TATE LAB

TATE

CRYOMAGNETICS, INC.

OPERATING INSTRUCTION MANUAL

FOR

MODEL CIM - COMPUTER INTERFACE MODULE

Serial No L295E

WARNING: DO NOT ATTEMPT TO OPERATE THIS EQUIPMENT BEFORE YOU HAVE THOROUGHLY READ THIS INSTRUCTION MANUAL.

INNOVATION AND EXCELLENCE IN CRYOMAGNETICS

739 EMORY VALLEY ROAD
OAK RIDGE, TN
MAILING ADDRESS:
P.O. BOX 548 OAK RIDGE, TN 37831-0548

TELEPHONE 615-482-9551
TELEX 883-945
FAX 615-483-5891

INTRODUCTION

The **CIM** Computer Interface is a versatile module capable of providing a variety of the scanning, counting, and communications functions typically required in the laboratory. It is remotely programmable via both RS232 and GPIB rear panel connectors and can be used with laboratory computers or simply with a terminal. Eight rear panel analog ports can be programmed as inputs or outputs with a range of ±10.24 volts. Two rear panel digital input/output bits are provided for general use as well as an eight-bit digital input/output port accessible via the rear panel digital I/O connector. The digital ports can be used to interface the **CIM** with the IPS-100 magnet power system .A variety of simple, easily understood commands eliminate the need for extensive software development to integrate the **CIM** into existing experiments.

SPECIFICATIONS

ANALOG PORTS	
Inputs	1 MΩ input impedance, +10.24 VDC range, protected to 40 VDC. 13-bit resolution (2.5mV). Accuracy: 0.05%. Input offset less than 2.5 mV. Maximum A/D rate is 2 KHz.
Outputs	Output impedance less than 1Ω. Short circuit current limit is 20 mA. 13-bit resolution (2.5mV). Accuracy: 0.05%. Output offset less than 2.5mV.
DIGITAL PORTS	
Input Bits	Input impedance greater than 100KΩ. Minimum pulse width is 200nS. Maximum count rate is 4MHz. Logic one > 3 VDC. Logic zero < 0.7 VDC. Inputs protected to +10 VDC.
Output Bits	Can drive loads up to 50Ω to TTL logic levels. 2 definable input or output bits. 8 latched TTL output bits with strobe bit. 8-bit TTL input port with strobe bit.
INTERFACES	Both IEEE-488 Std Port and RS232-C(300 to 19.2Kbaud).
SYSTEM COMPONENTS	Z80-A µP @ 4 MHz. 8K ROM, 8K RAM. VLSI counters, UART, and IEEE-488 adapter.
GENERAL	
Power	117 / 220 VAC
Mechanical	Dimensions 19"x14.5" x 3.5" rack mount
Warranty	One year parts and labor on materials and workmanship.

SETUP & OPERATION

TRYING OUT THE CIM

Before attempting any detailed programming with the CIM it is best to get a feel for the type of functions the module provides by using it with a terminal. If one is available, connect a terminal with an RS232 port to the RS232 port on the rear of the CIM. The unit is shipped with its internal configuration switch set for 9.6 kbaud operation. If your terminal requires a different baud rate it will be necessary to open the top panel and readjust the configuration switch according to the values in Figure 1. After setting the correct baud rate and connecting the terminal, turn the unit on. The green ACTIVITY light will flash for a second or two indicating that the CIM is performing its internal hardware test routines and the sign-on message will appear at the terminal giving the version number of the ROM supplied with the unit. The red ERROR LED may also momentarily flash when the power is turned on; this does not indicate any problem with the unit. If the sign-on message is not displayed, consult Appendix A, Example1 for more detailed instructions. The last characters displayed will be the prompt 'OK->'. This indicates that the CIM is ready to accept commands.

Now type the command ?1 followed by a carriage return. The CIM responds by sending to the terminal the characters 0.000 indicating that the voltage at port 1 is 0 volts. In general, the command ?n causes the the CIM to send back the value at the nth analog port. Now we will use the unit to output a voltage we can read with the ? command. Type the command I4<cr>. This causes the CIM to configure the first 4 analog ports as inputs, with the remainder becoming analog output ports. For instance, if we wish to set the voltage at port 8 equal to 5.0 volts type S8=5.0<cr> (Set port 8 equal to five volts). We can read the voltage just set by typing ?8<cr> which will return with the value 5.000. Note that the ?<n> can be used to read the voltage at a port regardless of whether it is configured as an input or an output. The ability to use the eight analog ports as either inputs or outputs gives the CIM tremendous flexibility in laboratory situations. At this point, the user should review Figure 2 containing the complete command list for the CIM and experiment with a few of the commands.

