CCS MODEL 7710

APPLE II™ ASYNCHRONOUS SERIAL INTERFACE

OWNER'S MANUAL

Rev D
42000036-01

California Computer Systems
250 Caribbean Drive
Sunnyvale, CA 94086

APPLE II™ and APPLESOFT™ are trademarks of APPLE COMPUTER, INC.

TABLE OF CONTENTS

CHAPTER 1

INTRODUCTION

1.1 GENERAL DESCRIPTION

The CCS Model 7710 interfaces your APPLE computer to such peripheral devices as video terminals, line printers, modems, and even other computers. It is available in two versions: the 7710A, which includes the standard 7610C firmware for peripheral interfacing; and the 7710D, which includes the 7610D firmware for interfacing the APPLE to another computer as well as to a peripheral. Each of these firmware packages is separately available, allowing you to add capabilities with a minimum of new hardware. In addition, firmware which allows 80 characters per line and 54 lines per page is available separately as product 7610B.

The 7710 features switch-selectable baud rates for compatibility with a wide variety of peripheral devices at optimum data rates.

1.2 SPECIFICATIONS

SYSTEM INTERFACE

Internal:	APPLE II Peripheral Slots 1-7
External:	EIA RS-232-C (1969) Configurations A through E Primary Circuits Only DCE (DTE Adaptor Available) Failsafe Input Circuits (Shorts and Opens)
TRANSFER MODE	Asynchronous Serial 7 or 8 Data Bits Odd, Even, or No Parity Bits 1 or 2 Stop Bits

DATA RATES	50 baud	75 baud	110 baud
	134.5 baud	150 baud	200 baud
	300 baud	600 baud	1200 baud
	1800 baud	2400 baud	4800 baud
	9600 baud	19200 baud	External

PROGRAM MEMORY	256 Bytes ROM, Auto Power Down Replaceable with 2112 RAMs
REQUIRED POWER	+5VDC +12VDC -12VDC
FEATURES	Supports Interrupt Daisy Chain with On-Board Arbitration Logic DMA Daisy Chain Pass-Through Glass Epoxy (FR-4) PC Board Gold Plated Connector Fingers Solder Mask Both Sides of Board Component Silkscreen

1.3 USING THIS MANUAL

All users of the 7710 should read carefully Chapters 2 and 3, which deal with the configuration, installation, and operation of the 7710. The rest of the manual may be read or ignored as the user desires. The balance of Chapter 1 is an introduction to serial interfaces, and is provided for those users who are not familiar with the principles of serial interfaces in general and the RS-232-C standards in particular. Chapter 4 contains information required only by those users who plan to write their own assembly-language drivers for the 7710. Chapter 5, a discussion of the hardware design of the 7710, is intended for the curious and for those who need to trouble-shoot or modify the 7710. Appendix A contains miscellaneous technical information, including a schematic/logic diagram and a list of user-replaceable parts. Listings for the 7610C and 7610D are provided in Appendix B. Those using the 7710 as a printer interface can find cable-connection requirements for a number of common printers in Appendix A. ROM, RAM, and interface check-out procedures, for those who wish to run them, are given in Appendix D.

Throughout this manual, low-active signals are indicated by a minus before the signal name/mnemonic (e.g., -DEV SEL). Control characters are indicated by a ^ before the character (e.g., ^P). The symbol <cr> represents a carriage return.

1.4 SERIAL INTERFACES: BASIC THEORY

A computer is a very expensive do-nothing unless you can give it data, instruct it what to do with the data, and then have it present the results. To help do this, peripheral devices were designed. But because computers could communicate with peripherals in a number of formats, another kind of device was necessary: interfaces. Interfaces translate between a computer, which inputs and outputs in one format, and a peripheral, which inputs and/or outputs in another format.

There are two major types of computer data transfer: parallel and serial. Parallel interfaces transfer words of data simultaneously on parallel data lines. In parallel communications, all data bits must be not only transmitted but also received simultaneously. Simultaneous reception can be assured only if the distance of transmission remains below a certain limit, typically about ten feet for a computer-to-peripheral cable.

The simultaneous reception problem is not present in serial data transfer, in which the transmitter disassembles words and sends them a bit at a time over a single wire and the receiver reassembles the arriving stream of bits into words. For this scheme to work, the receiver and transmitter must make identical assumptions about which part of the data stream represents which bit of which word. Two ways have been devised to do this: synchronous and asynchronous modes of transmission. During synchronous transmission, a pre-defined pattern of synchronization bits is sent out first. When the receiver finds this pattern, it divides the subsequent data into pre-defined word-length groups. This requires

highly precise synchronization of transmitter and receiver timing. Asynchronous interfaces handle the problem differently. They add a start bit to the front of each word, plus one or more stop bits to the end. The total group of bits is then transmitted one bit at a time. The receiver can identify the data bits even if it is slightly out of sync with the transmitter because it resynchronizes at the beginning of each word.

In both synchronous or asynchronous modes, the receiver must know how fast the data is being sent. Generally, as long as the receiver expects data at the rate the sender is sending it, the actual rate of transfer does not matter. Common usage has defined many standard signal rates. Usually, they are an even multiple of 75 baud (bits per second, including overhead bits such as start and stop bits). A few other rates, 50, 110, and 134.5, are used by the industry giants. The RS-232-C standard for serial communication does not specify any standard signal rates, but suggests a practical upper limit of 20 kilobaud and indirectly establishes a theoretical upper limit of 50 kilobaud.

It is not enough for computers and peripherals to be able to exchange data; each must also be able to tell when the other is ready to transmit or receive. This is done with "handshaking" signals. Because a wide variety of handshaking schemes are possible, the EIA created the RS-232-C interface specifications to let manufacturers know what to expect. Two "sides" were defined. Because one side of the interface is usually connected to some type of computer terminal, equipment at that end is called Data Terminal Equipment (DTE). Equipment at the other end is called Data Communications Equipment (DCE), because to transmit serial data over long distances via telephone wires a Modulator/

Demodulator, or Modem, is needed. For short distances (less than several hundred feet), modems and telephone wires are not needed, but one side of the interface must be made to think it's DCE.

RS-232-C defines the necessary protocol between the DCE and the DTE for many possible configurations (see Sections A.4 and A.5). Of these, the first five (A, B, C, D, and E) are most commonly used. Configuration A defines a one-way, transmit-only interface, as might be used by a simple serial keyboard. Configuration B is also a one-way, transmit-only interface, but it has more handshaking. A paper tape reader might use this type of interface. Configuration C is also one-way, but receives only, and might be used by a serial printer.

With Configuration D, we start getting into two-way traffic. Configuration D is HALF DUPLEX; although it can carry traffic both ways, it can only carry one way at a time. When a modem is used in this configuration, it is often called a "two-wire modem" because the telephone line is connected with only two wires. This type of interface is not often used. Configuration E is a FULL DUPLEX link. This means that the link can support traffic in both directions at the same time. CRT terminals most often use this type of link. A modem used in this configuration is called a "four-wire modem." The 7710 supports all five of these configurations. In addition, a Data Terminal Ready (DTR) handshake line has been provided. Normally used only on synchronous interfaces, this line is also used by some asynchronous printers.

CHAPTER 2

SET-UP AND INSTALLATION

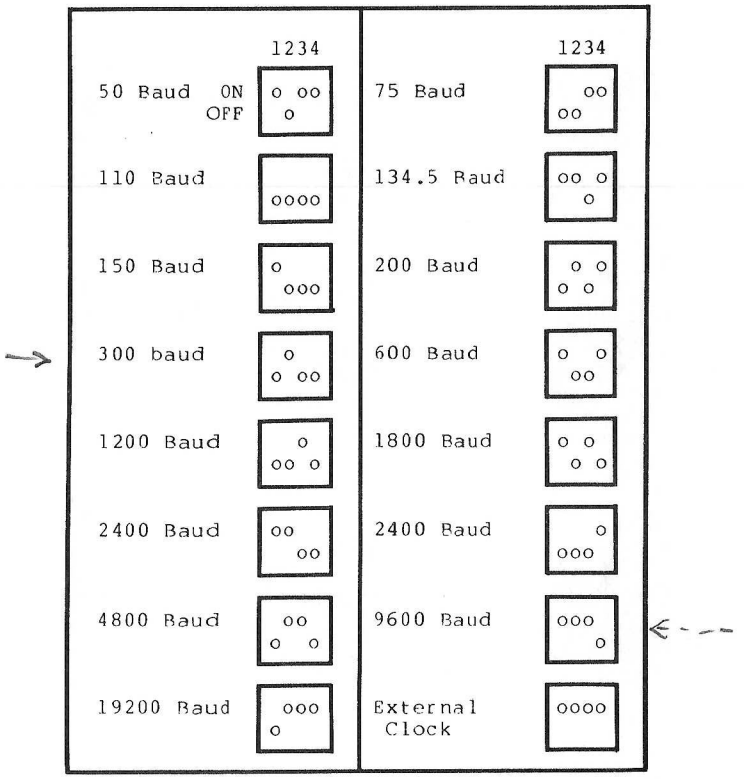
2.1 BOARD CONFIGURATION

Before you install and operate your 7710 Asynchronous Interface, you will need to configure it and your peripheral to ensure that the 7710, your peripheral, and your APPLE meet each other's expectations. Unless you install RAMs, configuration of the 7710 consists of setting the baud rate switches. Use of RAMs requires the installation of a jumper wire. (NOTE: You will use RAMs only if you write your own machine language driver program to replace the PROM driver that comes with the card.) Also, you will have to acquire or construct the appropriate cable for your peripheral.

2.1.1 SETTING THE BAUD RATE

Your peripheral manual should tell you at which baud rate or rates the peripheral will operate. If the peripheral can handle several baud rates, choose the highest one available which is common to the 7710 interface card. Set your terminal for that baud rate, following its instructions. Then set the rocker switches on the 7710 for that rate as shown in Table 2.1 (o means to push that side of the switch down).

