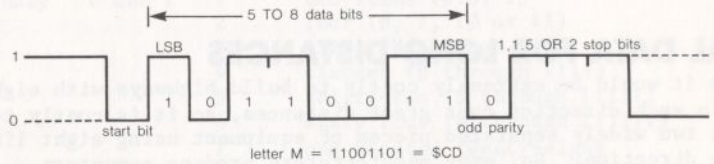


Figure 4-4. RS-232-C Serial Data Format

What is this mysterious parity bit all about? It is an optional extra bit set to Ø or 1 to make the total number of data and stop bits set to 1 an odd number (odd parity) or an even number (even parity); or this extra bit can always be set to Ø (called SPACE parity) or to 1 (MARK parity).

The combined total of data and parity bits set to 1 in Figure 4-4 is 5, an odd number (and the parity bit is 1), so it qualifies as a correct character if odd parity (or MARK parity) has been agreed upon by sender and receiver. However, if that same character were received under even parity (or SPACE parity), the receiving device would signal that a transmission error had occurred. If one bit in a character changes during transmission, parity checking will detect the error. If two bits change, the error will go undetected.

RS-232-C SIGNALS

Since the RS-232-C standard stems from the early days of telephone and telegraph, the names given to its signals may sound quaint to our "modern" ears. However, the signals correspond to familiar conditions that we take for granted when using a telephone. Table 4-2 lists the basic signals required by the RS-232-C standard, and what conditions they correspond to in a telephone call that you originate. Think of yourself as the Data Terminal (a terminus or end point of the conversation), and the phone as the Data Set (the communication device). Note: not is indicated by a bar above a signal name.

RS-232-C Signal	Abbrev.	Similar to
Data Terminal Ready	DTR	you pick up the phone
Data Set Ready	DSR	the phone is working
Request To Send	RTS	you want to talk
Clear To Send	CTS	the phone has established a connection and the person at the other end is ready to listen
Transmit Data	TxD	you speak into the phone
not Request To Send	$\overline{\text{RTS}}$	you've finished talking and are ready to listen or to hang up
not Clear To Send	$\overline{\text{CTS}}$	the phone has sent your words and is ready for your next request to send a message
not Data Terminal Rdy	$\overline{\text{DTR}}$	you hang up

Table 4-2. RS-232-C Signals As Interpreted by the Sender

Here are the RS-232-C signals and how you would interpret them if you were to answer a telephone call (Table 4-3).

RS-232-C Signal	Abbrev.	Similar to
Ring Indicator	RI	the phone rings (optional)
Data Set Ready	DSR	you pick up the phone; it works
Data Carrier Detect	DCD	you hear background noise
Receive Data	RxD	you hear what is said
not Data Set Ready	$\overline{\text{DSR}}$	the other party has hung up

Table 4-3. RS-232-C Signals As Interpreted by the Receiver

Modems

All of the above signals refer to the interaction between what RS-232-C calls Data Terminal Equipment (DTE--end points of data transfers, such as the Apple II or a printer) and what it calls Data Communication Equipment (DCE--transmitting or receiving devices, such as modems).

What is a modem? The name is short for MOdulator/DEModulator. As a modulator it takes electrical signals from a computer or printer (or other device) that it is connected to, and turns them into musical tones over a telephone line. As a demodulator it takes the musical tones it detects on a telephone line and turns them back into electrical signals for use by the printer or computer (or other device) that it is connected to. It also handles the RS-232-C control signals to and from that device (Figure 4-5).

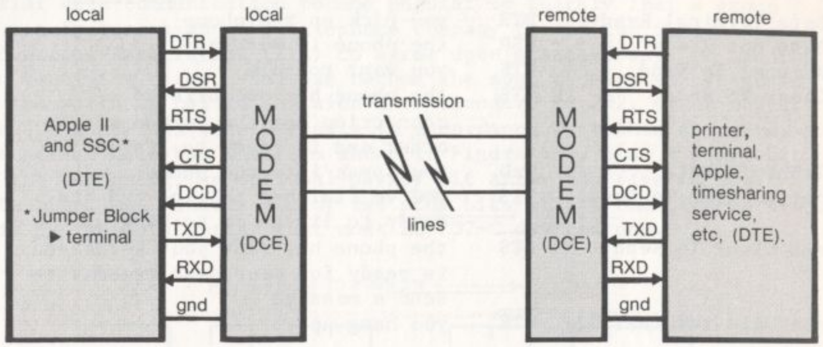


Figure 4-5. An RS-232-C Setup with Modems

By convention, the calling (originate) modem produces a fairly high tone (let's say LA) as the background or carrier signal that it sends; it then modulates (changes) that tone to SO to mean Ø and TI to mean 1. Meanwhile, the called (answer) modem plays a lower tone, MI, as a carrier signal, and modulates that tone to RE to indicate Ø or FA to indicate 1. In this way, both modems can send and receive information along the same wires without interpreting what they send as received messages and vice versa. (All their voices sound alike.)

Modem Eliminators

RS-232 signals are designed for the interactions of two DTE's, two DCE's, and telephone lines, as shown in Figure 4-5. What if you just want to connect two DTE's together in the same room, directly (for example, an Apple II and a printer)? You can use what is called a null modem or modem eliminator. The jumper block on the SSC does just that when it is connected with its triangle pointing toward the word TERMINAL.

By using different tones to send and receive information, modems can make sure that what comes from the "mouthpiece" (transmit register) of one DTE gets routed to the "earpiece" (receive register) of the other. A null modem simply crosses those two wires (Figure 4-6).

To simulate the other signal exchanges that modems would perform, the null modem interconnects the signal wires as shown in Figure 4-6. Thus RTS gets turned back to the sender as CTS as though the phone had instantly established a connection; RTS is also connected to DCD on the other side to pretend that a carrier signal has been detected. Finally, connecting DTR (willing to transfer data) from one side to both RI and DSR (a call arriving) on the other side completes the simulated telephone connection. (RI is optional.) The jumper block does it all!

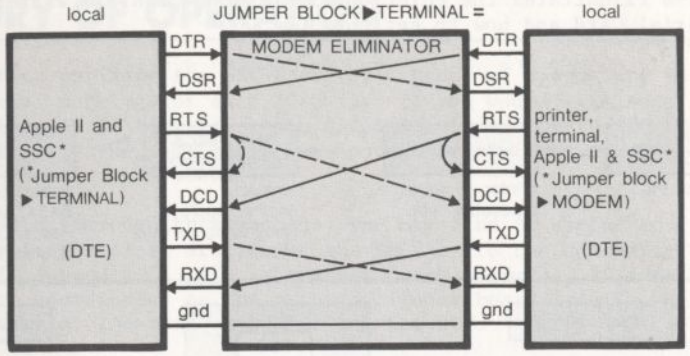


Figure 4-6. An RS-232-C Setup with a Modem Eliminator

SSC MODES AND CONFIGURATIONS

Figure 4-7 outlines the possible operating modes of the Super Serial Card and their relationships to each other.

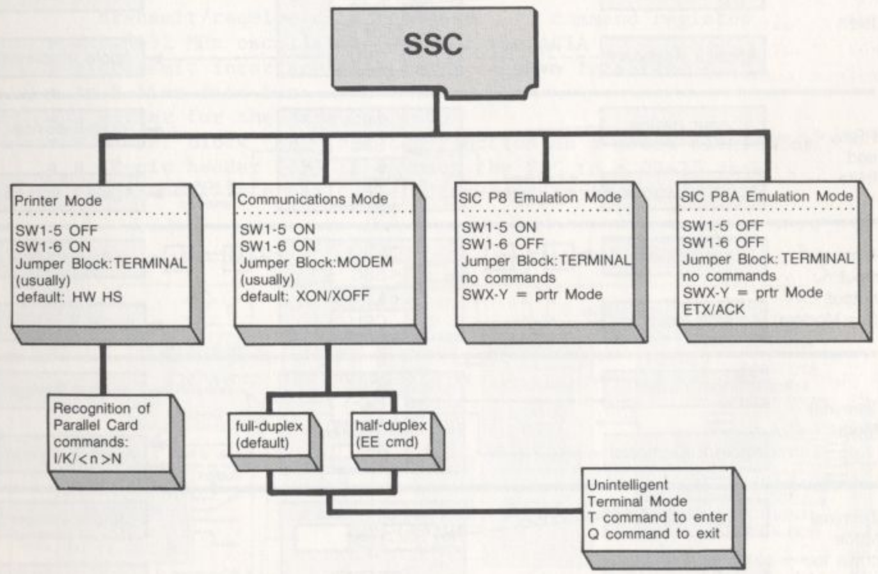


Figure 4-7. SSC Operating Modes

Figure 4-8 illustrates the chief configurations possible with the Super Serial Card and how to set them up.

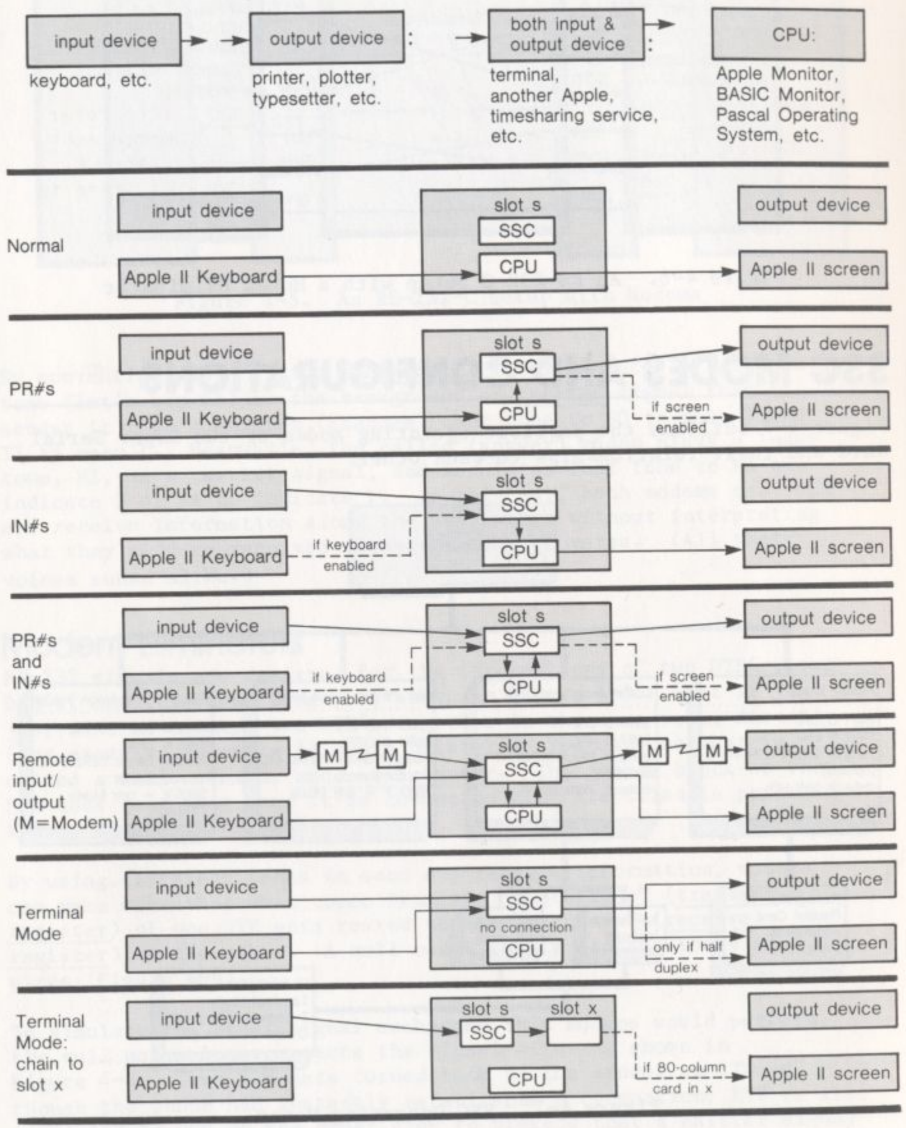


Figure 4-8. SSC Configurations

THEORY OF OPERATION

This section explains the SSC's overall theory of operation, but not the internal workings of each IC chip. If you would like such information, it is best to obtain specifications from the IC manufacturers. The most complex component is the ACIA, which is a Synertek 6551 or equivalent.

While reading through this section, you may find it useful to refer to Figure 4-9, a block diagram of the SSC, or to the schematic diagram in Appendix C. All references in the form 1A, 3C, etc., pertain to coordinates on the printed circuit board itself. Here is an inventory of the main components of the SSC:

- 5Ø-pin connection to the Apple II peripheral connector slot
- a 12-line address bus
- addressing and control logic (1B, 1C, 2C, 3C)
- a 2K-by-8-bit ROM (4B-5C)
- jumpers and bow ties for optional substitution of RAM (3-4A)
- two blocks of 7 switches each (1A, 2A)
- two registers for reading the switch settings (2B, 3B)
- an Asynchronous Communications Interface Adapter (ACIA; 4-5A) with its internal registers:

  status/reset register — control register
  transmit/receive data register — command register
- a 1.8432 MHz oscillator (3A) for the ACIA
- a transmit interface (6A) and a receive interface (7A)
- an 8-line data bus
- a buffer for the data bus (6C)
- a jumper block (6B) that can function as a modem eliminator
- a 1Ø-pin header (7B) to connect the SSC to a DB-25 jack via a short internal cable (discussed in Appendix C)

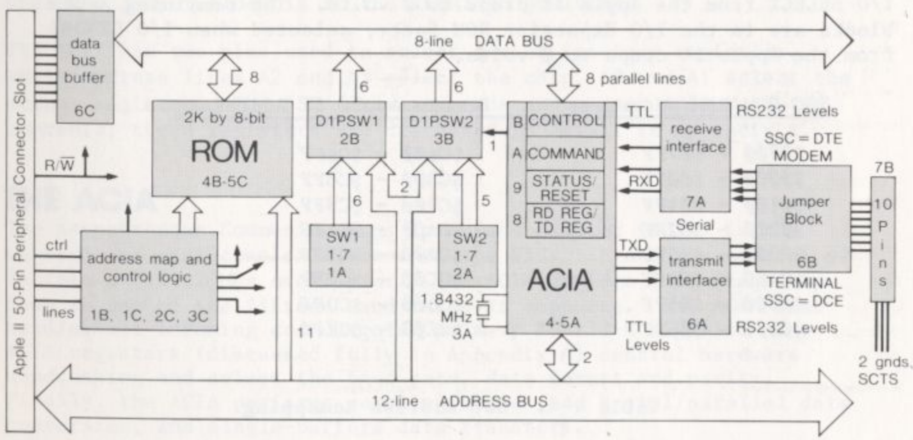


Figure 4-9. Overall Block Diagram of the SSC

ADDRESSING AND CONTROL LOGIC

The twelve address lines (AØ - A11) from the Apple II provide all the necessary $CØØØ addressing on the SSC. Control logic at 1B, 1C, 2C and 3C, plus the signals RESET, DEVICE SELECT, I/O SELECT, and I/O STROBE, ensure the routing of signals to the appropriate addresses.

The SSC follows the Apple II protocol in its use of the $C8ØØ address space. An LS279 (1B) serves as a NAND gate, a pair of inverters, and a set-reset latch. The latch is set by an access to the $Csxx space, and is reset by access to the $CFxx space or by a reset. When this set-reset latch is set, the Apple II can access the $C8ØØ space on the SSC. A small RC filter prevents the latch from being reset by spurious noise.

ROM/RAM Space

The 2K ROM (4B-5C) containing the SSC driver firmware resides in the $C8ØØ - $CFFF address space. However, an LSØØ (2C) and an LS32 (3C) remap the addresses from the range $CsØØ - $CsFF to the range $CFØØ - $CFFF, since the $CFxx addresses are unusable. (Access to them disables use of the $C8ØØ address space.) As a result of this remapping, only one ROM is required, and none of the ROM space is wasted.

The SSC can use a 2K-by-8-bit RAM in place of the ROM. Between columns 3 and 4 and rows A and B on the SSC, there are three jumper pads and three bow ties. If you solder the jumper pads and cut the bow ties, pins 18, 2Ø and 21 will be, respectively, chip enable, output enable and read-write control (instead of ROM enables).

The ROM (or RAM) addresses are mapped as follows (Table 4-4). The first 256-byte block is the Peripheral Card ROM Space, selected when I/O SELECT from the Apple II drops to Ø volts. The remaining seven blocks are in the I/O Expansion ROM Space, selected when I/O STROBE from the Apple II drops to Ø volts.

SSC ROM/RAM Addresses	Become Apple II Addresses
$Ø7ØØ - $Ø7FF	$CsØØ - $CsFF
$ØØØØ - $ØØFF	$C8ØØ - $C8FF
$Ø1ØØ - $Ø1FF	$C9ØØ - $C9FF
$Ø2ØØ - $Ø2FF	$CAØØ - $CAFF
$Ø3ØØ - $Ø3FF	$CBØØ - $CBFF
$Ø4ØØ - $Ø4FF	$CCØØ - $CCFF
$Ø5ØØ - $Ø5FF	$CDØØ - $CDFF
$Ø6ØØ - $Ø6FF	$CEØØ - $CEFF

Table 4-4. SSC Address Remapping

Registers in Peripheral I/O Space

Whenever DEVICE SELECT drops to Ø volts, the Apple II is addressing the SSC's Peripheral I/O Space (the sixteen bytes starting at $CØ8Ø + sØ). This signal is combined logically with address lines AØ through A3 to select one of the six registers that reside in that space (Table 4-5).

Chip selected	Address(+sØ)	Purpose of register
LS365 (2B)	$CØ81	store state of SW1 (1A) (read)
LS365 (3B)	$CØ82	store state of SW2 (2A) and state of CTS (read)
ACIA (4-5A)	$CØ88	receive (read), transmit (write)
ACIA (4-5A)	$CØ89	status (read), reset (write)
ACIA (4-5A)	$CØ8A	command (read and write)
ACIA (4-5A)	$CØ8B	control (read and write)

Table 4-5. Registers in SSC Peripheral I/O Space

The two LS365 chips act as buffers so that the state of eleven of the fourteen available switches, plus the state of RS-232-C signal Clear To Send (CTS), can be read. There are 3.3K ohm pullup resistors at the switch inputs of the LS365 chips. A closed switch pulls down an input, and it is read as zero.

Three switches are not connected to the LS365s. Switch SW2-6, when ON, passes interrupt requests from the ACIA to the Apple II. (The Apple II, however, currently does not support interrupts.) Setting switches SW1-7 ON and SW2-7 OFF connects DB-25 pin 8 (DCD) to the DCD input of the ACIA. Setting SW1-7 OFF and SW2-7 ON splices pin 19, Secondary Clear To Send (SCTS), onto the DCD input of the ACIA when the jumper block is in the TERMINAL position.

The ACIA has two pins used to select one of its four registers. While address lines A2 and A3 select the chip, AØ and A1 select the actual register. The SSC firmware reads and writes ACIA register contents; these registers are discussed in detail in Appendix A.

THE ACIA

The Asynchronous Communications Interface Adapter (ACIA) is the central and most complex element of the SSC. It and the crystal at 3A form a 1.8432 MHz oscillator. The ACIA divides this frequency down to one of the fifteen baud rates it supports. The ACIA also handles all incoming and outgoing primary RS-232-C signals. The ACIA registers (discussed fully in Appendix A) control hardware handshaking and select the baud rate, data format and parity. Finally, the ACIA performs parallel/serial and serial/parallel data conversion, and single-buffers data transfers.

DATA INPUT AND OUTPUT

The MC1489 at 7A converts the incoming serial data from RS-232-C to TTL voltage levels. The MC1488 at 6A converts the outgoing serial data from TTL to RS-232-C voltage levels, and in conjunction with three capacitors limits the output slew rate. Three of the received handshake lines (Clear To Send, Data Carrier Detect, and Data Set Ready) have 15K ohm pullup resistors so the SSC will work with devices that do not assert those signals.

DATA BUS

The 8-bit data bus on the SSC is, of course, a parallel bus. The ACIA takes output from it and gives input to it in parallel form. Also connected to the bus are the two switch detection registers (2B and 3B) and the ROM or RAM chip.

An LS245 (6C) buffers the output to the data bus, and minimizes input loading. The data bus has a 3.3K ohm pullup resistor on each line so the data inputs on the LS245 are not floating when it turns on in output mode.

JUMPER BLOCK

The jumper block has two positions: when its arrow points toward MODEM, the SSC looks like Data Terminal Equipment (DTE); that is, the SSC is prepared to talk to Data Communication Equipment (DCE), such as a modem. When installed with its arrow pointing toward TERMINAL, the jumper block acts as a modem eliminator (null modem); that is, the SSC looks like the DCE on the other device's side of a serial communication connection. In this position, the SSC can talk directly to a printer or any other DTE. Figure 4-6 shows the signal swapping that the jumper block in the TERMINAL position performs.

APPENDIX A
FIRMWARE

This appendix contains the following information:

- an explanation of the Pascal 1.1 firmware card protocol
- a firmware memory map
- a description of the SSC's use of its peripheral slot scratchpad RAM addresses
- a description of the ACIA registers and switch detection registers in the SSC's peripheral I/O space
- a list of firmware entry points and 65Ø2 register values
- the actual SSC firmware listings

PASCAL 1.1 FIRMWARE PROTOCOL

The old Apple II Serial Interface Card (SIC) ran under Pascal 1.Ø with three direct firmware entry points, one for each of the three I/O functions it supported:

Address	Contains
$C8ØØ	initialization routine entry point
$C84D	read routine entry point
$C9AA	write routine entry point

New peripheral cards can be "accepted" into the Pascal 1.Ø system by appearing to be a SIC; that is, with these same three entry points and with $38 at $CsØ5 and $18 at $CsØ7 (see Device ID section below).

Pascal 1.1, on the other hand, has a more flexible setup, and also supports more I/O functions. It can make indirect calls to the firmware in a (new) peripheral card through addresses in a branch table in the card's firmware. It also has facilities for uniquely identifying new peripheral I/O devices.

I/O ROUTINE ENTRY POINTS

The I/O routine entry point branch table is located near the beginning of the CsØØ address space (s being the slot number where the peripheral card is installed). This space was chosen instead of the $C8ØØ space, since under BASIC protocol the $CsØØ space is required, while the $C8ØØ space is optional.

The branch table locations that Pascal 1.1 uses are:

Address	Contains
$CsØD	initialization routine offset (required)
$CsØE	read routine offset (required)
$CsØF	write routine offset (required)
$Cs1Ø	status routine offset (required)
$Cs11	$ØØ if optional offsets follow; non-zero if not
$Cs12	control routine offset (optional)
$Cs13	interrupt handling routine offset (optional)

Notice that $Cs11 contains $ØØ only if the control and interrupt handling routines are supported by the firmware. (For example, the SSC does not support these two routines, and so location $Cs11 contains a (non-zero) firmware instruction.) Apple II Pascal 1.Ø and 1.1 do not support control and interrupt requests, but such requests may be implemented in future versions of the Pascal BIOS and other future Apple II operating systems.

Here are the entry point addresses, and the contents of the 65Ø2 registers on entry to and on exit from Pascal 1.1 I/O routines:

Addr.	Offset for	X Register	Y Register	A Register
$CsØD	Initialization			
	On entry	$Cs	$sØ	
	On exit	error code	(unchanged)	(unchanged)
$CsØE	Read			
	On entry	$Cs	$sØ	
	On exit	error code	(unchanged)	character read
$CsØF	Write			
	On entry	$Cs	$sØ	char. to write
	On exit	error code	(unchanged)	(unchanged)
$Cs1Ø	Status			
	On entry	$Cs	$sØ	request (Ø or 1)
	On exit	error code	(changed)	(unchanged)

Notes: Request code Ø means, "Are you ready to accept output?"
Request code 1 means, "Do you have input ready?"
On exit, the reply to the status request is in the carry bit: carry clear means "No"; carry set means "Yes."

Table A-1. I/O Routine Offsets and Registers under Pascal 1.1

DEVICE IDENTIFICATION

Pascal 1.1 uses four firmware bytes to identify the peripheral card. Both the identifying bytes and the branch table are near the beginning of the $CsØØ ROM space. The identifiers are listed in Table A-2.

Address	Value
$CsØ5	$38 (like the old Serial Interface Card)
$CsØ7	$18 (like the old Serial Interface Card)
$CsØB	$Ø1 (the Generic Signature of new FW cards)
$CsØC	$ci (the Device Signature; see below)

Table A-2. Bytes Used for Device Identification

The first digit, c, of the Device Signature byte identifies the device class as listed in Table A-3.

Digit	Class
$Ø	reserved
$1	printer
$2	joystick or other X-Y input device
$3	serial or parallel I/O card
$4	modem
$5	sound or speech device
$6	clock
$7	mass storage device
$8	8Ø-column card
$9	network or bus interface
$A	special purpose (none of the above)
$B-F	reserved for future expansion

Table A-3. Device Class Digit

The second digit, i, of the Device Signature byte is a unique identifier for the card, assigned by Apple Technical Support. For example, the SSC has a Device Signature of $31: the 3 signifies that it is a serial or parallel I/O card, and the 1 is the low-order digit supplied by Apple Technical Support.

Although version 1.1 of Pascal ignores the Device Signature, applications programs can use them to identify specific devices.

