OWNER'S MANUAL

Model 7720 Parallel Interface

California Computer Systems

CALIFORNIA COMPUTER SYSTEMS

APPLE II™ PARALLEL INTERFACE

MODEL 7720

OWNER'S MANUAL

89000-07720
REV D

California Computer Systems
250 Caribbean Drive
Sunnyvale, CA 94086
USA

TABLE OF CONTENTS

GREETINGS

and welcome to the world of parallel interfaces. With the CCS 7720 Parallel Interface board, you have the means to interface to your APPLE II computer a variety of peripheral devices, such as printers, paper tape equipment, or even another computer.

Before you rush off and start plugging into high-speed peripherals, take time to study this manual and the manuals of the devices you want to interface to your computer. Once you feel comfortable with the hardware and software needs of each, plug them in, program the interface, and then have fun using the additional capability of your computer.

NOTE: The CCS Parallel Interface comes in two versions, the 7720A and the 7720B. The 7720A ROMs provide a standard I/O driver, while the 7720B ROMs provide a special driver for the Centronics printer. Otherwise the 7720A and the 7720B are identical. Special instructions regarding the 7720B are given in this manual when necessary.

CHAPTER 1

THEORY OF OPERATION

There are two major types of peripheral interfaces: serial and parallel. Serial interfaces transfer information to and from the peripherals as streams of binary digits (bits), one bit at a time. This form of information transfer requires a minimum number of data paths (sometimes only two wires), but is usually limited to slow-to-moderate transfer rates. Typical rates vary from a few hundred to several thousand bits per second. Parallel interfaces, on the other hand, use multiple data paths, thus transferring many bit streams simultaneously. Though more data paths (i.e., wires) are required, moderate to very high data transfer rates are possible with parallel interfaces. Rates well in excess ofone million bits per second are routinely achieved.

In spite of their differences, serial and parallel interfaces have at least one characteristic in common. Both require control signals to tell the unit at one end of the data path what the unit at the other end is doing. These "handshake" signals signify such things as "I'm busy; wait a bit," "I've got the data you requested; take it," or "I've got it; get me some more." Many of these signals are very general in nature and therefore apply to almost all peripherals you can hook up to your computer. This signal generality, plus the fact that peripheral manufacturers cannot afford to design a unique interface for each computer which might interface with their product, led to the development of general-purpose interface chips such as the 6821 Peripheral Interface Adapter (PIA).

1.1 THE 6821 PIA

The PIA integrated circuit is the heart of the 7720A interface. It provides the necessary compatibility between your computer's I/O Connectors and two 8-bit parallel busses to the outside world. It also provides the handshaking signals which make everything work together correctly.

The 6851 PIA chip can be divided functionally into three sections. The processor interface section contains the chip selection and control logic as well as the data bus drivers and receivers for interface with the computer's Peripheral I/O conectors. Peripheral sections A and B are almost alike and contain all of the necessary data storage and control logic for interfacing with a broad range of peripheral devices. The operation of each of these sections will be discussed in turn.

1.1.1 Processor Interface Section

The logic in this section provides control for the unit (CS0, CS1, -CS2),register selection (RS0, RS1), read/write selection (R/W), processor interrupt (-IRQA, -IRQB), register reset (-RES), and timing (E). It also contains the drivers and receivers for the eight bi-directional data lines to the processor (D0- D7). The PIA has, however, a limited ability to drive the data lines of the computer. Therefore, to ensure that data is successfully passed to and from the computer and the PIA, a bi-directional line driver (the 8304) has been placed between them. This chip can drive the bus with sufficient power to allow good data transfer. In doing so, however, the chip consumes a significant portion of your computer's power supply. Since it is used only for brief periods of time, a power-down feature has been provided. A transistor and associated circuitry monitor the device select line from the computer (-DEVICE SELECT). When -DEVICE SELECT (Pin 41) becomes active (low), the transistor turns power on for the 8304. When Pin 41 goes inactive (high), the transistor removes the power from the 8304. The power conservation feature will become more and more important to you as you add components to your system.

Because the APPLE II provides the logic to decode peripheral addresses and trigger -DEVICE SELECT to the appropriate peripheral I/O connector, no on-board device decoding is necessary. Therefore, the 7720 Interface board ties -DEVICE SELECT directly to -CS2 of the PIA. CS0 and CS1 are permanently enabled When the PIA is selected, the R/-W line is used to select between the read and write modes. R/-W is also used by the 8304 to determine the direction of data flow between the PIA and the computer.

The interrupt arbitration logic is just one link in the interrupt daisy chain. The entire daisy chain prioritizes peripheral-generated interrupts to ensure that only one device interrupts the computer at a

time. When the 7720 wants service from the processor, it issues an interrupt request through -IRQA or -IRQB. If no higher-priority interrupts are in progress (i.e., if connecter pin 28 is high), an interrupt request is issued to the computer through connecter pin 30. At the same time, a low signal sent through pin 23 tells all lower-priority devices that an interrupt is in progress and that they will have to hold off. When the PIA is serviced it will remove its interrupt request, letting the daisy chain signal go high again. Note that the highest priority device resides in peripheral slot 1, and the lowest in slot 7. Slot 0 does not support the daisy chain. Empty slots between cards break the daisy chain.

The -RESET line from the peripheral connector ties to -RES (pin 34) of the PIA. When -RES goes active (low), all of the PIA register bits are reset to logical zero. The -RESET line goes active every time you hit the Reset key on your computer's keyboard. Remember that after each time you hit Reset you have to reinitialize the interface before you can continue using it.

The PIA's timing is controlled by the phase 0 clock through PIA pin 25, Enable (E). This timing signal is used for many functions, including the strobing of data into various registers in the peripheral sections.

The last two signals within the processor section (RS0, RS1) are used in the PIA for register selection. These pins are connected to the two low-order Address lines at the peripheral I/O connector. Use of these lines will be discussed in some detail in the software programming section below.

1.1.2 Peripheral Sections

The PIA has two nearly-identical sections (A and B) for interfacing to your peripherals. Each section is made up of an eight-bit bi-directional data bus, two control lines, and three registers. The data bus is the parallel data transfer path, while the control lines are the handshaking signal lines between the interface and your peripheral. The operation of each of these elements is closely tied to the three types of registers in each section. One of these registers, the Output Register (OR), is used as temporary storage for data being transferred from the processor to the peripheral device (output). A second register is the Data Direction Register (DDR), which is used to condition each individual data line as either input or output. A logic one (1) in any specific bit of the DDR defines the corresponding peripheral data line to be

an output line, while a zero (0) specifies the corresponding line to be an input. Thus, if we wanted to define all the data lines as input, we would place a $00 in the DDR. Similarly, a $FF in the DDR conditions all peripheral data lines to be outputs. If we wanted the four high-order data lines to be inputs and the four low-order data lines to be outputs, we would put a $0F in the DDR, and so on.

The last register in each peripheral section is a Control Register (CR). This is the one that gives the PIA its versatility. For instance, one bit in the CR controls whether the computer talks to the OR or the DDR, both of which are on a "party line" (sharing the same address.) The CR also controls when and under what conditions interrupt requests to your computer are sent or inhibited. A look at what each CR bit means is presented in Section III of this manual.

1.2 CONTROL PROGRAM MEMORY

The APPLE II dedicates 256 bytes of memory space for each peripheral connector. This is enough space for most interface-unique software or firmware. Standard controller firmware for the 7720 is provided on two 256x4

ROMs, which come in two versions. The 7720A ROMs contain a standard driver; the 7720B ROMs contain a special driver for Centronics printers. The B ROMs are available sepatarely (CCS part no. 00000-7620B) for 7720A owners who wish to add Centronics interface capability to their systems.

Should you prefer to develop your own controller firmware, this board allows you to install two Random Access Memories. CCS offers a RAM-Pak to support this feature. If RAMs are used, the memory power-down feature must be disabled and the R/-W control line enabled to these RAMs. This is done by installing one jumper wire on the interface card. With the RAMs, you can develop and thoroughly test your controller program before committing it to ROMs.

CHAPTER 2

INSTALLATION AND CHECK-OUT

2.1 SETUP FOR THE PERIPHERAL

To make your terminal, interface, and computer work as a unit, several things must be set on the terminal. For instance, if your terminal allows you to select between half duplex and full duplex operation, select the FULL DUPLEX mode. Your computer's firmware expects full duplex keyboard/display. What this means is that the computer, after receiving a character from the keyboard, sends it back to the display. This "echo" feature allows quick verification that the computer received what you wanted it to. When a terminal is in a half duplex mode, on the other hand, it will display the character when it is typed in. Then, when the computer echoes, the character will be displayed a second time. For example, if you type in "RUN" with the terminal in the half duplex mode, you will see "RRUUNN" on the display.

Next, you must provide for Line Feed. Your computer does not generate Line Feed control characters. It expects your terminal to do this each time a Carriage Return (Control D) is sent. If your terminal has an Auto Line Feed option, use it. If not, the interface driver software will have to be programmed to do it. The driver listing in Chapter III shows one way to program a Line Feed after every Carriage Return.

The computer expects all alphabetic characters to be upper case. If the terminal has an Upper Case Only option, use it. Of course, the driver firmware can be programmed to convert all lower-case characters to upper case. One way to do this is also shown in the driver listing.

The PIA will neither generate nor check character parity for you. In most cases, parity checks are both unneeded and unwanted. Should your peripheral require parity bits, the driver software will have to be programmed to provide them.