TABLE 2.1. BAUD RATE SELECTION



[NOTE: If your peripheral's baud rate is available as an output and your cable makes the signal available to the 7710 at RS-232-C connector pin 24, you can set the 7710 to the peripheral's baud rate by setting the baud rate switches to the code for External. The incoming frequency must be 16 times the baud rate; e.g., 300 baud requires a 4800 Hz external clock signal.]

2.1.2 RAM JUMPER

This jumper is required only if you plan to write your own driver for the 7710 and wish to use RAMs when developing the driver. RAM contents will be continually destroyed by the ROM power-down feature unless a jumper wire is soldered between the pads marked RAM located just above the card connector fingers at the right side of the board.

2.2 PERIPHERAL CONFIGURATION

The following instructions apply to both the 7710A and the 7710D unless otherwise noted.

2.2.1 PERIPHERAL HALF/FULL DUPLEX (7710A Only)

If you are interfacing a video terminal that allows you to select between half and full duplex operation, select FULL DUPLEX. Your computer's

firmware expects a full duplex keyboard/display. This means that the computer, after receiving a character from the keyboard, sends that character back out to the display, allowing a quick check that the computer received what you wanted it to receive. When a terminal is in the half duplex mode, it displays the character as it is typed in, then, when the computer echoes, displays it a second time. For example, if you typed in RUN with the terminal in a half duplex mode, you would see RRUUNN on the display.

2.2.2 PERIPHERAL LINE FEED (7710A Only)

The 7710A firmware generates a Line Feed control character each time a Carriage Return (^D) is sent. If your terminal or printer has an Auto Line Feed option, make sure it is turned off. If it is turned on your texts will be double spaced.

2.2.3 PERIPHERAL UPPER CASE

Your APPLE expects all alphabetic characters to be in upper case. If your terminal has an Upper Case Only option, enable it. (While the standard firmware converts lower case to upper case, it is a good idea to select Upper Case Only if your terminal has that option.)

2.2.4 PERIPHERAL PARITY

On asynchronous links, data transfer is usually so reliable that no parity generation or checking is needed; the driver default command therefore selects no parity. This command may be altered, however, as described in Section 4.3.1. Whichever parity option you choose (even, odd, or none), the terminal must be set to match the command.

2.3 THE PERIPHERAL CABLE

The 7710 includes a cable to run from the card to the back of the computer. However, you will need to obtain a cable to connect your peripheral to the computer. In most cases a standard RS-232-C connector will be required, but you may have to make some modifications depending on your peripheral's handshaking. (Appendix C details the cable requirements for a number of printers.) The 7710 handshakes its output data (e.g., to a printer) on pin 4 (RTS) and handshakes its input data (e.g., from a keyboard) on pin 20 (DTR).

2.4 7710D DCE-TO-DTE CONVERSION

The RS-232-C standard for serial data communication defines the characteristics of the interchange signals between Data Terminal Equipment (DTE) and Data Communications Equipment (DCE), such as a MODEM. Each equipment type has

its own set of handshaking signals that are assigned to specified pins of a DB-25 connector. Since the 7710 was designed to interface with DTE equipment--terminals, printers, etc.--its connector configuration is that of a DCE device. Most users of the 7610D firmware, however, will want to interface the 7710 to a modem or computer so that the APPLE itself can be used as a terminal, a DTE device. In this application, the 7710's connector usually needs to be rewired for DTE configuration. (Some computers' RS-232-C connectors may take the DTE role, allowing the 7710 connector to be used as is. See your computer's manual.)

A DCE-to-DTE adapter can be ordered from CCS or built by the user. To make a DTE Adapter, wire together male and female DP-25 connectors as described below. The male end plugs into the connector at the back of your APPLE. Only a few wires are needed to create the necessary signals on the female adapter connector: the primary DCE handshaking lines (CTS, DSR) from the 7710 must be switched to the corresponding DTE handshaking lines (RTS, DTR); the serial data transfer lines must be exchanged; and the ground wires must be connected. The male-to-female pin connections are as follows:

PIN 5 (CTS) to PIN 4 (RTS)
PIN 6 (DSR) to PIN 20 (DTR)
PIN 2 to PIN 3 (Received data from DCE)
PIN 3 to PIN 2 (Transmit data to DCE)
PIN 1 to PIN 1 (Protective Ground)
PIN 7 to PIN 7 (Signal Ground)

If you will be using the External Baud Rate option on the 7710, you must also wire the 7710's External Clock input to the DTE External Clock line:

PIN 24 to PIN 15 (External Clock from DCE)

2.5 CARD INSTALLATION

Before installing the interface in your computer, align pin 1 of the I/O cable connector with pin 1 of the mating connector on the 7710. (Pin 1 of the cable connector can be identified by the outside stripe on the cable and by a triangular mark on the connector; pin 1 of the card connector is identified by a 1 on the silkscreen.) When all pins are properly aligned, push down firmly on the connector until you can no longer see the metal pins. Gently fold the ribbon cable on the diagonal so that the back panel connector is to the right of the board. Crease the fold only slightly; too much crease could fatigue and break the wires. This provides slack in the cable needed for strain relief.

The card is now ready to be inserted into the computer. Before installing the card, turn off the APPLE and disconnect the power cord.

```
****************************************
*                                      *
*  WARNING:  Do not remove the cover   *
*  of your computer if the power cord  *
*  is plugged in.  You may injure      *
*  yourself or damage your computer.   *
*                                      *
****************************************
```

Place the computer directly in front of you. Put the palms of your hands on the back of the

computer and curl your fingers around the rear edge. Gently but firmly pull up until you hear two distinct pops. Don't lift the cover any farther; slide it to the rear to remove it from the computer. Inside, toward the rear of the computer, you will see eight 50-pin connectors. They are numbered 0 through 7 from left to right. Place the 7710 in any of these connectors, with the exception of #0, the leftmost; slot 0 does not have the 256-byte program area available. Hold the card so that the component side is to the right, align the card edge in the chosen connector, and gently push the card down until it is firmly seated. Run the I/O cable through one of the slots in the back of the APPLE so that the RS-232-C connector is outside the computer. (We recommend that you use CCS's APPLECLIP--product #7300--to mount the connector securely on the back of the APPLE.) Replace the computer cover, and you are ready to try out the board.

CHAPTER 3

OPERATING INSTRUCTIONS

The 7710A is extremely simple to use. One command from either BASIC or the Monitor causes the APPLE to send output to or expect input from the peripheral interfaced through the 7710. Simple procedures also allow you to alter the characters per line and data format parameters.

The 7710D in the non-terminal mode operates the same as the 7710A, though it does not support the same parameters. Operation of the 7710D in the terminal mode is described in Section 3.2.

3.1 7710A OPERATION

3.1.1 OUTPUT

If you are using the 7710 as a printer interface, you will need to tell the APPLE when to stop sending output to the console and where to send it so that the printer receives it. To establish a printer or other peripheral interfaced through the 7710 as the destination of output from the APPLE, enter one of the following commands (n equals the 7710's slot number) from the keyboard.

PR#n	(> or] BASIC)
n^P	(Monitor)

All subsequent output from the computer will now be handled by the firmware's output routines. Anything typed at the keyboard will thus go to the printer or other peripheral, and will not appear on the screen. To restore the screen as the output destination, type in PR#0 from BASIC or 0^P from the Monitor.

Many users will want to select a printer as the output destination under program control. The following example shows how to do so. Note that the same commands are used as when the output destination is altered from the keyboard.

```
10 D$=CHR$(4)
100 INPUT "WHICH SLOT IS THE 7710 IN (1-7)?";S
110 PRINT D$;"PR#";S
120 PRINT "THIS SHOULD BE SENT TO THE PRINTER"
130 PRINT D$;"PR#0"
140 END
```


3.1.2 INPUT

If you are using the 7710 to interface a keyboard, paper tape reader, or other input device to the APPLE, you will need to tell the APPLE when to stop expecting input from the main keyboard and where to go for future input. To establish the peripheral interfaced through the 7710 as the source of input to the APPLE, enter one of the following commands (n = the 7710's slot number) from the main keyboard.

IN#n	(> or] BASIC)
n^K	(Monitor)

Either of these commands will cause your computer to go to the 7710 firmware's input routines for all subsequent input. The APPLE keyboard is now virtually "dead." The APPLE II will respond only to Reset and ^C entered from the keyboard; therefore, you should enter the PR# command before the IN# command if you wish to both input and output through the 7710. To restore the APPLE keyboard as the input device, enter IN#0 or 0^K from the new input device (if it has a keyboard) or under program control, or enter RESET or ^C from the APPLE keyboard.

3.1.3 CHANGING 7710A PARAMETERS

There are two default parameters in the standard driver which can be changed AFTER an IN#n/n^K or PR#n/n^P command has been executed and THE INTERFACE HAS BEEN USED ONCE.

3.1.3.1 Data Parameters

The default value for the ACIA operating mode command is $11, which specifies an 8 bit word, 2 stop bits, no parity, and no interrupts. To change the operating mode, POKE the desired command (derived from Table 4.1) into location 16256 + 16*n ($C080 + $n0), where n is the slot number. Remember, though, that any subsequent IN#n or OUT#n command will restore the default value.

3.1.3.2 Characters per Line

The firmware will automatically initiate a carriage return/line feed sequence after 255 characters have been printed on one line. If you want some other number of characters per line, simply POKE the desired number into location 1528 + the slot number. Make sure the number is in the range 0 < CPL <= 255. From BASIC, the keyboard command sequence (n = the slot number; c = characters per line) is as follows:

```
PR#n
POKE 1528+n,c
```

(NOTE: Some printers will not print characters until a complete line ending with a carriage return is received.)

The following example shows how to change the characters-per-line parameter under program control. Line 130 prints a null character without a carriage return, initializing the interface before the new value is POKEd to the characters-per-line location. Any PRINT statement

could be used in this line, but non-null characters would be printed, which would in most cases be undesirable.

```
10 D$=CHR$(4)
100 INPUT "NUMBER OF CHARACTERS PER LINE?";C
110 INPUT "SLOT NUMBER?";S
120 PRINT D$;"PR#";S
130 PRINT CHR$(0);
140 POKE 1528+S,C
150 REM -REPLACE THE FOLLOWING LINES WITH
    YOUR OWN PROGRAM
160 FOR I=4*C/10
170 PRINT *;
180 FOR J=1 TO 9
190 PRINT J;
200 NEXT J
210 NEXT I
220 PRINT D$;"PR#0"
230 END
```

3.1.4 FREEZING TERMINAL DISPLAY (^S)

If you use the 7710A with a device capable of both input and output (usually a keyboard/display console), you can freeze the display of a long listing or text at any point by entering a ^S from the device's keyboard. The firmware will stop outputting characters to the screen until another character is entered from the keyboard, after which output to the screen will resume as normal.