SSC FIRMWARE MEMORY USAGE

Table A-4 is an overall map of the locations that the SSC uses, both in the Apple II and in the SSC's own firmware address space.

Addresses	Name of area	Contents
$ØØØØ-$ØØFF	Page Zero	Monitor pointers, I/O hooks, and temporary storage (Table A-5)
$Ø4xx-$Ø7xx (selected locations)	Peripheral Slot Scratchpad RAM	Locations (8 per slot) in Apple's pages $Ø4 through $Ø7. SSC uses all eight of them (Table A-6)
$CØ(8+s)Ø - $CØ(8+s)F	Peripheral Card I/O Space	Locations (16 per slot) for general I/O; SSC uses 6 bytes (Table A-7)
$CsØØ-$CsFF	Peripheral Card ROM Space	One 256-byte page reserved for card in slot s; first page of SSC FW
$C8ØØ-$CFFF	Expansion ROM	Eight 256-byte pages reserved for a 2K ROM or PROM; SSC maps its FW onto $C8ØØ-$CEFF (Table 4-4)

Table A-4. Memory Usage Map

ZERO PAGE LOCATIONS

The SSC makes use of these zero-page locations (Table A-5):

Address	Name	Description
* $24	CH	Monitor pointer to current position of cursor on screen
$26	SLOT16	Usually (slot# x 16); that is, $sØ
$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.

Table A-5. Zero-Page Locations Used by SSC

SCRATCHPAD RAM LOCATIONS

The SSC uses the Scratchpad RAM locations as listed in Table A-6.

Address	Field name	Bit(s)	Interpretation
$Ø478+s	DELAYFLG	Ø - 1	<FF> delay selection
		2 - 3	<LF> delay selection
		4 - 5	<CR> delay selection
		6 - 7	Translate option
$Ø4F8+s	HANDSHKE	Ø - 7	Buffer count for handshake (P8A Mode)
	PARAMETER	Ø - 7	Accumulator for FW's command processor
$Ø578+s	STATEFLG	Ø - 2	Command mode when not Ø (Printer and Communications Modes only)
		Ø - 4	Enquire character (P8A Mode); dflt ETX
		3 - 5	Slot to chain to (Communications Mode)
		6	Set to 1 after lowercase input character
		7	Terminal Mode when 1 (Comm Mode)
		7	Enable <CR> gen. when 1 (other 3 modes)
$Ø5F8+s	CMDBYTE	Ø - 6	Printer Mode default is <CTRL-I>; Comm Mode default is <CTRL-A>
		7	Set to 1 to Zap control commands
$Ø678+s	STSBYTE		Status and IORESULT byte (Appendix F)
$Ø6F8+s	CHNBYTE	Ø - 2	Current Apple screen slot (Comm Mode); when slot = Ø, chaining is enabled
		3 - 7	$CsØØ space entry point (Comm Mode)
	PWDBYTE	Ø - 7	Current printer width (other modes); for listing compensation, auto-<CR>
$Ø778+s	BUFBYTE	Ø - 6	One-byte input buffer (Comm Mode); used in conjunction with XOFF recognition
		7	Set to 1 when buffer full (Comm Mode)
	COLBYTE	Ø - 7	Current-column counter for tabbing, etc. (other 3 modes)
$Ø7F8+s	MISCFLG	Ø	Generate <LF> after <CR> when 1
		1	Printer Mode when Ø; Comm Mode when 1
		2	Keyboard input enabled when 1
		3	<CTRL-S> (XOFF), <CTRL-R> and <CTRL-T> input checking when 1
		4	Pascal Op Sys when 1; BASIC when Ø
		5	Discard <LF> input when 1
		6	Enable lowercase and special character generation when 1 (Comm Mode)
		6	Tabbing option on when 1 (Printer Mode)
		7	Echo output to Apple screen when 1

Table A-6. Scratchpad RAM Locations Used by SSC

PERIPHERAL CARD I/O SPACE

There are 16 bytes of I/O space allocated to each slot in the Apple II. Each set begins at address $C∅8∅ + (slot x 16); for example, if the SSC is in slot 3, its group of bytes extends from $C∅B∅ to $C∅BF. Table A-7 interprets the 6 bytes the SSC uses.

Address	Register	Bit(s)	Interpretation
$C∅81+s∅	DIPSW1	∅	SW1-6 is OFF when 1, ON when ∅
	(SW1-x)	1	SW1-5 is OFF when 1, ON when ∅
		4 - 7	same as above for SW1-4 through SW1-1
$C∅82+s∅	DIPSW2	∅	Clear To Send (CTS) is true (-) when ∅
	(SW2-x)	1 - 3	same as above for SW2-5 through SW2-3
		5 & 7	same as above for SW2-2 & SW2-1
$C∅88+s∅	TDREG	∅ - 7	ACIA Transmit Register (write)
	RDREG	∅ - 7	ACIA Receive Register (read)
$C∅89+s∅	STATUS		ACIA Status/Reset Register
		1 ∅	Parity error detected when 1
		2 1	Framing error detected when 1
		4 2	Overrun detected when 1
		8 3	ACIA Receive Register full when 1
		16 4	ACIA Transmit Register empty when 1
		32 5	Data Carrier Detect (DCD) true when ∅
		64 6	Data Set Ready (DSR) true when ∅
		128 7	Interrupt (IRQ) has occurred when 1
$C∅8A+s∅	COMMAND		ACIA Command Register (read/write)
		∅	Data Terminal Ready (DTR): enable (1) or disable (∅) receiver and all interrupts
		1	When 1, allow STATUS bit 3 to cause IRQ
		2 - 3	Control transmit interrupt, Request To Send (RTS) level, and transmitter
		4	When ∅, normal mode for receiver; when 1, echo mode (but bits 2 and 3 must be ∅)
		5 - 7	Control parity (values: Table 2-7)
$C∅8B+s∅	CONTROL		ACIA Control Register (read/write)
		∅ - 3	Baud rate: $∅ = 16 times external clock; $1 - $F = decimal in Table 2-5
		4	When 1, use baud rate generator; when ∅, use external clock (not supported)
		5 - 6	Number of data bits: 8 (bit 5 and 6 = ∅) 7 (5 = 1, 6 = ∅), 6 (5 = ∅, 6 = 1) or 5 (bit 5 and 6 both = 1)
		7	Number of stop bits: 1 (bit 7 = ∅); if bit 7 = 1, then 1-1/2 (with 5 data bits, no parity), 1 (8 data plus parity) or 2

Table A-7. SSC Registers in Peripheral Card I/O Space

SSC ENTRY POINTS

This section contains the SSC firmware entry points for the Apple II Monitor, BASIC, Pascal 1.∅ and Pascal 1.1. The Pascal 1.1 entry point offsets conform to the Firmware card protocol outlined in the first section of this appendix.

MONITOR ROM ENTRY POINTS

The SSC uses these entry points in the Monitor ROM, unless Pascal initializes the SSC.

Address	Name	Description
$FDED	COUT	sends a character to output hook (chaining) used for chaining
$FE89	SETKBD	sets KSW to point to keyboard (reset)
$FE93	SETSCR	sets CSW to point to Apple screen (reset)
$FF58	IORTS	known position of an RTS instruction
$FDF6	VIDOUT	sends a character to the Apple screen

Table A-8. Monitor ROM Entry Points Used by SSC

BASIC ENTRY POINTS

Here are the entry point addresses, and the contents of the 65∅2 registers on entry to and on exit from BASIC I/O routines:

Addr.	Routine	X Register	Y Register	A Register
$Cs∅∅	Initialization			
	On entry	anything	anything	anything
	On exit	(unchanged)	(unchanged)	character
Notes:	CSW and/or KSW points to $Cs∅∅. The character in the A register is output unless KSW points to $Cs∅∅ and CSW does not point to $Cs∅∅.			
$Cs∅5	Input			
	On entry	anything	anything	anything
	On exit	(unchanged)	(unchanged)	character in
Notes:	Character in is from ACIA or keyboard.			
$Cs∅7	Output			
	On entry	anything	anything	character out
	On exit	(unchanged)	(unchanged)	(changed)
Notes:	Character out is transmitted through the ACIA.			

Table A-9. BASIC Entry Points Used by SSC

PASCAL 1.0 ENTRY POINTS

There are three Pascal 1.Ø entry points: one for initialization, one for read operations, and one for write operations. These entry points are direct addresses.

Addr.	Routine	X Register	Y Register	A Register
$C8ØØ	Initialization			
	On entry	$Cs	$sØ	anything
	On exit	$Cs	$sØ	(unchanged)
Notes:	$C8ØØ space is enabled. Firmware initializes SSC to default values plus SW1 and SW2 selections.			
$C84D	Read			
	On entry	$Cs	$sØ	anything
	On exit	$Cs	$Cs	character in
Notes:	$C8ØØ space is enabled. Pascal returns ACIA or keyboard data in the A Register and location $678+s with high bit cleared.			
$C9AA	Write			
	On entry	$Cs	$sØ	character out
	On exit	error code	$Cs	(changed)
Notes:	$C8ØØ space is enabled. Output character is transmitted through the ACIA. Pascal posts error code to IORESULT.			

Table A-1Ø. Pascal 1.Ø Entry Points Used by SSC

PASCAL 1.1 ENTRY POINTS

The Pascal 1.1 entry point protocol is outlined in the first section of this appendix. The values given here are the addresses of the routines. Unlike Pascal 1.Ø, Pascal 1.1 enters these routines using indirect addressing.

Addr.	Offset for	Value	X Register	Y Register	A Register
$CsØD	Initialization	$(Cs)8E			
	On entry		$Cs	$sØ	anything
	On exit		$ØØ	$sØ	(changed)
Notes:	$C8ØØ space is enabled. Firmware initializes SSC to default values plus SW1 and SW2 selections.				
$CsØE	Read	$(Cs)94			
	On entry		$Cs	$sØ	anything
	On exit		error code	$Cs	char. in
Notes:	$C8ØØ space is enabled. Character in from ACIA or keyboard is returned in the A Register.				
$CsØF	Write	$(Cn)97			
	On entry		$Cs	$sØ	char. out
	On exit		error code	$Cs	(changed)
Notes:	$C8ØØ space is enabled. The byte in the A Register is sent out through the ACIA.				
$Cs1Ø	Status	$(Cs)9A			
	On entry		$Cs	$sØ	request (Ø or 1)
	On exit		error code	$sØ	error code
Notes:	$C8ØØ space is enabled. Request = Ø asks ACIA whether it is ready to transmit another byte; request = 1 asks ACIA whether it has an input character available. On exit, carry bit = Ø for Yes or 1 for No.				

Table A-11. Pascal 1.1 Offsets Used by SSC

OTHER SPECIAL FIRMWARE LOCATIONS

The SSC firmware uses several other addresses for predefined purposes. Table A-12 lists these locations.

Address	Value	Purpose
$CsØ5	$38	Pascal serial/firmware card identifier (as well as BASIC input entry point)
$CsØ7	$18	Pascal serial/firmware card identifier (as well as BASIC output entry point)
$CsØB	$Ø1	Pascal 1.1 generic signature byte ($Ø1 = firmware card)
$CsØC	$31	Pascal 1.1 Device Signature byte ($31 = serial or parallel I/O card #1)
$Cs11	$85	Pascal 1.1 optional routines flag (nonzero value = not supported)
$CsFF	$Ø8	Firmware revision level