Parallel interfaces have no guiding standard that defines the connector types or pin assignments for the interconnecting cable. Because of this, you will need to prepare your own custom signal cable. Consult both your peripheral manual and Chapter 5 of this manual to obtain sufficient information to design and build this cable.

Caution is advised in the preparation of this cable. Parallel interfaces, unlike serial interfaces, require that special attention be given to data line length in order to avoid a problem with "data skew." Data skew is the situation in which n (8, in our case) bits of data, sent simultaneously, arrive at their destination at significantly different times. This problem can be caused by unequal data path (wire) lengths, as well as by such things as differences in line-driving capacity and cross-coupling between data paths. To help avoid data skew, the length of the parallel lines between the 7720 Interface board and the device you are connecting to it should be kept to four feet or less. Lengths greater than this, though they often work satisfactorily, may cause problems.

2.2 CARD INSTALLATION

Now let's go back to the interface card and put it into the computer. First install the back panel connector cable onto the card. You must align pin 1 of the cable connector with pin 1 of J2, the mating connector on the board. Pin 1 on the cable connector can be identified by the outside colored strip on the cable or by a triangular mark or other type tick on the dual 13-pin connector. Pin 1 of J2 is marked on the board. When all the pins are properly aligned, push down firmly on the connector until you can no longer see the metal connector pins. Next gently fold the ribbon cable at a 45 degree angle towards the ROMs/RAMs. Crease the fold only slightly; too much crease might fatigue and break the wires in the ribbon. Now gently fold the ribbon back under itself. The slack in the cable is needed for strain relief. Note that the back panel connector points to the right of the board. Your card is now ready to be inserted into the computer.

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

Now place the computer directly in front of you. Remove the top cover by laying the palms of your hands on the back edge of the computer, with your fingers hanging over the rear. Curl your fingers around the rear edge until you feel the ridge at your fingertips. Gently but firmly pry up until you hear two distinct pops. Don't lift the cover any further; slide it to the rear to remove it from the computer. Toward the inside rear of the computer, you will see eight 50-pin connecters. They are numbered 1 through 7 from left to right. Place the CCS Parallel Interface card into any of these connectors except #0, the leftmost; it is reserved for other use. We suggest you use slot #2, if it isn't already occupied. Insert the card by holding it and the cable so that the component side of the card is to the right and the cable connector is to the rear. Align the card edge into the chosen connector; then gently push the card edge down until it is firmly seated. Now slide the cable connector through the nearest back-panel slot, and replace the cover on the computer. Plug one end of your signal cable to the external connector and the other end to the appropriate place on your peripheral. Finally, plug in the line power cord, and you're ready to test the interface.

2.3 CHECKOUT

The best test for any computer hardware is everyday use. There are a two tests you should perform first, though, to become confident that the interface will really work. One tests the on-board memory, the other the PIA itself. Two memory tests are listed below. Which one you use depends on the type of memory you are using. If you are using the 7720 as shipped, you have ROMs on-board, not RAMs. Use the ROM test below. If you have installed RAMs in place of the ROMs, use the RAM test.

For these tests we have assumed that the 7720 is in slot #2. If you put it in a different slot, you will need to modify the tests accordingly.

2.3.1 ROM Test

This test displays the contents of the ROMs on the TV screen. You can compare the display with the ROM program listing. This verifies that the ROMs are properly installed and can be read by the computer.

NOTE: Your computer's disassembler can't recreate any assembler pseudo-operation codes, such as ORG or EQU. Occasionally, use of the ORG instruction may hide an instruction from the disassembler. For instance:

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

will disassemble as

```
          BCS     *+$38
```

Watch out for this kind of programming trick when you are comparing the listings. It is valid code, but may make you think you have bad ROMs. Programming tricks like this are used to save memory in tight situations.

Procedure:

a. Turn on and Reset your computer.

b. Type in C200L (CR)

c. Compare the 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 are read.

f. If problems result, compare the hexadecimal values of the memory locations. The ROMs may be reversed on the card. If this isn't the case, see your dealer.

2.3.2 RAM Test

This test verifies that you can read and write to all locations of the controller RAMs. It copies a 256 byte segment of your system's firmware into the RAMs, then compares this copy to the original. Errors, if any, are displayed on the TV screen.

Procedure:

a. Turn on and Reset the 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 this does not occur and you have made sure that the RAM jumper is installed and the RAMs are properly seated, see your CCS dealer.

2.3.3 Parallel Data Loop Test.

This test checks out the PIA and the line drivers. It does this by sending out a known byte of data from one side of the PIA, looping it back to the other PIA section, reading the data, and comparing the result.

For the test, we need a "loop-back" test fixture. This may be made by taking a standard DB-25P plug that mates into the back panel connector and wiring all the signal lines together as shown in Section 5.5, page 5-6. This will allow the output data to be looped back into the PIA input section.

Procedure:

a. With the power off, disconnect the signal cable from the back panel.

b. Install the loop-back test fixture on the back panel connector.

c. Turn on and Reset your computer.

d. Type in C0A1:00 (CR) to access the DDRA.

e. Type in C0A3:00 (CR) to access the DDRB.

f. Type in C0A0:00 24 FF 24 (CR) to complete initialization.

g. Type in C0A2:55 (CR) to write an alternate bit pattern to the PIA B side.

h. Type in C0A0 (CR) to read the receive data from PIA A side.

i. Compare to see if the data from h matches what was sent out in step g.

j. Repeat steps g, h, and i using AA for the 55 in step g.

k. Repeat steps g, h, and i using different data patterns until you are satisfied that the interface works.

l. Experiment with different commands until you are comfortable with the working of of the PIA.

m. If you have any problems, see your dealer.

After you have completed this test, turn off the power, disconnect the test fixture, and reconnect the peripheral. If you are using the programmed ROMs, you are ready to use the CCS Parallel Interface. If not, you are ready to start developing your controller software.

CHAPTER 3

INTERFACE SOFTWARE/FIRMWARE

The 7720 Parallel Interface is quite a versatile device. It gets its versatility by striking a balance between the hardware and its controlling software. The hardware takes care of most of the tasks which are unchanged regardless of use. The software is left to do what it does best, the performance of unique tasks. With this "personality" contained in the software, it is impossible to write one program which is everything to everybody. CCS offers two standard ROM-Paks which should meet most requirements. If neither ROM-Pak suits your needs, use the information in this chapter to write your own software.

3.1 REGISTER ADDRESSES

Your computer dedicates 16 memory addresses to each of the peripheral connector slots (except slot #0) for the memory-mapped input or output. These 16 memory addresses are above and beyond the 256 dedicated program memory addresses. The I/O addresses are located at $C0xy, where: x = 8 + n; n = the peripheral slot number (1, 2,...,7); and y = the specific address (0, 1,...,$E, $F). The PIA register addresses are as follows:

C0x0 = the A side data (direction) register;

C0x1 (read) = the A side status register;

C0x1 (write) = the A side command register;

C0x2 = the B side data (direction) register;

C0x3 (read) = the B side status register;

C0x3 (write) = the B side command register.

3.2 PIA COMMANDS

The PIA functions are controlled by a command byte. Bits 0-2 have individual meanings, while bits 3-5 form a three-bit code.

Bit 0 = 0	No interrupts from this side
= 1	Interrupt if Status Bit 7 set
Bit 1 = 0	CA(B)1 low sets Status Bit 7
= 1	CA(B)1 high sets Status Bit 7
Bit 2 = 0	Enable Data Direction Register
= 1	Enable Peripheral Register

The next four commands program CA(B)2 as an Interrupt Input.

Bits 543	
0x0	No interrupt
0x1	Interrupt when Bit 6 is set.
00x	CA(B)2 low sets Bit 6.
01x	CA(B)2 high sets Bit 6.

The next three commands program CA2 as an output line. Note that they assume that PIA-A is an input port to the computer.

Bits 543	
100	+BUSY signal: -READY FOR DATA.
101	-ACK strobe (1us pulse).
11y	CA2 = y

The next three commands program CB2 as an output line. Note that they assume that PIA-B is an output port from the computer.

Bits 543

```
     100   -DATA READY signal; made
           active by writing data into
           PIA-B Data Register.
     101   -OUTPUT STROBE (1us pulse).
     11y   CB2 = y.
```

Note that if Command Bit 2 equals 0, the Data Direction Register may be accessed through the corresponding data port. This register establishes, on a line-by-line basis, whether a line will be an input or an output line. If you want a line to be used to input data, put a 0 into the corresponding Data Direction Register bit. If you want it to output, put in a 1.

3.3 PIA STATUS

In the command structure, note that handshake lines CA1, CA2, CB1, and CB2 cause status bits 6 and 7 to be set or reset. These two bits are read only; we cannot alter these two bits. They are the only true status bits. Status bits 0 to 5 will show us only the last written command.

3.4 PORT PROGRAMMING

In the standard drivers, the PIA's A side has been programmed for input and the B side for output. To set up the A side, a $00 is loaded into the command register. This gives us access to DDR-A. Then a $00 is loaded into DDR-A side to establish the A port as an input port. Finally, a $24 is loaded into the A side command register to set up the A side's operating mode. This command sets up the CA1 input line to load data into the PIA from the peripheral on a positive to negative transition. CA2 then goes high to tell the peripheral not to send any more data for a while and not to interrupt the computer. The computer will soon come around and read the status port. Since the CA1 caused status bit 7 to be set, the computer knows that data is waiting in the PIA's A side data register. The computer now reads the data, which in turn causes CA2 to go low, notifying the peripheral that it is free to send another byte of data.