3.2 7710D OPERATION

The 7710 Asynchronous Serial Interface with 7610D firmware allows the APPLE II to function as an ordinary computer terminal when connected to either a modem or another computer. In the terminal mode of operation, the firmware supports both full- and half-duplex modes and the BREAK signal. The user can switch from a non-terminal mode to the terminal mode with a few commands entered from the APPLE II keyboard or from an external device. In the non-terminal mode, the 7610D firmware allows the APPLE II to interface to such peripherals as printers and terminals. Input to the APPLE II can be taken from the 7710 instead of the APPLE's keyboard and output from the APPLE II can be sent to the 7710 as well as to the APPLE's display.

3.2.1 NORMAL I/O MODE

In the non-terminal mode, the 7710D responds to the PR# and IN# commands, as well as the Monitor equivalents, as decribed in Section 3.1.

3.2.2 TERMINAL MODE

After the IN# and PR# commands, the APPLE II still functions as a computer; i.e. its "brain" is still connected. In the Terminal Mode, the APPLE II simulates a computer terminal, its brain in effect disconnected. The Terminal Mode can be

entered and exited by a number of commands, either entered on the APPLE's keyboard or sent by the external device.

3.2.2.1 Keyboard Commands

The Terminal Commands entered from the keyboard are always prefixed by ^A. They can be used only after an IN#n command and are the only keyboard commands, besides Reset and ^C, that the Apple II will recognize after IN#n has been entered.

Full-Duplex Command (^A^F)

The Full-Duplex command allows your Apple II to operate as a full-duplex terminal. In the full duplex mode, when a character is entered on the keyboard, it is sent to the external device that the terminal is communicating with. Unless the external device sends it back (echoes), the character does not appear on the screen. The full-duplex mode with echoing is often preferred since it confirms the communication's reliability. However, it requires that the receiving device has the capability of echoing the character. Enter ^A^F to enable the full-duplex mode.

Half-Duplex Command (^A^H)

The Half-Duplex command causes your APPLE II to mimic a half-duplex terminal. In the half-duplex mode, a character entered on the keyboard is sent to both the APPLE's screen and the external device. Thus the user sees exactly

what he has entered. If you use the half-duplex mode with a receiving device designed for full-duplex, every character on the screen will be ddoouubblleedd, since the receiving device will be echoing the characters it receives. To enable the half-duplex mode, enter ^A^H.

Break Command (^A^S)

Many terminals in time-sharing systems have a BREAK key; the APPLE II does not. This command allows the APPLE II to send out the BREAK signal. It is needed only when communicating with an external device that requires the BREAK signal. Turn on the BREAK signal by entering ^A^S. Turn it off by pressing any key other than Control or Shift.

Exit Command (^A^X)

To exit from the terminal mode, enter ^A^X. A backslash will appear on the screen. If a PR# command was in effect, it does not change. Input, however, will now be taken from the APPLE II's keyboard.

Reset and ^C would also return the APPLE to normal operations; however, any program running would be terminated. After Reset, both PR# and IN# are set to their default values, IN#0 and PR#0.

3.2.2.2 Commands from External Devices

Remote Mode (^R)

When the APPLE II is operating in the terminal mode, an external device can connect and disconnect the APPLE's brain. If the APPLE receives ^R while in the terminal mode, it operates in a Remote Mode to the external device. Any input coming from the external device goes to the APPLE's brain. Thus the external device can control the APPLE II as if its instructions had come from the APPLE II's keyboard. If the PR#n command is in effect (where n = the 7710's slot number) the APPLE will echo what it receives from the external device.

Terminal Mode (^T)

After ^R has been issued, ^T returns the APPLE to the terminal mode. In effect, it cancels the Remote Mode command and reinstates the APPLE as a half-duplex or full-duplex terminal. If the 7710 has been accessed by an IN#n command, but the terminal mode has not yet been invoked, ^T sent by an external device turns the APPLE into a half-duplex terminal.

3.2.3 CHANGING 7710D PARAMETERS

Some of the 7710D firmware parameters (for example, serial data format) can be changed by BASIC POKE commands or monitor commands. To change the parameters, you must first initialize the 7710

board by using the IN#n or PR#n commands or their monitor equivalents, n^P or n^K. Any subsquent invoking of the IN#n or PR#n commands or their monitor equivalents will reinitialize the 7710, causing the parameters to return to their default values. Note that the 7610D firmware does NOT support the standard 7610C parameter for characters-per-line.

3.2.3.1 Serial Data Format

Changing this parameter for the 7710D firmware is exactly the same as changing it for the 7710C firmware; see Section 3.3.1.3.

3.2.3.2 Lower-Case Conversion

The APPLE II converts lower-case characters received from the external device into upper case characters. To stop the APPLE from doing so, zero out the contents of location $638+n. (From BASIC, POKE 1592+n,0.) If you load this location with $A0 instead (POKE 1592+n,160), lower-case characters are printed as upper-case, reverse video characters.

3.2.3.3 Output to the APPLE's Screen

The 7710 will echo output characters to the APPLE II's screen only if Bit 7 of $6F8+n is 0. From BASIC, POKE 1784+n,128 to disable screen echo; POKE 1784+n,0 to disable screen echo.

CHAPTER 4

PROGRAMMING INFORMATION

This chapter contains the information you will need if you plan to write your own assembly-language driver for the 7710 interface. Included are discussions of the 6850 Asynchronous Communication Interface Adapter (ACIA) registers, register addressing, and general driver requirements. Instructions are also provided for saving the driver on disk and loading it into RAM. See Section 2.1.2 for instructions on replacing the ROMs with RAMs.

4.1 ACIA REGISTERS

Your computer dedicates 16 memory addresses to each peripheral connector slot 1-7 for memory-mapped input/output. These 16 addresses are above and beyond the 256 dedicated program memory addresses. The I/O addresses are located at $C0xy, where x = 8 + n, n = the peripheral slot

number (1-7), and y = $1-F. On the 7710 the I/O addresses are occupied by the ACIA registers as follows:

$C0x0 (WR) = ACIA command register
$C0x0 (RD) = ACIA status register
$C0x1 (WR) = ACIA transmit register
$C0x1 (RD) = ACIA receive register

NOTE: The last address digit is not decoded beyond even or odd. This means that data can be passed through any odd address within the range, while the ACIA's command/status registers can be accessed at any even address.

4.1.1 ACIA COMMAND REGISTER

ACIA operation is controlled by one-byte commands written to the Command Register. The commands are defined in Table 4.1. Because the baud rate generator outputs a clock 16 times the baud rate, bits 1 and 0 should be set to 01.

4.1.2 ACIA STATUS REGISTER

The status bits, when set, have the meanings given in Table 4.2.

TABLE 4.1. ACIA COMMANDS

7654 3210	
xxxx xx00	The clock is 1x the baud rate
xxxx xx01	The clock is 16x the baud rate
xxxx xx10	The clock is 64x the baud rate
xxxx xx11	ACIA Master Reset
xxx0 00xx	7 data + Even parity + 2 stop bits
xxx0 01xx	7 data + Odd parity + 2 stop bits
xxx0 10xx	7 data + Even parity + 1 stop bit
xxx0 11xx	7 data + Odd parity + 1 stop bit
xxx1 00xx	8 data + No parity + 2 stop bits
xxx1 01xx	8 data + No parity + 1 stop bit
xxx1 10xx	8 data + Even parity + 1 stop bit
xxx1 11xx	8 data + Odd parity + 1 stop bit
x00x xxxx	Set RS-232-C CTS Disable transmit interrupts
x01x xxxx	Set RS-232-C CTS Enable transmit interrupts
x10x xxxx	Clear RS-232-C CTS Disable transmit interrupts
x11x xxxx	Set RS-232-C CTS Transmit break on transmit data Disable transmit interrupts
0xxx xxxx	Disable Receive Interrupts
1xxx xxxx	Enable Receive Interrupts when: Receiver data register full Receive data overrun DTR signal inactive

TABLE 4.2. ACIA STATUS

Bit 0	Receive data is ready for the computer.
Bit 1	The transmit register can accept data.
Bit 2	RS-232-C DTR inactive; don't send.
Bit 3	RS-232-C RTS inactive; don't send.
Bit 4	Received data improperly framed.
Bit 5	Data received before previous byte read.
Bit 6	Parity Error in received data.
Bit 7	ACIA-generated transmit or receive interrupt.

4.2 INPUT AND OUTPUT ROUTINES

Your computer looks at two page 0 locations to find out where the current keyboard handler and console output driver programs are located. The addresses are:

$36 - $37: console output handler
$38 - $39: keyboard input handler

When you type in the BASIC command IN#n, the firmware writes a $00 in location $38 and a $Cn in location $39; the equivalent monitor command, n^K, does the same thing. This creates an effective address of $CnOO for the keyboard handler initialization program. The next time keyboard input is wanted, the initialization routine gets called. It must set everything up, then pass control to the input routine to actually do the input. One of the initializer's tasks is to change location $38 to identify the input handler's correct entry point. Then, the next time input is wanted, we can go straight to the input routine. Likewise, when PR#n is typed in, location $36 is set to $00 and $37 is set to $Cn. On the first output, control will be passed to $CnOO for output initialization. Location $36 should then be set to match the output handler's entry point for all subsequent console output.

Be aware that if any peripheral control options are allowed in the program, invoking an IN#n or PR#n command will cause the default options to be selected for both input and output unless the options are initialized after the input-or-output initialization decision is made.