Table A-12. SSC Special Firmware Locations

SSC FIRMWARE LISTINGS

```
0000:        ***********************************
0000:        *                                 *
0000:        * APPLE II SSC FIRMWARE           *
0000:        *                                 *
0000:        *    BY LARRY KENYON              *
0000:        *    -JANUARY 1981-               **************
0000:        *                                              *
0000:        * (C) COPYRIGHT 1981 BY APPLE COMPUTER, INC.   *
0000:        *                                              *
0000:        ************************************************
0000:        *                                 *
0000:        * VARIABLE DEFINITIONS            *
0000:        *                                 *
0000:        ***********************************
0000:        ******************
0000:        * ZERO PAGE EQUS *
0000:        ******************
0024:        CH        EQU  $24         ;CURSOR HORIZONTAL POSITION
0026:        SLOT16    EQU  $26         ;SAVE $NO TO FREE UP Y-REG
0027:        CHARACTER EQU $27          ;OUTPUT, SCREEN AND INPUT CHARS
0028:        BASL      EQU  $28         ;BASE SCREEN ADDRESS POINTER
0035:        ZPTEMP    EQU  $35         ;WORKHORSE TEMPORARY
002A:        ZPTMP1    EQU  $2A         ;WHEN ZPTEMP ISN'T ENOUGH
002B:        ZPTMP2    EQU  $2B         ;TEMPORARIES, TEMPORARIES!
0036:        CSWL      EQU  $36         ;CHAR OUT VECTOR
0037:        CSWH      EQU  $37
0038:        KSWL      EQU  $38         ;CHAR IN VECTOR
0039:        KSWH      EQU  $39
003C:        A1L       EQU  $3C         ;BATCH MOVE POINTER
004E:        RNDL      EQU  $4E         ;RANDOM NUMBER SEED
004F:        RNDH      EQU  $4F
0000:        *******************
0000:        * GENERAL EQUATES *
0000:        *******************
0100:        STACK     EQU  $100        ;SYSTEM STACK BLOCK
0200:        INBUFF    EQU  $200        ;SYSTEM INPUT BUFFER
C000:        KBD       EQU  $C000       ;KEYBOARD INPUT
C010:        KBDSTRB EQU  $C010         ;KEYBOARD CLEAR
CFFF:        ROMSOFF EQU  $CFFF         ;DISABLES C0-RES. $C800 ROMS
0000:        ***********************
0000:        * SSC CARD ADDRESSES *
0000:        ***********************
C081:        DIPSW1    EQU  $C081       ;(+$N0) DIPSWITCH BLOCK 1
C082:        DIPSW2    EQU  $C082       ;(+$N0) DIPSWITCH BLOCK 2
C088:        TDREG     EQU  $C088       ;(+$N0) TRANSMIT DATA REG (WRITE)
C088:        RDREG     EQU  $C088       ;(+$N0) READ DATA REG (READ)
C089:        STREG     EQU  $C089       ;(+$N0) STATUS REGISTER (READ)
C089:        RESET     EQU  $C089       ;(+$N0) SOFTWARE RESET (WRITE)
C08A:        CMDREG    EQU  $C08A       ;(+$N0) COMMAND REGISTER (R/W)
C08B:        CTLREG    EQU  $C08B       ;(+$N0) CONTROL REGISTER (R/W)
```



```
0000:        ***************************************************************
0000:        *  BIT-> B7  B6  B5  B4  B3  B2  B1  B0
0000:        *       +------------------------------+
0000:        * DIPSW1 S1  S2  S3  S4  Z   Z   S5  S6  (LEFT DIPSWITCH)
0000:        *
0000:        * (S1-S4 USED FOR BAUD RATE, S5-S6 FOR FIRMWARE MODE)
0000:        *
0000:        * DIPSW2 S1  Z   S2  Z   S3  S4  S5  CTS (RIGHT DIPSWITCH)
0000:        *
0000:        * STREG  INT DSR DCD TDR RDR OVR FE  PE
0000:        *
0000:        * CTLREG STB << WL >> CK << BAUD RATE >>
0000:        *
0000:        * CMDREG <<PARITY >> ECH <<XMIT>> RE DTR
0000:        *
0000:        ****************************************************************
0000:        ******************************************
0000:        * SCREEN VARIABLES: PPC AND SIC MODES *
0000:        ******************************************
0538:        CMDBYTE EQU $5F8-$C0  ;HOLDS COMMAND CHARACTER (PPC & CIC)
0438:        HANDSHKE EQU $4F8-$C0 ;SIC P8A CHAR COUNTER FOR ETX/ACK
0438:        PARAMETER EQU $4F8-$C0 ;ACCUMULATOR FOR CMD PARAMETER
04B8:        STATEFLG EQU $578-$C0  ;
0000:        * B7=CR GEN ENB FLAG  B6=AFTER LC INPUT FLG
0000:        * B2-B0=COMMAND INTERPRETER STATES
0000:        *    0 0 0  IDLE
0000:        *    0 0 1  CMD CHAR RECEIVED
0000:        *    0 1 0  COLLECT <N> UNTIL CHAR THEN DO COMMAND
0000:        *    0 1 1  SKIP UNTIL SPACE, THEN GOTO STATE 4
0000:        *    1 0 0  E/D COMMANDS
0000:        *    1 0 1  UNUSED
0000:        *    1 1 0  WAIT UNTIL CR THEN SET STATE TO ZERO
0000:        *    1 1 1  WAIT UNTIL CR THEN DO PROC INDICATED BY PARM
0000:        *
0000:        * (B4-B0 DETERMINE ENQUIRE CHAR FOR P8A MODE)
0000:        *
03B8:        DELAYFLG EQU $478-$C0
0000:        * B7-B6=SCREEN TRANSLATION OPTIONS
0000:        *   0 0    LC->UC
0000:        *   0 1    NO TRANSLATION
0000:        *   1 0    LC->UC INVERSE
0000:        *   1 1    LC->UC, UC->UC INVERSE
0000:        *   (1-3 WILL ALLOW LC CHARS TO PASS THRU MONITOR)
0000:        *
0000:        * B5-B4=CR DELAY   0 0 = NO DELAY
0000:        * B3-B2=LF DELAY   0 1 = 32 MILLISEC
0000:        * B1-B0=FF DELAY   1 0 = 1/4 SEC
0000:        *                  1 1 = 2 SEC
0000:        *
05B8:        STSBYTE EQU   $678-$C0  ;STATUS/IORESULT/INPUT BYTE
0638:        PWDBYTE EQU   $6F8-$C0  ;PRINTER (FORMAT) WIDTH
06B8:        COLBYTE EQU   $778-$C0  ;COLUMN POSITION COUNTER
0738:        MISCFLG EQU   $7F8-$C0  ;
0000:        * B7=ECHO BIT          B6=TABBING OPTION ENABLE
0000:        * B5=LINEFEED EAT      B4=PASCAL/BASIC FLAG
0000:        * B3=XOFF ENB FLAG     B2=KEYBOARD ENB
0000:        * B1=PPC/CIC MODE      B0=LF GENERATE ENB
0000:        *
```

```
0000:          ****************************************
0000:          * TEMP SCREEN VARS (SLOT INDEPENDENT) *
0000:          ****************************************
07F8:          MSLOT   EQU  $7F8     ;BUFFER FOR HI SLOT ADDR ($CN)
0000:          ************************************
0000:          * SCREEN VARIABLES: CIC MODE       *
0000:          ************************************
0000:          *
0000:          * STATEFLG: B7=TERMINAL MODE FLAG
0000:          *       B3-B5=CHAIN SLOT
0000:          *
0638:          CHNBYTE EQU  $6F8-$C0 ;CURRENT OUTPUT SCREEN ($CN00 ENTRY)
0000:          *
0000:          * B0-B7=CN00 ENTRY
0000:          *
06B8:          BUFBYTE EQU  $778-$C0 ;BUFFER FOR ONE
0000:          *                       INPUT BYTE: HIGH BIT IS SET
0000:          *                       WHEN BUFFER IS FULL
0000:          *
0000:          * MISCFLG:           B6=TERM MODE SHIFT ENB
0000:          *
0000:          * OTHER SLOT VARIABLES AS DEFINED FOR PPC AND SIC MODES
0000:          *
0000:          ***********************
0000:          * MONITOR SUBROUTINES *
0000:          ***********************
FDED:          COUT    EQU  $FDED    ;CHARACTER OUT (THRU CSW)
FE89:          SETKBD  EQU  $FE89    ;SETS KSW TO APPLE KEYBOARD
FF58:          IORTS   EQU  $FF58    ;KNOWN "RTS" LOCATION
FCBA:          NXTA1   EQU  $FCBA    ;INCREMENT A1H,L AND CMP TO A2H,L
FE93:          SETSCR  EQU  $FE93    ;SETS CSW TO APPLE SCREEN
FDF6:          VIDOUT  EQU  $FDF6    ;OUTPUT A CHAR TO APPLE SCREEN
0000:                  CHN  SSC.CN00
0000:          *********************************
0000:          *                               *
0000:          * APPLE II SSC FIRMWARE         *
0000:          *                               *
0000:          *     BY LARRY KENYON           *
0000:          *                               *
0000:          *      -JANUARY 1981-           **************
0000:          *                                            *
0000:          * (C) COPYRIGHT 1981 BY APPLE COMPUTER, INC. *
0000:          *                                            *
0000:          **********************************************
0000:          *                               *
0000:          * CN00 SPACE CODE               *
0000:          *                               *
0000:          *********************************
----- NEXT OBJECT FILE NAME IS SSC.DCLS.OBJ0
C700:                  ORG  $C700
C700:          *
C700:2C 58 FF  BINIT   BIT  IORTS    ;SET THE V-FLAG
C703:70 0C             BVS  BENTRY   ;<ALWAYS>
C705:38        IENTRY  SEC           ;BASIC INPUT ENTRY
C706:90                DFB  $90      ;OPCODE FOR BCC
C707:18        OENTRY  CLC           ;BASIC OUTPUT ENTRY
C708:B8                CLV
C709:50 06             BVC  BENTRY   ;<ALWAYS> SKIP AROUND PASCAL 1.1 ENTRY
```



```
C70B:01                DFB  $01           ;GENERIC SIGNATURE BYTE
C70C:31                DFB  $31           ;DEVICE SIGNATURE BYTE
C70D:8E                DFB  >PINIT
C70E:94                DFB  >PREAD
C70F:97                DFB  >PWRITE
C710:9A                DFB  >PSTATUS
C711:85 27     BENTRY  STA  CHARACTER
C713:86 35             STX  ZPTEMP        ;INPUT BUFFER INDEX
C715:8A                TXA                ;SAVE X AND Y REGS ON STACK
C716:48                PHA
C717:98                TYA
C718:48                PHA
C719:08                PHP                ;SAVE ENTRY FLAGS
C71A:78                SEI                ;NO RUPTS DURING SLOT DETERMINATION
C71B:8D FF CF          STA  ROMSOFF       ;SWITCH OUT OTHER $C800 ROMS
C71E:20 58 FF          JSR  IORTS
C721:BA                TSX
C722:BD 00 01          LDA  STACK,X       ;RECOVER $CN
C725:8D F8 07          STA  MSLOT
C728:AA                TAX                ;X-REG WILL GENERALLY BE $CN
C729:0A                ASL  A
C72A:0A                ASL  A             ;DETERMINE $N0
C72B:0A                ASL  A
C72C:0A                ASL  A
C72D:85 26             STA  SLOT16
C72F:A8                TAY                ;Y-REG WILL GENERALLY BE $N0
C730:28                PLP                ;RESTORE RUPTS
C731:50 29             BVC  NORMIO
C733:          *
C733:          * BASIC INITIALIZATION
C733:          *
C733:1E 38 05          ASL  CMDBYTE,X     ;ALWAYS ENABLE COMMANDS
C736:5E 38 05          LSR  CMDBYTE,X
C739:B9 8A C0          LDA  CMDREG,Y      ;JUST HAD A POWER-ON OR PROGRAM RESET?
C73C:29 1F             AND  #$1F
C73E:D0 05             BNE  BINIT1
C740:A9 EF             LDA  #$EF          ;IF SO, GO JOIN INIT IN PROGRESS
C742:20 05 C8          JSR  INIT1
C745:          *
C745:E4 37     BINIT1  CPX  CSWH
C747:D0 0B             BNE  FROMIN
C749:A9 07             LDA  #>OENTRY
C74B:C5 36             CMP  CSWL          ;IF CSW IS ALREADY POINTING TO OENTRY,
C74D:F0 05             BEQ  FROMIN        ; THEN WE MUST HAVE COME FROM KSW
C74F:85 36             STA  CSWL          ;OTHERWISE, SET CSW TO OENTRY
C751:18        FROMOUT CLC                ;INDICATE WE ARE CALLED FOR OUTPUT
C752:90 08             BCC  NORMIO        ;<ALWAYS>
C754:E4 39     FROMIN  CPX  KSWH          ;MAKE SURE KSW POINTS HERE
C756:D0 F9             BNE  FROMOUT       ;
C758:A9 05             LDA  #>IENTRY
C75A:85 38             STA  KSWL          ;SET UP KSW (NOTE CARRY SET FROM CPX)
C75C:          *
C75C:          * BRANCH TO APPROPRIATE BASIC I/O ROUTINE
C75C:          *
C75C:BD 38 07  NORMIO  LDA  MISCFLG,X     ;SEPARATE CIC MODE FROM OTHERS
C75F:29 02             AND  #$02          ;NOT ZERO FOR CIC MODE
C761:08                PHP                ;SAVE CIC MODE INDICATION
C762:90 03             BCC  BOUTPUT
```

```
C764:4C BF C8          JMP  BINPUT
C767:          *
C767:BD B8 04  BOUTPUT LDA  STATEFLG,X ;CHECK FOR AFTER LOWERCASE INPUT
C76A:48                PHA
C76B:0A                ASL  A
C76C:10 0E             BPL  BOUTPUT1  ;SKIP IF NOT
C76E:A6 35             LDX  ZPTEMP
C770:A5 27             LDA  CHARACTER
C772:09 20             ORA  #$20
C774:9D 00 02          STA  INBUFF,X  ;RESTORE LOWERCASE IN BUFFER
C777:85 27             STA  CHARACTER ;AND FOR OUTPUT ECHO
C779:AE F8 07          LDX  MSLOT
C77C:68        BOUTPUT1 PLA
C77D:29 BF             AND  #$BF      ;ZERO THE FLAG
C77F:9D B8 04          STA  STATEFLG,X
C782:28                PLP            ;RETRIEVE CIC MODE INDICATION
C783:F0 06             BEQ  BOUTPUT2  ;BRANCH FOR PPC, SIC MODES
C785:20 63 CB          JSR  OUTPUT    ;CIC MODE OUTPUT
C788:4C B5 C8          JMP  CICEXIT   ;FINISH BY CHECKING FOR TERM MODE
C78B:          *
C78B:4C FC C8  BOUTPUT2 JMP SEROUT
C78E:          ***********************************
C78E:          *                                 *
C78E:          * NEW PASCAL INTERFACE ENTRIES    *
C78E:          *                                 *
C78E:          ***********************************
C78E:20 00 C8  PINIT   JSR  PASCALINIT ;
C791:A2 00             LDX  #0        ;NO ERROR POSSIBLE
C793:60                RTS
C794:4C 9B C8  PREAD   JMP  PASCALREAD ;
C797:4C AA C9  PWRITE  JMP  PASCALWRITE ;
C79A:
C79A:          * NEW PASCAL STATUS REQUEST
C79A:          *
C79A:          * A-REG=0 -> READY FOR OUTPUT?
C79A:          * A-REG=1 -> HAS INPUT BEEN RECEIVED?
C79A:          *
C79A:4A        PSTATUS LSR  A         ;SAVE REQUEST TYPE IN CARRY
C79B:20 9B C9          JSR  PENTRY    ;(PRESERVES CARRY)
C79E:B0 08             BCS  PSTATIN
C7A0:20 F5 CA          JSR  SROUT     ;READY FOR OUTPUT?
C7A3:F0 06             BEQ  PSTATUS2
C7A5:18                CLC
C7A6:90 03             BCC  PSTATUS2  ;CARRY CLEAR FOR NOT READY
C7A8:          *
C7A8:20 D2 CA  PSTATIN JSR  SRIN      ;SETS CARRY CORRECTLY
C7AB:BD B8 05  PSTATUS2 LDA STSBYTE,X ;GET ERROR FLAGS
C7AE:AA                TAX
C7AF:60                RTS
C7B0:          ***************************************************
C7B0:          * ROUTINE TO SEND A CHARACTER TO ANOTHER CARD *
C7B0:          ***************************************************
C7B0:A2 03     SENDCD  LDX  #3
C7B2:B5 36     SAVEHOOK LDA CSWL,X
C7B4:48                PHA
C7B5:CA                DEX
C7B6:10 FA             BPL  SAVEHOOK
C7B8:          *
```



```
C7B8:          * NOW PUT CARD ADDRESS IN HOOK
C7B8:          *
C7B8:AE F8 07          LDX  MSLOT
C7BB:BD 38 06          LDA  CHNBYTE,X
C7BE:85 36             STA  CSWL
C7C0:BD B8 04          LDA  STATEFLG,X ;GET SLOT #
C7C3:29 38             AND  #$38
C7C5:4A                LSR  A
C7C6:4A                LSR  A
C7C7:4A                LSR  A
C7C8:09 C0             ORA  #$C0      ;FORM $CN
C7CA:85 37             STA  CSWH
C7CC:          *
C7CC:          * OUTPUT TO THE PERIPHERAL
C7CC:          *
C7CC:8A                TXA            ;SAVE $CN
C7CD:48                PHA
C7CE:A5 27             LDA  CHARACTER
C7D0:48                PHA
C7D1:09 80             ORA  #$80      ;80 COL BOARDS WANT HI-BIT ON
C7D3:20 ED FD          JSR  COUT
C7D6:          *
C7D6:          * NOW RESTORE EVERYTHING THE OTHER CARD MAY HAVE CLOBBERED
C7D6:          *
C7D6:68                PLA
C7D7:85 27             STA  CHARACTER
C7D9:68                PLA
C7DA:8D F8 07          STA  MSLOT
C7DD:AA                TAX
C7DE:0A                ASL  A
C7DF:0A                ASL  A
C7E0:0A                ASL  A
C7E1:0A                ASL  A
C7E2:85 26             STA  SLOT16
C7E4:8D FF CF          STA  ROMSOFF
C7E7:          *
C7E7:          * PUT BACK CSWL INTO CHNBYTE
C7E7:          *
C7E7:A5 36             LDA  CSWL
C7E9:9D 38 06          STA  CHNBYTE,X
C7EC:          *
C7EC:A2 00             LDX  #0
C7EE:68        RESTORHOOK PLA
C7EF:95 36             STA  CSWL,X
C7F1:E8                INX
C7F2:E0 04             CPX  #4
C7F4:90 F8             BCC  RESTORHOOK
C7F6:          *
C7F6:AE F8 07          LDX  MSLOT
C7F9:60                RTS
C7FA:          *
C7FA:C1 D0 D0          ASC  "APPLE"
C7FD:CC C5
C7FF:08                DFB  $8
C800:          *
```

```
C800:              CHN  SSC.C800
C800:          ***********************************
C800:          *                                 *
C800:          * APPLE II SSC FIRMWARE           *
C800:          *                                 *
C800:          *   BY LARRY KENYON               *
C800:          *                                 *
C800:          *   -JANUARY 1981-                **************
C800:          *                                              *
C800:          * (C) COPYRIGHT 1981 BY APPLE COMPUTER, INC. *
C800:          *                                              *
C800:          ************************************************
C800:          *                                 *
C800:          * C800 SPACE: HIGH LEVEL STUFF    *
C800:          *                                 *
C800:          ***********************************
C800:          * PASCAL 1.0 INIT ENTRY *
C800:          *************************
----- NEXT OBJECT FILE NAME IS SSC.DCLS.OBJ1
C800:                   ORG  $C800
C800:20 9B C9  PASCALINIT JSR PENTRY   ;PASCAL 1.0 INITIALIZATION ENTRY
C803:A9 16              LDA  #$16      ;NO XOFF, ECHO, LF EAT, OR LF GEN
C805:48        INIT1    PHA            ;GOES TO MISCFLG AFTER MODIFICATION
C806:A9 00              LDA  #0
C808:9D B8 04           STA  STATEFLG,X
C80B:9D B8 03           STA  DELAYFLG,X
C80E:9D 38 04           STA  HANDSHKE,X
C811:9D B8 05           STA  STSBYTE,X
C814:9D 38 06           STA  PWDBYTE,X
C817:9D B8 06           STA  COLBYTE,X
C81A:B9 82 C0           LDA  DIPSW2,Y  ;SET LF GEN OPTION FROM D2-S5
C81D:85 2B              STA  ZPTMP2    ;SAVE FOR LATER
C81F:4A                 LSR  A         ;S5-> CARRY
C820:4A                 LSR  A         ;IF S5=ON=0 THEN LEAVE MISCFLG ALONE
C821:90 04              BCC  INIT1A
C823:68                 PLA            ;OTHERWISE, MAKE SURE LF GEN
C824:29 FE              AND  #$FE      ; ENABLE IS RESET
C826:48                 PHA            ;
C827:B8        INIT1A   CLV            ;V WILL BE CLEAR FOR CIC MODE
C828:B9 81 C0           LDA  DIPSW1,Y
C82B:4A                 LSR  A         ;SIC MODES SET CARRY
C82C:B0 07              BCS  INIT2     ;BRANCH FOR SIC MODES
C82E:4A                 LSR  A
C82F:B0 0E              BCS  INIT2B    ;PPC MODE BRANCH
C831:A9 01              LDA  #$01      ;CTL-A
C833:D0 3D              BNE  INIT5     ;<ALWAYS> CIC MODE BRANCH
C835:          *
C835:4A        INIT2    LSR  A         ;SET CARRY FOR P8A
C836:A9 03              LDA  #$03      ;SET ETX AS DEFAULT INQUIRY CHAR
C838:B0 02              BCS  INIT2A    ;BRANCH FOR P8A
C83A:A9 80              LDA  #$80      ;FOR P8 SET AUTO CR GEN
C83C:9D B8 04  INIT2A   STA  STATEFLG,X
C83F:2C 58 FF  INIT2B   BIT  IORTS     ;SET V-FLAG FOR PPC, SIC MODES
C842:A5 2B              LDA  ZPTMP2
C844:29 20              AND  #$20      ;SET CR DELAY
C846:49 20              EOR  #$20      ;SO 1=ENB, 0=DISABLE
C848:9D B8 03           STA  DELAYFLG,X ; FROM D2-S2
C84B:          *
C84B:70 0A              BVS  INIT3     ;<ALWAYS> BRANCH AROUND PASCAL
C84D:          *************************
C84D:          * PASCAL 1.0 READ ENTRY *
C84D:          * (MUST BE AT $C84D)    *
C84D:          *************************
C84D:20 9B C8  PREAD0   JSR  PASCALREAD ;DO PASCAL 1.1 READ
C850:AE F8 07           LDX  MSLOT     ;MODIFY FOR 1.0
C853:9D B8 05           STA  STSBYTE,X ;CHARACTER READ
C856:60                 RTS
C857:          ************************
C857:          * NOW WHERE WERE WE??? *
C857:          ************************
C857:          *
C857:A5 2B     INIT3    LDA  ZPTMP2    ;PPC, SIC MODES USE SWITCHES
C859:4A                 LSR  A         ; TO SET PWIDTH, CR DELAY
C85A:4A                 LSR  A
C85B:29 03              AND  #$03
C85D:A8                 TAY
C85E:F0 04              BEQ  INIT4
C860:          *
C860:68                 PLA            ;RESET VIDEO ENABLE FOR PWIDTH#40
C861:29 7F              AND  #$7F
C863:48                 PHA
C864:          *
C864:B9 A6 C9  INIT4    LDA  PWDTBL,Y
C867:9D 38 06           STA  PWDBYTE,X
C86A:A4 26              LDY  SLOT16
C86C:          *
C86C:68                 PLA            ;CLEAR CIC BIT IN FUTURE MISCFLG
C86D:29 95              AND  #$95      ; (AND TABBING, XOFF AND LF EAT BITS)
C86F:48                 PHA
C870:A9 09              LDA  #$09      ;CTL-I
C872:          *
C872:9D 38 05  INIT5    STA  CMDBYTE,X ;CMD ESC CHAR (IGNORED FOR SIC MODES)
C875:68                 PLA
C876:9D 38 07           STA  MISCFLG,X ;SET MISCFLG FLAGS
C879:          *
C879:          * NOW FOR THE ACIA INITIALIZATION  ROUTINE
C879:          *
C879:A5 2B     INITACIA LDA  ZPTMP2    ;DIPSW2
C87B:48                 PHA
C87C:29 A0              AND  #$A0      ;DATA BIT OPTIONS FOR CIC MODE
C87E:50 02              BVC  INITACIA1 ;BRANCH FOR CIC MODE
C880:29 80              AND  #$80      ;8 DATA, 1 OR 2 STOP FOR SIC, PPC
C882:20 A1 CD  INITACIA1 JSR DATACMD1  ;SET CONTROL REG
C885:20 81 CD           JSR  BAUDCMD1  ;SET DIPSWITCH BAUD RATE
C888:68                 PLA
C889:29 0C              AND  #$0C      ;PARITY OPTIONS FOR CIC MODE
C88B:50 02              BVC  INITACIA2 ;BRANCH FOR CIC MODE
C88D:A9 00              LDA  #$0       ;DISABLE PARITY FOR SIC, PPC MODES
C88F:0A        INITACIA2 ASL A
C890:0A                 ASL  A
C891:0A                 ASL  A
C892:09 0B              ORA  #$0B
C894:99 8A C0           STA  CMDREG,Y
C897:B9 88 C0           LDA  RDREG,Y   ;THROW OUT THE STRANGE STUFF
C89A:60                 RTS
C89B:          ************************
```

```
C89B:                * PASCAL READ ROUTINE *
C89B:                ***********************
C89B:20 9B C9        PASCALREAD JSR PENTRY ;SHARED BY BOTH PASCAL VERSIONS
C89E:20 AA C8        PASCALREAD1 JSR GETCHAR ;GET ACIA/KBD DATA
C8A1:29 7F                  AND  #$7F     ;CLEAR HIGH BIT FOR PASCAL
C8A3:AC F8 07        PASEXIT LDY  MSLOT
C8A6:BE B8 05               LDX  STSBYTE,Y ;ERROR STATUS-> X-REG
C8A9:60                     RTS
C8AA:                ******************************
C8AA:                * GETCHAR ROUTINE WAITS FOR *
C8AA:                *  THE NEXT CHAR FROM EITHER *
C8AA:                *  THE ACIA OR KEYBOARD (IF  *
C8AA:                *  ENABLED).  USED BY PASCAL *
C8AA:                *  READ ROUTINE, XON WAIT,   *
C8AA:                *  AND ACK WAIT. DATA IS RE- *
C8AA:                *  TURNED IN THE A-REGISTER  *
C8AA:                ******************************
C8AA:20 FF CA        GETCHAR JSR  INPUT        ;ACIA DATA?
C8AD:B0 05                  BCS  GETCHAR1
C8AF:20 2C CC               JSR  CKKBD        ;KEYBOARD INPUT?
C8B2:90 F6                  BCC  GETCHAR
C8B4:60              GETCHAR1 RTS             ;EXIT WHEN WE HAVE SOMETHING
C8B5:                *
C8B5:                       CHN  SSC.HILEV
```

```
C8B5:                ************************************
C8B5:                *                                  *
C8B5:                * APPLE II SSC FIRMWARE            *
C8B5:                *                                  *
C8B5:                *   BY LARRY KENYON                *
C8B5:                *                                  *
C8B5:                *    -FEBRUARY 1981-               *************
C8B5:                *                                              *
C8B5:                * (C) COPYRIGHT 1981 BY APPLE COMPUTER, INC. *
C8B5:                *                                              *
C8B5:                ************************************************
C8B5:                *                                  *
C8B5:                * CIC, SIC, PPC MODE HIGH-LEVEL    *
C8B5:                *                                  *
C8B5:                ************************************
C8B5:                * CIC EXIT ROUTINE  .              *
C8B5:                ************************************
C8B5:20 1E CA        CICEXIT JSR  CHECKTERM ;SEE IF WE'VE ENTERED TERMINAL MODE
C8B8:                ***********************
C8B8:                * BASIC EXIT ROUTINE *
C8B8:                ***********************
C8B8:68              BASICEXIT PLA
C8B9:A8                     TAY
C8BA:68                     PLA
C8BB:AA                     TAX
C8BC:A5 27                  LDA  CHARACTER
C8BE:60                     RTS
C8BF:                *************************
C8BF:                * BASIC  INPUT ROUTINE *
C8BF:                *************************
C8BF:F0 29           BINPUT  BEQ  BINACIA    ;BRANCH IF NOT CIC MODE
C8C1:BD B8 06               LDA  BUFBYTE,X ;INPUT BUFFER FULL?
C8C4:10 05                  BPL  BINKBD
C8C6:5E B8 06               LSR  BUFBYTE,X ;RESET BUFFER FULL
C8C9:D0 24                  BNE  BINACIA1  ;<ALWAYS>
C8CB:                *
C8CB:20 3E CC        BINKBD  JSR  GETKBD     ;KEYBOARD DATA?
C8CE:90 1A                  BCC  BINACIA
C8D0:                *
C8D0:BD B8 03        BINEND  LDA  DELAYFLG,X
C8D3:29 C0                  AND  #$C0       ;TRANSLATE LOWERCASE TO UPPERCASE?
C8D5:F0 0E                  BEQ  BINEND1    ;IF SO, LET THE MONITOR DO IT
C8D7:A5 27                  LDA  CHARACTER  ;IF NOT, SET FLAG IF
C8D9:C9 E0                  CMP  #$E0       ; THIS IS A LOWERCASE CHAR
C8DB:90 08                  BCC  BINEND1    ; FOR INPUT BUFFER CORRECTION
C8DD:BD B8 04               LDA  STATEFLG,X ; (CIRCUMVENT APPLE MONITOR)
C8E0:09 40                  ORA  #$40
C8E2:9D B8 04               STA  STATEFLG,X
C8E5:                *
C8E5:28              BINEND1 PLP
C8E6:F0 D0                  BEQ  BASICEXIT ;BRANCH IF NOT CIC MODE
C8E8:D0 CB                  BNE  CICEXIT   ;<ALWAYS> CHECK TO SEE IF WE
C8EA:                *                      ENTERED TERM MODE (VIA KYBD ESCAPE
C8EA:20 FF CA        BINACIA JSR  INPUT      ;ACIA DATA?
C8ED:90 DC                  BCC  BINKBD
C8EF:20 11 CC        BINACIA1 JSR RESTORE    ;DO BASIC CURSED DUTY
C8F2:28                     PLP
C8F3:08                     PHP             ;GET CIC MODE INDICATOR
```

```
C8F4:F0 DA                BEQ  BINEND   ;SKIP IF NOT CIC MODE
C8F6:20 D1 C9             JSR  CKINPUT  ;LOOK FOR INPUT STREAM SPECIAL CHARS
C8F9:4C D0 C8             JMP  BINEND   ;
C8FC:           ***********************************
C8FC:           * SIC, PPC BASIC OUTPUT ROUTINE *
C8FC:           ***********************************
C8FC:20 1A CB   SEROUT    JSR  CMDSEQCK  ;CHECK FOR A COMMAND SEQUENCE
C8FF:B0 B7                BCS  BASICEXIT ;BRANCH IF WE WERE IN COMMAND MODE
C901:A5 27                LDA  CHARACTER ;SAVE CHAR ON STACK
C903:48                   PHA
C904:BD 38 07             LDA  MISCFLG,X ;IF VIDEO OR TABBING ENABLED,
C907:29 C0                AND  #$C0      ; DON'T MESS WITH THE CURSOR
C909:D0 16                BNE  TABCHECK
C90B:           *
C90B:A5 24                LDA  CH       ;CHECK FOR COMMA TABBING
C90D:F0 42                BEQ  NOTAB    ;IF CH=0, THERE WAS NO TAB OR COMMA
C90F:C9 08                CMP  #8       ;INTEGER BASIC COMMA?
C911:F0 04                BEQ  COMMA
C913:C9 10                CMP  #16      ;APPLESOFT COMMA?
C915:D0 0A                BNE  TABCHECK
C917:09 F0      COMMA     ORA  #$F0
C919:3D B8 06             AND  COLBYTE,X ;SET COL TO PREVIOUS TAB
C91C:18                   CLC
C91D:65 24                ADC  CH       ;THEN INCREMENT TO NEXT TAB
C91F:85 24                STA  CH
C921:           *
C921:           *
C921:BD B8 06   TABCHECK  LDA  COLBYTE,X
C924:C5 24                CMP  CH       ;IS TABBING NEEDED?
C926:F0 29                BEQ  NOTAB    ;IF EQUAL THEN NO TAB NEEDED
C928:A9 A0                LDA  #$A0     ;SPACE FOR FORWARD TAB
C92A:90 08                BCC  TAB1
C92C:BD 38 07             LDA  MISCFLG,X ;DON'T BACKSPACE UNLESS TABBING
C92F:0A                   ASL  A        ; OPTION IS ENABLED
C930:10 1F                BPL  NOTAB
C932:A9 88                LDA  #$88     ;BACKSPACE FOR BACKTAB
C934:85 27      TAB1      STA  CHARACTER
C936:2C 58 FF             BIT  IORTS    ;SET V=1 TO INDICATE TABBING
C939:08                   PHP           ;SAVE TABBING INDICATOR
C93A:70 0C                BVS  TAB2     ;<ALWAYS> AROUND BATCH MOVE ENTRY
C93C:EA                   NOP
C93D:           ***************************
C93D:           * SHORT BATCH MOVE:       *
C93D:           *  LOCATE AT $C93D FOR    *
C93D:           *  COMPATIBILITY WITH     *
C93D:           *  SIC P8 BLOCK MOVE.     *
C93D:           ***************************
C93D:2C 58 FF   BATCHIN   BIT  IORTS
C940:50                   DFB  $50      ;DUMMY BVC
C941:B8         BATCHOUT  CLV           ;V=0 FOR OUTPUT ENTRY
C942:AE F8 07             LDX  MSLOT
C945:4C EF C9             JMP  BATCHIO
C948:           *********************
C948:           * BURP . . .        *
C948:           *********************
C948:20 B5 C9   TAB2      JSR  ADJUST   ;ADJUST COLUMN COUNT
C94B:20 6B CB             JSR  OUTPUT2  ;DON'T GO TO SCREEN WHEN TABBING
C94E:4C 68 C9             JMP  FORCECR  ;SHARE SOME CODE. . .
```



```
C951:           *
C951:68         NOTAB     PLA
C952:B8                   CLV
C953:08                   PHP           ;SAVE 'NO TAB' INDICATION
C954:85 27      NOTAB1    STA  CHARACTER ;(FORCE CR REENTRY)
C956:48                   PHA
C957:20 68 CB             JSR  OUTPUT1  ;ENTER AFTER CMD SEQ CHECK
C95A:20 B5 C9             JSR  ADJUST
C95D:68                   PLA
C95E:49 8D                EOR  #$8D     ;WAS IT A CR?
C960:0A                   ASL  A
C961:D0 05                BNE  FORCECR
C963:9D B8 06             STA  COLBYTE,X ;IF SO, RESET COLUMN TO 0
C966:85 24                STA  CH
C968:           *
C968:BD B8 04   FORCECR   LDA  STATEFLG,X ;FORCE CR DISABLED?
C96B:10 0D                BPL  SEREND
C96D:BD 38 06             LDA  PWDBYTE,X ;FORCE CR IF LIMIT REACHED
C970:F0 08                BEQ  SEREND   ;(FOR P8 POKE COMPATIBILITY)
C972:18                   CLC
C973:FD B8 06             SBC  COLBYTE,X
C976:A9 8D                LDA  #$8D
C978:90 DA                BCC  NOTAB1   ;BRANCH TO FORCE CR
C97A:           *
C97A:28         SEREND    PLP
C97B:70 A4                BVS  TABCHECK ;BRANCH IF TABBING
C97D:           *
C97D:BD 38 07             LDA  MISCFLG,X ;DON'T MESS WITH CURSOR
C980:30 16                BMI  SEREND2  ; WHEN VIDEO IS ON
C982:BC B8 06             LDY  COLBYTE,X
C985:0A                   ASL  A
C986:30 0E                BMI  SETCH    ;SET CH TO VALUE OF COL FOR TABBING
C988:98                   TYA
C989:A0 00                LDY  #0
C98B:38                   SEC
C98C:FD 38 06             SBC  PWDBYTE,X ;
C98F:C9 F8                CMP  #$F8     ;WITHIN 8 CHARS OF PWIDTH?
C991:90 03                BCC  SETCH
C993:69 27                ADC  #$27     ;IF SO, ADJUST TO WITHIN 8 OF 40
C995:A8                   TAY
C996:84 24      SETCH     STY  CH
C998:           *
C998:4C B8 C8   SEREND2   JMP  BASICEXIT ;THAT'S ALL
C99B:           *
C99B:           ****************************
C99B:           * PASCAL ENTRY ROUTINE     *
C99B:           ****************************
C99B:8E F8 07   PENTRY    STX  MSLOT
C99E:84 26                STY  SLOT16
C9A0:A9 00                LDA  #0
C9A2:9D B8 05             STA  STSBYTE,X
C9A5:60                   RTS
C9A6:           *
C9A6:           **********************************
C9A6:           * SIC MODE PRINTER WIDTH TABLE *
C9A6:           **********************************
C9A6:29         PWDTBL    DFB  $29      ;40 COLUMNS
C9A7:48                   DFB  $48      ;72 COLUMNS
```

```
C9A8:50                  DFB  $50      ;80 COLUMNS
C9A9:84                  DFB  $84      ;132 COLUMNS
C9AA:           ************************
C9AA:           * PASCAL WRITE ROUTINE *
C9AA:           * (DOUBLES AS PASCAL   *
C9AA:           *   1.0 ENTRY POINT)   *
C9AA:           *  -MUST BE AT $C9AA-  *
C9AA:           ************************
C9AA:85 27      PASCALWRITE STA CHARACTER
C9AC:20 9B C9            JSR  PENTRY
C9AF:20 63 CB            JSR  OUTPUT
C9B2:4C A3 C8            JMP  PASEXIT  ;LOAD X-REG WITH ERROR BYTE & RTS
C9B5:           *
C9B5:           *****************************
C9B5:           * COLUMN ADJUST ROUTINE     *
C9B5:           * (PPC, SIC MODES ONLY)     *
C9B5:           *****************************
C9B5:A5 27      ADJUST   LDA  CHARACTER
C9B7:49 08               EOR  #$08     ;BACKSPACE?
C9B9:0A                  ASL  A
C9BA:F0 04               BEQ  DECRCOL  ;IF SO, DECREMENT COLUMN
C9BC:49 EE               EOR  #$EE     ;DELETE? ($FF, RUB)
C9BE:D0 09               BNE  CTRLTST
C9C0:DE B8 06   DECRCOL  DEC  COLBYTE,X ;DECREMENT COLUMN COUNT
C9C3:10 03               BPL  ADJRTS
C9C5:9D B8 06            STA  COLBYTE,X ;DON'T ALLOW TO GO BELOW 0
C9C8:60         ADJRTS   RTS
C9C9:C9 C0      CTRLTST  CMP  #$C0     ;DON'T INCREMENT COLUMN COUNT FOR
C9CB:B0 FB               BCS  ADJRTS   ; CONTROL CHARACTERS
C9CD:FE B8 06            INC  COLBYTE,X
C9D0:60                  RTS
C9D1:           *******************************************
C9D1:           * ROUTINE TO PROCESS SPECIAL INPUT CHARS *
C9D1:           *******************************************
C9D1:BD 38 07   CKINPUT  LDA  MISCFLG,X
C9D4:29 08               AND  #$08     ;INPUT CTL CHARS ENABLED?
C9D6:F0 16               BEQ  CIEND
C9D8:           *
C9D8:BD B8 04            LDA  STATEFLG,X
C9DB:A4 27               LDY  CHARACTER
C9DD:C0 94               CPY  #$94     ;CTL-T?
C9DF:D0 04               BNE  CKINPUT1
C9E1:09 80               ORA  #$80     ;SET TERMINAL MODE
C9E3:D0 06               BNE  CKINPUT2 ;<ALWAYS>
C9E5:           *
C9E5:C0 92      CKINPUT1 CPY  #$92     ;CONTROL-R?
C9E7:D0 05               BNE  CIEND
C9E9:29 7F               AND  #$7F     ;RESET TERMINAL MODE
C9EB:9D B8 04   CKINPUT2 STA  STATEFLG,X
C9EE:60         CIEND    RTS
C9EF:           *
```



```
C9EF:                    CHN  SSC.TERM
C9EF:           ***********************************
C9EF:           *                                 *
C9EF:           * APPLE II SSC FIRMWARE           *
C9EF:           *                                 *
C9EF:           *   BY LARRY KENYON               *
C9EF:           *                                 *
C9EF:           *      -APRIL 1981-               *************
C9EF:           *                                             *
C9EF:           * (C) COPYRIGHT 1981 BY APPLE COMPUTER, INC. *
C9EF:           *                                             *
C9EF:           ***********************************************
C9EF:           * SHORT BLOCK MOVE        *
C9EF:           ***************************
C9EF:8A         BATCHIO  TXA
C9F0:0A                  ASL  A
C9F1:0A                  ASL  A
C9F2:0A                  ASL  A
C9F3:0A                  ASL  A
C9F4:85 26               STA  SLOT16
C9F6:A9 00               LDA  #0
C9F8:9D B8 05            STA  STSBYTE,X ;ZERO ERROR INDICATION
C9FB:70 0F               BVS  MOVIN
C9FD:           *
C9FD:A0 00      MOVOUT   LDY  #0
C9FF:B1 3C               LDA  (A1L),Y   ;GET BUFFER DATA
CA01:85 27               STA  CHARACTER
CA03:20 02 CC            JSR  ACIAOUT   ;SEND IT OUT THE ACIA
CA06:20 BA FC            JSR  NXTA1
CA09:90 F2               BCC  MOVOUT
CA0B:60                  RTS
CA0C:           *
CA0C:20 D2 CA   MOVIN    JSR  SRIN
CA0F:90 FB               BCC  MOVIN
CA11:B9 88 C0            LDA  RDREG,Y
CA14:A0 00               LDY  #0
CA16:91 3C               STA  (A1L),Y   ;PUT ACIA DATA INTO BUFFER
CA18:20 BA FC            JSR  NXTA1
CA1B:90 EF               BCC  MOVIN
CA1D:60                  RTS
CA1E:           *
CA1E:           ***********************************
CA1E:           *                                 *
CA1E:           *  TERMINAL MODE ROUTINES         *
CA1E:           *                                 *
CA1E:           ***********************************
CA1E:BD B8 04   CHECKTERM LDA STATEFLG,X ;HAVE WE ENTERED TERMINAL MODE?
CA21:10 31               BPL  TERMRTS   ;IF NOT, A SIMPLE RTS WILL DO. . .
CA23:           *
CA23:           * WE ENTER THE WORLD OF TERMINAL MODE
CA23:           *
CA23:A9 02      TERMMODE LDA  #$02      ;START IN SHIFT-LOCK STATE
CA25:48                  PHA            ;SHIFT STATE IS SAVED ON STACK
CA26:A9 7F               LDA  #$7F
CA28:20 E2 CD            JSR  KCMD1     ;RESET ECHO (DEFAULT TO FULL DUP)
CA2B:           *
CA2B:A4 24      TERMNEXT LDY  CH
CA2D:B1 28               LDA  (BASL),Y
```

```
CA2F:85 27              STA  CHARACTER ;SAVE SCREEN CHARACTER
CA31:A9 07     TERMNEXT1 LDA #$07     ;IMPLEMENT A FLASHING UNDERLINE
CA33:25 4F              AND  RNDH      ; FOR A CURSOR
CA35:D0 10              BNE  TERMNEXT3
CA37:A4 24              LDY  CH
CA39:A9 DF              LDA  #$DF
CA3B:D1 28              CMP  (BASL),Y  ;IS UNDERLINE ON THE SCREEN?
CA3D:D0 02              BNE  TERMNEXT2 ;IF NOT, PUT IT THERE
CA3F:A5 27              LDA  CHARACTER ;OTHERWISE USE TRUE SCREEN CHAR
CA41:91 28     TERMNEXT2 STA (BASL),Y
CA43:E6 4F              INC  RNDH      ;MAKE IT FLASH, BUT
CA45:E6 4F              INC  RNDH      ;NOT TOO SLOW AND NOT TOO FAST
CA47:          *
CA47:BD B8 04  TERMNEXT3 LDA STATEFLG,X ;ARE WE STILL IN TERM MODE?
CA4A:30 09              BMI  TERMACIAIN ;IF SO, GO CHECK ACIA
CA4C:          *
CA4C:20 11 CC  TERMEXIT JSR RESTORE    ;ALWAYS REPLACE OUR CURSOR
CA4F:68                 PLA            ;CLEAN UP THE STACK
CA50:A9 8D              LDA  #$8D      ;RETURN A <CR> TO COVER UP
CA52:85 27              STA  CHARACTER
CA54:60        TERMRTS RTS
CA55:          *
CA55:20 FF CA  TERMACIAIN JSR INPUT    ;ACIA INPUT?
CA58:90 0C              BCC  TERMKBDIN ;IF NOT, GO CHECK KEYBOARD
CA5A:20 11 CC           JSR  RESTORE   ;RESTORE CURSOR, INPUT->CHARACTER
CA5D:20 D1 C9           JSR  CKINPUT   ;CHECK FOR CTL-T, CTL-R
CA60:20 A3 CC           JSR  SCREENOUT1 ;INPUT->SCREEN ALWAYS
CA63:4C 2B CA           JMP  TERMNEXT  ;
CA66:          *
CA66:20 3E CC  TERMKBDIN JSR GETKBD    ;KEYPRESS?
CA69:90 C6              BCC  TERMNEXT1 ;SKIP IF NOT
CA6B:70 BE              BVS  TERMNEXT  ;BRANCH IF WE DID A KBD ESCAPE SEQ.
CA6D:BD 38 07           LDA  MISCFLG,X ;SHIFTING ENABLED?
CA70:0A                 ASL  A
CA71:10 22              BPL  TERMSEND1
CA73:68                 PLA            ;RECOVER TERMSTATE
CA74:A8                 TAY
CA75:A5 27              LDA  CHARACTER
CA77:C0 01              CPY  #1        ;1 = SHIFT LETTERS, XLATE NUMBERS
CA79:F0 20              BEQ  TERMCAP
CA7B:B0 34              BCS  TERMLOCK  ;2 MEANS CAPS LOCK MODE
CA7D:          *
CA7D:C9 9B     TERMNORM CMP #$9B       ;ESC?
CA7F:D0 06              BNE  TERMLETTER
CA81:          *
CA81:C8        TERMINC INY             ;INCREMENT STATE
CA82:98        TERMINC1 TYA
CA83:48                 PHA            ;PUT BACK ON STACK
CA84:4C 2B CA           JMP  TERMNEXT
CA87:          *
CA87:C9 C1     TERMLETTER CMP #$C1     ;<A?
CA89:90 08              BCC  TERMSEND
CA8B:C9 DB              CMP  #$DB      ;>Z?
CA8D:B0 04              BCS  TERMSEND
CA8F:09 20              ORA  #$20      ;IT'S A LETTER SO TRANSLATE TO LC
CA91:85 27              STA  CHARACTER
CA93:          *
CA93:98        TERMSEND TYA
```



```
CA94:48                 PHA            ;PUT STATE BACK ON STACK
CA95:20 68 CB  TERMSEND1 JSR OUTPUT1   ;GO OUTPUT
CA98:4C 2B CA           JMP  TERMNEXT
CA9B:          *
CA9B:C9 9B     TERMCAP CMP #$9B        ;TWO ESCAPES?
CA9D:F0 E2              BEQ  TERMINC
CA9F:C9 B0              CMP  #$B0      ;<0?
CAA1:90 0A              BCC  TERMCAP1
CAA3:C9 BB              CMP  #$BB      ;>COLON?
CAA5:B0 06              BCS  TERMCAP1
CAA7:          *
CAA7:          * ESC <NUMBER> SO TRANSLATE INTO MISSING ASCII CHAR
CAA7:          *
CAA7:A8                 TAY
CAA8:B9 09 CA           LDA  TRANSLATE-$B0,Y
CAAB:85 27              STA  CHARACTER
CAAD:A0 00     TERMCAP1 LDY #0         ;BACK TO STATE 0
CAAF:F0 E2              BEQ  TERMSEND  ;<ALWAYS>
CAB1:          *
CAB1:C9 9B     TERMLOCK CMP #$9B       ;ESC?
CAB3:D0 DE              BNE  TERMSEND
CAB5:A0 00              LDY  #0
CAB7:F0 C9              BEQ  TERMINC1;<ALWAYS>
CAB9:          *
CAB9:          ******************************
CAB9:          * TRANSLATE TABLE            *
CAB9:          ******************************
CAB9:9B        TRANSLATE DFB $9B       ;ESC
CABA:9C                 DFB  $9C       ;FS
CABB:9F                 DFB  $9F       ;US
CABC:DB                 DFB  $DB       ;LEFT BRACKET
CABD:DC                 DFB  $DC       ;LEFT SLASH
CABE:DF                 DFB  $DF       ;UNDERSCORE
CABF:FB                 DFB  $FB       ;LEFT ENCLOSE
CAC0:FC                 DFB  $FC       ;VERTICAL BAR
CAC1:FD                 DFB  $FD       ;RIGHT ENCLOSE
CAC2:FE                 DFB  $FE       ;TILDE
CAC3:FF                 DFB  $FF       ;RUB
CAC4:          *
CAC4:                   CHN  SSC.CORE
```

```
CAC4:            ***********************************
CAC4:            *                                 *
CAC4:            * APPLE II SSC FIRMWARE           *
CAC4:            *                                 *
CAC4:            *    BY LARRY KENYON              *
CAC4:            *                                 *
CAC4:            *     -JANUARY 1981-              **************
CAC4:            *                                              *
CAC4:            * (C) COPYRIGHT 1981 BY APPLE COMPUTER, INC.   *
CAC4:            *                                              *
CAC4:            ************************************************
CAC4:            *                                 *
CAC4:            * CORE SUBROUTINES                *
CAC4:            *                                 *
CAC4:            ***********************************
CAC4:            ***********************************
CAC4:            * GENERAL PURPOSE WAIT ROUTINE    *
CAC4:            ***********************************
CAC4:            *
CAC4:            * WAITMS WAITS FOR [A-REG] MILLISECONDS (256 IF A-REG=0)
CAC4:            *
CAC4:A2 CA       WAITMS   LDX  #202
CAC6:CA          WAITMS1  DEX              ;<DON'T LET THIS LOOP CROSS A PAGE>
CAC7:D0 FD                BNE  WAITMS1     ;5 MICROSECOND LOOP
CAC9:38                   SEC
CACA:E9 01                SBC  #01
CACC:D0 F6                BNE  WAITMS
CACE:AE F8 07             LDX  MSLOT
CAD1:60                   RTS
CAD2:            ****************************************
CAD2:            * ACIA STATUS REGISTER READ ROUTINES *
CAD2:            ****************************************
CAD2:            *
CAD2:            *   SRIN USED TO CHECK ACIA INPUT STATUS
CAD2:            *
CAD2:A4 26       SRIN     LDY  SLOT16      ;SLOT16=$N0
CAD4:B9 89 C0             LDA  STREG,Y
CAD7:48                   PHA
CAD8:29 20                AND  #$20        ;DCD?
CADA:4A                   LSR  A           ;AN ERROR IF NOT
CADB:4A                   LSR  A
CADC:85 35                STA  ZPTEMP
CADE:68                   PLA
CADF:29 0F                AND  #$0F
CAE1:C9 08                CMP  #$08        ;SET CARRY IF RDR FULL, ELSE CLEAR
CAE3:90 04                BCC  SRIN1
CAE5:29 07                AND  #$07        ;PE, FE, OVR VALID ONLY WHEN RDR=1
CAE7:B0 02                BCS  SRIN2       ;<ALWAYS>
CAE9:A5 35       SRIN1    LDA  ZPTEMP
CAEB:05 35       SRIN2    ORA  ZPTEMP      ;GET DCD ERROR BIT
CAED:F0 05                BEQ  SRIN3       ;BRANCH IF NO ERRORS FOUND
CAEF:09 20                ORA  #$20        ;ELSE SET BIT 5 TO OFFSET FOR PASCAL
CAF1:9D B8 05             STA  STSBYTE,X   ;AND SAVE IN STATUS TEMP
CAF4:60          SRIN3    RTS              ;CY=1 MEANS DATA IS AVAILABLE
CAF5:            *
CAF5:            * SROUT CHECKS IF TDR IS EMPTY + HARDWARE HANDSHAKE IS OK
CAF5:            *
CAF5:A4 26       SROUT    LDY  SLOT16
CAF7:B9 89 C0             LDA  STREG,Y
CAFA:29 70                AND  #$70
CAFC:C9 10                CMP  #$10        ;EQU IF TDR EMPTY, DCD, DSR, & CTS
CAFE:60                   RTS
CAFF:            *
CAFF:            **************************
CAFF:            * GENERAL INPUT ROUTINE *
CAFF:            **************************
CAFF:20 D2 CA    INPUT    JSR  SRIN
CB02:90 15                BCC  NOINPUT1
CB04:            *
CB04:B9 88 C0             LDA  RDREG,Y     ;GET THE ACIA INPUT
CB07:09 80                ORA  #$80        ;SET HI BIT FOR BASIC
CB09:C9 8A                CMP  #$8A        ;LINEFEED?
CB0B:D0 09                BNE  INPUT2
CB0D:            *
CB0D:A8                   TAY
CB0E:BD 38 07             LDA  MISCFLG,X   ;SEE IF WE SHOULD EAT IT
CB11:29 20                AND  #$20
CB13:D0 03                BNE  NOINPUT     ;IF SO, JUST KEEP IT A SECRET
CB15:98                   TYA
CB16:            *
CB16:38          INPUT2   SEC              ;INDICATE DATA
CB17:60                   RTS
CB18:            *
CB18:18          NOINPUT  CLC              ;CARRY CLEAR FOR NO INPUT
CB19:60          NOINPUT1 RTS
CB1A:            *
CB1A:            ***************************
CB1A:            * GENERAL OUTPUT ROUTINE *
CB1A:            ***************************
CB1A:            *
CB1A:            * START OF COMMAND CHECK ROUTINE
CB1A:            *
CB1A:A4 26       CMDSEQCK LDY  SLOT16
CB1C:B9 81 C0             LDA  DIPSW1,Y
CB1F:4A                   LSR  A
CB20:B0 36                BCS  NOCMD       ;DON'T WORRY ABOUT CMD SEQ FOR SIC
CB22:BD B8 04             LDA  STATEFLG,X
CB25:29 07                AND  #$07        ;ARE WE IN A COMMAND SEQUENCE?
CB27:F0 05                BEQ  ESCCHECK
CB29:20 FC CD             JSR  CMDPROC     ;IF SO, GOTO COMMAND CENTRAL
CB2C:38                   SEC              ;INDICATE COMMAND
CB2D:60                   RTS
CB2E:            *
CB2E:A5 27       ESCCHECK LDA  CHARACTER
CB30:29 7F                AND  #$7F        ;IGNORE HIGH BIT
CB32:DD 38 05             CMP  CMDBYTE,X   ;IS THIS BEGINNING OF A CMD SEQ?
CB35:D0 05                BNE  XOFFCK
CB37:FE B8 04             INC  STATEFLG,X  ;START UP COMMAND MODES
CB3A:38                   SEC              ;INDICATE COMMAND
CB3B:60                   RTS
CB3C:            *
CB3C:BD 38 07    XOFFCK   LDA  MISCFLG,X   ;IS XON ENABLED?
CB3F:29 08                AND  #$08
CB41:F0 15                BEQ  NOCMD       ;SKIP THIS IF NOT
CB43:            *
CB43:20 FF CA             JSR  INPUT       ;ANY INPUT?
```

```
CB46:90 10           BCC  NOCMD    ;IF NOT, GO OUTPUT
CB48:C9 93           CMP  #$93     ;IS IT AN XOFF?
CB4A:F0 0E           BEQ  XONWAIT  ;IF SO, GO WAIT FOR ANOTHER INPUT
CB4C:48              PHA
CB4D:BD 38 07        LDA  MISCFLG,X ;CIC MODE?
CB50:4A              LSR  A
CB51:4A              LSR  A
CB52:68              PLA
CB53:90 04           BCC  ANRTS
CB55:9D B8 06        STA  BUFBYTE,X ;IF SO, WE HAVE A BUFFER
CB58:18     NOCMD    CLC           ;INDICATE NOT A CMD SEQ
CB59:60     ANRTS    RTS
CB5A:       *
CB5A:20 AA C8 XONWAIT JSR GETCHAR  ;GET ACIA/KBD DATA
CB5D:C9 91           CMP  #$91     ;IS IT AN XON?
CB5F:D0 F9           BNE  XONWAIT  ;IF NOT, WAIT
CB61:18              CLC           ;OTHERWISE, INDICATE NOT A CMD SEQ
CB62:60              RTS           ; AND RETURN
CB63:       ***************************************************
CB63:       * NOW THE OUTPUT ROUTINE YOU'VE BEEN WAITING FOR *
CB63:       ***************************************************
CB63:20 1A CB OUTPUT  JSR  CMDSEQCK
CB66:B0 F1           BCS  ANRTS    ;DON'T OUTPUT COMMAND SEQUENCES
CB68:       *
CB68:20 9E CC OUTPUT1 JSR SCREENOUT
CB6B:       *
CB6B:A4 26  OUTPUT2  LDY  SLOT16
CB6D:B9 81 C0        LDA  DIPSW1,Y
CB70:4A              LSR  A
CB71:90 4E           BCC  OUTPUT3  ;SKIP ETX/ACK FOR NATIVE MODES
CB73:4A              LSR  A
CB74:90 4B           BCC  OUTPUT3  ;BRANCH IF NOT P8A EMULATION
CB76:       *
CB76:       *********************
CB76:       * P8A ETX/ACK STUFF*
CB76:       *********************
CB76:       * AFTER 148 CHARACTERS BUT NOT WITHIN AN ESCAPE SEQUENCE
CB76:       * OF UP TO 5 CHARACTERS, THE HANDSHAKE IS PERFORMED
CB76:       * (WILL DELAY UNTIL 'NOT ESC' AND THEN 4 MORE CHARS
CB76:       *  OR UNTIL AN 'ESC')
CB76:       *
CB76:A5 27  P8AOUT1  LDA  CHARACTER ;SAVE CHAR ON STACK
CB78:48              PHA
CB79:BD 38 04        LDA  HANDSHKE,X ;CHAR COUNT FOR BUFFER FULL
CB7C:C9 67           CMP  #103     ;IF <103 THEN 153 CHARS IN BUFFER
CB7E:90 10           BCC  ETX
CB80:C9 6C           CMP  #108     ;IF >=108 THEN LESS THAN 149 CHARS
CB82:B0 22           BCS  P8AOUT2  ; SO NO HANDSHAKE IS NEEDED YET
CB84:C9 6B           CMP  #107     ;SETS CARRY IF 107 (149 SENT)
CB86:68              PLA
CB87:48              PHA
CB88:49 9B           EOR  #$9B     ;ESC?
CB8A:29 7F           AND  #$7F     ;IGNORE HI-BIT
CB8C:D0 18           BNE  P8AOUT2  ;COUNT AS 1 OF 5 IF NOT 'ESC'
CB8E:B0 19           BCS  P8AOUT3  ;DON'T COUNT IF 149TH CHAR IS 'ESC'
CB90:       *
CB90:BD B8 04 ETX    LDA  STATEFLG,X ;SEND QUERY CHAR TO PRINTER
CB93:29 1F           AND  #$1F     ;(DEFAULT IS ETX)
```



```
CB95:09 80           ORA  #$80
CB97:85 27           STA  CHARACTER
CB99:20 02 CC        JSR  ACIAOUT
CB9C:20 AA C8 ACK    JSR  GETCHAR  ;GET ACIA/KBD DATA
CB9F:49 86           EOR  #$86     ;ACK?
CBA1:D0 ED           BNE  ETX      ;IF NOT ACK, REPEAT HANDSHAKE
CBA3:9D 38 04        STA  HANDSHKE,X ;INIT CHAR COUNT TO 255
CBA6:       *
CBA6:DE 38 04 P8AOUT2 DEC HANDSHKE,X
CBA9:68     P8AOUT3  PLA           ;GET REAL CHAR TO OUTPUT
CBAA:85 27           STA  CHARACTER
CBAC:49 8D           EOR  #$8D     ;IF CR AND CR DELAY MODE
CBAE:0A              ASL  A
CBAF:D0 0A           BNE  P8AOUT4  ; THEN FAKE CHAR COUNT TO LESS THAN
CBB1:BD B8 03        LDA  DELAYFLG,X ; 48 TO FORCE HANDSHAKE ON NEXT
CBB4:29 30           AND  #$30     ; CHARACTER OUT
CBB6:F0 03           BEQ  P8AOUT4
CBB8:9D 38 04        STA  HANDSHKE,X
CBBB:       *
CBBB:20 02 CC P8AOUT4 JSR ACIAOUT
CBBE:4C EA CB        JMP  LFGEN    ;(SKIP DELAYS)
CBC1:       ********************************
CBC1:       * AND BACK TO NORMAL OUTPUT *
CBC1:       ********************************
CBC1:20 02 CC OUTPUT3 JSR ACIAOUT  ;OUTPUT THE CHARACTER
CBC4:       *
CBC4:       * NOW CHECK FOR CR, LF, AND FF DELAYS
CBC4:       *
CBC4:0A              ASL  A
CBC5:A8              TAY
CBC6:BD B8 03        LDA  DELAYFLG,X ;GET DELAY FLAGS
CBC9:C0 18           CPY  #$18     ;FORM FEED?
CBCB:F0 0C           BEQ  OUTDLY1
CBCD:4A              LSR  A
CBCE:4A              LSR  A        ;RIGHT JUSTIFY LF DELAY
CBCF:C0 14           CPY  #$14     ;LINE FEED?
CBD1:F0 06           BEQ  OUTDLY1
CBD3:4A              LSR  A
CBD4:4A              LSR  A        ;RIGHT JUSTIFY CR DELAY
CBD5:C0 1A           CPY  #$1A     ;CARRIAGE RETURN?
CBD7:D0 25           BNE  OUTPUTEND
CBD9:29 03  OUTDLY1  AND  #$03     ;JUST WANT LOWEST 2 BITS
CBDB:F0 0D           BEQ  LFGEN    ;NO DELAY INDICATED
CBDD:A8              TAY
CBDE:B9 FE CB        LDA  DLYTBL-1,Y
CBE1:A8              TAY           ;DELAY IN 32 MSEC INCREMENTS
CBE2:A9 20  OUTDLYLP LDA  #32      ;
CBE4:20 C4 CA        JSR  WAITMS
CBE7:88              DEY
CBE8:D0 F8           BNE  OUTDLYLP
CBEA:       *
CBEA:       * CHECK ON LF GENERATION OPTION
CBEA:       *
CBEA:A5 27  LFGEN    LDA  CHARACTER
CBEC:0A              ASL  A
CBED:C9 1A           CMP  #$1A     ;CARRIAGE RETURN?
CBEF:D0 0D           BNE  OUTPUTEND
CBF1:BD 38 07        LDA  MISCFLG,X ;IS LF GENERATE ENABLED?
```

```
CBF4:6A              ROR  A
CBF5:90 07           BCC  OUTPUTEND
CBF7:A9 8A           LDA  #$8A
CBF9:85 27           STA  CHARACTER ;LINE FEED
CBFB:4C 6B CB        JMP  OUTPUT2   ;(DON'T ECHO IT)
CBFE:60         OUTPUTEND RTS
CBFF:           *
CBFF:01         DLYTBL  DFB  $01         ;32 MSEC
CC00:08                 DFB  $08         ;1/4 SEC
CC01:40                 DFB  $40         ; 2  SEC
CC02:           ***************************
CC02:           * ACIA OUTPUT ROUTINE     *
CC02:           ***************************
CC02:20 F5 CA   ACIAOUT JSR  SROUT       ;READY FOR OUTPUT?
CC05:D0 FB              BNE  ACIAOUT
CC07:98                 TYA
CC08:09 89              ORA  #$89        ;PREPARE TO ADDRESS ACIA,
CC0A:A8                 TAY              ; CAUSING 6502 FALSE READ TO OCCUR
CC0B:A5 27              LDA  CHARACTER ; ON PAGE $BF (AVOIDING RDR READ)
CC0D:99 FF BF           STA  $BFFF,Y     ;HERE YOU ARE ACIA
CC10:60                 RTS
CC11:           *
CC11:           *************************************
CC11:           * RESTORE CURSOR (NOT FOR PASCAL) *
CC11:           * (A-REG SHOULD CONTAIN NEW CHAR) *
CC11:           *************************************
CC11:48         RESTORE PHA              ;SAVE NEW CHARACTER
CC12:A4 24              LDY  CH
CC14:A5 27              LDA  CHARACTER ;OLD CHARACTER
CC16:91 28              STA  (BASL),Y
CC18:68                 PLA
CC19:           *
CC19:C9 95              CMP  #$95        ;SCREEN PICK?
CC1B:D0 0C              BNE  RESTOREND
CC1D:A5 27              LDA  CHARACTER ;IF SO, USE SCREEN CHAR
CC1F:C9 20              CMP  #$20        ;INVERSE?
CC21:B0 06              BCS  RESTOREND
CC23:20 DF CC           JSR  GETXLATE ;REVERSE THE TRANSLATION
CC26:59 DB CC           EOR  REVMASK,Y
CC29:85 27      RESTOREND STA CHARACTER
CC2B:60                 RTS
CC2C:           *
CC2C:                   CHN  SSC.UTIL
```



```
CC2C:           ************************************
CC2C:           *                                  *
CC2C:           * APPLE II SSC FIRMWARE            *
CC2C:           *                                  *
CC2C:           *    BY LARRY KENYON               *
CC2C:           *                                  *
CC2C:           *    -JANUARY 1981-                **************
CC2C:           *                                               *
CC2C:           * (C) COPYRIGHT 1981 BY APPLE COMPUTER, INC. *
CC2C:           *                                               *
CC2C:           *************************************************
CC2C:           *                                  *
CC2C:           * UTILITY ROUTINES                 *
CC2C:           *                                  *
CC2C:           ************************************
CC2C:           * PASCAL-BASIC KEYBOARD FETCH *
CC2C:           ************************************
CC2C:18         CKKBD   CLC              ;RETURN CARRY CLEAR FOR NO DATA
CC2D:BD 38 07           LDA  MISCFLG,X
CC30:29 04              AND  #$04        ;ANSWER NO IF KEYBOARD IS DISABLED
CC32:F0 09              BEQ  CKKBDXIT
CC34:           *
CC34:AD 00 C0   CKKBD1  LDA  KBD
CC37:10 04              BPL  CKKBDXIT
CC39:8D 10 C0           STA  KBDSTRB
CC3C:38                 SEC              ;INDICATE DATA
CC3D:60         CKKBDXIT RTS
CC3E:           *************************************************
CC3E:           * GET A CHAR FROM KEYBOARD FOR BASIC ONLY *
CC3E:           *************************************************
CC3E:E6 4E      GETKBD  INC  RNDL        ;MIX UP RANDOM # SEED
CC40:D0 02              BNE  GETKBD1     ; FOR BASIC
CC42:E6 4F              INC  RNDH
CC44:20 2C CC   GETKBD1 JSR  CKKBD       ;KEYBOARD FETCH ROUTINE
CC47:B8                 CLV              ;INDICATE NO ESCAPE SEQUENCE
CC48:90 F3              BCC  CKKBDXIT ;EXIT IF NO KEY PRESS
CC4A:20 11 CC           JSR  RESTORE     ;DO BASIC CURSED DUTY
CC4D:29 7F              AND  #$7F
CC4F:DD 38 05           CMP  CMDBYTE,X ;IS IT THE START OF A COMMAND?
CC52:D0 3D              BNE  GETKBDONE ;IF NOT, EXIT INDICATING DATA
CC54:A4 26              LDY  SLOT16
CC56:B9 81 C0           LDA  DIPSW1,Y    ;ONLY DO CMD ESC FOR PPC, SIC MODES
CC59:4A                 LSR  A
CC5A:B0 35              BCS  GETKBDONE
CC5C:           ****************************
CC5C:           * KEYBOARD ESCAPE HANDLER *
CC5C:           ****************************
CC5C:A0 0A      KBDESC  LDY  #$A         ;FIRST PRINT A PROMPT
CC5E:B9 93 CC   PROMPTLOOP LDA PROMPTBL,Y
CC61:85 27              STA  CHARACTER
CC63:98                 TYA
CC64:48                 PHA
CC65:20 A3 CC           JSR  SCREENOUT1 ;ALWAYS SEND TO SCREEN
CC68:68                 PLA
CC69:A8                 TAY
CC6A:88                 DEY
CC6B:10 F1              BPL  PROMPTLOOP
CC6D:           *
```

```
CC6D:A9 01                     LDA  #1           ;START OUT IN COMMAND STATE 1
CC6F:20 7B CE                  JSR  SETOSTATE
CC72:              *
CC72:20 34 CC      GETCMD      JSR  CKKBD1       ;WAIT FOR KEYBOARD CHARACTER
CC75:10 FB                     BPL  GETCMD
CC77:C9 88                     CMP  #$88         ;BACKSPACE?
CC79:F0 E1                     BEQ  KBDESC       ;IF SO, THEN START OVER
CC7B:85 27                     STA  CHARACTER
CC7D:              *
CC7D:20 A3 CC                  JSR  SCREENOUT1
CC80:20 1A CB                  JSR  CMDSEQCK     ;PUMP THRU CMD INTERPRETER
CC83:              *
CC83:BD B8 04                  LDA  STATEFLG,X   ;ARE WE DONE?
CC86:29 07                     AND  #$07
CC88:D0 E8                     BNE  GETCMD       ;IF NOT, GO AGAIN
CC8A:              *
CC8A:A9 8D                     LDA  #$8D         ;FORCE BACK A CARRIAGE RETURN
CC8C:85 27                     STA  CHARACTER
CC8E:2C 58 FF                  BIT  IORTS        ;INDICATE THAT A CMD SEQ HAS OCCURRED
CC91:38            GETKBDONE   SEC               ;INDICATE SUCCESS
CC92:60                        RTS
CC93:              *
CC93:              *
CC93:BA C3 D3      PROMPTBL    ASC  ":CSS         ELPPA"
CC96:D3 A0 C5
CC99:CC D0 D0
CC9C:C1
CC9D:8D                        DFB  $8D
CC9E:              *
CC9E:              *********************************************************
CC9E:              * ROUTINE TO PRINT A CHARACTER ON THE CURRENT DISPLAY *
CC9E:              *********************************************************
CC9E:BD 38 07      SCREENOUT   LDA  MISCFLG,X
CCA1:10 13                     BPL  NOOUT        ;IF SCREEN DISABLED
CCA3:              *
CCA3:BD 38 07      SCREENOUT1  LDA  MISCFLG,X    ;ENTRY AFTER ECHO CHECK
CCA6:29 02                     AND  #$02         ;IF IT ISN'T CIC MODE,
CCA8:F0 0D                     BEQ  ASCREEN      ;ALWAYS USE THE APPLE SCREEN
CCAA:BD B8 04                  LDA  STATEFLG,X   ;CURRENT SCREEN = APPLE SCREEN?
CCAD:29 38                     AND  #$38
CCAF:F0 06                     BEQ  ASCREEN      ;SLOT 0= APPLE SCREEN
CCB1:              *
CCB1:8A                        TXA               ;JUMP TO CN00 SPACE
CCB2:48                        PHA
CCB3:A9 AF                     LDA  #>SENDCD-1   ; TO VECTOR TO THE PERIPHERAL
CCB5:48                        PHA               ; IN THE CHAIN SLOT
CCB6:60            NOOUT       RTS
CCB7:              *
CCB7:              * APPLE 40-COL SCREEN DRIVER
CCB7:              *
CCB7:20 DF CC      ASCREEN     JSR  GETXLATE     ;GET THE TRANSLATE OPTIONS
CCBA:09 80                     ORA  #$80         ;SET HIGH BIT OF CHAR
CCBC:C9 E0                     CMP  #$E0         ;LOWERCASE?
CCBE:90 06                     BCC  TESTLETTER
CCC0:59 D3 CC                  EOR  LCMASK,Y     ;DO LOWERCASE TRIP
CCC3:4C F6 FD      TOSCREEN    JMP  VIDOUT       ;ALL REGS ARE PRESERVED
CCC6:              *
CCC6:              * IF UPPERCASE, WE ONLY MAP LETTERS
CCC6:              *
CCC6:C9 C1         TESTLETTER  CMP  #$C1         ;<A?
CCC8:90 F9                     BCC  TOSCREEN
CCCA:C9 DB                     CMP  #$DB         ;>Z?
CCCC:B0 F5                     BCS  TOSCREEN
CCCE:59 D7 CC                  EOR  UCMASK,Y
CCD1:90 F0                     BCC  TOSCREEN     ;<ALWAYS>
CCD3:              *
CCD3:              * MASKS FOR CASE TRANSLATION
CCD3:20 00 E0      LCMASK      DFB  $20,$00,$E0,$20
CCD6:20
CCD7:00 00 00      UCMASK      DFB  $00,$00,$00,$C0
CCDA:C0
CCDB:00 00 E0      REVMASK     DFB  $00,$00,$E0,$C0
CCDE:C0
CCDF:              *
CCDF:BD B8 03      GETXLATE    LDA  DELAYFLG,X   ;TRANSLATE OPTIONS IN B6-B7
CCE2:2A                        ROL  A
CCE3:2A                        ROL  A
CCE4:2A                        ROL  A
CCE5:29 03                     AND  #$03
CCE7:A8                        TAY
CCE8:A5 27                     LDA  CHARACTER
CCEA:60                        RTS
CCEB:              *
```