The B side of the PIA is programmed in a similar manner. Here, though, the DDR-B is loaded with a $FF to make the B side an output port. The relative roles of CB1 and CB2 change a little bit to account for the differences between input and output. CB1 is now used as a negative logic peripheral "READY FOR DATA" line. When the peripheral is ready to accept another byte of data, it makes CA1 go negative. This causes the B side Status Bit 7 to be set. When the computer has an output character ready, it checks the status bit to find out if the peripheral is ready. Since it is, the computer then writes the character to the B side data register. This write causes CB2 to go negative and at the same time resets the B side's Status Bit 7. When the peripheral detects that CB2 is low, it should make CB1 go high and grab the data. After the peripheral has done whatever it is going to do with the character it will make CB1 go low again, and the cycle repeats.

3.5 INPUT/OUTPUT HANDLERS

Your APPLE II looks at two Page Zero locations to find out where the current keyboard input and console output control programs are located. These locations are:

$36-$37: console output handler;

$38-$39: keyboard input handler.

Whenever you type in the BASIC command IN#n, the firmware writes $00 in location $38 and $Cn in location $39. The equivalent monitor command, n[Ctrl]K, does the same thing. This makes an effective address of $Cn00 for the input handler initialization program. Thus the next time any keyboard input is wanted, the initialization routine gets called. The initializer must set everything up and then pass control to the input routine to actually do the input. Part of the initializer's task is to change location $38 to identify the input driver entry point. Then the next time input is wanted we can go straight to the input routine. We do not need to set everything up again. Likewise, when OUT#n (or n[Ctrl]P) is entered, location $36 is set to 0 and $37 set to $Cn. On the first output, control is passed to $Cn00 for output initialization. Location $36 must then be set to match the output handler's entry point for all subsequent console output.

Your computer handles input and output on a byte-by-byte basis. The data is passed between the handler and the calling program through the accumulator (A register). Your input routine should leave the data in the accumulator when control is returned to the caller. In the output routine the handler can find the data in the accumulator.

The input and output routines will be called as subroutines. Control can be returned to the caller by issuing an "RTS" (Return from Subroutine) instruction. Good programming practice says to save, upon entering a subroutine, all register contents, then to restore the register's original contents just before leaving the subroutine. (This does not apply to parameter-passing registers, of course.)

3.6 SCRATCHPAD MEMORY

The video display refresh memory locations (addresses $400 to $7FF) use only the first 120 of every 128 locations for the display data. The left-over 64 addresses can be used for other purposes. Use them carefully and be sure to test your routines thoroughly, though. Some other programmer may have beaten you to them. Two sets of locations which are available include $6F(n+8) and $77(n+8), where n is the slot number. For most programs, this should be enough space in which to save, for instance, the last issued PIA command, etc.

Although we do not need it in the standard programs, one other scratch location merits mention. Address $07F8 is often used to hold the page address of the current console. The page address is $C0 + n, where n is the slot number of the active interface board.

3.7 WRITING THE DRIVER

We now have enough information to program an easy remote console controller program. Our program will consist of three parts: initialization, input, and output. For initialization, we must:

a. Save the registers

b. Reset the PIA

c. Give the PIA its proper command

d. Set the proper input or output entry point

e. Initialize any special pointers or counters

f. Go to Step b of the appropriate routine, depending on whether input or output was wanted.

For input, we must:

a. Save the registers

b. Wait until the input data is ready

c. Read the input data

d. Do any special data conversion needed (set bit 7 = 1, convert lower case to upper, etc.)

e. Restore the registers

f. Return to the caller.

For output, we must:

a. Save the registers

b. Do desired preprint control (tabs, etc.)

c. Wait until the PIA can take more data

d. Write the data to the PIA

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

f. Restore the registers

g. Return to the caller.

Several of the above tasks are common to all the routines. To stretch our 256 bytes of space as far as possible, we must make as much code as possible common to all of the routines. Since we cannot predict what absolute

addresses will contain this code, we cannot create any subroutine calls. This means that we must use relocatable code throughout. This also means that we cannot use any absolute addressing unless that absolute address is fixed and will always be there when we need it. Otherwise-unused status flags may be used to indicate which entry point we came in from. This allows us to make some code common to all routines, yet go to the right unique code streams when we need to. We use the V (overflow) flag to indicate whether we are initializing or not, and the Carry flag to indicate whether we are inputting or outputting. After the flags have served their purpose they can be reused to indicate such things as a tab in progress.