Input and output by your computer are handled

on a byte-by-byte basis. As a result, data can be passed between the I/O routines and the computer in the Accumulator (A Register). Input data should be left in the Accumulator when control is returned to the caller. The output routine can find its data in the Accumulator.

The input and output routines should be called as subroutines. Control can be returned to the caller with a simple RTS (Return from Subroutine) instruction. All register contents should be saved on entry to subroutines, then restored to their original contents just before leaving (except for parameter-passing registers). This practice saves headaches and program-debugging time later on.

Several scratchpad memory locations are available. The video-display-refresh memory locations (addresses $400-$7FF) use only the first 120 of every 128 locations for the display data. The remaining 24 addresses can be used for other purposes. Two sets of available locations are $6Fx and $77x, where x = 8 + slot number. For most programs, these locations should be sufficient. But one other scratch location merits identification. Address $7F8 is often used to hold the page address of the currently-active peripheral. The page address is $Cn, where n is the slot number.

You now have enough information to program a simple remote console interface driver. Your program should consist of three parts: initialization, input, and output. For initialization, you must:

a. Save the registers;

b. Reset the ACIA;

c. Initialize the ACIA with the default command word;

d. Establish input and/or output entry points;

e. Allocate and/or initialize any special pointers, counters, etc;

f. Go to Step b of the appropriate I/O routine.

For input, you must:

a. Save the registers;

b. Check the ACIA status and wait until the input data is ready;

c. Read the input data;

d. Do any special data conversion as needed--e.g., lower-to-upper case;

e. Restore the registers;

f. Return to the caller.

For output, you must:

a. Save the registers;

b. Do any special preprint control (tabs, form feeds, etc.);

c. Wait until the ACIA can take more data;

d. Write the data to the ACIA;

e. Do any postprint control (line or page control, insert a line feed after a carriage return, etc.);

f. Restore the registers;

g. Return to the caller.

As you can see, several tasks are common to all the routines. To stretch 256 bytes of space as far as possible, you should make as much code as possible common. Unless your driver is slot-dependent you can't predict what absolute addresses will contain the common code, so you can't use subroutine calls, but must use relative code throughout. Unused status flags can be used to indicate the entry point. The standard driver uses the V (Overflow) flag to indicate initializing or not, and the C (Carry) flag to indicate input or output. This allows you to use common segments of code, then branch to the task-unique code according to the state of the relevant flag. After the flags have served their purposes, they can be reused to indicate a tab in progress or other function.

The program listing for the 7610C firmware in Appendix B provides examples of the handling of specific driver tasks. We encourage you to use as much of the code as suits your application.

4.3 LOADING YOUR DRIVER INTO RAM

If you have installed RAM, it must be loaded every time you turn your computer on and want to use the interface. The following procedure was devised for floppy disk systems; if you use some other storage medium, you'll need to devise your own scheme.

4.3.1 SAVING THE DRIVER ON DISK

The first chore is to get the interface software initially into memory. The firmware miniassembler works nicely for this; see your Red Book for details on how to use it. Assemble the driver directly into the interface's memory. For instance, if the interface is in slot #1, use address $C100 as the base address. After you have assembled your driver into memory, save a copy of it on disk. To do this, first move a copy down into the "lower 32". Your disk can only load and save programs from the lower 32K of memory. Location $A00 is a good spot for the copy; it will not interfere with the Integer BASIC or the Disk Operating System (DOS). This Monitor command performs the move nicely:

*A00<C100.C1FFM<cr>

Next, transfer control over to >BASIC under the DOS. If the DOS is already in memory, just type in:

*3D0G

Otherwise, do a disk boot:

*6^P<cr>

Finally, you are ready to actually save the driver on disk:

>BSAVE ASYN1.0,A$A00,L$100<cr>

Your driver software is now saved on your disk with a file name of ASYN1.0, and you are ready to check out the driver and modify it as necessary. Don't forget to save a copy after each modification. There's nothing more frustrating than to try to check out a routine only to have it bomb out and erase itself in the process. When you're happy with the routine, save it one last time.

4.3.2 POWER-ON LOADING OF THE DRIVER

Two methods of loading the software from disk are outlined here: a) using direct commands, and b) under program control.

a. Direct Commands:

To load the driver with direct commands, perform the following sequence:

1. Boot in the DOS:

6^P<cr>

2. Read in the driver file:

>BLOAD ASYN1.0<cr>

3. Return control to the Monitor:

>CALL-155<cr>

4. Finally, upload the driver into the interface's RAM (n = the slot number):

*CnOO<A00.AFFM<cr>

b. Loading Under Program Control:

This alternate method combines steps 2 and 4 above into one automated step. A simple >BASIC program to perform this is:

```
10 INPUT "ASYNC INTERFACE SLOT IS: ",S
20 IF S<1 OR S>7 THEN GOTO 10
30 DEST = -16384 + 256 * S
40 PRINT CHR$(4);"BLOAD ASYN1.0,A$A00"
50 FOR I=0 TO 255
60 POKE DEST + I,PEEK(2560 + I)
70 NEXT I
80 END
```

Assuming that this program has been saved on disk under the file name of ASYN, all you have to do now is:

1. Boot in the DOS:

*6^P<cr>

2. Execute the ASYN program:

>RUN ASYN<cr>

3. Answer the question that appears:

ASYNC INTERFACE SLOT IS: ?2<cr>

CHAPTER 5

HARDWARE DESIGN

The 7710 card hardware can be divided into four sections: a) the transmitter/receiver section; b) the baud rate generator; c) the control section; and d) the program memory. Each section is discussed separately in the following paragraphs.

5.1 TRANSMITTER/RECEIVER SECTION

The major component of this section and the heart of the 7710 is the 6850 Asynchronous Communications Interface Adapter, or ACIA. This device performs the parallel-to-serial and serial-to-parallel data conversion, adds start and stop bits when transmitting and removes them when receiving, makes available status information, and controls handshaking with the peripheral or modem. A programmable control register allows specification of word length, number of stop bits, parity type or inhibition, and clock division

ratios, besides initiating transmitter and receiver cycles and enabling or disabling interrupts. A status register provides operation and error status bits as detailed in Table 4.2.

On the serial side, the ACIA provides three handshake lines for peripherals/modems. One of these lines, -DATA CARRIER DETECT (-DCD), allows the peripheral to control the ACIA receiver section and initiate an interrupt when the peripheral is not ready, in effect allowing the peripheral to tell the computer to slow down. The -DCD input is tied to the inverted Data Terminal Ready (DTR) signal (pin 20) of the RS-232-C connector. The ACIA thus will stop transmitting when DTR goes low. When DTR goes high, the ACIA will resume transmitting.

The other two ACIA handshake signals are -Request to Send (-RTS) and -Clear to Send (-CTS). These signals correspond to the primary RS-232-C handshake lines of the same names; however, the ACIA signals were named for the ACIA used as a DTE device; on the 7710 it is used as a DCE device. Thus ACIA output -RTS is inverted to control RS-232-C line CTS, a high on which tells the peripheral that the ACIA is ready to transmit, while RS-232-C line RTS inverted controls ACIA input -CTS; RTS active indicates that the peripheral is ready to receive data from the ACIA.

A 75154 and a 75150 perform the line receiver and line driver functions respectively, translating the standard Transistor-Transistor Logic (TTL) signals of the ACIA to the signal levels required by RS-232-C. They also provide for fail-safe operation should your interface cable short or disconnect accidentally.

5.2 BAUD RATE GENERATOR

The ACIA needs a clock to tell it how often to send or sample signals. Different serial devices require data to be clocked at different rates. The 7710 employs a 4702 baud rate generator to supply standard baud rates from 50 to 19200. This chip contains an oscillator for the on-board quartz crystal and a controllable frequency divider circuit. The oscillator generates a highly stable 2.4576 MHz square wave signal. This signal is counted down by a divisor determined by the four switches on the card. The output of the 4702, which controls the ACIA Transmitter and Receiver Clock pins, is 16 times the actual selected baud rate; thus the ACIA must be programmed to expect a x16 clock.

The select inputs to the 4702 provide 16 four-digit codes. The 4702, by itself, generates 13 different baud rates, from 50 to 9600, which account for 13 of the codes. One of the three remaining codes is a second code to select 2400 baud. Another is used to select 19,200 baud. Normally, the 4702 prescales the 2.4576 MHz signal by dividing it first by 16 to yield a 9600 (x16) signal. The prescaler, however, allows connection to the signal after it has been divided by 8; thus a 19,200 baud signal is available. The last code is used on the 7710 to select a signal from pin 24 of the RS-232-C connector, allowing the peripheral (or another source) to generate the baud rate.

5.3 CONTROL SECTION

The control section of the interface consists primarily of a buffer between the ACIA and the APPLE data bus, ACIA control logic, and interrupt arbitration logic. The ACIA has only limited ability to drive the data lines of the computer. An 8304B bi-directional line buffer (the 8304B) between the computer and the ACIA ensures successful data transfer. To reduce power comsumption by this device, a power-down feature has been included. When -DEVICE SELECT goes low, a transistor turns on power to the 8304B, allowing data transfer. When -DEVICE SELECT goes high, the transistor turns off power to the 8304B. R/-W determines the direction of data transfer through the 8304B.

The ACIA control logic is relatively straight-forward. When -DEV SEL is low, the ACIA is selected. Address bit A0 determines whether a data or command/status register is accessed, while R/-W determines whether a read (receiver or status) register or a write (transmitter or command) register is accessed.

The interrupt arbitration logic is one link in the interrupt daisy chain, which prioritizes peripheral-generated interrupts to ensure that only one device interrupts at a time. If no higher-priority interrupt is in progress (pin 28, INT IN, high), a low from ACIA output -IRQ will force the INT OUT line low, telling the lower-priority devices that an interrupt is pending, forcing them to wait, and will force pin 30, -IRQ, low. After being serviced, the ACIA removes its interrupt request, and the arbitration logic allows INT OUT to go high again.

Remember that the device in the leftmost slot in the group 1 through 7 has the highest interrupt piority; slot 0 does not support the daisy chain. Also keep in mind that empty slots between cards break the daisy chain, and only cards to the left of the empty slot(s) will be part of the chain.