(listings continued on next page)

```
CCEB:                     CHN  SSC.CMD
CCEB:            ***********************************
CCEB:            *                                 *
CCEB:            * APPLE II SSC FIRMWARE           *
CCEB:            *                                 *
CCEB:            *   BY LARRY KENYON               *
CCEB:            *                                 *
CCEB:            *     -JANUARY 1981-              **************
CCEB:            *                                              *
CCEB:            * (C) COPYRIGHT 1981 BY APPLE COMPUTER, INC.   *
CCEB:            *                                              *
CCEB:            ************************************************
CCEB:            *                                 *
CCEB:            * SSC COMMAND PROCESSOR           *
CCEB:            *                                 *
CCEB:            ***********************************
CCEB:            ********************************************************
CCEB:            * COMMAND TABLE (USED BY COMMAND PROCESSER ROUTINE *
CCEB:            ********************************************************
CCEB:42          CMDTBL  DFB  $42         ;B(REAK)
CCEC:67                  DFB  $67         ;CIC     PAS  NS=7
CCED:C0                  DFB  >BREAKCMD-1
CCEE:54                  DFB  $54         ;T(ERMINAL)
CCEF:47                  DFB  $47         ;CIC          NS=7
CCF0:A6                  DFB  >TERMCMD-1
CCF1:43                  DFB  $43         ;C(R GENERATE)
CCF2:87                  DFB  $87         ;    PPC      NS=7
CCF3:A6                  DFB  >TERMCMD-1
CCF4:51                  DFB  $51         ;Q(UIT)
CCF5:47                  DFB  $47         ;CIC          NS=7
CCF6:B8                  DFB  >QUITCMD-1
CCF7:52                  DFB  $52         ;R(ESET)
CCF8:C7                  DFB  $C7         ;CIC PPC      NS=7
CCF9:AC                  DFB  >RESETCMD-1
CCFA:5A                  DFB  $5A         ;Z COMMAND
CCFB:E7                  DFB  $E7         ;CIC PPC PAS  NS=7
CCFC:F3                  DFB  >ZCMD-1
CCFD:49                  DFB  $49         ;I COMMAND
CCFE:90                  DFB  $90         ;    PPC      NS=0
CCFF:D3                  DFB  >ICMD-1
CD00:4B                  DFB  $4B         ;K COMMAND
CD01:90                  DFB  $90         ;    PPC      NS=0
CD02:DF                  DFB  >KCMD-1
CD03:            *
CD03:45                  DFB  $45         ;E(CHO)
CD04:43                  DFB  $43         ;CIC          NS=3
CD05:80                  DFB  $80
CD06:46                  DFB  $46         ;F(ROMKYBD)
CD07:E3                  DFB  $E3         ;CIC PPC PAS  NS=3
CD08:04                  DFB  $04
CD09:4C                  DFB  $4C         ;L(F GENERATE)
CD0A:E3                  DFB  $E3         ;CIC PPC PAS  NS=3
CD0B:01                  DFB  $01
CD0C:58                  DFB  $58         ;X(OFF)
CD0D:E3                  DFB  $E3         ;CIC PPC PAS  NS=3
CD0E:08                  DFB  $08
CD0F:54                  DFB  $54         ;T(ABBING)
CD10:83                  DFB  $83         ;    PPC      NS=3
CD11:40                  DFB  $40
CD12:53                  DFB  $53         ;S(HIFTING)
CD13:43                  DFB  $43         ;CIC          NS=3
CD14:40                  DFB  $40
CD15:4D                  DFB  $4D         ;M(UNCH LF)
CD16:E3                  DFB  $E3         ;CIC PPC PAS  NS=3
CD17:20                  DFB  $20
CD18:            *
CD18:00                  DFB  $0C         ;END OF FIRST PART MARKER
CD19:            *
CD19:42          CMDTBL1 DFB  $42         ;B(AUD)
CD1A:F6                  DFB  $F6         ;CIC PPC PAS  NS=6
CD1B:7C                  DFB  >BAUDCMD-1
CD1C:50                  DFB  $50         ;P(ARITY)
CD1D:F6                  DFB  $F6         ;CIC PPC PAS  NS=6
CD1E:9A                  DFB  >PARITYCMD-1
CD1F:44                  DFB  $44         ;D(ATA)
CD20:F6                  DFB  $F6         ;CIC PPC PAS  NS=6
CD21:9B                  DFB  >DATACMD-1
CD22:46                  DFB  $46         ;F(F DELAY)
CD23:F6                  DFB  $F6         ;CIC PPC PAS  NS=6
CD24:46                  DFB  >FFCMD-1
CD25:4C                  DFB  $4C         ;L(F DELAY)
CD26:F6                  DFB  $F6         ;CIC PPC PAS  NS=6
CD27:40                  DFB  >LFCMD-1
CD28:43                  DFB  $43         ;C(R DELAY)
CD29:F6                  DFB  $F6         ;CIC PPC PAS  NS=6
CD2A:3A                  DFB  >CRCMD-1
CD2B:54                  DFB  $54         ;T(RANSLATE)
CD2C:D6                  DFB  $D6         ;CIC PPC      NS=6
CD2D:34                  DFB  >TRANCMD-1
CD2E:4E                  DFB  $4E         ;N COMMAND
CD2F:90                  DFB  $90         ;    PPC      NS=0
CD30:E8                  DFB  >NCMD-1
CD31:53                  DFB  $53         ;S(CREENSLOT)
CD32:56                  DFB  $56         ;CIC          NS=6
CD33:60                  DFB  >SSLOTCMD-1
CD34:            *
CD34:00                  DFB  $00         ;END OF TABLE MARKER
CD35:            *
CD35:            ***********************
CD35:            * COMMAND ROUTINES    *
CD35:            * (CALLED BY PARSER)  *
CD35:            * (MUST START IN      *
CD35:            *   PAGE $CD .      ) *
CD35:            ***********************
CD35:A9 3F       TRANCMD LDA  #$3F        ;SET SCREEN TRANSLATE OPTIONS
CD37:A0 07               LDY  #$7
CD39:D0 10               BNE  DELAYSET    ;<ALWAYS>
CD3B:A9 CF       CRCMD   LDA  #$CF        ;SET CR DELAY
CD3D:A0 05               LDY  #$5
CD3F:D0 0A               BNE  DELAYSET    ;<ALWAYS>
CD41:            *
CD41:A9 F3       LFCMD   LDA  #$F3        ;SET LF DELAY
CD43:A0 03               LDY  #$3
CD45:D0 04               BNE  DELAYSET    ;<ALWAYS>
CD47:            *
CD47:A9 FC       FFCMD   LDA  #$FC        ;SET FF DELAY
```

```
CD49:A0 01             LDY   #$1
CD4B:3D B8 03  DELAYSET AND  DELAYFLG,X ;DON'T DISTURB THE OTHER FLAGS
CD4E:85 2A             STA   ZPTMP1
CD50:BD 38 04          LDA   PARAMETER,X
CD53:29 03             AND   #$03      ;JUST USE TWO BITS
CD55:18                CLC
CD56:6A                ROR   A         ;ONCE FOR FUN
CD57:2A        ROTATE  ROL   A         ;CHANGE DIRECTIONS
CD58:88                DEY
CD59:D0 FC             BNE   ROTATE    ;PREPARE IT TO OR INTO THE FLAGS
CD5B:          *
CD5B:05 2A             ORA   ZPTMP1
CD5D:9D B8 03          STA   DELAYFLG,X
CD60:60                RTS
CD61:          *
CD61:29 07     SSLOTCMD AND  #$7       ;SET SLOT COMMAND
CD63:0A                ASL   A
CD64:0A                ASL   A
CD65:0A                ASL   A
CD66:85 2A             STA   ZPTMP1
CD68:0A                ASL   A
CD69:C5 26             CMP   SLOT16    ;MAKE SURE WE DON'T SET IT
CD6B:F0 0F             BEQ   SSLOTCMD1 ; TO OUR OWN SLOT
CD6D:BD B8 04          LDA   STATEFLG,X
CD70:29 C7             AND   #$C7      ;PUT NEW SLOT NUMBER IN BITS 3-5
CD72:05 2A             ORA   ZPTMP1    ; OF CMDBYTE,X
CD74:9D B8 04          STA   STATEFLG,X
CD77:A9 00             LDA   #0        ;STORE ZERO INTO
CD79:9D 38 06          STA   CHNBYTE,X ;SLOT OFFSET (SET TO CN00 ENTRY)
CD7C:60        SSLOTCMD1 RTS
CD7D:          *
CD7D:29 0F     BAUDCMD AND   #$0F      ;SET NEW BAUD RATE
CD7F:D0 07             BNE   BAUDCMD2
CD81:B9 81 C0  BAUDCMD1 LDA  DIPSW1,Y  ;ZERO PARM = RELOAD FROM SWITCHES
CD84:4A                LSR   A
CD85:4A                LSR   A
CD86:4A                LSR   A
CD87:4A                LSR   A
CD88:09 10     BAUDCMD2 ORA  #$10      ;SET INT. BAUD RATE GENERATOR
CD8A:85 2A             STA   ZPTMP1
CD8C:A9 E0             LDA   #$E0
CD8E:85 2B     CTLREGSET STA ZPTMP2
CD90:B9 8B C0          LDA   CTLREG,Y
CD93:25 2B             AND   ZPTMP2
CD95:05 2A             ORA   ZPTMP1
CD97:99 8B C0          STA   CTLREG,Y
CD9A:60                RTS
CD9B:          *
CD9B:88        PARITYCMD DEY           ;TRICK: SO CTLREG,Y ACTUALLY
CD9C:          *                       ADDRESSES THE COMMAND REG
CD9C:          *
CD9C:0A        DATACMD ASL   A         ;SET NEW # OF DATA BITS
CD9D:0A                ASL   A
CD9E:0A                ASL   A
CD9F:0A                ASL   A
CDA0:0A                ASL   A
CDA1:85 2A     DATACMD1 STA  ZPTMP1
CDA3:A9 1F             LDA   #$1F
CDA5:D0 E7             BNE   CTLREGSET ;<ALWAYS>
CDA7:          *
CDA7:1E B8 04  TERMCMD ASL   STATEFLG,X ;SET TERMINAL MODE
CDAA:38                SEC
CDAB:B0 10             BCS   QCMD1     ;<ALWAYS>
CDAD:          *
CDAD:99 89 C0  RESETCMD STA  RESET,Y   ;DROP RTS, DTR
CDB0:20 93 FE          JSR   SETSCR    ;PR#0
CDB3:20 89 FE          JSR   SETKBD    ;IN#0
CDB6:AE F8 07          LDX   MSLOT
CDB9:1E B8 04  QUITCMD ASL   STATEFLG,X ;CLEAR TERMINAL MODE
CDBC:18                CLC
CDBD:7E B8 04  QCMD1   ROR   STATEFLG,X
CDC0:60                RTS
CDC1:          *
CDC1:B9 8A C0  BREAKCMD LDA  CMDREG,Y  ;SEND BREAK SIGNAL
CDC4:48                PHA             ; FOR 233 MILLISECONDS
CDC5:09 0C             ORA   #$0C
CDC7:99 8A C0          STA   CMDREG,Y
CDCA:A9 E9             LDA   #233      ;DELAY FOR 233 MICROSEC.
CDCC:20 C4 CA          JSR   WAITMS
CDCF:68                PLA             ;RESTORE OLD COMMAND REG CONTENTS
CDD0:99 8A C0          STA   CMDREG,Y
CDD3:60                RTS
CDD4:          *
CDD4:A9 28     ICMD    LDA   #$28
CDD6:9D 38 06          STA   PWDBYTE,X ;SET PRINTER WIDTH TO 40
CDD9:A9 80             LDA   #$80
CDDB:1D 38 07          ORA   MISCFLG,X ;SET SCREEN ECHO
CDDE:D0 05             BNE   KCMD2     ;<ALWAYS>
CDE0:          *
CDE0:A9 FE     KCMD    LDA   #$FE      ;RESET THE LF GENERATE FLAG
CDE2:3D 38 07  KCMD1   AND   MISCFLG,X
CDE5:9D 38 07  KCMD2   STA   MISCFLG,X
CDE8:60                RTS
CDE9:          *
CDE9:C9 28     NCMD    CMP   #40       ;>=40?
CDEB:90 0E             BCC   ZCMDRTS   ;IF NOT, JUST EXIT
CDED:9D 38 06          STA   PWDBYTE,X ;SET NEW PRINTER WIDTH
CDF0:A9 3F             LDA   #$3F      ;DISABLE SCREEN, SET LISTING MODE
CDF2:D0 EE             BNE   KCMD1     ;<ALWAYS>
CDF4:          *
CDF4:1E 38 05  ZCMD    ASL   CMDBYTE,X ;DISABLE COMMAND RECOGNITION
CDF7:38                SEC
CDF8:7E 38 05          ROR   CMDBYTE,X
CDFB:60        ZCMDRTS RTS
CDFC:          *
CDFC:          *************************************
CDFC:          * VECTOR ACCORDING TO COMMAND STATE *
CDFC:          *************************************
CDFC:A8        CMDPROC TAY             ;A-REG=COMMAND STATE
CDFD:A5 27             LDA   CHARACTER
CDFF:29 7F             AND   #$7F
CE01:          *
CE01:C9 20             CMP   #$20      ;SKIP SPACES FOR ALL MODES
CE03:D0 09             BNE   CMDPROC2
CE05:C0 03             CPY   #$3       ;EXCEPT MODE 3
CE07:F0 01             BEQ   CMDPROC1
```

```
CE09:60              RTS
CE0A:A9 04     CMDPROC1 LDA #$4
CE0C:D0 6D           BNE  SETOSTATE ;<ALWAYS>
CE0E:          *
CE0E:C9 0D     CMDPROC2 CMP #$0D      ;CARRIAGE RETURN?
CE10:D0 12           BNE  CMDPROC4  ;
CE12:20 79 CE        JSR  ZEROSTATE ;ABORT FOR STATES 0-5, EXIT FOR 6,7
CE15:C0 07           CPY  #$07      ;IN STATE 7 WE VECTOR TO THE PROC
CE17:F0 01           BEQ  CMDPROC3  ;
CE19:60              RTS            ;OTHERWISE, JUST EXIT
CE1A:          *
CE1A:A9 CD     CMDPROC3 LDA #$CD      ;ALL PROCS MUST START IN PAGE $CD
CE1C:48              PHA
CE1D:BD 38 04        LDA  PARAMETER,X
CE20:48              PHA
CE21:A4 26           LDY  SLOT16    ;NEEDED BY BREAK CMD
CE23:60              RTS
CE24:          *
CE24:85 35     CMDPROC4 STA ZPTEMP
CE26:A9 CE           LDA  #$CE      ;ALL ROUTINES MUST START
CE28:48              PHA            ; IN PAGE $CE
CE29:B9 30 CE        LDA  STATETBL,Y
CE2C:48              PHA
CE2D:A5 35           LDA  ZPTEMP
CE2F:60              RTS            ;RTS TO COMMAND PROCEDURE
CE30:          *
CE30:          * NOW THE STATE ROUTINES
CE30:          *
CE30:          ***********************
CE30:          *  STATE BRANCH TABLE *
CE30:          ***********************
CE30:A7        STATETBL DFB >STATERR-1 ;BAD STATE
CE31:37              DFB  >CSTATE1-1 ;<CMD> SEEN
CE32:61              DFB  >CSTATE2-1 ;ACCUMULATE PARAMETER
CE33:89              DFB  >CDONE-1   ;SKIP UNTIL SPACE
CE34:8A              DFB  >CSTATE4-1 ;E/D SOMETHING
CE35:A7              DFB  >STATERR-1 ;ILLEGAL STATE
CE36:89              DFB  >CDONE-1   ;SKIP UNTIL CR
CE37:89              DFB  >CDONE-1   ;SKIP UNTIL CR THEN DO CMD
CE38:          *******************
CE38:          * COMMAND STATE 1 *
CE38:          *******************
CE38:DD 38 05  CSTATE1 CMP CMDBYTE,X ;IS IT <CMD>?
CE3B:D0 06           BNE  CSTATE1A
CE3D:DE B8 04        DEC  STATEFLG,X ;SET STATE BACK TO ZERO
CE40:4C 02 CC        JMP  ACIAOUT   ;OUTPUT <CMD> IF SO
CE43:          *
CE43:C9 30     CSTATE1A CMP #$30      ;>=0?
CE45:90 0D           BCC  CSTATE1B
CE47:C9 3A           CMP  #$3A      ;<=9?
CE49:B0 09           BCS  CSTATE1B
CE4B:29 0F           AND  #$0F      ;IT'S A NUMBER
CE4D:9D 38 04        STA  PARAMETER,X
CE50:A9 02           LDA  #2
CE52:D0 27           BNE  SETOSTATE ;<ALWAYS> SET MODE 2 AND RETURN
CE54:          *
CE54:C9 20     CSTATE1B CMP #$20      ;IS IT A CONTROL CHAR?
CE56:B0 06           BCS  CSTATE1C
```



```
CE58:9D 38 05        STA  CMDBYTE,X ;SET NEW COMMAND CHARACTER
CE5B:4C 79 CE        JMP  ZEROSTATE ;RESET STATE TO ZERO
CE5E:          *
CE5E:A0 00     CSTATE1C LDY #0        ;USE COMMAND TABLE
CE60:F0 4D           BEQ  CMDSEARCH ;<ALWAYS>
CE62:          *****************************************
CE62:          * COMMAND STATE 2: ACCUMULATE PARAMETER *
CE62:          *****************************************
CE62:49 30     CSTATE2 EOR #$30       ;CONVERT $30-$39 TO 0-9
CE64:C9 0A           CMP  #$A       ;0-9?
CE66:B0 0D           BCS  CSTATE2A
CE68:A0 0A           LDY  #$A       ;IT'S A NUMBER, SO ADD
CE6A:7D 38 04  ACCLOOP ADC PARAMETER,X ; IT TO 10*PARAMETER
CE6D:88              DEY
CE6E:D0 FA           BNE  ACCLOOP
CE70:9D 38 04        STA  PARAMETER,X
CE73:F0 15           BEQ  CDONE     ;<ALWAYS>
CE75:          *
CE75:A0 2E     CSTATE2A LDY #CMDTBL1-CMDTBL ;USE COMMAND TABLE
CE77:D0 36           BNE  CMDSEARCH ;<ALWAYS>
CE79:          *********************
CE79:          * SET COMMAND STATE *
CE79:          *********************
CE79:A9 00     ZEROSTATE LDA #0
CE7B:85 2A     SETOSTATE STA ZPTMP1
CE7D:AE F8 07        LDX  MSLOT
CE80:BD B8 04        LDA  STATEFLG,X
CE83:29 F8           AND  #$F8
CE85:05 2A           ORA  ZPTMP1
CE87:9D B8 04        STA  STATEFLG,X
CE8A:60        CDONE RTS
CE8B:          *************************
CE8B:          * COMMAND STATE 4 (E/D) *
CE8B:          *************************
CE8B:A8        CSTATE4 TAY              ;E/D -> Y-REG
CE8C:BD 38 04        LDA  PARAMETER,X
CE8F:C0 44           CPY  #$44      ;D(ISABLE)?
CE91:F0 09           BEQ  CSTATE4A
CE93:C0 45           CPY  #$45      ;E(NABLE)?
CE95:D0 11           BNE  STATERR   ;IF NOT, IGNORE THIS COMMAND
CE97:1D 38 07        ORA  MISCFLG,X ;SET FLAG
CE9A:D0 05           BNE  CSTATE4B  ;<ALWAYS>
CE9C:49 FF     CSTATE4A EOR #$FF      ;INVERT FOR DISABLE
CE9E:3D 38 07        AND  MISCFLG,X ;RESET FLAG
CEA1:9D 38 07  CSTATE4B STA MISCFLG,X
CEA4:          *********************
CEA4:          * ESCAPE TO STATE 6 *
CEA4:          *********************
CEA4:A9 06     SETSTATE6 LDA #6
CEA6:D0 D3           BNE  SETOSTATE ;<ALWAYS>
CEA8:A9 20     STATERR LDA #32        ;CODE FOR BAD COMMAND
CEAA:9D B8 05        STA  STSBYTE,X
CEAD:D0 F5           BNE  SETSTATE6 ;<ALWAYS>
CEAF:          *************************************
CEAF:          * TABLE DRIVEN COMMAND PROCESSOR *
CEAF:          *************************************
CEAF:B9 EB CC  CMDSEARCH LDA CMDTBL,Y ;GET CANDIDATE CHARACTER
CEB2:F0 F4           BEQ  STATERR   ;A ZERO MARKS THE END OF A SUBTABLE
```

```
CEB4:C5 35             CMP  ZPTEMP     ;MATCH?
CEB6:F0 05             BEQ  CMDMATCH
CEB8:C8                INY
CEB9:C8       CMDSEARCH1 INY           ;REENTRY FOR WRONG MODES
CEBA:C8                INY             ;ENTRY LENGTH = 3
CEBB:D0 F2             BNE  CMDSEARCH  ;<ALWAYS>
CEBD:         *
CEBD:C8       CMDMATCH INY
CEBE:B9 EB CC          LDA  CMDTBL,Y
CEC1:85 2A             STA  ZPTMP1
CEC3:29 20             AND  #$20       ;CHECK PASCAL ENABLE
CEC5:D0 07             BNE  CMDMATCH1  ;IT'S ON SO DONT CHECK P-BIT
CEC7:BD 38 07          LDA  MISCFLG,X  ;OFF SO MAKE SURE
CECA:29 10             AND  #$10       ;  THAT WE AREN'T IN PASCAL
CECC:D0 EB             BNE  CMDSEARCH1 ;BRANCH IF WE ARE
CECE:         *
CECE:BD 38 07 CMDMATCH1 LDA MISCFLG,X  ;GET CIC/PPC BIT
CED1:4A                LSR  A          ;SHIFT CIC/PPC MODE BIT TO CARRY
CED2:4A                LSR  A
CED3:24 2A             BIT  ZPTMP1     ;PPC->N CIC->V
CED5:B0 04             BCS  CMDMATCH2  ;BRANCH IF CIC MODE
CED7:10 E0             BPL  CMDSEARCH1 ;NOT OK FOR PPC
CED9:30 02             BMI  CMDEXEC    ;AND OK
CEDB:50 DC    CMDMATCH2 BVC CMDSEARCH1 ;NOT OK FOR CIC
CEDD:         *
CEDD:A5 2A    CMDEXEC  LDA  ZPTMP1     ;RETRIEVE TABLE MODE BYTE
CEDF:48                PHA
CEE0:29 07             AND  #$07
CEE2:20 7B CE          JSR  SETOSTATE  ;SET NEXT STATE
CEE5:C8                INY
CEE6:68                PLA
CEE7:29 10             AND  #$10       ;
CEE9:D0 07             BNE  CMDEXEC1   ;IF BIT 4 IS SET, VECTOR TO ROUTINE
CEEB:B9 EB CC          LDA  CMDTBL,Y
CEEE:9D 38 04          STA  PARAMETER,X
CEF1:60                RTS
CEF2:         *
CEF2:A9 CD    CMDEXEC1 LDA  #$CD       ;ROUTINES MUST BE IN PAGE $CD
CEF4:48                PHA
CEF5:B9 EB CC          LDA  CMDTBL,Y
CEF8:48                PHA
CEF9:A4 26             LDY  SLOT16
CEFB:BD 38 04          LDA  PARAMETER,X ;LOT OF ROUTINES NEED THIS
CEFE:60                RTS
CEFF:         *
CEFF:00                DFB  $00
```