A listing of the standard 7720A controller follows. Study it carefully. It contains special line and page length features which were not explained above. Depending on your needs, you can use the program as it is, or write your own using ours as a point of departure.

```
* 4.0 STANDARD DRIVER PROGRAM FOR THE CCS APPLE II
* PARALLEL INTERFACE CARD ("A" ROM Version)
*
*   This program is an example of an input and output
* driver for the Parallel Interface card.  It
* contains all the necessary coding to allow the direct
* logical replacement of your computer's keyboard and TV
* output.  The output defaults to 80 characters per line,
* the same as the TV output.  Three entry points are defined:
* one for initialization, one for input, and one for output.
* When the I/O commands, IN#n or PR#n (n = slot number) are
* issued, the computer sets up a jump vector address to the
* initialization entry point.  The initialization routine
* will then adjust the input or output entry point vector
* to the correct address when the first input or output
* occurs.
*
* System Equates
*
MAXLN   EQU     $36         54 line default
MAXCHR  EQU     $50         80 char/line default
BKSP    EQU     $87         ASCII Back Space - 1 (for carry)
LNFD    EQU     $8A         ASCII Line Feed
FF      EQU     $8C         ASCII Form Feed
CARRET  EQU     $8D         ASCII Carriage Return
FS      EQU     $95         Forward Space
SPACE   EQU     $A0         ASCII space
CPL     EQU     $21         Characters per line
LPP     EQU     $23         Lines per page
CH      EQU     $24         Tab column
CSWL    EQU     $36         Location of output driver vector
KSWL    EQU     $38         Location of input driver vector
```

```
                KSWH    EQU     $39
                RANDL   EQU     $4E         Random number seed location
                RANDH   EQU     $4F
                LOCASE  EQU     $6F8-$C0    Hold for UC conversion mask
                LNCNT   EQU     $778        Line counter
                CHCNT   EQU     $778-$C0    Character counter
                DATAA   EQU     $C080       Add 16*slot for the PIA Data port A
                CMDA    EQU     DATAA+1     PIA-A command port
                STATA   EQU     CMDA        PIA-A Status port
                DATAB   EQU     DATAA+2     PIA-B Data port
                CMDB    EQU     DATAB+1     PIA-B command port
                STATB   EQU     CMDB        PIA-B Status port
                WAIT    EQU     $FCA8       Time killer routine
                RETURN  EQU     $FFCB       Used to find the slot address
                *
                * The common code
                *
                        ORG     $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         BCS     DFB     $B0         Always skip the 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
0014 BC 00 01           LDY    $100,X     Y=Slot Page Number
0017 68                 PLA               Recover the output data
0018 68                 PLA                 (if any)
0019 68                 PLA
001A 68                 PLA
001B 9A                 TXS               Restore the Stack Pointer
001C 48                 PHA               Save the data on top of stack
001D 98                 TYA               Get the Slot Page Number
001E AA                 TAX               X=Slot Page Number ($CN)
001F 0A                 ASL    A          Multiply by 16 to get $NO
0020 0A                 ASL    A            (where N=Slot number)
0021 0A                 ASL    A
0022 0A                 ASL    A
0023 A8                 TAY               Y=$NO
0024 E6 4E              INC    RANDL      Increment the Random Number Seed
0026 D0 02              BNE    COMA
0028 E6 4F              INC    RANDH
002A 68        COMA     PLA               Get the saved status codes
002B 28                 PLP
002C 48                 PHA
002D 50 19              BVC    IO         Routine IO, branch
               *
               * Initialization Routine
               *
               * This code handles the first input or the first output
               * request after invoking the IN#n or the PR#n commands.
               * It checks to see whether input or output is wanted,
               * then goes to the appropriate code to initialize that
               * half of the PIA.
               *
```

```
002F B8                 CLV               Clear the initialization flag
0030 A5 38              LDA    KSWL       See if input is wanted
0032 D0 24              BNE    OINIT      No, branch for output init
0034 E4 39              CPX    KSWH       Maybe, make sure
0036 D0 20              BNE    OINIT      Branch if not
0038 99 81 C0           STA    CMDA,Y     Prepare access to DDR
003B 99 80 C0           STA    DATAA,Y    Set DDR-A for input
003E A9 24              LDA    #$24       Turn PIA on
0040 99 81 C0           STA    CMDA,Y
0043 A9 07              LDA    #7         Set normal input entry point
0045 85 38              STA    KSWL
0047 38                 SEC               Fall through next branch
0048 90 2F     IO       BCC    OUT        Normal output
               *
               * Input Routine
               *
004A B9 81 C0  INPUT    LDA    STATA,Y    Get PIA-A status
004D 2A                 ROL    A          Isolate Receive Ready Bit
004E 90 FA              BCC    INPUT      Loop until data is ready
0050 68                 PLA               Get rid of data on stack top
0051 B9 80 C0           LDA    DATAA,Y    Read the new data
0054 09 80              ORA    #$80       Set Bit 7 for normal video
0056 30 78              BMI    DONA       Always branch
               *
               * Output Initialization
               *
0058 A9 05     OINIT    LDA    #5         Set normal output entry point
005A 85 36              STA    CSWL
005C A9 50              LDA    #MAXCHR    Set defaults
005E 85 21              STA    CPL        Characters per line
0060 A9 36              LDA    #MAXLN     Lines per page
0062 85 23              STA    LPP
```

```
0064 A9 00              LDA    #0         Zero Counters
0066 8D 78 07           STA    LNCNT
0069 9D B8 06           STA    CHCNT,X
006C 99 83 C0           STA    CMDB,Y     Enable DDR-B access
006F A9 FF              LDA    #$FF       Set the DDR-B for output
0071 99 82 C0           STA    DATAB,Y
0074 A9 24              LDA    #$24       Normal control command
0076 99 83 C0           STA    CMDB,Y
                *
                * Output Routine
                *
                * This routine does the actual output of the data.  It
                * expects to find the data for output on the top of the
                * stack (where the common code put it).
                *
0079 BD B8 06   OUT     LDA    CHCNT,X    See if tab wanted
007C C5 24              CMP    CH
007E B0 03              BCS    OUTA       Branch if no tab
0080 A9 A0              LDA    #SPACE     Space out to column
0082 48                 PHA               Save on stack
0083 68         OUTA    PLA               Retrieve character
0084 48         OUTD    PHA               Resave it
0085 08                 PHP               Save Tab Flag
0086 C9 8C              CMP    #FF        Check for form feed
0088 D0 09              BNE    CRTLU      Branch if not
008A 28                 PLP               Restore tab flag
008B AD 78 07           LDA    LNCT       Develop # lines
008E E5 23              SBC    LPP          to end of page
0090 38                 SEC               Insure no tab on form feed
0091 D0 2D              BNE    LFA        Go finish the FF
0093 C9 95      CRTLU   CMP    #FS        See if Control U
0095 F0 04              BEQ    CHCTI      Branch if so
```

```
0097 29 60              AND    #$60       Test for other ctrl char
0099 F0 03              BEQ    OUTB       Skip counter increment
009B FE B8 06   CHCTI   INC    CHCNT,X    Bump counter
009E 28         OUTB    PLP               Reget tab flag
009F B9 83 C0   OUTC    LDA    STATB,Y    Get PIA-B status
00A2 29 80              AND    #$80       Isolate Acceptor Status bit
00A4 F0 F9              BEQ    OUTC       Wait if not ready
00A6 68                 PLA               Get data for output
00A7 99 82 C0           STA    DATAB,Y    Output it
00AA 90 CD              BCC    OUT        Branch if tab
00AC 70 01              BVS    LF         Branch if self-gen char
00AE 48                 PHA               Resave character
00AF C9 8A      LF      CMP    #LNFD      See if line feed
00B1 D0 13              BNE    BS         No, branch
00B3 EE 78 07           INC    LNCNT      Bump line count
00B6 30 33              BMI    MAKELF     Branch if eject in progress
00B8 AD 78 07           LDA    LNCNT      Get the line count
00BB E5 23              SBC    LPP        Ready for eject?
00BD 30 10              BMI    DONE       No, done
00BF E9 0C      LFA     SBC    #$0C       Allow for margin
00C1 8D 78 07           STA    LNCNT      Reset line count
00C4 D0 25              BNE    MAKELF     Do the eject
00C6 E9 87      BS      SBC    #BKSP      See if back space
00C8 D0 13              BNE    CR         No, branch
00CA DE B8 06           DEC    CHCNT,X    Yes, adjust counter
00CD 30 12              BMI    CRA        Disallow negative count
                *
                *  Final Common Code
                *
                * This part of the code restores the registers and
                * returns to the caller.  It is used by all of the routines.
                *
```

```
00CF 68          DONE    PLA                Set the stack straight
00D0 BA          DONA    TSX                Modify A register value in
00D1 E8                  INX                  stack to insure it is
00D2 E8                  INX                  restored to the right value
00D3 E8                  INX
00D4 9D 00 01            STA    $100,X
00D7 68                  PLA                Restore registers
00D8 A8                  TAY
00D9 68                  PLA
00DA AA                  TAX
00DB 68                  PLA
00DC 60                  RTS                Done!
00DD E9 06       CR      SBC    #6          See if a Carr Ret
00DF D0 12               BNE    AUTOCR      No, branch
00E1 9D B8 06    CRA     STA    CHCNT,X     Zero out counter
00E4 85 24               STA    CH            and tab pointer
00E6 A9 C0               LDA    #$C0        Wait for print head
00E8 20 A8 FC            JSR    WAIT          to return
00EB A9 8A       MAKELF  LDA    #LNFD       Make a line feed
00ED 2C CB FF    SELF    BIT    RETURN      V=1 for self-gen character
00F0 38                  SEC                C=1 for no tab
00F1 B0 91               BCS    OUTD        Always branch
00F3 BD B8 06    AUTOCR  LDA    CHCNT,X     End of line yet?
00F6 C5 21               CMP    CPL           yet?
00F8 30 D5               BMI    DONE        No, done
00FA A9 8D               LDA    #CARRET     Yes, get a Carriage Return
00FC D0 EF               BNE    SELF        Process it
00FD                     END
```

CHAPTER 4

OPERATION

In this section, we will give you a description of how to generate the controller driver routine and how to route the console input or output through this card. Little more can be said since the rest of the operating procedures are determined by the software you load use. We suggest that after you install the programs you attach a copy of the unique operating instructions to this chapter of the manual. Should you opt to use the sample driver program above, instructions are provided to guide you in selecting some non-default parameters in real time.

4.1 DRIVER GENERATION

The controller software has to be loaded into the computer before you can use it. Of course if you are installing ROMs on the card, the firmware is already there. But if you selected RAMs, they must be loaded every time you turn your computer on and want to use the interface. The following procedure is devised for floppy disks; if you use some other storage media, you will need to devise your own scheme.

The first chore is to get the controller software initially into memory. The firmware mini-assembler works nicely for this. See your Red Book for details of how to use it. Assemble the driver directly into the interface 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 load and save programs only from the lower 32K of memory. Location $A00 is a good spot for the copy. It won't interfere with the Integer BASIC or the Disk Operating System (DOS). This Monitor command performs the move nicely:

```
*A00<C100.C1FFM(cr)
```

Now 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(ctrl P)(cr)
```

Finally, we are ready to save the driver on disk:

```
>BSAVE PAR1.0,A$A00,L$100(cr)
```

Your driver software is now saved on your disk, with a file name of PAR1.0 if you followed the above example. You are now ready to test out the driver and modify it as necessary until you are happy with its performance. Do not forget to save a copy after each modification. There is nothing more frustrating than to try to check a routine only to have it bomb out and erase itself in the process. After you are happy with it, save it for one last time. You are now ready to routinely use the driver.

4.2 POWER-ON DRIVER LOADING

Two methods of loading the software from disk are outlined here. The first method uses direct commands, while the second does it under program control.

4.2.1 Direct Commands

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

1. Boot in the DOS:

   ```
   *6(ctrl P)(cr)
   ```

2. Read in the driver file:

   ```
   >BLOAD PAR1.0(cr)
   ```

3. Return control to the monitor:

   ```
   >CALL-155(cr)
   ```

4. Finally, upload the driver to the interface RAM. Assuming that the interface is in slot #2:

   ```
   *C200<A00.AFFM(cr)
   ```

4.2.2 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 "PARALLEL INTERFACE SLOT IS: ",S
20 IF S<1 OR S>7 THEN GOTO 10
30 DEST = -16384 + 256 * S
40 PRINT "(ctrl D)BLOAD PAR1.0,A$A00"
50 FOR I = 0 TO 255
60 POKE DEST + I, PEEK (2650 + I)
70 NEXT I
80 END
```

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

1. Boot in the DOS:

   ```
   *6(ctrl P)(cr)
   ```

2. Execute the PAR program:

   ```
   >RUN PAR(cr)
   ```

3. Answer the question when it appears:

   ```
   PARALLEL INTERFACE SLOT IS: ?2(cr)
   ```

The program is now loaded and ready to use.

4.3 INPUT

The programs you install in the RAM/ROMs won't do any good unless control is passed to them for input or output. To do this, type in (n=the slot number):

```
IN#n (>or ] BASIC)

n(ctrl K)  (Monitor)
```

Either of these commands will cause your computer to go to the installed input program on the card for all subsequent input to the computer. On the very first input, the PIA input side will be initialized if the driver program is, or is like, the standard drivers. Initializing the PIA input side doesn't alter the output side, even if the output function has already been invoked. Be aware that invoking the IN#n command may cause the driver to reselect the default options for both input and output, unless these options are initialized after the input vs output initialization decision is made in the driver. The sample program, for instance, waits until after the decision is made before it initializes the character and line counters. In this way, the IN command has no affect on the counters.

4.4 OUTPUT

To cause all console output to be controlled by the programs on the card, type in one of the following commands (n=the slot number):

```
PR#n  (> or ] BASIC)
n(ctrl P)  (Monitor)
```

All subsequent output from the computer will be routed to the interface's driver program. Again, be aware of possible re-selection of the default options.

4.5 DEFAULT PARAMETERS

Several default parameters of the standard drivers can be changed after the IN#n or OUT#n (as appropriate) commands have been executed.