5.4 PROGRAM MEMORY

Your computer dedicates 256 bytes of memory space to each peripheral connector, addressable at $Cn00-CnFF where n is the slot number. The high byte of the address activates the -IO SEL line for the appropriate slot; the low byte selects the actual location on the on-board memory. Your 7710 includes a driver loaded on two 256 x 4 ROMs enabled when -IO SEL is low and R/-W is high. Because of the relatively high power consumption of the ROMs, a power-down circuit has been included to remove power from the ROMs when -IO SEL is inactive.

Should you desire to develop your own software, you may substitute 2112 static RAMs for the ROMs. Using RAMs allows full development and testing of a program before it is committed to ROMs. If RAMs are installed, a jumper wire must be soldered between the pads labeled RAM to disable the power-down feature, which would otherwise destroy the RAM contents as fast as they could be written in. Pin 14 of the ROM/RAM is controlled by R/-W, providing the necessary RAM input as well as deselecting the ROM during write operations.

APPENDIX A

TECHNICAL INFORMATION

A.1 USER-REPLACEABLE PARTS

QTY	REF	CCS P/N	DESCRIPTION
5	C1-4,7	42034-21046	CAPACITOR, MONOLYTHIC .1uf, 50vdc
2	C5,6	42215-55605	CAPACITOR, MICA 56pf, 500vdc, 10%
1	J2	56004-02013	HEADER, DUAL 13 PIN, STRAIGHT ENTRY
2	Q1,2	36100-02907	TRANSISTOR, SI; PN2907 GENERAL PURPOSE
1	R1	40002-01005	RESISTOR, FIXED, COMP 10 ohm, 1/4W, 10%
1	R2	40002-06815	RESISTOR, FIXED, COMP 680 ohm, 1/4W, 10%
1	R3	40003-01015	RESISTOR, FIXED, COMP 100 ohm, 1/2W, 10%
4	R4-7	40002-02215	RESISTOR, FIXED, COMP 220 ohm, 1/4W, 10%
1	R8	40002-02725	RESISTOR, FIXED, COMP 2.7K, 1/4 W, 10%
1	R9	40002-01055	RESISTOR, FIXED, COMP 1M, 1/4W, 10%
1	U1	30300-00150	IC, INTERFACE; 75150 DUAL RS-232-C DRIVER
1	U2	31100-06850	IC, DIGITAL, MOS; 6850 ACIA
1	U3	30300-00154	IC, INTERFACE; 75154 QUAD RS-232 RCVR
1	U4	31000-04702	IC, DIGITAL, CMOS; 4702 BAUD RATE GENERATOR
1	U7	30900-08304	IC, DIGITAL, TTL; 8304B OCTAL BUS DRVR/RCVR
1	U8	30000-00009	IC, DIGITAL, TTL; 74LS09 QUAD 2-IN AND, OC

QTY	REF	CCS P/N	DESCRIPTION
1	U9	30000-00136	IC, DIGITAL, TTL; 74LS136 QUAD 2-IN EX-OR, OC
1	U10	30000-00003	IC, DIGITAL, TTL; 74LS03 QUAD 2-IN NAND, OC
1	Y1	48132-45762	XTAL, QUARTZ 2.4576MHz, HC-6
1	XU1	58102-00080	SOCKET, IC; LOW PROFILE 8 PIN DIP
3	XU8-10	58102-00140	SOCKET, IC; LOW PROFILE 14 PIN DIP
2	XU3-6	58102-00160	SOCKET, IC; LOW PROFILE 16 PIN DIP
1	XU7	58102-00200	SOCKET, IC; LOW PROFILE 20 PIN DIP
1	XU2	58102-00240	SOCKET, IC; LOW PROFILE 24 PIN DIP
1		50026-01051	CABLE, 9" I/O, RS-232-C CONNECTOR
1	S1	27111-41010	SWITCH, DIP, 4PST

A.2 RS-232-C CONNECTOR PINOUTS

FRONT VIEW

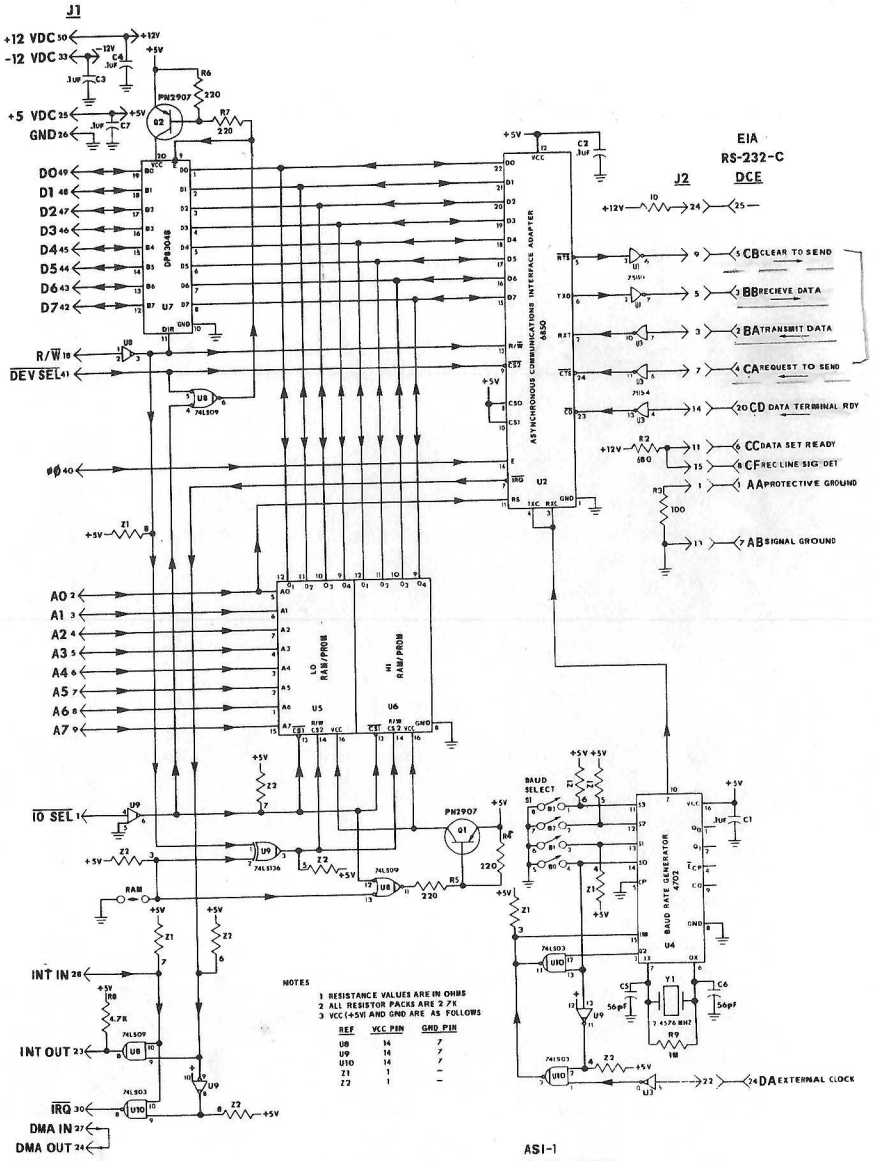
PROTECTIVE GROUND	AA	1 •			—	UNASSIGNED
TRANSMIT DATA	BA	2 •	×	14	SBA	SEC TRANSMIT DATA
RECEIVE DATA	BB	3 •	×	15	DB	TRANSMIT SIG ELE CLK (DCE)
REQUEST TO SEND	CA	4 •	×	16	SBB	SEC RECEIVE DATA
CLEAR TO SEND	CB	5 •	×	17	DD	RECEIVE SIG ELE CLK (DCE)
DATA SET READY	CC	6 ○		18	—	UNASSIGNED
SIGNAL GROUND	AB	7 •	×	19	SCA	SEC REQUEST TO SEND
REC LINE SIG DET	CF	8 ○	•	20	CD	DATA TERMINAL READY
RESERVED	—	9	×	21	CG	SIGNAL QUALITY DETECTOR
RESERVED	—	10	×	22	CE	RING INDICATOR
UNASSIGNED	—	11	×	23	CH/CI	DATA RATE SELECT
SEC REC LINE SIG DET	SCF	12 ×	•	24	DA	TRANSMIT SIG ELE CLK (DTE)
SEC CLEAR TO SEND	SCB	13 ×		25	—	UNASSIGNED

DB-25S (FEMALE)

7710
• FULL/ACTIVE SUPPORT
○ PASSIVE SUPPORT
× NOT SUPPORTED

EIA RS-232C
DCE TYPE CONNECTOR PIN ASSIGNMENT

A.3 SCHEMATIC/LOGIC DIAGRAM



ASI-1
MODEL 7710A
ASYNCHRONOUS SERIAL INTERFACE

A.4 DEFINITION OF RS-232-C CONFIGURATIONS

A	Transmit Only
B	Transmit Only*
C	Receive Only
D	Half Duplex; or Duplex*
→ E	Full Duplex
F	Primary Channel Transmit Only* /Secondary Channel Receive Only
G	Primary Channel Receive Only / Secondary Channel Transmit Only*
H	Primary Channel Transmit Only / Secondary Channel Receive Only
I	Primary Channel Receive Only / Secondary Channel Transmit Only
J	Primary Channel Transmit Only* / Half Duplex Secondary Channel
K	Primary Channel Receive Only / Half Duplex Secondary Channel
L	Half Duplex Primary Channel / Half Duplex Secondary Channel; or
L	Duplex Primary Channel* / Duplex Secondary Channel*
M	Duplex Primary Channel / Duplex Secondary Channel
Z	Special (Circuits specified by supplier)

*Note the inclusion of Request to Send in a Transmit Function, where it would not ordinarily be expected, but could indicate a non-transmit mode to the data communications equipment (DCE) to permit it to remove a line signal or to send synchronizing or framing signals as required.