SYMBOL TABLE SORTED BY SYMBOL

```
   3C A1L             CE6A ACCLOOP        CC02 ACIAOUT       ?CB9C ACK
 C9C8 ADJRTS          C9B5 ADJUST         CB59 ANRTS          CCB7 ASCREEN
 C8B8 BASICEXIT         28 BASL          ?C93D BATCHIN        C9EF BATCHIO
?C941 BATCHOUT        CD7D BAUDCMD        CD81 BAUDCMD1       CD88 BAUDCMD2
 C711 BENTRY          C8EF BINACIA1       C8EA BINACIA        C8E5 BINEND1
 C8D0 BINEND          C745 BINIT1        ?C700 BINIT          C8CB BINKBD
 C8BF BINPUT          C77C BOUTPUT1       C767 BOUTPUT        C78B BOUTPUT2
 CDC1 BREAKCMD        06B8 BUFBYTE        CE8A CDONE            24 CH
   27 CHARACTER       CA1E CHECKTERM      0638 CHNBYTE        C8B5 CICEXIT
 C9EE CIEND           C9D1 CKINPUT        C9E5 CKINPUT1       C9EB CKINPUT2
 CC3D CKKBDXIT        CC2C CKKBD          CC34 CKKBD1         0538 CMDBYTE
 CEF2 CMDEXEC1        CEDD CMDEXEC        CECE CMDMATCH1      CEBD CMDMATCH
 CEDB CMDMATCH2       CE0A CMDPROC1       CE0E CMDPROC2       CE1A CMDPROC3
 CE24 CMDPROC4        CDFC CMDPROC        C08A CMDREG         CEAF CMDSEARCH
 CEB9 CMDSEARCH1      CB1A CMDSEQCK       CD19 CMDTBL1        CCEB CMDTBL
 06B8 COLBYTE         C917 COMMA          FDED COUT           CD3B CRCMD
 CE43 CSTATE1A        CE54 CSTATE1B       CE5E CSTATE1C       CE38 CSTATE1
 CE75 CSTATE2A        CE62 CSTATE2        CE9C CSTATE4A       CEA1 CSTATE4B
 CE8B CSTATE4           37 CSWH             36 CSWL           C08B CTLREG
 CD8E CTLREGSET       C9C9 CTRLTST        CDA1 DATACMD1       CD9C DATACMD
 C9C0 DECRCOL         03B8 DELAYFLG       CD4B DELAYSET       C081 DIPSW1
 C082 DIPSW2          CBFF DLYTBL         CB2E ESCCHECK       CB90 ETX
 CD47 FFCMD           C968 FORCECR        C754 FROMIN         C751 FROMOUT
 C8B4 GETCHAR1        C8AA GETCHAR        CC72 GETCMD         CC3E GETKBD
 CC44 GETKBD1         CC91 GETKBDONE      CCDF GETXLATE       0438 HANDSHKE
 CDD4 ICMD            C705 IENTRY         0200 INBUFF         C805 INIT1
 C827 INIT1A          C835 INIT2          C83C INIT2A         C83F INIT2B
 C857 INIT3           C864 INIT4          C872 INIT5         ?C879 INITACIA
 C882 INITACIA1       C88F INITACIA2      CB16 INPUT2         CAFF INPUT
 FF58 IORTS           C010 KBDSTRB        C000 KBD            CC5C KBDESC
 CDE2 KCMD1           CDE5 KCMD2          CDE0 KCMD             39 KSWH
   38 KSWL            CCD3 LCMASK         CD41 LFCMD          CBEA LFGEN
 0738 MISCFLG         CA0C MOVIN          C9FD MOVOUT         07F8 MSLOT
 CDE9 NCMD            CB58 NOCMD          CB19 NOINPUT1       CB18 NOINPUT
 CCB6 NOOUT           C75C NORMIO         C954 NOTAB1         C951 NOTAB
 FCBA NXTA1           C707 OENTRY         CBD9 OUTDLY1        CBE2 OUTDLYLP
 CB68 OUTPUT1         CB6B OUTPUT2        CB63 OUTPUT         CBC1 OUTPUT3
 CBFE OUTPUTEND      ?CB76 P8AOUT1        CBA6 P8AOUT2        CBA9 P8AOUT3
 CBBB P8AOUT4         0438 PARAMETER      CD9B PARITYCMD      C800 PASCALINIT
?C89E PASCALREAD1     C89B PASCALREAD     C9AA PASCALWRITE    C8A3 PASEXIT
 C99B PENTRY          C78E PINIT         ?C84D PREAD0         C794 PREAD
 CC93 PROMPTBL        CC5E PROMPTLOOP     C7A8 PSTATIN        C79A PSTATUS
 C7AB PSTATUS2        0638 PWDBYTE        C9A6 PWDTBL         C797 PWRITE
 CDBD QCMD1           CDB9 QUITCMD        C088 RDREG          C089 RESET
 CDAD RESETCMD        CC11 RESTORE        CC29 RESTOREND      C7EE RESTORHOOK
 CCDB REVMASK           4F RNDH             4E RNDL           CFFF ROMSOFF
 CD57 ROTATE          C7B2 SAVEHOOK       CC9E SCREENOUT      CCA3 SCREENOUT1
 C7B0 SENDCD          C998 SEREND2        C97A SEREND         C8FC SEROUT
 C996 SETCH           FE89 SETKBD         CE7B SETOSTATE      FE93 SETSCR
 CEA4 SETSTATE6         26 SLOT16         CAE9 SRIN1          CAEB SRIN2
 CAD2 SRIN            CAF4 SRIN3          CAF5 SROUT          CD7C SSLOTCMD1
 CD61 SSLOTCMD        0100 STACK          04B8 STATEFLG       CEA8 STATERR
 CE30 STATETBL        C089 STREG          05B8 STSBYTE        C934 TAB1
 C948 TAB2            C921 TABCHECK      ?C088 TDREG          CA55 TERMACIAIN
 CAAD TERMCAP1        CA9B TERMCAP        CDA7 TERMCMD       ?CA4C TERMEXIT
 CA82 TERMINC1        CA81 TERMINC        CA66 TERMKBDIN      CA87 TERMLETTER
 CAB1 TERMLOCK       ?CA23 TERMMODE       CA2B TERMNEXT       CA31 TERMNEXT1
 CA41 TERMNEXT2       CA47 TERMNEXT3     ?CA7D TERMNORM       CA54 TERMRTS
 CA93 TERMSEND        CA95 TERMSEND1      CCC6 TESTLETTER     CCC3 TOSCREEN
 CD35 TRANCMD         CAB9 TRANSLATE      CCD7 UCMASK         FDF6 VIDOUT
 CAC4 WAITMS          CAC6 WAITMS1        CB3C XOFFCK         CB5A XONWAIT
 CDFB ZCMDRTS         CDF4 ZCMD           CE79 ZEROSTATE        35 ZPTEMP
   2A ZPTMP1            2B ZPTMP2
```