4.5.1 PIA operating mode

You can change the PIA settings by selecting the appropriate values as defined in Section III above and POKEing them into the following locations, depending on the effect wanted:

$-16256 + 16*n ($C080 + n) for DDR-A (The A command must be set for DDR access)
$-16255 + 16*n ($C081 + n) for A side command register
$-16254 + 16*n ($C082 + n) for DDR-B (The B command must be set for DDR access)
$-16253 + 16*n ($C083 + n) for B side command register

Remember, though, that any subsequent IN#n or OUT#n command will re-command the PIA to its default values for that side.

4.5.2 Lines Per Page

The standard drivers will automatically issue 12 line feeds after every 54 lines have been printed. To change the number of printed lines which trigger the automatic 12 line feeds, POKE the new maximum lines per page into location $6F8 (1784d) + slot number. This number should be in the range of 1 < LPP < 127 or funny results will happen!

4.5.3 Characters Per Line

The sample driver will automatically initiate a carriage return, line feed sequence after every 80 characters have been printed. If you want some other number of characters per line, simply

POKE your value into location $5F8 (1528d) + slot number. Make sure it is in the range 0 < CPL <= 255.

NOTE: Although the standard drivers will respond to >BASIC's TAB function, no attempt has been made to allow for the]BASIC functions of HTAB or VTAB. There is not enough room in 256 bytes of memory to allow for this.

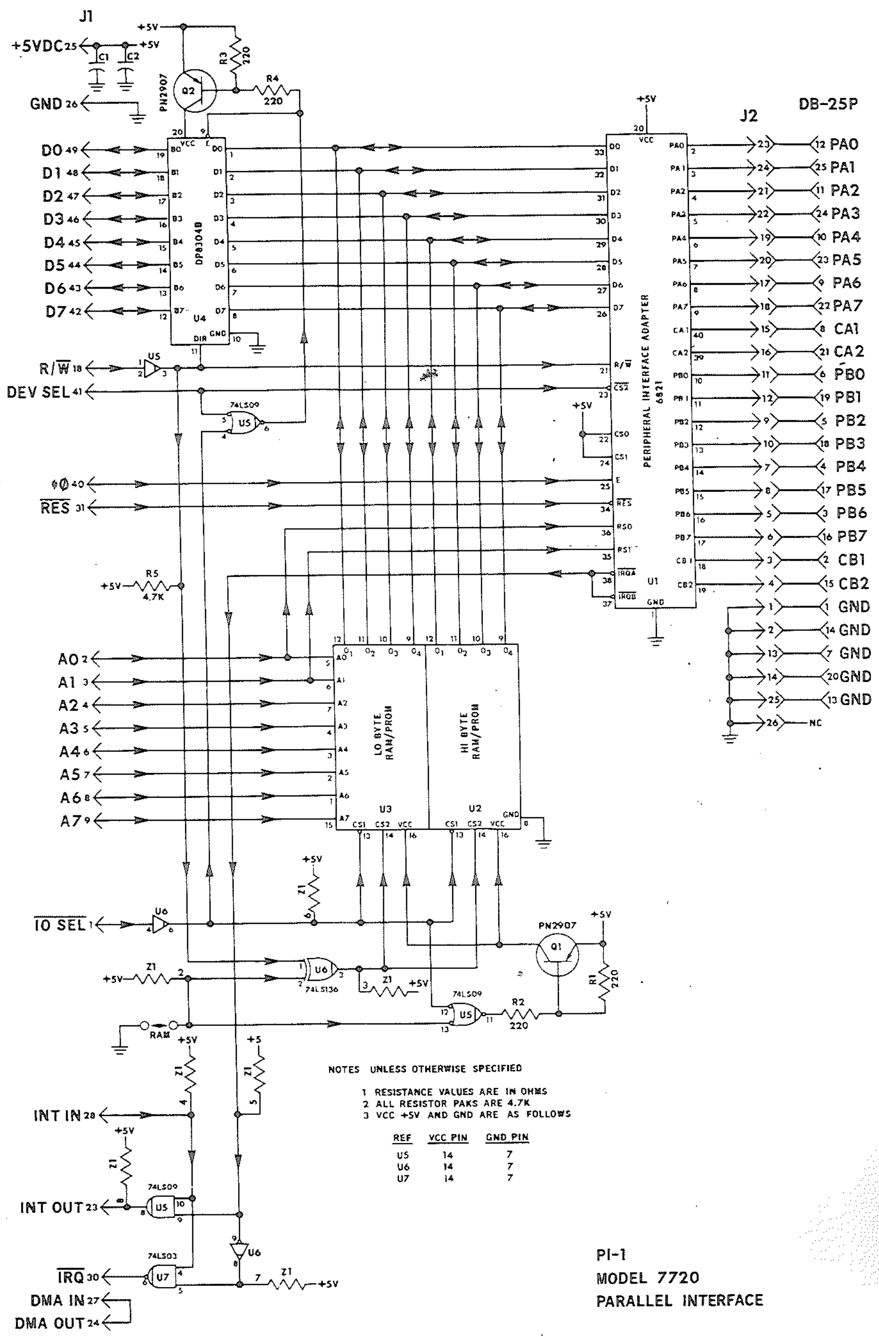
CHAPTER 5

TECHNICAL INFORMATION

5.1 SPECIFICATIONS

SIZE:	5" l x 2.75" h x 0.75" w (max)
WEIGHT:	less than 5 oz.
SYSTEM INTERFACE:	
Internal:	APPLE II Peripheral slots 1 through 7
External:	Two 8-bit bi-directional parallel ports Four handshake lines TTL compatible Side A and B CMOS drive capability Side A All external lines via DB-25 connector
MEMORY:	ROM (Mask) PROM (Fuse Link) RAM (Static: two 2112's)
Size:	256 Bytes Note: ROM/PROM Auto Powered Down
REQUIRED POWER:	+5 volts DC
FEATURES:	Supports Daisy Chain Interrupts with On-Board Arbitration Logic Allows DMA Daisy Chain Pass-Through Glass Epoxy (FR-4) PC Board Gold Plated Connector Fingers Solder Mask Both Sides of Board Component Silkscreen For details of other features, see a 6821 Data Sheet.

5.2 SCHEMATIC/LOGIC DIAGRAM



NOTES UNLESS OTHERWISE SPECIFIED
1 RESISTANCE VALUES ARE IN OHMS
2 ALL RESISTOR PAKS ARE 4.7K
3 VCC +5V AND GND ARE AS FOLLOWS

REF	VCC PIN	GND PIN
U5	14	7
U6	14	7
U7	14	7

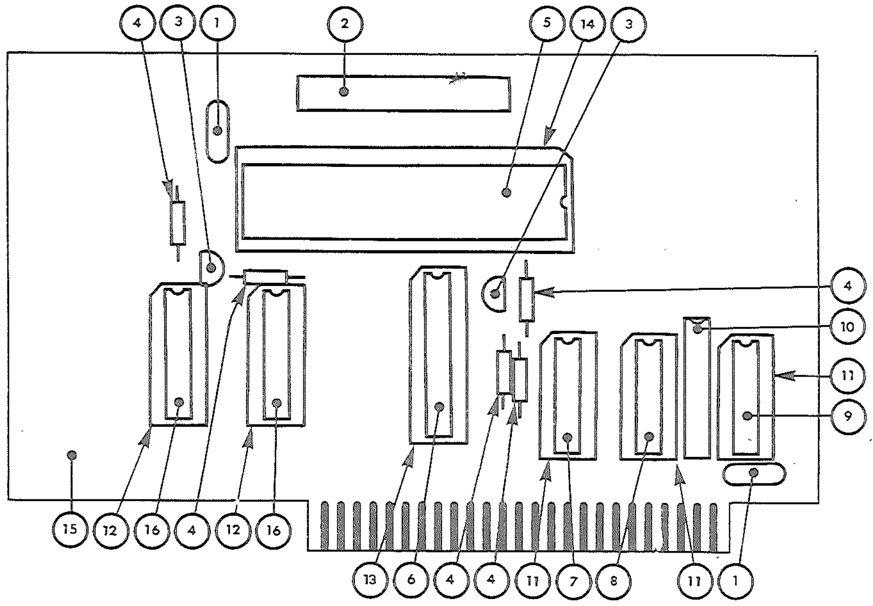
PI-1
MODEL 7720
PARALLEL INTERFACE

5.3 PARTS LIST

CCS 7720 (Assy 00000-7720A or B)