COMMUNICATING WITH THE CIM

The CIM is programmed by sending it strings of ASCII characters via the RS232 or GPIB connectors. The choice of RS232 or GPIB, as well as the details of the actual communications interface used (parity bits, GPIB address, baud rate, etc.) is made by setting the 8 configuration switches inside the unit according to the table in Figure 1 The configuration switch is read only when the unit is turned on or after a GPIB 'device clear' message is received. Thus, changing the value of the switch without clearing the unit or turning it off and on again has no effect. Once the CIM receives characters over either bus, it looks for the ASCII carriage return character, which signals the end of a line of commands, and processes the commands. Within a given line, there may be many individual commands; these must be delimited by the semicolon (;) character. For instance if we wanted to know the voltages on all of the analog ports we could send the command string:

?1;?2;?3;?4;?5;?6;?7;?8<cr>

and the unit would respond by returning all the port values. It is not necessary in most cases to wait until the CIM has finished processing a given line of commands before sending the next: the unit automatically queues the commands and executes each as it is ready. (An exception to this occurs when the unit is operated in synchronous mode.)

The various commands cause the unit to either alter its internal state or send various values back over the bus. Values which are sent by the CIM are also in the form of strings of ASCII characters followed by a string of terminating characters, usually carriage returns and line feeds. For instance, in responding to the ?1 command the unit might send the string 2.357<cr> to indicate the value 2.357 volts. The choice of terminating characters is determined by whether the CIM is used in the RS232 or GPIB mode and whether the 'echo' feature is in use. In addition, special terminating sequences may be specified by using the Z command. The default terminating sequences for each of the various modes is shown below. Note that in the GPIB mode the final terminating character is accompanied by the EOI (End or Identify) message.

Setting The Configuration Switch

RS232 Operation

Bit	Setting	Explanation
8	OFF	Set RS232 operation
7	ON	Echo mode (terminal operation)
	OFF	Disables echo (computer operation)
6	ON	Two stop bits
	OFF	One stop bit
5	ON	Enable parity (8 data bits and 1 parity bit)
	OFF	Disables parity (8 data bits and no parity bit)
4	ON	Even parity
	OFF	Odd parity
3, 2, 1		RS232 Baud Rate (see below)

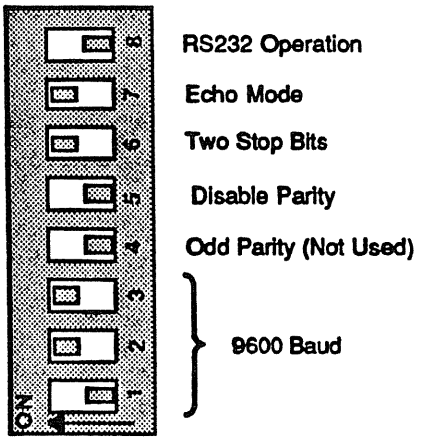
Baud Rate	Bit 3	Bit 2	Bit 1
300 Baud	OFF	OFF	ON
600 Baud	OFF	ON	OFF
1200 Baud	OFF	ON	ON
2400 Baud	ON	OFF	OFF
9600 Baud	ON	ON	OFF
19.2 Kbaud	ON	ON	ON

GPIB Operation

Bit	Setting	Explanation
8	ON	Set GPIB Operation
7	ON	RS232 Echo mode for GPIB
	OFF	GPIB operation without Echo
6	ON	2 stop bits for RS232 echo.
	OFF	1 stop bit for RS232 echo.
5, 4, 3, 2, 1		GPIB address for module (see below)

GPIB Address	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1
0	OFF	OFF	OFF	OFF	OFF
1	OFF	OFF	OFF	OFF	ON
2	OFF	OFF	OFF	ON	OFF
.			.		
.			.		
.			.		
28	ON	ON	ON	OFF	OFF
29	ON	ON	ON	OFF	ON
30	ON	ON	ON	ON	OFF