SYMBOL TABLE SORTED BY ADDRESS

```
   24 CH                26 SLOT16           27 CHARACTER        28 BASL
   2A ZPTMP1            2B ZPTMP2           35 ZPTEMP           36 CSWL
   37 CSWH              38 KSWL             39 KSWH             3C A1L
   4E RNDL              4F RNDH           0100 STACK          0200 INBUFF
 03B8 DELAYFLG        0438 HANDSHKE       0438 PARAMETER      04B8 STATEFLG
```

```
 0538 CMDBYTE      05B8 STSBYTE      0638 PWDBYTE      0638 CHNBYTE
 06B8 COLBYTE      06B8 BUFBYTE      0738 MISCFLG      07F8 MSLOT
 C000 KBD          C010 KBDSTRB      C081 DIPSW1       C082 DIPSW2
?C088 TDREG        C088 RDREG        C089 STREG        C089 RESET
 C08A CMDREG       C08B CTLREG      ?C700 BINIT        C705 IENTRY
 C707 OENTRY       C711 BENTRY       C745 BINIT1       C751 FROMOUT
 C754 FROMIN       C75C NORMIO       C767 BOUTPUT      C77C BOUTPUT1
 C78B BOUTPUT2     C78E PINIT        C794 PREAD        C797 PWRITE
 C79A PSTATUS      C7A8 PSTATIN      C7AB PSTATUS2     C7B0 SENDCD
 C7B2 SAVEHOOK     C7EE RESTORHOOK   C800 PASCALINIT   C805 INIT1
 C827 INIT1A       C835 INIT2        C83C INIT2A       C83F INIT2B
?C84D PREAD0       C857 INIT3        C864 INIT4        C872 INIT5
?C879 INITACIA     C882 INITACIA1    C88F INITACIA2    C89B PASCALREAD
?C89E PASCALREAD1  C8A3 PASEXIT      C8AA GETCHAR      C8B4 GETCHAR1
 C8B5 CICEXIT      C8B8 BASICEXIT    C8BF BINPUT       C8CB BINKBD
 C8D0 BINEND       C8E5 BINEND1      C8EA BINACIA      C8EF BINACIA1
 C8FC SEROUT       C917 COMMA        C921 TABCHECK     C934 TAB1
?C93D BATCHIN     ?C941 BATCHOUT     C948 TAB2         C951 NOTAB
 C954 NOTAB1       C968 FORCECR      C97A SEREND       C996 SETCH
 C998 SEREND2      C99B PENTRY       C9A6 PWDTBL       C9AA PASCALWRITE
 C9B5 ADJUST       C9C0 DECRCOL      C9C8 ADJRTS       C9C9 CTRLTST
 C9D1 CKINPUT      C9E5 CKINPUT1     C9EB CKINPUT2     C9EE CIEND
 C9EF BATCHIO      C9FD MOVOUT       CA0C MOVIN        CA1E CHECKTERM
?CA23 TERMMODE     CA2B TERMNEXT     CA31 TERMNEXT1    CA41 TERMNEXT2
 CA47 TERMNEXT3   ?CA4C TERMEXIT     CA54 TERMRTS      CA55 TERMACIAIN
 CA66 TERMKBDIN   ?CA7D TERMNORM     CA81 TERMINC      CA82 TERMINC1
 CA87 TERMLETTER   CA93 TERMSEND     CA95 TERMSEND1    CA9B TERMCAP
 CAAD TERMCAP1     CAB1 TERMLOCK     CAB9 TRANSLATE    CAC4 WAITMS
 CAC6 WAITMS1      CAD2 SRIN         CAE9 SRIN1        CAEB SRIN2
 CAF4 SRIN3        CAF5 SROUT        CAFF INPUT        CB16 INPUT2
 CB18 NOINPUT      CB19 NOINPUT1     CB1A CMDSEQCK     CB2E ESCCHECK
 CB3C XOFFCK       CB58 NOCMD        CB59 ANRTS        CB5A XONWAIT
 CB63 OUTPUT       CB68 OUTPUT1      CB6B OUTPUT2     ?CB76 P8AOUT1
 CB90 ETX         ?CB9C ACK          CBA6 P8AOUT2      CBA9 P8AOUT3
 CBBB P8AOUT4      CBC1 OUTPUT3      CBD9 OUTDLY1      CBE2 OUTDLYLP
 CBEA LFGEN        CBFE OUTPUTEND    CBFF DLYTBL       CC02 ACIAOUT
 CC11 RESTORE      CC29 RESTOREND    CC2C CKKBD        CC34 CKKBD1
 CC3D CKKBDXIT     CC3E GETKBD       CC44 GETKBD1      CC5C KBDESC
 CC5E PROMPTLOOP   CC72 GETCMD       CC91 GETKBDONE    CC93 PROMPTBL
 CC9E SCREENOUT    CCA3 SCREENOUT1   CCB6 NOOUT        CCB7 ASCREEN
 CCC3 TOSCREEN     CCC6 TESTLETTER   CCD3 LCMASK       CCD7 UCMASK
 CCDB REVMASK      CCDF GETXLATE     CCEB CMDTBL       CD19 CMDTBL1
 CD35 TRANCMD      CD3B CRCMD        CD41 LFCMD        CD47 FFCMD
 CD4B DELAYSET     CD57 ROTATE       CD61 SSLOTCMD     CD7C SSLOTCMD1
 CD7D BAUDCMD      CD81 BAUDCMD1     CD88 BAUDCMD2     CD8E CTLREGSET
 CD9B PARITYCMD    CD9C DATACMD      CDA1 DATACMD1     CDA7 TERMCMD
 CDAD RESETCMD     CDB9 QUITCMD      CDBD QCMD1        CDC1 BREAKCMD
 CDD4 ICMD         CDE0 KCMD         CDE2 KCMD1        CDE5 KCMD2
 CDE9 NCMD         CDF4 ZCMD         CDFB ZCMDRTS      CDFC CMDPROC
 CE0A CMDPROC1     CE0E CMDPROC2     CE1A CMDPROC3     CE24 CMDPROC4
 CE30 STATETBL     CE38 CSTATE1      CE43 CSTATE1A     CE54 CSTATE1B
 CE5E CSTATE1C     CE62 CSTATE2      CE6A ACCLOOP      CE75 CSTATE2A
 CE79 ZEROSTATE    CE7B SETOSTATE    CE8A CDONE        CE8B CSTATE4
 CE9C CSTATE4A     CEA1 CSTATE4B     CEA4 SETSTATE6    CEA8 STATERR
 CEAF CMDSEARCH    CEB9 CMDSEARCH1   CEBD CMDMATCH     CECE CMDMATCH1
 CEDB CMDMATCH2    CEDD CMDEXEC      CEF2 CMDEXEC1     CFFF ROMSOFF
 FCBA NXTA1        FDED COUT         FDF6 VIDOUT       FE89 SETKBD
 FE93 SETSCR       FF58 IORTS
```


APPENDIX B

APPLE INTERFACE CARD EMULATION

The SSC emulates both the P8 and the P8A versions of the Apple II Serial Interface Card (SIC), although the SSC is not completely POKE-compatible with either. In addition, the SSC supports several Apple II Communications Card and Parallel Card software commands.

OLD SERIAL INTERFACE CARD EMULATION

The SSC replaces the P8 and P8A versions of the Apple II Serial Interface Card (SIC) and it has two switch-selectable modes to emulate them, as explained below. However, because of firmware space limitations, the SSC does not support all functions of the older interface cards, and various POKE locations are different. This section explains these functional differences.

It is best to use Printer Mode rather than one of the emulation modes, except under these circumstances:

- if you have extensive existent applications that use PEEKs and POKEs to modify SIC operating characteristics
- if you need SIC P8A mode's ETX/ACK (or other-character/ACK) handshaking capabilities

What the SSC does NOT support that the old SIC does:

- P8 SIC block moves
- baud rates other than the 15 listed in the various baud rate tables in this manual (ACIA hardware generates only those 15)
- data formats other than 5 - 8 data bits and 1, 1-1/2 or 2 stop bits (ACIA characteristic; other formats rarely used anyway)
- <ESC>U and <ESC>L commands for upper and lowercase (but SSC's Translate command offers more options; POKEs also available)
- current-loop operation

To run the SSC in emulation of the old Apple II Serial Interface Card (SIC), prepare and install the SSC the same way as for Printer Mode (Chapters 1 and 2), with the following exceptions:

- Set mode switches SW1-5 ON and SW1-6 OFF to emulate the old SIC with a P8 ROM.
- Set mode switches SW1-5 OFF and SW1-6 OFF to emulate the old SIC with a P8A ROM.
- Install the SSC in whatever slot the old SIC was installed in for the application involved.
- Follow the instructions given in the next sections if the application program did PEEKs and POKEs.

P8 EMULATION POKES

Changing SIC parameters was done either by setting the seven switches located on the card, or by POKEing the SIC slot RAM locations where this configuration data was stored. BASIC programs that talked through the old SIC may be used with the new SSC; however, if the program POKEs at these slot RAM locations, those POKEs must be changed to be compatible with the SSC's use of the RAM. The P8 and P8A ROMs differ slightly in their use of these RAM locations. Tables B-1 and B-2 show the transformation for P8 mode; additional differences for P8A mode are noted in the following section. Other POKE possibilities are described in Appendix A.

In the tables, the letter s stands for the slot number (1-7) in which the SSC is installed; the other letters are used as variables whose values are noted in the table (sometimes further down).

There is no claim that making these changes is simple. In fact, whenever possible it is best to use Printer Mode and its software commands to change SSC operating variables.

Here is an example of how to use the tables: let's say that the SSC is in slot #3. You want: a baud rate of 11Ø; data format of 5 data bits and 2 stop bits, even parity; line width of 4Ø with video on, no automatic <LF> after <CR>; no translation of lowercase to uppercase; and no 1/4-second delay after <CR>. The PEEKs and POKEs:

```
POKE 49339, 243    (49291 + 3*16; 3 + 24Ø)
POKE 49338, 1Ø7    (4929Ø + 3*16; p = 1Ø7)
POKE 2Ø43, 132     (plug in magic number)
POKE 1147, 64      (plug in magic number)
```

The same thing in Printer Mode with appropriate switch settings is:

```
SW1-1 to SW1-7: ON   ON  OFF OFF OFF ON  ON
SW2-1 to SW2-7: --   OFF ON  ON  OFF OFF OFF
```


Then to set 5 data and 2 stop bits, use <CTRL-I>7D<RETURN>; for even parity, use <CTRL-I>3P<RETURN>; to leave lowercase alone, use <CTRL-I>1T<RETURN>. You can use commands to change baud rate, etc.