#	QTY	REF	CCS PART #	DESCRIPTION
1	3	C1-3	42034-21046	CAPACITOR, MONOLYTHIC .1uf, 50vdc
2	1	P2	56004-02013	HEADER, DUAL 13 PIN
3	2	Q1,2	36100-02907	TRANSISTOR, SI; PNP GENERAL PURPOSE, PN2907
4	4	R1-4	40002-02215	RESISTOR, FIXED, COMP 220ohm, 1/4W, 10%
5	1	U1	31100-06821	IC, DIGITAL, MOS; 6821 PIA
6	1	U4	30900-08304	IC, DIGITAL, TTL; 8304B OCTAL BUS DRVR/RCVR
7	1	U5	30000-00009	IC, DIGITAL, TTL; 74LS09 QUAD 2 IN AND (OC)
8	1	U6	30000-00136	IC, DIGITAL, TTL; 74LS136 QUAD 2 IN EX-OR (OC)
9	1	U7	30000-00003	IC, DIGITAL, TTL; 74LS03 QUAD 2 IN NAND (OC)
10	1	Z1	40930-72726	RESISTOR NETWORK, SIP 2.7K x 7
11	3	XU5-7	58102-00140	SOCKET, IC; LOW PROFILE 14-PIN DIP
12	4	XU2,3	58102-00160	SOCKET, IC; LOW PROFILE 16-PIN DIP
13	1	XU4	58102-00200	SOCKET, IC; LOW PROFILE 20-PIN DIP
14	1	XU1	58102-00400	SOCKET, IC; LOW PROFILE 40-PIN DIP
15	1	-	07720-00002	BOARD, PC PI-1, REV A
16	2	U5,6	00000-7620A	ROM-PAK, 7720A STD
		or	00000-7620B	ROM-PAK, CENTRONICS
-	1	-	00000-7325A	CABLE ASSEMBLY, 9" DUAL 13 TO DB-25P
-	1	-	89000-07720	MANUAL

5.4 PARTS BREAKDOWN



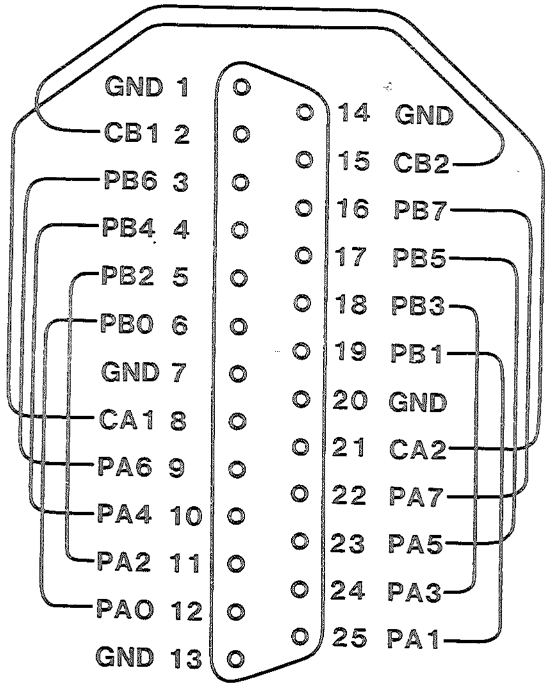
5.5 PARALLEL ECHOPLEX CONNECTOR

PARALLEL ECHO-BACK CONNECTOR

The Parallel Echo-back Connector is a Loop-Back testing module in which a peripheral can "talk" with itself to determine proper functioning of the peripheral. This Parallel Echo-back Connector can be constructed using the information below. Hand shake functions may also be tested with this configuration.

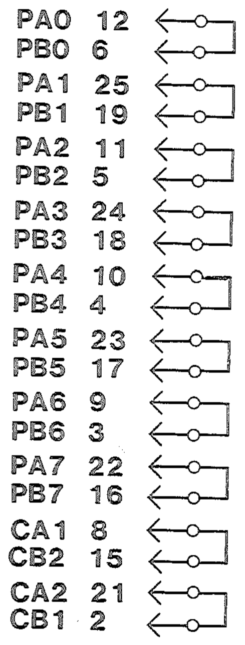
QTY	PARTS LIST
1	DB-25S Connector
A/R	Insulated 24g wire
A/R	Solder