A.5 RS-232-C STANDARD CONFIGURATIONS

	Interchange Circuit		Interface Configuration												
			A	B	C	D	E	F	G	H	I	J	K	L	M	Z
1	AA	Protective Ground	-	-	-	-	-	-	-	-	-	-	-	-	-	-
7	AB	Signal Ground / Common Return	X	X	X	X	X	X	X	X	X	X	X	X	X	X
2	BA	Transmitted Data	X	X		X	X	X		X		X		X	X	o
3	BB	Received Data			X	X	X		X		X		X	X	X	o
4	CA	Request to Send		X		X		X				X		X		o
5	CB	Clear to Send	X	X		X	X	X		X		X		X	X	o
6	CC	Data Set Ready	X	X	X	X	X	X	X	X	X	X	X	X	X	o
20	CD	Data Terminal Ready	S	S	S	S	S	S	S	S	S	S	S	S	S	o
22	CE	Ring Indicator	s	s	s	s	s	s	s	s	s	s	s	s	s	o
8	CF	Received Line Signal Detector			X	X	X		X		X		X	X	X	o
21	CG	Signal Quality Detector														o
23	CH/CI	Data Signalling Rate Selector (DTE/DCE)														o
15 24	DA/DB	Transmit Signal Element Timing (DTE/DCE)	T	T		T	T	T		T		T	T	T	T	o
17	DD	Receiver Signal Element Timing (DCE)			+	+	+		+		+		+	+	+	o
14	SBA	Secondary Transmitted Data							x		x	x	x	x	x	o
16	SBB	Secondary Received Data						x		x		x	x	x	x	o
19	SCA	Secondary Request to Send							x			x	x	x		o
13	SCB	Secondary Clear to Send							x		x	x	x	x	x	o
12	SCF	Secondary Received Line Signal Detector						x		x		x	x	x	x	o

Notes: Upper case indicates a line supported by the CCS 7710.
Lower case indicates a line not supported by the CCS 7710.
X = Basic interchange circuits, all systems
T = Additional interchange circuits required for synchronous channel
S = Additional interchange circuits required for switched service
O = Specified by supplier as required
- = Optional; supported by the CCS 7710