Selection	SSC switches and settings	PEEKs and POKES to use for: P8 Serial Card	PEEKs and POKES to use for: Super Serial Card
P8 Mode:	SW1-5 ON, SW1-6 OFF		
P8A Mode:	SW1-5 OFF, SW1-6 OFF		
Baud Rate:	SW1-1 to SW1-4 same as Printer Mode	POKE 1144+s,r r =	POKE 49291+s*16,r r = b + d; b =
5Ø		(not available)	1 dec/$Ø1 hex
75		Ø dec/$ØØ hex	2 dec/$Ø2 hex
11Ø		176 dec/$BØ hex	3 dec/$Ø3 hex
135		144 dec/$9Ø hex	4 dec/$Ø4 hex
15Ø		128 dec/$8Ø hex	5 dec/$Ø5 hex
3ØØ		64 dec/$4Ø hex	6 dec/$Ø6 hex
6ØØ		32 dec/$2Ø hex	7 dec/$Ø7 hex
12ØØ		16 dec/$1Ø hex	8 dec/$Ø8 hex
18ØØ		11 dec/$ØB hex	9 dec/$Ø9 hex
24ØØ		8 dec/$Ø8 hex	1Ø dec/$ØA hex
36ØØ		5 dec/$Ø5 hex	11 dec/$ØB hex
48ØØ		4 dec/$Ø4 hex	12 dec/$ØC hex
72ØØ		(not available)	13 dec/$ØD hex
96ØØ		2 dec/$Ø2 hex	14 dec/$ØE hex
192ØØ		1 dec/$Ø1 hex	15 dec/$ØF hex
Data Format:		POKE 1912+s,r POKE 1272+s,t	(to get r above, add d to b) d =
8 data,1 stop	SW2-1 ON	r = 9; t = 1*	16 dec/$1Ø hex
7 data,1 stop		r = 8; t = 1*	48 dec/#3Ø hex
6 data,1 stop		r = 7; t = 1*	8Ø dec/$5Ø hex
5 data,1 stop		r = 6; t = 1*	112 dec/$7Ø hex
8 data,2 stop	SW2-1 OFF	r = 9; t = 2*	144 dec/$9Ø hex
7 data,2 stop		r = 8; t = 2*	176 dec/$BØ hex
6 data,2 stop		r = 7; t = 2*	2Ø8 dec/$DØ hex
5 data,2 stop		r = 6; t = 2*	24Ø dec/$FØ hex
		* add 1 if p = 1 or Ø	
Parity:		POKE 14ØØ+s,p	POKE 4929Ø+s*16,p
none		p = 2	p = 11 ($ØB hex)
odd		p = 1	p = 43 ($2B hex)
even		p = Ø	p = 1Ø7 ($6B hex)
MARK		(not available)	(not available)
SPACE		(not available)	(not available)

Table B-1. SIC Switch Settings, PEEKs and POKEs, Part I

Selection	SSC switches and settings	PEEKs and POKES to use for: P8 Serial Card	PEEKs and POKES to use for: Super Serial Card
Line Width:	SW2-3 & SW2-4, same as Printer Mode	POKE 1784+s,r r=1 to 255; for no <CR>,r=0	POKE 1784+s,r r=40 to 255; for no <CR>, PEEK 1400+s, POKE 1400+s, (old value + 128)
Video/ Generate <LF>/ Translate/ <CR> Delay:	SW2-3 & SW2-4 SW2-5 (no switch) SW2-2 (all switches same as in Printer Mode)	V = Video on? G = Gen. <LF>? T = LC to UC? D = Dly 1/4 s? POKE 2040+s,r r= dec hex V G T D 4 $04 Y N Y Y 5 $05 Y Y Y Y 36 $24 Y N N Y 37 $25 Y Y N Y 68 $44 Y N Y N 69 $45 Y Y Y N 100 $64 Y N N N 101 $65 Y Y N N 132 $84 N N Y Y 133 $85 N Y Y Y 164 $A4 N N N Y 165 $A5 N Y N Y 196 $C4 N N Y N 197 $C5 N Y Y N 228 $E4 N N N N 229 $E5 N Y N N	V = Video on? G = Gen. <LF>? POKE 2040+s,r r= dec hex V G 4 $04 N N 5 $05 N Y 132 $84 Y N 133 $85 Y Y T = LC to UC? D = Dly 1/4 s? POKE 1144+s,r r = dec hex T D 0 $00 Y N 16 $10 Y Y 64 $40 N N 80 $50 N Y

Table B-2. SIC Switch Settings, PEEKs and POKEs, Part II

P8A EMULATION POKES

The P8A ROM differs from the P8 ROM in several ways:

1) The <CR> delay switch now determines whether an ETX/ACK handshake is performed after each <CR> that is transmitted. The corresponding RAM bit was not the same as the P8 <CR> delay bit, but was kept in bit 2 of location 1400+s. For SSC emulation, the control is the same as the <CR> delay bit as noted above (in location 1144+s).

2) The number of stop bits was always 2; for SSC P8A mode this is configured via switch SW2-1 and can also be set via software by POKEing location 4929 as noted above.

3) The printer width information was kept in the same location that the P8 ROM kept the number of stop bits; the P8 printer width byte was zeroed to avoid automatic generation of carriage returns. The SSC P8A emulation code keeps the printer width information in the same place as for P8 emulation and uses the high-order bit at location 1400+s to control automatic generation of carriage returns.

4) Lowercase input is enabled by default for the P8A ROM; in P8A emulation, however, it is enabled by the POKE shown in Table B-2.

5) In contrast to the P8 ROM, the P8A ROM and the SSC do not support batch moves.

6) The enquire character for the SIC P8A ROM was ETX (ASCII 3); for SSC P8A mode, this can be changed to another control character by a POKE to location 1400+s. For example, to change the enquire character to ENQ (ASCII 5), which is used by many RS-232 devices, use this POKE: POKE 1400+s,5. Note that this also disables the automatic generation of carriage returns. Actually, any character between 0 and 31 can be used, although only 3 and 5 are used much.

OTHER EMULATION MODE DIFFERENCES

If your old programs, written to control one of the old Serial Interface Card ROMs, still don't work after you've followed all this handy advice, then read on.

The SSC always monitors the RS-232-C handshake lines to determine whether or not the device is ready to accept data. If your device fails to assert one of these lines, the SSC will wait patiently forever.

When the arrow on the jumper block is pointing toward TERMINAL, your device sees DCD and DSR asserted as soon as the SSC is initialized, and the SSC sees CTS whenever the device sends RTS. If the device does not assert both RTS and DTR, the SSC will assume it is not ready to receive data. This can be used as a hardware handshake to prevent buffer overflow at the device (e.g., when your printer runs out of paper it can stop asserting one of these lines and the SSC will wait while you put in more paper). If you do not connect these lines, the SSC will always treat them as if they were asserted.

The Serial Interface Card tied RTS to CTS, and DTR to DCD and DSR; if your RS-232 device depended upon this, you may want to make a special connector which does this.

Your device may have depended upon the half-duplex nature of the SIC. The ACIA on the SSC is able to send and receive at the same time and is always configured to do so.

The SIC was initialized each time it was called at location $Cs00 (for example, by a PR#s or IN#s). The SSC is only reintialized after the ACIA has been reset (either by resetting the Apple or by exiting from Printer or Communication Mode via a Reset command).

OLD COMMUNICATIONS CARD COMMANDS

The SSC supports all the functions supported by the old Apple II Communications Interface Card (CIC), although the two ACIAs' registers are not the same on a bit-by-bit level. The SSC also supports the CIC commands: <CTRL-T>, <CTRL-R>, and <CTRL-S>.

SWITCH TO TERMINAL MODE—<CTRL-T>

In Communication Mode, the SSC is initialized to recognize the remote-control command <CTRL-T> arriving in the stream of incoming data. This character causes the SSC to enter Terminal Mode (the same as the T(erminal command (Chapter 3). You can disable <CTRL-T> recognition by issuing an X(OFF D(isable command.

BYPASS TERMINAL MODE—<CTRL-R>

When the SSC is in Terminal Mode and X(OFF E(nable (the default in this mode) is in effect, the SSC recognizes the remote control command <CTRL-R> arriving in the input data stream, and responds by bypassing (exiting from) Terminal Mode. This is the same as the Q(uit Terminal Mode command (Chapter 3).

XOFF—<CTRL-S>

The SSC interprets <CTRL-S> as the ASCII XOFF character. When it receives <CTRL-S> from a remote device, it stops transmitting data until it receives an XON character from that device.

PARALLEL CARD COMMANDS

The SSC is not hardware compatible with the Apple II Parallel Cards. However, for the sake of compatibility with software written for parallel interface applications, the SSC supports the following commands. You do not need to follow these commands with <RETURN>.

LINE WIDTH n AND VIDEO OFF—<CTRL-I><n>N

This command turns off the Apple II video screen and generates a <CR> after n characters (if automatic <CR> generation is enabled via the C command (Chapter 2); n can be any value from 4Ø through 255.

LINE WIDTH 40 AND VIDEO ON—<CTRL-I>I

This command turns on the Apple II video screen and sets the line width to 4Ø.

DISABLE AUTOMATIC LINEFEED—<CTRL-I>K

This command has the same effect as L(inefeed D(isable (Chapter 2): it turns off automatic generation of <LF> after <CR>.

APPENDIX C

SPECIFICATIONS AND SCHEMATICS

This appendix contains the SSC specifications, connector pin assignments, jumper block wiring, and a schematic diagram. Use the schematic diagram with the Theory of Operation section in Chapter 4.

SSC SPECIFICATIONS

PHYSICAL CHARACTERISTICS

Dimensions	2-3/4" x 7" (68.8 mm x 177.8 mm)
Weight	3 oz. (9Ø gm), approximately
Cables required	internal cable from 1Ø-pin header on SSC to DB-25 connector on case of Apple II (supplied); shielded RS-232-C cable to external device (not supplied)
Controls	2 blocks of 7 switches each, set by user before installation
Special Tools	none required

ENVIRONMENT

Operating temperature	4Ø° F to 95° F (5° C to 35° C)
Storage temperature	-4Ø° F to 122° F (-4Ø° C to 5Ø° C)
Operating relative humidity	5% to 95% (noncondensing)
Storage relative humidity	5% to 95% (noncondensing)

SPECIAL CIRCUITS

SY6551	Asynchronous Communications Interface Adapter
2316	Read Only Memory (2,Ø48 by 8 bits) with SSC firmware
	The SSC has the usual power supply bypassing capacitors

APPLE II SLOT LOCATION

BASIC programs	any slot except slot #Ø
APPLESOFT programs	any slot except slot #Ø
PASCAL programs	slot #1 for use with printer, etc.
	slot #2 for use with modem
	slot #3 for use with terminal

SOFTWARE COMPATIBILITY

The SSC is compatible with the following languages and operating systems:

Integer BASIC	DOS 3.2	Pascal 1.Ø	65Ø2 Assembler
Applesoft BASIC	DOS 3.3	Pascal 1.1	

Under BASIC, input sent to the SSC at high baud rates may be lost, since the SSC can only buffer two characters at a time and BASIC may not be fast enough to read characters before they are overlaid.

In any software environment, characters may be lost when sent to the video screen in scrolling mode at greater than 3ØØ baud. There are at least three solutions to this problem: lower the baud rate to 3ØØ baud; reduce the scrolling window size (using 2 fewer lines already makes 12ØØ baud possible), or use an 8Ø-column card with automatic hardware scrolling.

CONNECTOR PIN ASSIGNMENTS

Table C-1 lists the signals assigned to the connector pins on the 1Ø-pin header at location 7B on the SSC, and the corresponding pins on the DB-25 connector that you attach to the back of the Apple II case.

1Ø-pin Header	DB-25 Connector	Signal name
1	1	Frame Ground
2	2	Transmit Data (TXD)
3	3	Receive Data (RXD)
4	4	Request To Send (RTS)
5	5	Clear To Send (CTS)
6	6	Data Set Ready (DSR)
7	19	Secondary Clear To Send (SCTS)
8	7	Signal Ground
9	2Ø	Data Terminal Ready (DTR)
1Ø	8	Data Carrier Detect (DCD)

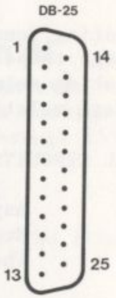


Table C-1. Connector Pin Assignments

JUMPER BLOCK WIRING

Table C-2 lists the signals that the jumper block connects to the SSC when the arrow points toward the word MODEM and when it points toward the word TERMINAL. In the latter case, the jumper block acts as a modem eliminator.

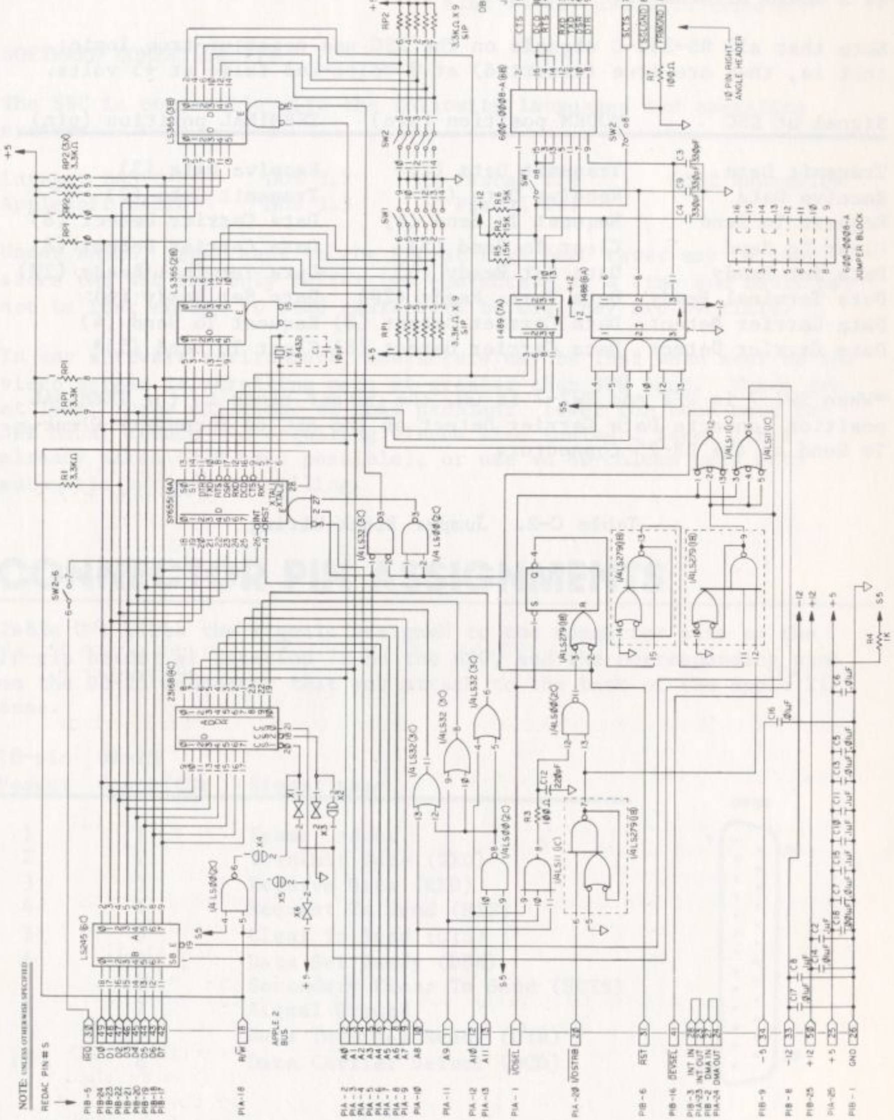
Note that all RS-232-C signals on the SSC use negative-true logic; that is, they are true (asserted) at Ø volts and false at +5 volts.

Signal at SSC	MODEM position (pin)	TERMINAL position (pin)
Transmit Data	Transmit Data (2)	Receive Data (3)
Receive Data	Receive Data (3)	Transmit Data (2)
Request To Send	Request To Send (4)	Data Carrier Detect (8)
Clear To Send	Clear To Send (5)	Data Carrier Detect (8)
Data Set Ready	Data Set Ready (6)	Data Terminal Ready (2Ø)
Data Terminal Ready	Data Term. Ready (2Ø)	Data Set Ready (6)
Data Carrier Detect	Data Carrier Detect (8)	Request To Send (4)
Data Carrier Detect	Data Carrier Detect (8)	Clear To Send (5)*

*When SW1-7 is OFF and SW2-7 is ON, the jumper block in the TERMINAL position connects Data Carrier Detect on the SSC to Secondary Clear To Send on the DB-25 connector.

Table C-2. Jumper Block Wiring

SCHEMATIC DIAGRAM



APPENDIX D

ASCII CODE TABLE

The table below shows the entire ASCII character set, and how to generate each character. Not all characters are available directly from the Apple II keyboard. However, in Terminal Mode (Chapter 3) you can generate all of the lowercase and special ASCII characters not accessible directly from the Apple II keyboard.

Here is how to interpret this table:

- The BINARY column has the 7-bit code for each ASCII character.
- The LOW DEC column gives the decimal equivalent of the 7-bit binary value. This value is the same if the binary code has 8 bits and the high-order bit is Ø (SPACE parity; Pascal).
- The LOW HEX column gives the corresponding hexadecimal value.
- The HI DEC column gives the decimal equivalent of the 7-bit binary value if a high-order bit equal to 1 is appended to it (MARK parity; BASIC); for example, 11ØØ1ØØØ for the letter H.
- The HI HEX column gives the corresponding hexadecimal value.
- The ASCII CHAR column gives the ASCII character name.
- The INTERPRETATION column spells out the meaning of special symbols and abbreviations where necessary.
- The WHAT TO TYPE column indicates what keystrokes generate the ASCII character from the NORMAL (unaided) Apple II keyboard, and from the TERMINAL Mode (firmware assisted) keyboard. Characters not accessible are labeled "n/a." The numbers between columns refer to footnotes.
- Angle brackets enclose the names of single keys (like <ESC> for the ESC key), or enclose keystrokes involving more than one key (like <CTRL-SHIFT-M>, which means "hold down CTRL and SHIFT while pressing M.") But <ESC>9 means "type ESC, THEN type 9" because the 9 is outside the angle brackets.

To put the SSC in Terminal Mode, set SW1-5 and SW1-6 both ON; then use the T command or the remote-control <CTRL-T> command. When the SSC first enters Terminal Mode, the keyboard is locked in uppercase. Press <ESC> once for lowercase. This also prepares the SSC for the special <ESC>-plus-number keystrokes. Press <ESC> twice in a row to lock the keyboard in uppercase again.

7-BIT BINARY	LOW DEC	LOW HEX	HI DEC	HI HEX	ASCII CHAR	INTERPRETATION	WHAT TO TYPE NORMAL	TERMINAL
ØØØØØØØ	Ø	ØØ	128	8Ø	NUL	Blank (null)	<CTRL-@>	
ØØØØØØ1	1	Ø1	129	81	SOH	Start of Header	<CTRL-A> [1]	
ØØØØØ1Ø	2	Ø2	13Ø	82	STX	Start of Text	<CTRL-B>	
ØØØØØ11	3	Ø3	131	83	ETX	End of Text	<CTRL-C> [2]	
ØØØØ1ØØ	4	Ø4	132	84	EOT	End of Transm.	<CTRL-D>	
ØØØØ1Ø1	5	Ø5	133	85	ENQ	Enquiry	<CTRL-E> [3]	
ØØØØ11Ø	6	Ø6	134	86	ACK	Acknowledge	<CTRL-F> [4]	
ØØØØ111	7	Ø7	135	87	BEL	Bell	<CTRL-G>	
ØØØ1ØØØ	8	Ø8	136	88	BS	Backspace	<CTRL-H> [5]	
ØØØ1ØØ1	9	Ø9	137	89	HT	Horizontal Tab	<CTRL-I> [6]	
ØØØ1Ø1Ø	1Ø	ØA	138	8A	LF	Linefeed	<CTRL-J>	
ØØØ1Ø11	11	ØB	139	8B	VT	Vertical Tab	<CTRL-K>	
ØØØ11ØØ	12	ØC	14Ø	8C	FF	Form Feed	<CTRL-L>	
ØØØ11Ø1	13	ØD	141	8D	CR	Carriage Return	<CTRL-M> [7]	
ØØØ111Ø	14	ØE	142	8E	SO	Shift Out	<CTRL-N>	
ØØØ1111	15	ØF	143	8F	SI	Shift In	<CTRL-O>	
ØØ1ØØØØ	16	1Ø	144	9Ø	DLE	Data Link Escape	<CTRL-P>	
ØØ1ØØØ1	17	11	145	91	DC1	Device Control 1	<CTRL-Q> [8]	
ØØ1ØØ1Ø	18	12	146	92	DC2	Device Control 2	<CTRL-R> [9]	
ØØ1ØØ11	19	13	147	93	DC3	Device Control 3	<CTRL-S> [1Ø]	
ØØ1Ø1ØØ	2Ø	14	148	94	DC4	Device Control 4	<CTRL-T> [11]	
ØØ1Ø1Ø1	21	15	149	95	NAK	Neg. Acknowledge	<CTRL-U> [12]	
ØØ1Ø11Ø	22	16	15Ø	96	SYN	Synchronization	<CTRL-V>	
ØØ1Ø111	23	17	151	97	ETB	End of Text Blk.	<CTRL-W>	
ØØ11ØØØ	24	18	152	98	CAN	Cancel	<CTRL-X>	
ØØ11ØØ1	25	19	153	99	EM	End of Medium	<CTRL-Y>	
ØØ11Ø1Ø	26	1A	154	9A	SUB	Substitute	<CTRL-Z>	
ØØ11Ø11	27	1B	155	9B	ESC	Escape	<ESC> [13]	<ESC>Ø

1. Normal command character in Communication Mode.
2. Used in ETX/ACK protocol (SIC P8A Emulation Mode).
3. Used in ENQ/ACK protocol (SIC P8A Emulation Mode).
4. Used in ETX/ACK or ENQ/ACK protocol (SIC P8A Emulation Mode).
5. Or use ← key.
6. Normal Command character in Printer Mode.
7. Or use <RETURN> key.
8. XON in XON/XOFF protocol (usually in Communication Mode).
9. Remote-control command to Exit from Terminal Mode.
1Ø. XOFF in XON/XOFF protocol (usually in Communication Mode).
11. Remote-control command to Enter Terminal Mode.
12. Or use → key.
13. Use the ESC key to generate the Escape character with the normal Apple II keyboard. In Terminal Mode, use <ESC>Ø.