RS232 Example



GPIB Example

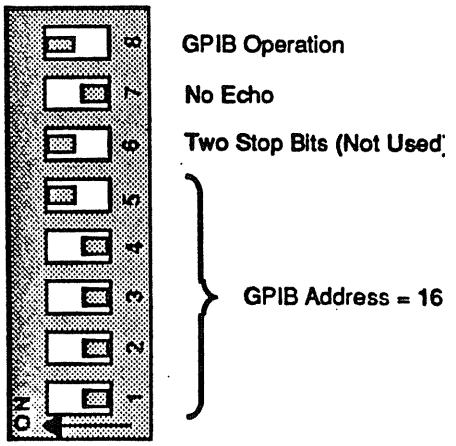


Figure 1 - The Configuration Switch

DEFAULT TERMINATION CHARACTERS

Interface	Echo	No Echo
RS232	CR, LF	CR
GPIB	CR,LF (EOI)	CR,LF (EOI)

Note that the terminating characters are sent with each value returned by the **CIM**. Thus, in responding to the command string ?1;?B1;?3<cr> while in the RS232 non-echo mode, the unit would send a string such as 2.000<cr>1<cr> 4.875<cr>.

Some commands deal with byte quantities, such as the internal 8-bit digital input/output ports. These values are both sent and received as ASCII coded decimal quantities. For instance, if the user wished to set the value of the digital output port to:

Bit: 7 6 5 4 3 2 1 0
Setting: 0 0 0 1 0 1 1 0

he would send the command SD=22<cr> because 00010110 binary is equal to 22 decimal. Likewise, if the same binary value were present at the digital input port and the command ?D was sent, the unit would respond by sending the string 22<cr>.

FRONT PANEL LEDS

The green front panel LED flashes when the **CIM** is sending or receiving information over the RS232 or GPIB interfaces. If the green light remains on while the **CIM** is not receiving commands, it indicates that the **CIM** is trying to send data over the bus but cannot. This will occur, for instance, if the RS232 'clear to send' line is never asserted, or in the GPIB mode if the unit is never made a talker after requesting data. The red front panel LED flashes whenever an error has occurred, such as an undefined or illegal command string, an A/D overflow, or missed data.

ECHO AND NO ECHO OPERATION

In order to allow the **CIM** to be operated from a terminal, an echo feature has been added which causes the unit to echo back commands received over the RS232 port. This feature is enabled by setting bit 7 of the internal configuration switch ON. In this mode the **CIM** will send linefeeds in addition to carriage returns with each value and will also send the prompts 'OK->' and 'ERR->' to indicate that the previous command was either processed or contained an error. When the unit is controlled by a computer, the echo feature should be turned off to prevent the sending of spurious characters. The echo feature can also be used with the GPIB as explained in the section on GPIB operation.

DTE AND DCE OPERATION

The **CIM** contains an internal jumper plug which allows it to emulate either a DTE (terminal) or a DCE (modem) while operating in the RS232 mode. The unit is shipped as a DCE so that it may immediately be connected to a terminal for check out. If it is desired to configure the unit as DTE (connection to many computers will require this) simply pull out the jumper plug from the socket marked 'DCE' on the circuit board and plug it into the socket marked 'DTE'. The plug is located at the rear of the board directly behind the RS232 connector.

PROGRAMMING

In this section, each of the **CIM** commands will be discussed in detail. Commands are denoted by boldface characters. The brackets < > indicate that the item named inside the brackets is to be sent with the command. The brackets are never sent as part of an actual command string. Figure 2 provides a brief summary of all the **CIM** commands. The symbol <cr> denotes an ASCII carriage return character.

INPUT / OUTPUT COMMANDS

The **CIM** contains three types of input/output ports accessible to the user: the eight analog input/output ports which are referred to simply by their number, 1 through 8; the two front panel bit input/output ports, referred to as B1 and B2; and the 8 bit digital input/output port which is referred to as D. The basic format for interrogating and setting the value of each of these types of ports is the same. To interrogate a port, the command ?<port name> is sent. For instance, to interrogate the value of B2 the user would send ?B2<cr> and the **CIM** would respond by sending back either '1' or '0' depending on whether the value at B2 was TTL high or low. To set the value of a port, the command S<port name>=<value><cr> is sent. Thus, to set port 6 to 3.456 volts, we would send the command S6=3.456<cr>. The analog ports may have any value between -10.237 volts and +10.237 volts, the digital bits B1 and B2 are either TTL low or high, and the value of the digital port D may range from 0 to 255.