APPENDIX B

FIRMWARE LISTINGS

```
0000                ;
0000                ; DRIVER PROGRAM FOR THE
0000                ; CCS 7710A ASYNCHRONOUS SERIAL INTERFACE
0000                ;
0000                ; COPYRIGHT 1979 CALIFORNIA COMPUTER SYSTEMS
0000                ;
0000                ;This program contains the necessary code to allow direct
0000                ;logical replacement of your computer's keyboard and TV
0000                ;output.  Output defaults to 255 characters per line and
0000                ;no paging.  Three entry points are defined: initialization,
0000                ;input, and output.  When the > commands IN#n or PR#n are
0000                ;issued, the computer sets up a jump vector address to the
0000                ;initialization entry point.  The initialization routine
0000                ;will adjust the input or output entry point vector to the
0000                ;correct address when the first input or output occurs.
0000                ;
0000                ; SYSTEM EQUATES
0000                        ORG     0
0000                MAXCHR  EQU     $FF         ;255 characters per line default
0000                BKSP    EQU     $87         ;ASCII Back Space - 1 (for carry) Control H
0000                LNFD    EQU     $8A         ;ASCII Line Feed - Control J
0000                FF      EQU     $8C         ;ASCII Form Feed
0000                CARRET  EQU     $8D         ;ASCII Carriage Return
0000                FS      EQU     $95         ;Forward Space - Control U
0000                SPACE   EQU     $A0
```



```
0000                CH      EQU     $24         ;Tab column pointer ;24 lignes
0000                CSWL    EQU     $36         ;Location of output driver vector
0000                KSWL    EQU     $38         ;Location of input driver vector
0000                KSWH    EQU     $39
0000                CPL     EQU     $5F8-$C0    ;MAXCHR SAVE LOCATION
0000                LOCASE  EQU     $6F8-$C0    ;LC conversion mask hold
0000                LNCNT   EQU     $7F8-$C0    ;Line counter
0000                CHCNT   EQU     $778-$C0    ;Character counter
0000                CMD     EQU     $C080       ;+ $n0 for the ACIA Command port
0000                STATUS  EQU     CMD         ;This is the ACIA Status Port
0000                DATA    EQU     CMD + 1     ;This is the ACIA Data Port
0000                WAIT    EQU     #FCA8       ;Address of wait routine
0000                RETURN  EQU     $FFCB       ;Used to find the slot address
0000                ;
0000                ;The common code
0000                ;
0000 2C CB FF       INIT    BIT     RETURN      ;Set V = 1
0003 70 04                  BVS     COM
0005 18             OUTEP   CLC                 ;Clear the carry for output
0006 B0                     DFB     $B0         ;Make a BCS to skip next instruction
0007 38             INEP    SEC                 ;Set the carry for input
0008 B8                     CLV                 ;Clear the V Flag for I/O
0009 48             COM     PHA                 ;Save the registers and status
000A 8A                     TXA
000B 48                     PHA
```

```
000C 98                 TYA
000D 48                 PHA
000E 08                 PHP
000F 78                 SEI             ;Disable interrupts
0010 20 CB FF           JSR     RETURN  ;Put slot address on the stack
0013 BA                 TSX             ;Put the Slot Page
0014 BC 00 01           LDY     $100,X  ;   Number into Y
0017 68                 PLA             ;Recover the output data (if any)
0018 68                 PLA             ;
0019 68                 PLA
001A 68                 PLA
001B 9A                 TXS             ;Restore the Stack Pointer
001C 48                 PHA             ;Save the data in the stack top
001D 98                 TYA             ;Get the Slot Page Number
001E AA                 TAX             ;Establish X Index
001F 0A                 ASL     A       ;Multiply by 16 to get the
0020 0A                 ASL     A       ;  $n0 index to access the ACIA
0021 0A                 ASL     A
0022 0A                 ASL     A
0023 A8                 TAY             ;Establish Y Index
0024 68         COMA    PLA             ;Get the saved status codes
0025 28                 PLP
0026 48                 PHA
0027 50 1B              BVC     IO      ;Routine IO branch
0029
```



```
0029            ; Initialization Routine
0029            ;
0029            ;This code handles the first input or the first output request
0029            ;after the IN#n or the PR#n command.  It initializes the ACIA,
0029            ;sets up the lower case conversion  mask  and  the entry point
0029            ;vector, and finally goes to the appropriate input  or  output
0029            ;routine.
0029            ;
0029 A9 23              LDA     #$23    ;Reset the ACIA
002B 99 80 C0           STA     CMD,Y
002E A9 11              LDA     #$11    ;Set for 8 + 2, No parity, No int
0030 99 80 C0           STA     CMD,Y
0033 B8                 CLV             ;Clear the initialization flag
0034 A5 38              LDA     KSWL    ;See if input is wanted
0036 D0 29              BNE     OINIT   ;No, branch for output initialization
0038 E4 39              CPX     KSWH    ;Maybe, make sure
003A D0 25              BNE     OINIT   ;No, branch for output
003C 9D 38 06           STA     LOCASE,X ;Set no lower case conversion mask
003F A9 07              LDA     #7      ;Set normal input entry point vector
0041 85 38              STA     KSWL
0043 38                 SEC             ;Fall through next branch
0044 90 31      IO      BCC     OUT     ;Go to normal output
0046            ;
```

```
0046                ; Input Routine
0046                ;
0046                ;This routine expects no parameters  from  calling  routines.
0046                ;It waits until data has been typed in from the keyboard, and
0046                ;then returns that data in the accumulator to the caller.  It
0046                ;ignores all  line feeds  and converts all lower case letters
0046                ;to upper case (if the locase mask = #$20).
0046                ;
0046 B9 80 C0       INPUT   LDA     STATUS,Y  ;Get ACIA status
0049 4A                     LSR     A         ;Isolate Rx Rdy bit
004A 90 FA                  BCC     INPUT     ;Loop until data is ready
004C 68                     PLA               ;Get rid of data on stack top
004D B9 81 C0               LDA     DATA,Y    ;Read the new data
0050 09 80                  ORA     #$80      ;Set Bit 7 for normal video
0052 48                     PHA               ;Save data on stack
0053 C9 8A                  CMP     $LNFD     ;Ignore Line Feeds
0055 F0 EF                  BEQ     INPUT
0057 68                     PLA
0058 C9 E0                  CMP     #$E0      ;See if it is lower case
005A 90 74                  BCC     DONA      ;Go finish up if not
005C 5D 38 06               EOR     LOCASE,X  ;Convert to upper if mask = $20
005F B0 6F                  BCS     DONA      ;Now go finish up
0061                        ;
0061                        ;Output Initialization
0061                        ;
```



```
0061 A9 05          OINIT   LDA     #5        ;Set normal output entry point vector
0063 85 36                  STA     CSWL
0065 A9 FF                  LDA     #MAXCHR   ;Set defaults
0067 9D 38 05               STA     CPL,X
006A A9 00                  LDA     #0
006C 9D B8 06               STA     CHCNT,X
006F                ;
006F                ; Output Routine
006F                ;
006F                ;This routine does the actual output of the data.  It expects
006F                ;to find the data for output  on the top of the stack  (where
006F                ;the common code put it).
006F                ;
006F BD B8 06       OUT     LDA     CHCNT,X   ;See if tab wanted
0072 C5 24                  CMP     CH
0074 B0 03                  BCS     OUTA      ;Branch if no tab
0076 A9 A0                  LDA     #SPACE    ;Space out to column
0078 48                     PHA               ;Save on stack
0079 68             OUTA    PLA               ;Retrieve character
007A 48             OUTD    PHA               ;Resave it
007B 08                     PHP               ;Save Tab Status
007C C9 95          CRTLU   CMP     #FS       ;Check for ^U
007E F0 04                  BEQ     CHCTI     ;Branch if so
0080 29 60                  AND     #$60      ;Test for other cntrl char
0082 F0 03                  BEQ     OUTB      ;Skip counter increment
```

```
0084 FE B8 06   CHCTI   INC     CHCNT,X   ;Bump counter
009D 28         OUTB    PLP               ;Reget tab flag
009E B9 80 C0   OUTC    LDA     STATUS,Y  ;Get ACIA status
008B 29 03              AND     #3        ;Isolate Tx buffer bit
008D F0 F9              BEQ     OUTC      ;Wait if not ready
008F 29 01              AND     #1        ;Check for input char
0091 D0 46              BNE     INCHR     ;Get char if there is one
0093 68                 PLA               ;Get data for output
0094 99 81 C0           STA     DATA,Y    ;Output it
0097 90 D6              BCC     OUT       ;Branch if tab
0099 70 01              BVS     LF        ;Branch if self generating character
009B 48                 PHA               ;Resave character
009C C9 8A      LF      CMP     #LNFD     ;See if line feed
009E F0 09              BEQ     DONE
00A0 E9 87      BS      SBC     #BKSP     ;See if back space
00A2 D0 13              BNE     CR        ;No, branch
00A4 DE B8 06           DEC     CHCNT,X   ;Yes, adjust counter
00A7 30 12              BMI     CRA       ;Disallow negative count
00A9            ;
00A9            ; Final Common Code
00A9            ;
00A9            ;This part of the code restores the registers and returns to
00A9            ;the caller.  It is used by all of these routines.
00A9            ;
00A9 68         DONE    PLA               ;Set the stack straight
```



```
00AA BA         DONA    TSX               ;Modify A register value in
00AB E8                 INX               ;  Stack to insure it is
00AC E8                 INX               ;  Restored to the right value
00AD E8                 INX
00AE 9D 00 01           STA     $100,X
00B1 68                 PLA               ;Restore registers
00B2 A8                 TAY
00B3 68                 PLA
00B4 AA                 TAX
00B5 68                 PLA
00B6 60                 RTS               ;Done!
00B7            ;
00D7            ; Rest of output code
00B7            ;
00B7 E9 06      CR      SBC     #6        ;See if a Carriage Return
00B9 D0 12              BNE     AUTOCR    ;No, branch
00BB 9D B8 06   CRA     STA     CHCNT,X   ;Zero out counter
00BE 85 24              STA     CH        ;  and tab pointer
00C0 A9 C0              LDA     #$C0      ;Wait for print head to return
00C2 20 A8 FC           JSR     WAIT      ;
00C5 A9 8A      MAKELF  LDA     #LNFD     ;Make a line feed
00C7 2C CB FF   SELF    BIT     RETURN    ;V=1 for self generating character
00CA 38                 SEC               ;C=1 for no tab
00CB B0 AD              BCS     OUTD      ;Always branch
00CD BD B8 06   AUTOCR  LDA     CHCNT,X   ;End of line yet?
```

```
00D0 DD 38 05            CMP     CPL,X
00D3 90 D4               BCC     DONE        ;No, done
00D5 A9 8D               LDA     #CARRET     ;Yes, get a Carriage Return
00D7 D0 EE               BNE     SELF        ;Process it
00D9                ;
00D9                ; Cntrl S Interrupt Code
00D9                ;
00D9                ;This routine processes the cntrl S (DC3) to interrupt
00D9                ;sending.  Any character will cause sending to resume.
00D9                ;
00D9 08             INCHR   PHP             ;Save psw on stack
00DA B9 81 C0            LDA     DATA,Y     ;Get input char
00DD 29 7F               AND     #$7F       ;Strip msb
00DF C9 13               CMP     #$13       ;Check for cntrl S
00E1 F0 03               BEQ     WAIT1      ;If so, wait for next char
00E3 28                  PLP                ;If not, clear stack
00E4 D0 A2               BNE     OUTC       ;   and go send char
00E6 B9 80 C0       WAIT1   LDA     STATUS,Y   ;Wait for input char
00E9 29 01               AND     #$1        ;Get input buffer status
00EB F0 F9               BEQ     WAIT1      ;Wait for input char
00ED B9 81 C0            LDA     DATA,Y     ;Read data to clear flag
00F0 28                  PLP                ;Get back old psw
00F1 D0 95               BNE     OUTC       ;Go send next char
```



```
 1 ************************************
 2 *                                  *
 3 *   7610D FIRMWARE DRIVER FOR THE  *
 4 *   CCS  7710 ASYNCHRONOUS SERIAL  *
 5 *            INTERFACE             *
 6 *                                  *
 7 *       WRITTEN BY FRED VILES      *
 8 *                                  *
 9 ************************************
10 *
11 *
12 *                                  ;ACIA port addresses
13 STATUS       EQU     $C080
14 CONTRL       EQU     $C080
15 DATA         EQU     $C081
16 *                                  ; apple equates
17 RNDL         EQU     $4E           ; random # seed
18 RNDH         EQU     $4F
19 BASL         EQU     $28           ; screen line pointer
20 CH           EQU     $24           ; screen column pointer
21 STACK        EQU     $100
22 IORTS        EQU     $FF58         ; address of RTS instruction
23 SETKBD       EQU     $FE89         ; entry to do IN#0
24 KEYIN        EQU     $FD1B         ; entry to read APPLE keyboard
25 COUT1        EQU     $FDF0         ; entry sends char to screen
```

```
                        26 COUT         EQU     $FDED     ; char to screen and card
                        27 GETKEY       EQU     $FD26     ; entry gets key & resets flash
                        28 KEY          EQU     $C000     ; APPLE keyboard port
                        29 *
                        30 CN           EQU     $7F8      ; holds $CN where N=slot #
                        31 PRNT         EQU     $6F8      ; B7=0 if print is true
                        32 ASAVE        EQU     $578      ; saves A register
                        33 CASE         EQU     $638      ; UC/LC conversion byte
                        34 RAMCTL       EQU     $738      ; holds current ACIA control
                        35 *
                        36              ORG     $0000
                        37              OBJ     $0000
                        38 *
0000 2C 58 FF           39 HINIT        BIT     IORTS     ; sets V flag
0003 70 04              40              BVS     COMMON    ; skip soft entries
                        41 *                              ; print entry
0005 18                 42 PRNTRY       CLC
0006 B0 FE              43              BCS     *
                        44              ORG     *-1
                        45              OBJ     *
                        46 *                              ; input entry
0007 38                 47 INNTRY       SEC
0008 B8                 48              CLV               ; flag soft entry
                        49 *                              ; common entry code
0009 8D 78 05           50 COMMON       STA     ASAVE     ; save A register
```



```
000C 98                 51              TYA
000D 48                 52              PHA               ; save Y register
000E 8A                 53              TXA
000F 48                 54              PHA               ; save X register
0010 08                 55              PHP               ; save status
0011 78                 56              SEI               ; disable interrupts
0012 20 58 FF           57              JSR     IORTS     ; while playing with
0015 BA                 58              TSX               ; the stack to get
0016 BC 00 01           59              LDY     STACK,X   ; $CN
0019 8C F8 07           60              STY     CN
001C 98                 61              TYA
001D 0A                 62              ASL
001E 0A                 63              ASL
001F 0A                 64              ASL
0020 0A                 65              ASL
0021 4E F8 06           66              LSR     PRNT      ; set print flag true
0024 28                 67              PLP               ; enable interrupts ASAP
0025 AA                 68              TAX               ; $N0 -> X
0026 50 21              69              BVC     SOFT      ; skip inits if soft entry
                        70 *
0028 A9 20              71              LDA     #$20
002A 99 38 06           72              STA     CASE,Y    ; case=$20 for LC -> UC
002D A9 A3              73              LDA     #$A3
002F 9D 80 C0           74              STA     CONTRL,X  ; ACIA master reset
0032 4A                 75              LSR               ; set full handshake
```

```
0033 99 38 07    76            STA     RAMCTL,Y  ; with 8 data, 2 stop bits
0036 A5 36       77            LDA     $36       ; test for PR#n or IN#n
0038 D0 0B       78            BNE     SETIN     ; hard entry
003A C4 37       79            CPY     $37
003C D0 07       80            BNE     SETIN
003E A9 05       81            LDA     #$5       ; PR#n--set soft entry
0040 85 36       82            STA     $36       ; address in vector
0042 18          83            CLC
0043 90 04       84            BCC     SOFT
0045 A9 07       85 SETIN      LDA     #$07      ; IN#n--set soft entry
0047 85 38       86            STA     $38       ; address in vector
                 87 *
                 88 *
0049 B8          89 SOFT       CLV
                 90 *
004A AC F8 07    91 RSTCTL     LDY     CN        ; par moi c'est C3
004D B9 38 07    92            LDA     RAMCTL,Y
0050 29 BF       93            AND     #$BF      ; set RTS signal 10|| ||||
0052 9D 80 C0    94 SETCTL     STA     CONTRL,X  ;
0055 90 7D       95            BCC     OUT2
0057 A4 24       96 IN         LDY     CH
0059 B1 28       97            LDA     (BASL),Y
005B 48          98            PHA               ; save screen character
005C AD 00 C0    99 IN1        LDA     KEY       ; check keyboard
005F 0A         100            ASL
```



```
0060 68         101            PLA               ; get screen char
0061 90 0F      102            BCC     IN3       ; branch if no key
0063 08         103            PHP
0064 20 26 FD   104            JSR     GETKEY    ; reset screen & get key
0067 28         105            PLP
0068 C9 81      106            CMP     #$81      ; control-A?
006A F0 47      107            BEQ     COMAND    ; yes--go process command
006C 70 63      108            BVS     OUT1      ; send char if term mode
006E A4 24      109 IN2        LDY     CH
0070 B1 28      110            LDA     (BASL),Y  ; save screen char
0072 48         111 IN3        PHA               ; and set screen
0073 29 3F      112            AND     #$3F      ; to flashing
0075 09 40      113            ORA     #$40
0077 91 28      114            STA     (BASL),Y
0079 E6 4E      115 RBUMP      INC     RNDL
007B D0 02      116            BNE     CHKINP
007D E6 4F      117            INC     RNDH
007F BD 80 C0   118 CHKINP     LDA     STATUS,X  ; check ACIA input
0082 4A         119            LSR               ; status for char
0083 90 D7      120            BCC     IN1       ; no, go loop
0085 68         121            PLA               ; yes, reset screen
0086 91 28      122            STA     (BASL),Y
0088 AC F8 07   123            LDY     CN        ; update the ACIA
008B B9 38 07   124            LDA     RAMCTL,Y  ; control on exit
008E 9D 80 C0   125            STA     CONTRL,X  ; to turn RTS off
```

```
0091 BD 81 C0   126              LDA     DATA,X    ; get ACIA char
0094 09 80      127              ORA     #$80      ; set high bit
0096 C9 92      128              CMP     #$92      ; control-R?
0098 F0 AF      129              BEQ     SOFT
009A C9 94      130              CMP     #$94      ; control-T?
009C F0 18      131              BEQ     SETTRM
009E C9 8A      132              CMP     #$8A      ; LF?
00A0 F0 A8      133              BEQ     RSTCTL    ; yes--ignore
00A2 50 4C      134              BVC     RETRN1    ; return char if not term
00A4 C9 E0      135              CMP     #$E0      ; LC?
00A6 90 03      136              BCC     NOTLC     ; no
00A8 59 38 06   137              EOR     CASE,Y    ; yes--convert
00AB 08         138 NOTLC        PHP
00AC 20 ED FD   139              JSR     COUT      ; print char
00AF 28         140              PLP
00B0 38         141 RCLNK        SEC
00B1 70 97      142              BVS     RSTCTL
                143 *
                144 *
00B3 20 1B FD   145 COMAND       JSR     KEYIN     ; get char
00B6 2C 58 FF   146 SETTRM       BIT     IORTS     ; set terminal mode
00B9 C9 98      147              CMP     #$98      ; if control-X, go do IN#0
00BB F0 33      148              BEQ     RETRN1    ; and return with char
00BD 49 E3      149              EOR     #$E3
00BF C9 70      150              CMP     #$70      ; if control-S, go turn on
```



```
00C1 F0 8F      151              BEQ     SETCTL    ; break and loop
00C3 49 6B      152              EOR     #$6B      ; control-H?
00C5 F0 04      153              BEQ     SETPRT    ; yes
00C7 C9 0E      154              CMP     #$0E      ; control-F?
00C9 D0 E5      155              BNE     RCLNK     ; no--reset ACIA
00CB 6E F8 06   156 SETPRT       ROR     PRNT      ; yes--set print flag
00CE 70 E0      157              BVS     RCLNK     ; and loop
                158 *
                159 *
00D0 68         160 OUT          PLA
00D1 8D 78 05   161 OUT1         STA     ASAVE
00D4 BD 80 C0   162 OUT2         LDA     STATUS,X  ; loop until the
00D7 29 02      163              AND     #$02      ; ACIA transmit
00D9 F0 F9      164              BEQ     OUT2      ; buffer is empty
00DB AD 78 05   165              LDA     ASAVE
00DE 9D 81 C0   166              STA     DATA,X    ; output character
00E1 AC F8 06   167              LDY     PRNT      ; get flag and skip
00E4 30 05      168              BMI     OUT3      ; COUT1 if full duplex
00E6 08         169              PHP
00E7 20 F0 FD   170              JSR     COUT1
00EA 28         171              PLP
00EB 70 81      172 OUT3         BVS     IN2       ; loop if terminal mode
                173 *
00ED 0E F8 06   174              ASL     PRNT      ; restore print flag
                175 *
```

```
                        ASAVE
                        RETRN2
                        SETKBD

                        STA
                        BVC
                        JSR
                        PLA
                        TAX
                        PLA
                        TAY
                        LDA     ASAVE
                        RTS

00F0 8D 78 05   176 RETRN1
00F3 50 03      177
00F5 20 89 FE   178
00F8 68         179 RETRN2
00F9 AA         180
00FA 68         181
00FB A8         182
00FC AD 78 05   183
00FF 60         184
```