DB-25S



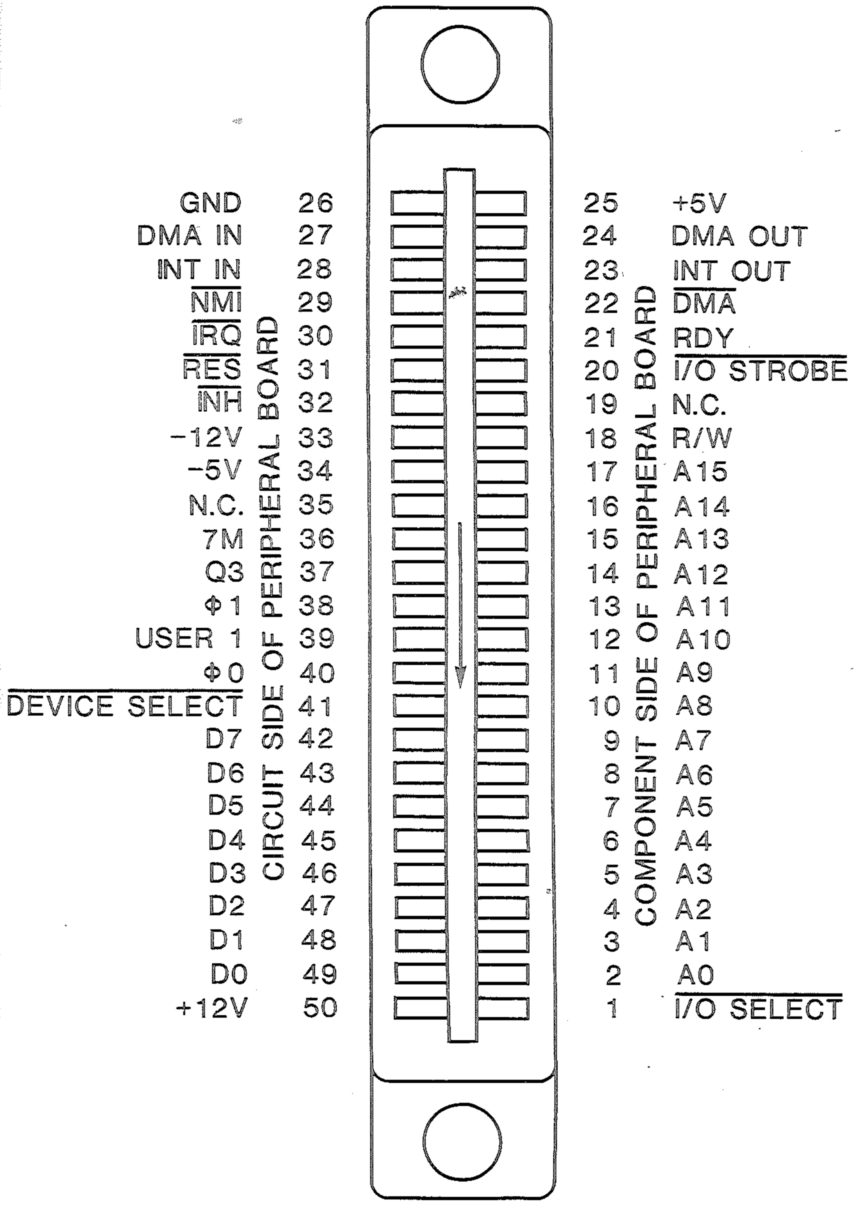
Back Side

SCHEMATIC



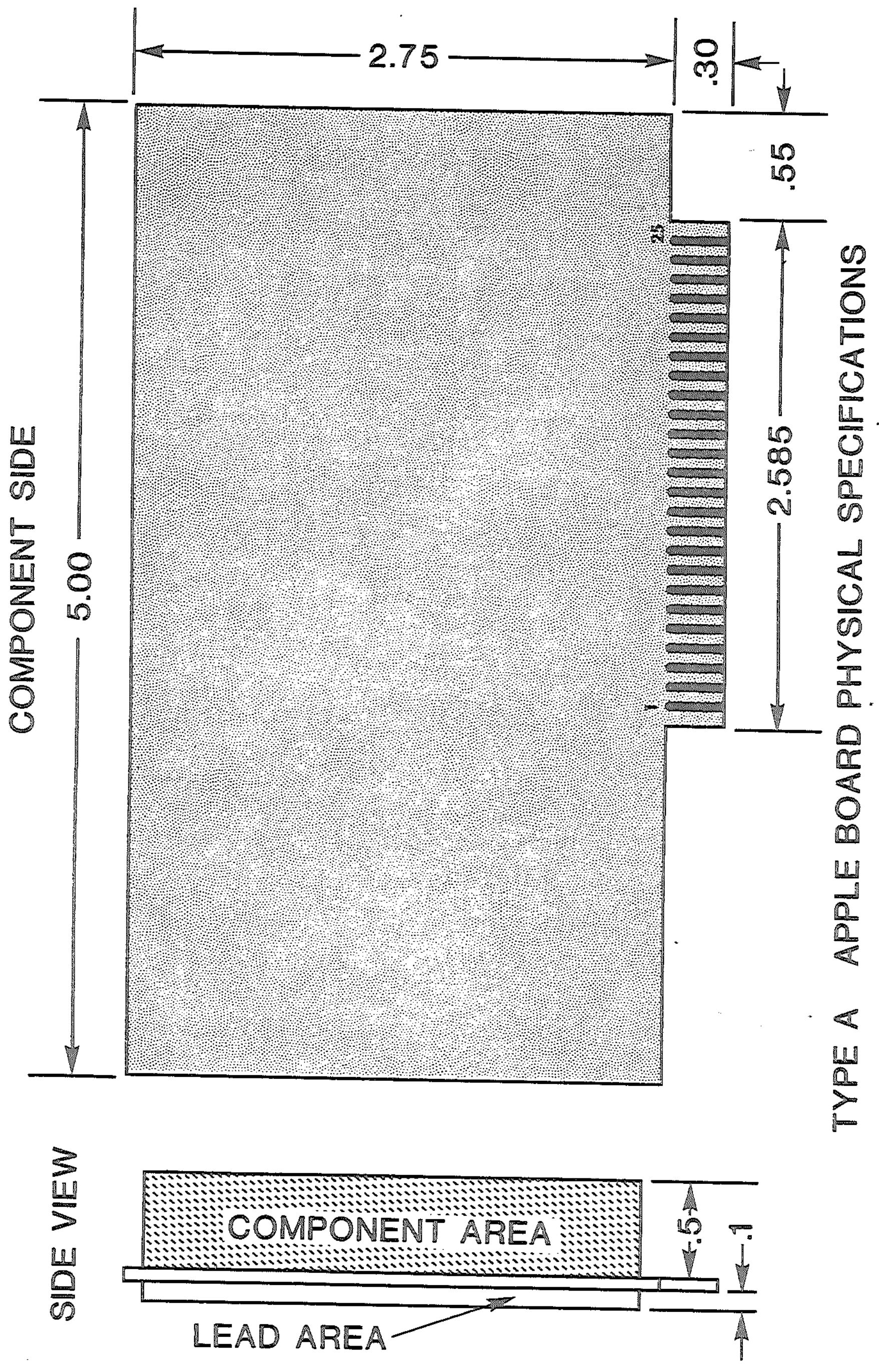
5.6 APPLE II I/O CONNECTOR PINOUT

TOP VIEW
BACK OF APPLE MAIN BOARD


GND 26
DMA IN 27
INT IN 28
$\overline{\text{NMI}}$ 29
$\overline{\text{IRQ}}$ 30
$\overline{\text{RES}}$ 31
$\overline{\text{INH}}$ 32
−12V 33
−5V 34
N.C. 35
7M 36
Q3 37
Φ1 38
USER 1 39
Φ0 40
$\overline{\text{DEVICE SELECT}}$ 41
D7 42
D6 43
D5 44
D4 45
D3 46
D2 47
D1 48
D0 49
+12V 50
CIRCUIT SIDE OF PERIPHERAL BOARD
25 +5V
24 DMA OUT
23 INT OUT
22 $\overline{\text{DMA}}$
21 RDY
20 $\overline{\text{I/O STROBE}}$
19 N.C.
18 R/W
17 A15
16 A14
15 A13
14 A12
13 A11
12 A10
11 A9
10 A8
9 A7
8 A6
7 A5
6 A4
5 A3
4 A2
3 A1
2 A0
1 $\overline{\text{I/O SELECT}}$
COMPONENT SIDE OF PERIPHERAL BOARD

FRONT OF APPLE MAIN BOARD

5.7 BOARD DIMENSIONS



APPENDIX A

INTERFACING THE CENTRONICS PRINTER

This firmware is a printer driver only and cannot be made to input data from a keyboard or any other source.

The firmware counts lines and will default to 54 lines per page and 12 lines between pages. You may change the lines per page by poking to location 1656 + the slot number. After the board is initialized, the number of characters per line defaults to 80. You may change the number of characters by poking to location 1528 + the slot number. The line count itself is in 1400 + slot number and the character count is in 1912 + slot number.

To use the Centronics firmware, you will have to build a cable to connect the board and the printer. The table below shows the pinouts for the Centronics 779 printer. Note that the firmware outputs on the A side of the 7720 Parallel Interface card. CA2 is the data strobe and CA1 is the acknowledge. PA0-PA7 are the data lines.

Pin Connections for the Centronics printer

Signal	7720 J2	7720 DB-25P	Centronics Connector
CA2	16	21	1
CA1	15	8	10
PA0	23	12	2
PA1	24	25	3
PA2	21	11	4
PA3	22	24	5
PA4	19	10	6
PA5	20	23	7
PA6	17	9	8
PA7	18	22	9
GND	1	1	14 and 16

```
0000               * CENTRONICS DRIVER PROGRAM FOR THE CCS 7720
0000               * ("B" ROM Version)
0000               *
0000               *
0000               * THIS PROGRAM IS AN OUTPUT DRIVER FOR THE PARALLEL
0000               * INTERFACE CARD FOR USE WITH CENTRONICS PRINTERS.
0000               *
0000               *
0000               *  STSTEM EQUATES
0000               MAXLN    EQU     $36             ;54 line default
0000               MAXCHR   EQU     $50             ;80 char/line default
0000               BKSP     EQU     $87             ;ASCII back space (for carry)
0000               LNFD     EQU     $8A             ;ASCII line feed
0000               FF       EQU     $8C             ;ASCII form feed
0000               CARRET   EQU     $8D             ;ASCII carriage return
0000               FS       EQU     $95             ;forward space
0000               SPACE    EQU     $A0             ;
0000               CPL      EQU     $5F8-$C0        ;max char save location
0000               LPP      EQU     $678-$C0        ;max line save location
0000               CH       EQU     $24             ;tab column
0000               CSWL     EQU     $36             ;location of output driver vector
0000               RANDL    EQU     $4E             ;random number seed location
0000               RANDH    EQU     $4F             ;
0000               LOCASE   EQU     $6F8-$C         ;hold for UC conversion mask
0000               LNCNT    EQU     $578-$C0        ;line counter
0000               CHCNT    EQU     $778-$C0        ;char counter
0000               DATAA    EQU     $C080           ;+$n0 for the PIA data port
0000               CMDA     EQU     DATAA+1         ;this is the PIA-A command port
0000               STATA    EQU     CMDA            ;this is the PIA-A status port
0000               DATAB    EQU     DATAA+2         ;+$n0 for the PIA data port
```

```
0000               CMDB     EQU     DATAB+1         ;this is the PIA-B command port
0000               STATB    EQU     CMDB            ;this is the PIA-B status port
0000               WAIT     EQU     $FCA8           ;time killer routine
0000               RETURN   EQU     $FFCB           ;used to find the slot address
0000               *
0000               *  THE COMMON CODE
0000               *
0000                        ORG     $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            BCS      DFB     $B0             ;always skip the 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 slot page number into Y
0014 BC 00 01               LDY     $100,X          ;
0017 8C F8 07               STY     $07F8           ;save slot high address
001A 68                     PLA                     ;recover the output data (if any)
001B 68                     PLA                     ;
001C 68                     PLA                     ;
001D 68                     PLA                     ;
001E 9A                     TXS                     ;restore the stack pointer
001F 48                     PHA                     ;save the data in the stack top
0020 98                     TYA                     ;get the slot page number
0021 AA                     TAX                     ;estab. X index
```

```
0022 0A                 ASL     A           ;multiply by 16 to get the $n0
0023 0A                 ASL     A           ;    index to access the PIA
0024 0A                 ASL     A           ;
0025 0A                 ASL     A           ;
0026 A8                 TAY                 ;estab. Y index
0027 E6 4E              INC     RANDL       ;increment the random number seed
0029 D0 02              BNE     COMA        ;
002B E6 4F              INC     RANDH       ;
002D 68         COMA    PLA                 ;get the saved status codes
002E 28                 PLP                 ;
002F 48                 PHA                 ;
0030 50 03              BVC     IO          ;routine IO
0032            *
0032            *  INITIALIZATION ROUTINE
0032            *
0032            *     THIS CODE HANDLES THE FIRST INPUT OR THE FIRST OUTPUT
0032            *  REQUEST AFTER INVOKING THE IN#N OR THE PR#N COMMANDS.
0032            *  IT CHECKS TO SEE WHETHER INPUT OR OUTPUT IS WANTED, THEN
0032            *  GOES TO THE APPROPRIATE CODE TO INITIALIZE THAT HALF OF
0032            *  THE PIA.
0032            *
0032 B8                 CLV                 ;clear the initialization flag
0033 50 02              BVC     OINIT       ;
0035 90 26      IO      BCC     OUT         ;normal output
0037            *
0037            *       OUTPUT INITIALIZATION
0037            *
0037 A9 05      OINIT   LDA     #5          ;set normal output entry point
0039 85 36              STA     CSWL        ;
003B A9 50              LDA     #MAXCHR     ;set defaults
003D 9D 38 05           STA     CPL,X       ;
0040 A9 36              LDA     #MAXLN      ;
0042 9D B8 05           STA     LPP,X       ;
```

```
0045 A9 00              LDA     #0          ;
0047 9D B8 04           STA     LNCNT,X     ;zero counters
004A 9D B8 06           STA     CHCNT,X     ;
004D 99 81 C0           STA     CMDA,Y      ;enable DDR-A access
0050 A9 FF              LDA     #$FF        ;set the DDR-B for output
0052 99 80 C0           STA     DATAA,Y     ;
0055 A9 3C              LDA     #$3C        ;normal control command
0057 99 81 C0           STA     CMDA,Y      ;
005A 99 83 C0           STA     CMDB,Y      ;
005D            *
005D            *  OUTPUT ROUTINE
005D            *
005D            *     THIS ROUTINE DOES THE ACTUAL OUTPUT OF THE DATA.  IT
005D            *  EXPECTS TO FIND THE DATA FOR OUTPUT ON THE TOP OF THE
005D            *  STACK (WHERE THE COMMON CODE PUTS IT).
005D            *
005D BD B8 06   OUT     LDA     CHCNT,X     ;see if tab wanted
0060 C5 24              CMP     CH          ;
0062 B0 03              BCS     OUTA        ;branch if no tab
0064 A9 A0              LDA     #SPACE      ;space out to column
0066 48                 PHA                 ;save on stack
0067 68         OUTA    PLA                 ;retrieve char
0068 48         OUTD    PHA                 ;resave it
0069 08                 PHP                 ;save tab flag
006A C9 8C              CMP     #FF         ;check for form feed
006C D0 0A              BNE     CRTLU       ;branch if not
006E 28                 PLP                 ;restore tab flag
006F BD B8 04           LDA     LNCNT,X     ;develop # lines
0072 FD B8 05           SBC     LPP,X       ;    to end of page
0075 38                 SEC                 ;ensure no tab on form feed
0076 D0 3C              BNE     LFA         ;go finish the FF
0078 C9 95      CRTLU   CMP     #FS         ;see if control U
007A F0 04              BEQ     CHCTI       ;branch if so
```

```
007C 29 60                  AND     #$60          ;test for other ctrl char
007E F0 03                  BEQ     OUTB          ;skip ctr increment
0080 FE B8 06    CHCTI      INC     CHCNT,X       ;bump counter
0083 28          OUTB       PLP                   ;reset tab flag
0084 68          OUTC       PLA                   ;
0085 48                     PHA                   ;
0086 99 80 C0               STA     DATAA,Y       ;
0089 A9 34                  LDA     #$34          ;
008B 99 81 C0               STA     CMDA,Y        ;
008E 49 08                  EOR     #$08          ;
0090 99 81 C0               STA     CMDA,Y        ;
0093 B9 81 C0    OUTE       LDA     STATA,Y       ;
0096 29 80                  AND     #$80          ;
0098 F0 F9                  BEQ     OUTE          ;
009A B9 80 C0               LDA     DATAA,Y       ;
009D 68                     PLA                   ;
009E 90 BD                  BCC     OUT           ;branch if tab
00A0 70 01                  BVS     LF            ;branch if self-gen char
00A2 48                     PHA                   ;resave char
00A3 C9 8A       LF         CMP     #LNFD         ;  +see if line feed
00A5 D0 14                  BNE     BS            ;no, branch
00A7 FE B8 04               INC     LNCNT,X       ;  bump line count
00AA 30 2F                  BMI     MAKELF        ;branch if eject in progress
00AC BD B8 04               LDA     LNCNT,X       ;  get the line count
00AF FD B8 05               SBC     LPP,X         ;ready for eject?
00B2 30 10                  BMI     DONE          ;no, done
00B4 E9 0C       LFA        SBC     #$0C          ;allow for margin
00B6 9D B8 04               STA     LNCNT,X       ;reset line count
00B9 D0 20                  BNE     MAKELF        ;do the eject
00BB E9 87       BS         SBC     #BKSP         ;see if backspace
00BD D0 13                  BNE     CR            ;no, branch
00BF DE B8 06               DEC     CHCNT,X       ;yes, adjust counter
00C2 30 12                  BMI     CRA           ;disallow negative count
```

```
00C4                *
00C4                *  FINAL COMMON CODE
00C4                *
00C4                *     THIS PART OF THE CODE RESTORES THE REGISTERS AND
00C4                *  RETURNS TO THE CALLER.  IT IS USED BY ALL OF THESE ROUTINES.
00C4                *
00C4 68             DONE    PLA                   ;set the stack straight
00C5 BA             DONA    TSX                   ;modify A register value in stack
00C6 E8                     INX                   ;   to ensure it is restored to
00C7 E8                     INX                   ;   right value
00C8 E8                     INX                   ;
00C9 9D 00 01               STA     $100,X        ;
00CC 68                     PLA                   ;restore registers
00CD A8                     TAY                   ;
00CE 68                     PLA                   ;
00CF AA                     TAX                   ;
00D0 68                     PLA                   ;
00D1 60                     RTS     DONE!         ;
00D2 E9 06          CR      SBC     #6            ;see if carr ret
00D4 D0 0D                  BNE     AUTOCR        ;no, branch
00D6 9D B8 06       CRA     STA     CHCNT,X       ;zero out counter
00D9 85 24                  STA     CH            ;   and tab pointer
00DB A9 8A          MAKELF  LDA     #LNFD         ;make a line feed
00DD 2C CB FF       SELF    BIT     RETURN        ;v=1 for self-gen
00E0 38                     SEC                   ;c=1 for no tab
00E1 B0 85                  BCS     OUTD          ;always branch
00E3 BD B8 06       AUTOCR  LDA     CHCNT,X       ;end of line
00E6 DD 38 05               CMP     CPL,X         ;   yet?
00E9 30 D9                  BMI     DONE          ;no, return
00EB A9 8D                  LDA     #CARRET       ;yes, get a carret
00ED D0 EE                  BNE     SELF          ;process it
00EF                        END
```