7-BIT BINARY	LOW DEC	LOW HEX	HI DEC	HI HEX	ASCII CHAR	INTERPRETATION	WHAT TO TYPE NORMAL	TERMINAL
ØØ111ØØ	28	1C	156	9C	FS	File Separator	n/a	<ESC>1
ØØ111Ø1	29	1D	157	9D	GS	Group Separator	<CTRL-SHIFT-M>	
ØØ1111Ø	3Ø	1E	158	9E	RS	Record Separator	<CTRL-SHIFT-N>	
ØØ11111	31	1F	159	9F	US	Unit Separator	n/a	<ESC>2
Ø1ØØØØØ	32	2Ø	16Ø	AØ	SP	Space	spacebar	
Ø1ØØØØ1	33	21	161	A1	!		!	
Ø1ØØØ1Ø	34	22	162	A2	"		"	
Ø1ØØØ11	35	23	163	A3	#		#	
Ø1ØØ1ØØ	36	24	164	A4	$		$	
Ø1ØØ1Ø1	37	25	165	A5	%		%	
Ø1ØØ11Ø	38	26	166	A6	&		&	
Ø1ØØ111	39	27	167	A7	'	Closing Quote	'	
Ø1Ø1ØØØ	4Ø	28	168	A8	(		(	
Ø1Ø1ØØ1	41	29	169	A9	)		)	
Ø1Ø1Ø1Ø	42	2A	17Ø	AA	*		*	
Ø1Ø1Ø11	43	2B	171	AB	+		+	
Ø1Ø11ØØ	44	2C	172	AC	,	Comma	,	
Ø1Ø11Ø1	45	2D	173	AD	-	Hyphen	-	
Ø1Ø111Ø	46	2E	174	AE	.	Period	.	
Ø1Ø1111	47	2F	175	AF	/		/	
Ø11ØØØØ	48	3Ø	176	BØ	Ø		Ø	
Ø11ØØØ1	49	31	177	B1	1		1	
Ø11ØØ1Ø	5Ø	32	178	B2	2		2	
Ø11ØØ11	51	33	179	B3	3		3	
Ø11Ø1ØØ	52	34	18Ø	B4	4		4	
Ø11Ø1Ø1	53	35	181	B5	5		5	
Ø11Ø11Ø	54	36	182	B6	6		6	
Ø11Ø111	55	37	183	B7	7		7	
Ø111ØØØ	56	38	184	B8	8		8	
Ø111ØØ1	57	39	185	B9	9		9	
Ø111Ø1Ø	58	3A	186	BA	:		:	
Ø111Ø11	59	3B	187	BB	;		;	
Ø1111ØØ	6Ø	3C	188	BC	<		<	
Ø1111Ø1	61	3D	189	BD	=		=	
Ø11111Ø	62	3E	19Ø	BE	>		>	
Ø111111	63	3F	191	BF	?		?	
1ØØØØØØ	64	4Ø	192	CØ	@		@	
1ØØØØØ1	65	41	193	C1	A		A	
1ØØØØ1Ø	66	42	194	C2	B		B	
1ØØØØ11	67	43	195	C3	C		C	
1ØØØ1ØØ	68	44	196	C4	D		D	
1ØØØ1Ø1	69	45	197	C5	E		E	
1ØØØ11Ø	7Ø	46	198	C6	F		F	
1ØØØ111	71	47	199	C7	G		G	
1ØØ1ØØØ	72	48	2ØØ	C8	H		H	
1ØØ1ØØ1	73	49	2Ø1	C9	I		I	
1ØØ1Ø1Ø	74	4A	2Ø2	CA	J		J	
1ØØ1Ø11	75	4B	2Ø3	CB	K		K	
1ØØ11ØØ	76	4C	2Ø4	CC	L		L	
1ØØ11Ø1	77	4D	2Ø5	CD	M		M	
1ØØ111Ø	78	4E	2Ø6	CE	N		N	

7-BIT BINARY	LOW DEC	LOW HEX	HI DEC	HI HEX	ASCII CHAR	INTERPRETATION	WHAT TO TYPE NORMAL	TERMINAL
1001111	79	4F	207	CF	O		O	
1010000	80	50	208	D0	P		P	
1010001	81	51	209	D1	Q		Q	
1010010	82	52	210	D2	R		R	
1010011	83	53	211	D3	S		S	
1010100	84	54	212	D4	T		T	
1010101	85	55	213	D5	U		U	
1010110	86	56	214	D6	V		V	
1010111	87	57	215	D7	W		W	
1011000	88	58	216	D8	X		X	
1011001	89	59	217	D9	Y		Y	
1011010	90	5A	218	DA	Z		Z	
1011011	91	5B	219	DB	[	Opening Bracket	n/a	<ESC>3
1011100	92	5C	220	DC	\	Reverse Slant	n/a	<ESC>4
1011101	93	5D	221	DD	]	Closing Bracket	<SHIFT-M>	
1011110	94	5E	222	DE	^	Circumflex	^	
1011111	95	5F	223	DF	_	Underline	n/a	<ESC>5
1100000	96	60	224	E0	`	Opening Quote	n/a [15]	
1100001	97	61	225	E1	a		n/a	a
1100010	98	62	226	E2	b		n/a	b
1100011	99	63	227	E3	c		n/a	c
1100100	100	64	228	E4	d		n/a	d
1100101	101	65	229	E5	e		n/a	e
1100110	102	66	230	E6	f		n/a	f
1100111	103	67	231	E7	g		n/a	g
1101000	104	68	232	E8	h		n/a	h
1101001	105	69	233	E9	i		n/a	i
1101010	106	6A	234	EA	j		n/a	j
1101011	107	6B	235	EB	k		n/a	k
1101100	108	6C	236	EC	l		n/a	l
1101101	109	6D	237	ED	m		n/a	m
1101110	110	6E	238	EE	n		n/a	n
1101111	111	6F	239	EF	o		n/a	o
1110000	112	70	240	F0	p		n/a	p
1110001	113	71	241	F1	q		n/a	q
1110010	114	72	242	F2	r		n/a	r
1110011	115	73	243	F3	s		n/a	s
1110100	116	74	244	F4	t		n/a	t
1110101	117	75	245	F5	u		n/a	u
1110110	118	76	246	F6	v		n/a	v
1110111	119	77	247	F7	w		n/a	w
1111000	120	78	248	F8	x		n/a	x
1111001	121	79	249	F9	y		n/a	y
1111010	122	7A	250	FA	z		n/a	z
1111011	123	7B	251	FB	{	Opening Brace	n/a	<ESC>6
1111100	124	7C	252	FC	|	Vertical Line	n/a	<ESC>7
1111101	125	7D	253	FD	}	Closing Brace	n/a	<ESC>8
1111110	126	7E	254	FE	~	Overline (Tilde)	n/a	<ESC>9
1111111	127	7F	255	FF	DEL	Delete/Rubout	n/a	<ESC>:

15. Use Closing Quote (39). For high value, use CHR$(96), etc.

APPENDIX E

TROUBLESHOOTING HINTS

This appendix contains two tables designed to help you diagnose problems that can occur when using the SSC to communicate with an RS-232-C device. The device can be a printer, or a plotter, or terminal, or another computer, or some other Data Terminal Equipment (DTE), and it can be connected either directly, or via a modem or some other Data Communication Equipment (DCE). Whenever two DTEs are connected together, there must be TWO modems (DCEs) or ONE modem eliminator (such as the jumper block when it points toward the word TERMINAL) between them.

When diagnosing problems, remember that there are many variables involved in the communications connection:

- the Apple II and its keyboard, screen, and software
- the SSC, the slot it is in, its switch settings (especially mode selection), its jumper block, cable, and software commands
- the external cable, with some number of wires (enough wires?) connected to pins (all the correct pins?) at each end
- possibly two modems connected by low-grade telephone lines, plus another cable from the remote modem to the remote device
- an RS-232-C device at the other end, with its own switch settings and needs (such as paper, ribbon, AC power...)

As you can see, making all these components work together correctly is no mean feat. If there are problems, the easiest way to resolve them is to start with very simple, sure communication between the Apple and the device. Once you have established basic communication (even if the characters are garbled), further troubleshooting becomes much easier. Be patient and methodical.

Trouble usually has characteristics visible on the Apple II screen (Table E-1), or at the device (Table E-2). If your troubleshooting efforts fail, consult your Apple dealer--but first record all the variables (as outlined above) and the symptoms you observed.

Problem	Symptom	Possible Cause	Solution
no data transfer	no sign of any communication at all	cable wires not connected OK; jumper block facing wrong way	check all cable connections, then pin assignments; try reversing jumper block
characters garbled	jh2 3g%$Q	wrong baud rate	change SW1-1 TO SW1-4 or use <n>B command
		wrong data format	change SW2-1 (and SW2-2 in Comm Mode) or use <n>D command to change format
		other device is off, out of paper, etc., off-line	turn on device, remedy its problems, put it on-line
paper not advancing	one line of smudge	printer needs line feeds from SSC	turn SW2-5 ON or use L(inefeed E(nable command
printer is skipping lines	lines look like this	printer and SSC both generating <LF> after <CR>	turn off SW2-5 in Printer Mode, or use L(inefeed D(isable command
missing characters	mssig caractrs	device buffer is overflowing	if device supports full RS-232-C handshaking, ensure all required cable wires are connected if device supports only ETX/ACK, set SIC P8A Mode if device supports XON/XOFF, set Printer Mode and use X(OFF E(nable cmd or set Comm Mode if device supports none of these, set delays with <n>C, <n>L and <n>F cmds
device sticks at line's end going nuts	one long OK line, smudge at right end	device doesn't generate own <CR>, and isn't getting enough from Apple	use SIC P8 Mode and <n>N command, or Printer Mode and C command plus appropriate SW2-3 and SW2-4 have software send <CR> before right margin

Table E-1. Problems Detected at the Device

Problem	Symptom	Possible Cause	Solution
Apple has occasional bad times	it works one minute & not next	ACIA interrupting the Apple when DCD or DSR changes	make sure that interrupt switch SW2-6 is OFF
Apple not working	dead kybd and screen	SSC in slot #3 under Pascal	Pascal expects external terminal to run the show
Apple kybd seems off	keystrokes all lost	echo off; keyboard zapped; IN# not 0	use E(cho E(nable cmd; unzap with POKE; IN#0
screen seems off	nothing typed is displayed	device not echoing (half duplex) or ACIA not sending to screen	in Comm or Terminal Mode, use E(cho E(nable; in SIC or Printer Mode, use I command or SW2-3 & -4 ON
screen is seeing double	eevveerryy tthhiinngg ttwwiiccee	device & SSC both echoing to Apple (full duplex)	use E(cho D(isable cmd in Comm Mode or use <n>N cmd in Printer Mode
screen is spacing double	lines look like this	device generating and sending <LF> after <CR>	use M(ask E(nable command to remove extra linefeeds
forced uppercase display	lowercase beCOMES UPPERCASE	Apple monitor changing letters in GETLINE routine	use <n>T command to allow lowercase to pass through (not possible in Pascal)
Apple misses some characters at the beginning of lines	pple sses ome racters t the bgnning lines	screen scrolling too slowly, or BASIC or Pascal program running too slowly, and so ACIA overruns	turn off screen (<n>N or SW2-3 & -4 in Prtr Mode); reduce scroll window; use assembly language or faster program routines; use lower baud rate (300 vs. 1200); use <n>C, <n>L or <n>F commands; in Comm Mode, chain (<n>S cmd) to 80-column card with its own scrolling hardware

Table E-2. Problems Detected at the Apple

APPENDIX F
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 F-1 lists the bit definitions of this byte:

\$678+s — Status Byte (STSBYTE)

7	6	5	4	3	2	1	Ø
Ø	Ø	Error	Ø	Car Lst	Overrun	Frm Err	Par Err

Bit	"1" Means	"Ø" Means
Ø	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

Table F-1. STSBYTE Bit Definitions

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

Bits Ø,1, and 2 are the same as the corresponding three bits of the ACIA Status Register (Appendix A). Bit 3 indicates whether or not the Data Carrier Detect (DCD; Chapter 4) 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 Ø, no error occurred.

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.

Any character--including the carriage return at the end of a WRITELN statement--will cause posting of a new value in IORESULT.

Table F-2 shows the possible combinations of error bits correspond to these decimal error codes.

BASIC PEEK $678+s or Pascal IORESULT	Carrier Lost	Overrun	Framing Error	Parity Error
Ø	(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
4Ø	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

Table F-2. Error Codes and Bits

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

GLOSSARY

To avoid lengthy or repetitive definitions, many terms used in one definition are themselves defined elsewhere in this glossary. Also for the sake of brevity, terms and expressions are spelled out, with their abbreviations immediately after them. In a glossary of this size, the reader will have little difficulty locating abbreviations.

ACK: An ASCII character (decimal 6; Appendix D) sent from a device to the Apple II in response to an ETX or ENQ character in SIC P8A Emulation Mode.

American Standard Code for Information Interchange (ASCII): A standard defining the codes to represent a 128-element character set (Appendix D) in a fixed way for devices of different manufacturers. It is the standard for digital communication over telephone lines.

Asserted: Made true (positive in positive-true logic; negative in negative-true logic). Usually refers to electrical signals, like the RS-232-C signal Clear To Send, etc.

Asynchronous: Having a variable time interval between characters.

Asynchronous Communications Interface Adapter (ACIA): In the SSC, a single chip (Synertek 6551 or equivalent) that converts data from parallel to serial form and vice versa, and handles serial transmission and reception and RS-232-C signals, under the control of internal registers set and changed by SSC firmware.

Baud: A unit of signalling speed equal to the number of discrete conditions or signal events per second. With the SSC, for example, using a data format of 1 start bit, 7 data bits, 1 parity bit and 1 stop bit (1Ø bits in all), 3ØØ baud is approximately equal to 3Ø characters per second.

Binary: A number system with two digits, "Ø" and "1," with each digit position moving from right to left representing a successive power of two. For example, 1 represents decimal 1; 1Ø represents 2; 1ØØ represents 4; 1ØØØ represents 8, etc.

Bit: A BInary digiT, either a Ø or a 1.

BREAK: A Ø.233 second SPACE (Ø) signal sent over a communication line to interrupt the sender. This signal is often used to end a session with a timesharing service.

Carriage Return (CR): An ASCII character (decimal 13; Appendix D) that ordinarily causes a printer or display screen to place the subsequent character on the left margin. On a manual typewriter, this movement is combined with linefeed (the advancement of the paper to the next line). With computers, carriage return and linefeed are separate, causing hair-raising problems for the user.

Carrier: The background signal on a communication channel that is modified to "carry" the information. Under RS-232-C, the carrier signal is equivalent to a continuous MARK or 1; a transition to Ø then represents a start bit.

Character: Any symbol that has a widely understood meaning. In the ASCII code, letters, numbers, punctuation marks, and so on, are all characters (Appendix D).

Chip: A tiny wafer of silicon, with conductive metallic impurities, that has layers of microscopic circuits etched on it.

Clear To Send (CTS): An RS-232-C signal from a DCE to a DTE that the SSC keeps false until the DCE makes it true, indicating that all circuits are ready to transfer data.

Command Character: An ASCII character, usually <CTRL-A> or <CTRL-I> (Appendix D), that causes the SSC firmware to interpret subsequent characters as a command.

Command Register: An ACIA location (at hexadecimal address $CØ8A+sØ) that stores parity type and RS-232-C signal characteristics.

Communications Interface Card (CIC): An Apple II interface card designed to connect the Apple II to a device via a DCE.

Communications Mode: An operating state in which the SSC is prepared to exchange data and signals with a DCE.

Control Character: Any character generated by holding down the key marked CTRL while pressing some other key.

Control Register: An ACIA location (at hexadecimal address $CØ8B+sØ) that stores data format and baud rate selections.

Daisy Chaining: A method of passing incoming signals and data from one peripheral connector slot to another, such as from the SSC slot to a slot containing an 8Ø-column-display card.

Data Bit: With the SSC, one of 5 to 8 bits representing a character.

Data Carrier Detect (DCD): An RS-232-C signal from a DCE to a DTE (such as the Apple II) indicating that a communication connection has been established. The SSC's internal circuits hold DCD false until the external device sets DCD true.

Data Communication Equipment (DCE): As defined by the RS-232-C standard, any device that transmits or receives information. Usually this is a modem. However, when a Modem Eliminator is used, the Apple II looks like a DCE to the other device, and the other device looks like a DCE to the Apple.

Data Conversion: Changing of data from parallel to serial form or from serial to parallel form.

Data Format: The form in which data is stored, manipulated or transferred. Serial data transmitted and received by the SSC has a data format of: one start bit, 5 to 8 data bits, an optional parity bit, and one, one and a half, or two stop bits.

Data Set Ready (DSR): An RS-232-C signal from a DCE to a DTE indicating that the DCE has established a connection.

Data Terminal Equipment (DTE): As defined by the RS-232-C standard, any device that generates or absorbs information, thus acting as a terminus of a communication connection.

Data Terminal Ready (DTR): An RS-232-C signal from a DTE to a DCE indicating a readiness to transmit or receive data.

Default Value: A value that is assumed or set in the absence of explicit instructions otherwise.

Device: A piece of equipment; usually a printer, plotter, terminal or computer. When the jumper block is in the MODEM position, the SSC expects the device to be a DCE (such as a modem).

Echo: To send an input character to a video screen, printer, or other output device. On a typewriter, what we strike on the keyboard appears on the page in the same step. With a computer, these two steps are controlled separately.

Electromagnetic Interference (EMI): Electrical or magnetic signals or noise that disturbs the operation of radio or television receivers. For example, a hair dryer often creates EMI that fuzzes up the picture on a nearby television set.

Emulation Mode: A manner of operating in which one computer or interface imitates another. For example, in SIC P8 Emulation Mode, the SSC acts very much like an Apple II Serial Interface Card with the P8 version of firmware.

ENQ: An ASCII character (decimal 5; Appendix D) used in the ENQ/ACK protocol (SIC P8A Emulation Mode).

ETX: An ASCII character (decimal 3; Appendix D) used in the ETX/ACK protocol (SIC P8A Emulation Mode).

Even Parity: Use of an extra bit set to Ø or 1 as necessary to make the total number of 1 bits an even number. For example, the 7-bit ASCII code for the letter A (1ØØØØØ1) has two 1 bits; for even parity, the transmitting device appends an eighth bit equal to Ø so that the total number of 1 bits remains even. The receiving device can count 1 bits as a way of checking for transmission errors.

False: Zero or negative voltage in positive-true logic; positive voltage in negative-true logic. Absence of an arbitrary signal or condition.

Firmware (FW): Software that resides in ROM and so is relatively unchangeable (firm) compared to software in RAM.

Form Feed (FF): An ASCII character (decimal 12; Appendix D) that causes a printer or other paper-handling device to advance to the top of the next page.

Framing Error (FRM): Absence of the expected stop bit(s) on a received character. The ACIA records this error by setting bit 1 (FRM) of its Status Register to 1. The ACIA checks and records each framing error separately: if the next character is OK, the FRM bit is cleared.

Full Duplex: Capable of simultaneous two-way communications.

Half Duplex: Capable of communications in one direction at a time.

Handshake : A kind of communication protocol in which the receiving device, when it has successfully gotten a character or block of characters, sends back an acknowledging signal, thereby triggering the next transmission.

Hardware: The actual physical switches, wires, chips, PC boards, and so on, of a computer system.

Header: A cable connector mounted on a PC board.

Hexadecimal: A numbering system that uses 16 digits; usually these are represented by the ten decimal digits, Ø through 9, plus the letters A through F (A representing decimal ten, F representing decimal fifteen, etc.). Each hexadecimal digit can represent a string of four binary digits.

High-order Bit: See Most Significant Bit.

Initialization: The process of setting up initial values and conditions. In the SSC, the firmware finds out the switch positions and the current operating system, and uses these

findings to initialize both the ACIA registers and the Scratchpad RAM locations for the slot the SSC is in.

Input: Data that flows from the outside world into the Apple II.

Interface: Some combination of hardware, firmware and software that makes possible the useful connection of two otherwise incompatible pieces of equipment.

Interrupt: A special control signal from an external source that diverts the Apple II from the program it is executing to a specific routine that handles the condition (such as a printer gone awry) that caused the interrupt.

Jumper Block: In the SSC, a plastic plug with pins connected in such a way that it passes RS-232-C signals between the SSC and the external device either unchanged (MODEM position) or permuted in the manner of a Modem Eliminator (TERMINAL position).

Least Significant Bit (LSB): The right-hand bit of a binary number as written down; its positional value is Ø or 1 (that is, Ø or 1 times 2 to the Ø power).

Linefeed (LF): An ASCII character (decimal 1Ø; Appendix D) that ordinarily causes a printer or video display to advance to the next line.

Local: Nearby; capable of direct connection using wires only.

Low-order Bit: See Least Significant Bit.

MARK Parity: A bit of value 1 appended to the high-order end of a binary number for transmission. The receiving device can then check for errors by looking for this value on each character.

Mode: Manner of operating. The SSC can operate in one of four chief modes, depending on the settings of switches SW1-5 and SW1-6: Printer Mode, Communications Mode, SIC P8 Emulation Mode, and SIC P8A Emulation Mode.

Modem: MOdulator/DEModulator; a DCE device that connects a DTE to communications lines. As used with the SSC, a device that exchanges RS-232-C signals with the ACIA to establish a communications connection, and then either converts data from RS-232-C voltages to RS-232-C tones for transmission, or performs the opposite conversion on received data.

Modem Eliminator: The physical crossing of wires that replaces a pair of modems for direct connection of two pieces of RS-232-C Data Terminal Equipment. In the SSC, the jumper block serves this purpose when installed in the TERMINAL position.

Most Significant Bit (MSB): The leftmost bit of a binary number as written down. This bit represents Ø or 1 times 2 to the power one less than the total number of bits in the binary number. For example, in the binary number 1ØØØØ, the 1 represents 1 times 2 to the fourth power, or sixteen.

Odd Parity: Use of an extra bit set to Ø or 1 as necessary to make the total number of 1 bits an odd number. For example, the 7-bit ASCII code for the letter A (1ØØØØØ1) has two 1 bits; for odd parity, the transmitting device appends an eighth bit equal to 1, making the total number of 1 bits odd. The receiving device can check for transmission errors by counting 1 bits.

Output: Data that flows from the Apple II to an external device.

Overrun (OVR): A condition that occurs when the Apple II processor does not retrieve a received character from the Receive Data Register before the subsequent character arrives. The ACIA automatically sets bit 2 (OVR) of its Status Register; subsequent characters are lost. The Receive Data Register contains the last valid data word received.

P8: One of two types of Programmable ROM (PROM) installed in the Apple II Serial Interface Card. This PROM performed batch moves, but had no provision for software handshaking.

P8A: One of two types of Programmable ROM (PROM) installed in the Apple II Serial Interface Card. This PROM provided the ENQ/ACK software handshaking required by several types of printers.

Parallel Interface: A connection between two devices where there is a separate wire for each bit of a character, so that an entire character can be transferred in a single instant.

Parity: Maintenance of a sameness of level or count, usually the count of 1 bits in each character, for error checking. In the SSC, the ACIA has a register that stores the type of parity selected (none, odd, even, MARK or SPACE). It automatically generates the parity bit when transmitting, and both checks and discards parity bits appended to received characters.

Parity Error (PAR): Absence of the correct parity bit value in a received character. The ACIA records this error by setting bit Ø (PAR) of its Status Register to 1.

Peripheral Connector Slot: One of eight 5Ø-pin slots inside the Apple II case near the back. Within certain restrictions, each slot can contain add-on memory, an adapter for 8Ø-column display, or an interface to an external device.

Polarized Header: On the SSC, a 1Ø-pin female connector for the internal cable; this connector has a slot on one side that receives a "key" on the cable's male connector.

Printed Circuit (PC) Board: A sheet of stiff nonconductive material with one or more thin layers of metal bonded to it. Unwanted areas of this metal are etched away, leaving the paths of the desired circuits. Electronic components can then be soldered to the board. Small PC boards are also called cards.

Printer Mode: An operating state in which the SSC is prepared to exchange data and signals with another DTE (such as a printer).

Protocol: A predefined exchange of control signals between devices enabling them to prepare for coordinated data transfer.

Radio Frequency Interference (RFI): Electromagnetic interference occurring at frequencies used for radio communications.

Random Access Memory (RAM): A series of storage locations that can be accessed directly (by means of horizontal and vertical coordinates) for both reading and writing.

Read Only Memory (ROM): A series of storage locations that can be read but cannot be written to; this protects the programs and data in the ROM from alteration or destruction.

Receive Data Register: A read-only register in the ACIA (at hexadecimal location $CØ88+sØ) that stores the most recent character successfully received.

Remote: Too distant for direct connection via wires or cables only.

Request To Send (RTS): An RS-232-C signal from a DTE to a DCE to prepare the DCE for data transmission.

Ring Indicator (RI): An optional RS-232-C signal from a DCE to a DTE that indicates the arrival of a call.

RS-232-C: A standard created by the Electronic Industries Association (EIA) to allow devices of different manufacturers to exchange serial data--particularly via telephone lines. The ACIA in the SSC implements all the required primary RS-232-C signals. These signals are true when at Ø volts.

Scratchpad RAM: Eight locations in the Apple's memory reserved for each of the 8 peripheral connector slots (64 bytes in all).

Secondary Clear To Send (SCTS): A secondary RS-232-C signal that some printers use instead of Clear To Send.

Serial Interface: A connection in which all the bits of a character are sent along a single wire one after the other.

Serial Interface Card (SIC): An Apple II product designed to connect an RS-232-C device directly to the Apple II.

SIC Emulation Mode: A state of operation in which the SSC imitates an Apple II Serial Interface Card.

SPACE Parity: A bit of value Ø appended to a binary number for transmission. The receiving device can look for this value on each character as a means of error checking.

Start Bit: A transition from a MARK signal to a SPACE signal for one bit-time, indicating that the next string of bits represents a character.

Status Register: An ACIA register (hexadecimal location $CØ89+sØ) that stores the state of two of the RS-232-C signals and of the Transmit and Receive Data Registers, as well as the outcome of the most recent character transfer.

Stop Bit: A MARK signal following a string of data bits to indicate the end of a character.

Super Serial Card (SSC): The interface card described in this manual. It is called "super" because it can simultaneously transmit and receive data in one of 35 formats at any of 15 speeds, honor several software protocols, communicate directly with either DTE or DCE, change operating characteristics in response to software commands, and dovetail with the chief operating environments offered with the Apple II.

Terminal: An input/output device, usually made up of a keyboard and video display and sometimes including its own printer and magnetic storage devices, that can act as a separate and even remote site for data transfer with a computer system.

Terminal Mode: An operating state of the SSC in which the firmware bypasses the Apple II's central processor, and makes the Apple act as a simple terminal capable of generating all of the ASCII characters.

Transmit Data Register: A write-only register in the ACIA (at hexadecimal location $CØ88+sØ) that holds the current character to be transmitted.

True: Positive voltage in positive-true logic; zero or negative voltage in negative-true logic. Assertion of an arbitrary signal or condition.

XOFF: An ASCII character (decimal 19; Appendix D) sent by a receiving device to a transmitting device to halt transmission of characters.

XON: An ASCII character (decimal 17; Appendix D) used in the XON/XOFF protocol as a go-ahead character from the receiving device to the sending device after an XOFF has been sent to halt transmission.

FIGURES AND TABLES

Chapter 1
GETTING STARTED

Chapter 2
PRINTER MODE

Chapter 3
COMMUNICATIONS MODE

Chapter 4
HOW THE SSC WORKS

Appendix A
FIRMWARE

Appendix B
APPLE INTERFACE CARD EMULATION

Appendix C
SPECIFICATIONS AND SCHEMATICS

Appendix E
TROUBLESHOOTING HINTS

Appendix F
ERROR CODES

INDEX

T

U

V

W

X

Y

Z