APPENDIX C

PRINTER INTERFACE REQUIREMENTS

Despite the RS-232-C standards, handshaking for serial printers is far from standardized. If you plan to use the 7710 as a serial printer interface, you must make sure that the 7710 finds the proper signals on the interface pins, altering the cable or adding jumper wires as required. Interface wiring for many of the common serial printers is given below. If your printer is not mentioned here, you will need to compare the 7710 handshaking with the printer's handshaking as described in the printer manual, then construct or alter your cable accordingly.

1. Anadex Printers

7710 pin	3	4	7
to			
Anadex pin	3	19	7

Connect 7710 pins 20 and 6 together at the RS-232-C connector.

2. Anderson Jacobson Printers

7710 pin	2	3	4	7	20
to					
AJ pin	2	3	20	7	4

Refer to the installation section of your printer manual for instructions on enabling the DTR signal. This signal is disabled when the printer is shipped, but must be enabled when the printer is used with the 7710.

3. Centronics Serial Interface Printers

7710 pin	3	4	6	7	8	20
to						
Cent pin	3	11*	6	7	8	20

*This Centronics printer signal is of the wrong polarity for use with the 7710 and must be inverted if the printer is to be operated at a high baud rate. Call Centronics for instructions.

If operation at 300 baud is acceptable, connect pin 4 of the 7710 interface to pin 4 of the printer.

4. C-ITOH Printers

7710 pin	3	4	7
to			
C-ITOH pin	3	20	7

Connect 7710 pins 20 and 6 together at the RS-232-C connector.

5. DECwriter

7710 pin	2	3	7
to			
DEC pin	2	3	7

Connect the following pins at the 7710 connector: 6 to 20, 4 to 8.

6. Diablo 630

7710 pin	2	3	4	5	6	7	8	20
to								
630 pin	2	3	20	5	6	7	8	4

Install a jumper plug connecting pins 5 and 6 of A60 on HPRO5 to enable the DTR signal.

7. Diablo 1600 and Xerox 1700 Series

Remove printer cover. Unplug connector at HPRO4 module. Use a paper clip to push out the wire on pin 3, switch it to pin 2. If you have problems, call a Diablo or Xerox service center and tell them you want to activate the DTR signal.

8. EPSON MX Series

7710 pin	3	4	6	7	20
to					
MX pin	3	11	6	7	20

9. IDS Paper Tiger

7710 pin	3	4	7
to			
IDS pin	3	20	7

Connect 7710 pins 20 and 6 together at the RS-232-C connector.

10. NEC Spinwriter

7710 pin	3	4	7
to			
NEC pin	3	19	7

Connect 7710 pins 20 and 6 together at the RS-232-C connector.

Connect NEC pins 20, 6, and 8 together at the RS-232-C connector.

11. Qume Printers

With the printer's Modem/No Modem switch in the Modem position, a Qume printer can be connected to the 7710 with a standard RS-232-C cable. The necessary connections are:

7710 pin	2	3	4	5	6	7	8	20
to								
Qume pin	2	3	4	5	6	7	8	20

12. Talley 1612

7710 pin	2	3	4	5	7	20
to						
1612 pin	2	3	19	5	7	20

13. Texas Instruments TI800 Series

7710 pin	3	4	6	7	8	20
to						
TI pin	3	11	6	7	8	20

APPENDIX D

CHECKOUT

Your 7710 has been fully tested, but you may for various reasons wish to test it yourself. The simple tests described in this section test most of the circuitry of the 7710. (The tests assume that the 7710 is in slot #2. If it is not, you will need to modify the tests accordingly.)

Please note that ALL 7710 MODULES ARE SHIPPED WITH ROMS. Unless you remove these ROMs and substitute RAMs, you should not run Test 2. Test 1 is valid for substitute ROMs, but you must of course compare the screen display with a correct program listing.

D.1 TEST 1: ROM TEST

This test displays the contents of the ROMs on the CRT screen, verifying that the ROMs can be read by the computer, and allowing you to compare

the contents with the program listing provided in Chapter 3.

a. Reset your computer.

b. Type in C200L<cr>.

c. Compare program listing to the TV display.

d. When you run out of screen display, type in L<cr>.

e. Repeat c and d until all 256 bytes of ROM have been read.

f. If problems result, compare the hexadecimal values of the memory locations. The ROMs may be reversed on your board. If not, see your CCS dealer.

Note: Your computer's disassembler cannot recreate assembler pseudo-operation codes, such as ORG or EQU. Occasionally, use of the ORG instruction could hide an instruction from the disassembler. For instance, the code:

```
BCS     *
ORG     *-1
SEC
```

will disassemble as

```
BCS     *+$38
```

Watch for this kind of programming trick when comparing the listings. It is valid code, but may

make you think you have bad ROMs. Programming tricks such as this are used to conserve memory in tight situations.

D.2 TEST 2: RAM TEST

This test verifies that you can read from and write to all locations of the program RAMs. A 256-byte segment of your computer's firmware is copied into the RAMs, then the copy is compared to the original. Errors are displayed on the screen.

a. Reset your computer.

b. Type in C200<F000.F0FFM<cr>.

c. Type in C200<F000.F0FFV<cr>.

d. A * should appear almost immediately on the screen if all is OK. If it doesn't, rerun the test to make sure that you didn't make a mistake. If you still have problems, see your CCS dealer.

D.3 TEST 3: SERIAL DATA LOOP TEST

This test checks out the ACIA, the clock, and the line drivers. It does this by transmitting a known byte of data, looping it back to the receiver, reading the data, and comparing the result. For this test, you will need a "loop-back" test fixture. To make one, obtain a

standard male DB-25S socket and wire pins 2 and 3 together, pins 4 and 5 together, and pins 8 and 20 together. This fixture allows transmitted data to be looped back into the ACIA receiver.

a. Disconnect the signal cable.

b. Plug the loop-back test fixture onto the end of the I/O cable.

c. Ensure that the External baud rate is NOT selected.

d. Turn on and reset your computer.

e. Type in C0A0:03<cr> to reset the ACIA.

f. Type in C0A0:11<cr> to initialize the ACIA.

g. Type in C0A1:55<cr> to write an alternating bit pattern.

h. Type in C0A1<cr> to read the received data.

i. Compare the display with what was sent.

j. Repeat Steps g, h, and i using AA in place of 55.

k. Repeat Steps f through i using different baud rates, ACIA commands, and data patterns until you are satisfied that the interface works properly. If you have problems, see your CCS dealer.