APPENDIX B

LIMITED WARRANTY

California Computer Systems (CCS) warrants to the original purchaser of its products that

(1) its CCS assembled and tested products will be free from materials defects for a period of one (1) year, and be free from defects of workmanship for a period of ninety (90) days; and

(2) its kit products will be free from materials defects for a period of ninety (90) days.

The responsibility of CCS hereunder, and the sole and exclusive remedy of the original purchaser for a breach of any warranty hereunder, is limited to the correction or replacement by CCS at CCS's option, at CCS's service facility, of any product or part which has been returned to CCS and in which there is a defect covered by this warranty; provided, however, that in the case of CCS assembled and tested products, CCS will correct any defect in materials and workmanship free of charge if the product is returned to CCS within ninety (90) days of original purchase from CCS; and CCS will correct defects in materials in its products and restore the product to an operational status for a labor charge of $25.00, provided that the product is returned to CCS within ninety (90) days in the case of kit products, or one (1) year in the case of CCS assembled and tested products. All such returned products shall be shipped prepaid and insured by original purchaser to:

Warranty Service Department
California Computer Systems
250 Caribbean Drive
Sunnyvale, California
94086

CCS shall have the right of final determination as to the existence and cause of a defect, and CCS shall have the sole right to decide whether the product should be repaired or replaced.

This warranty shall not apply to any product or any part thereof which has been subject to

(1) accident, neglect, negligence, abuse or misuse;

(2) any maintenance, overhaul, installation, storage, operation, or use, which is improper; or

(3) any alteration, modification, or repair by anyone other than CCS or its authorized representative.

THIS WARRANTY IS EXPRESSLY IN LIEU OF ALL OTHER WARRANTIES EXPRESSED OR IMPLIED OR STATUTORY INCLUDING THE WARRANTIES OF DESIGN, MERCHANTABILITY, OR FITNESS OR SUITABILITY FOR USE OR INTENDED PURPOSE AND OF ALL OTHER OBLIGATIONS OR LIABILITIES OF CCS. To any extent that this warranty cannot exclude or disclaim implied warranties, such warranties are limited to the duration of this express warranty or to any shorter time permitted by law.

CCS expressly disclaims any and all liability arising from the use and/or operation of its products sold in any and all applications not specifically recommended, tested, or certified by CCS, in writing. With respect to applications not specifically recommended, tested, or certified by CCS, the original purchaser acknowledges that he has examined the products to which this warranty attaches, and their specifications and descriptions, and is familiar with the operational characteristics thereof. The original purchaser has not relied upon the judgement or any representations of CCS as to the suitability of any CCS product and acknowledges that CCS has no knowledge of the intended use of its products. CCS EXPRESSLY DISCLAIMS ANY LIABILITY ARISING FROM THE USE AND/OR OPERATION OF ITS PRODUCTS, AND SHALL NOT BE LIABLE FOR ANY CONSEQUENTIAL OR INCIDENTAL OR COLLATERAL DAMAGES OR INJURY TO PERSONS OR PROPERTY.

CCS's obligations under this warranty are conditioned on the original purchaser's maintenance of explicit records which will accurately reflect operating conditions and

maintenance preformed on CCS's products and establish the nature of any unsatisfactory condition of CCS's products. CCS, at its request, shall be given access to such records for substantiating warranty claims. No action may be brought for breach of any express or implied warranty after one (1) year from the expiration of this express warranty's applicable warranty period. CCS assumes no liability for any events which may arise from the use of technical information on the application of its products supplied by CCS. CCS makes no warranty whatsoever in respect to accessories or parts not supplied by CCS, or to the extent that any defect is attributable to any part not supplied by CCS.

CCS neither assumes nor authorizes any person other than a duly authorized officer or representative to assume for CCS any other liability or extension or alteration of this warranty in connection with the sale or any shipment of CCS's products. Any such assumption of liability or modification of warranty must be in writing and signed by such duly authorized officer or representative to be enforceable. These warranties apply to the orginal purchaser only, and do not run to successors, assigns, or subsequent purchasers or owners; AS TO ALL PERSONS OR ENTITIES OTHER THAN THE ORIGINAL PURCHASER, CCS MAKES NO WARRANTIES WHATSOEVER, EXPRESS OR IMPLIED OR STATUTORY. The term "original purchaser" as used in this warranty shall be deemed to mean only that person to whom its product is originally sold by CCS.

Unless otherwise agreed, in writing, and except as may be necessary to comply with this warranty, CCS reserves the right to make changes in its products without any obligation to incorporate such changes in any product manufactured theretofore.

This warranty is limited to the terms stated herein. CCS disclaims all liability for incidental or consequential damages. Some states do not allow limitations on how long an implied warranty lasts and some do not allow the exclusion or limitation of incidental or consequential damages so the above limitations and exclusions may not apply to you. This warranty gives you specific legal rights, and you may also have other rights which vary from state to state.