Certain commands allocate the ports as either inputs or outputs. When the unit is turned on, or after a MR (master reset) command is received, analog ports 1 through 8 and both B1 and B2 are configured as inputs. To configure only the first n analog ports as inputs, with the remainder becoming outputs, the command I<n><cr> is sent. The two front panel digital bits are configured as outputs simply by setting them equal to the desired value, e.g. SB2=1<cr> configures B2 as an output and sets its value to TTL high. To reconfigure the bits as inputs, the command SB<n>=I<cr> is used. Note that certain commands automatically reconfigure the input/output status of B1 and B2 regardless of their previous value. For instance if the PB1 command is sent (Pulse Bit 1), B1 is reconfigured as an output even if SB1=I had been previously sent.

SYNCHRONOUS AND ASYNCHRONOUS MODES

The **CIM** can operate in one of two modes, called the synchronous and asynchronous modes, which dictate when the unit will send back the requested values in response to a ? command. In the asynchronous mode, the power on default mode, the unit returns each value immediately after it is requested. Setting synchronous mode operation allows the **CIM** to respond to certain commands only in response to an external trigger signal. When the synchronous mode is set using the MS command, front panel bit 1 (B1) is automatically configured as an input and reserved for use as a trigger. In this mode, when a ? command is sent, the **CIM** will wait until a trigger is received at B1, at which time it will sample the requested values and send them back over the bus. A 'trigger' consists of the falling edge of a standard TTL pulse. Note that all the commands on a given line are processed after the first trigger following the carriage return which delimits that line of commands. In other words, if the string ?1;?2;?3<cr> is sent in the synchronous mode the unit will wait for a trigger and then send the values of ports one, two, and three.

While the **CIM** is waiting for a trigger, commands may be sent to it normally. If operating in the RS232 echo mode, the next 'OK->' prompt will not be displayed until a trigger is received and the previously requested values are sent. Note that if another string containing ? commands is received while the unit is still waiting for a trigger to send back the result from a previous ? command, the first command is flushed and the next trigger will cause the unit to respond only to the latest ? command.

TRIGGER COMMANDS

Certain commands modify the properties of the trigger input. For instance, if the user wishes every nth pulse at B1 to trigger the unit, he can give the command T<n><cr> while in the synchronous mode. The value of n can range from 1 to 32767.

A trigger can be induced without externally pulsing B1 by sending the PB1 command (Note that this will leave B1 configured as an output.) The trigger can be masked by sending the command DT (disable trigger). To re-enable the trigger after this command has been sent, the command ET (enable trigger) is used.

PULSING COMMANDS

Both B1 and B2 can be used as general purpose TTL pulse sources using the P command. Sending PB1<cr> or PB2<cr> configures the appropriate bit as an output and outputs a 10µsec positive TTL pulse. The **CIM** may be programmed to act as a 'divide by n' counter by sending the command P/<n><cr> when the unit is in the synchronous mode. This command causes the **CIM** to output a 10µsec pulse on B2 every nth trigger. Note that the effects of the T command and the P/<n> commands are cumulative. Thus, if the user had set T10 and then gave the command P/5, a pulse would be output at B2 for every 50th input pulse at B1.

COUNTING COMMANDS

The **CIM** may be used as a general purpose counter using the C and ?C commands. Sending the C command configures B2 as an input and initializes the counter. Sending ?C<cr> causes the **CIM** to return the number of pulses counted at B2 and clears the counter in preparation for a new count. Note that the C command does not have to be sent before each ?C , but if B2 is ever reconfigured as an output the C command must be sent to reinitialize the counter before using ?C again. The maximum count rate is 4MHz. The maximum count is 65,535. The counter 'wraps around' to 0 and continues counting after the maximum count is reached.

SCAN COMMANDS

One of the most useful features of the **CIM** is its ability to independently read the input ports and store a series of data points. The scan command format is:

SC<port 1>,<port 2>,...<port n>:<num of triggers>

A maximum of 8 ports may be specified. This command causes the unit to sample and store the values of specified input ports each time a trigger is received. This process is repeated for the number of triggers specified in the command. For instance if we wanted to scan the values of ports 1 and 3 as well as the digital port for 100 triggers we would give the command:

SC1,3,D:100

The eight analog ports and the 8-bit digital port may be scanned in this manner. The two bits B1 and B2 may not be included in the scan command. When the required number of triggers has been received, the **CIM** stops scanning and sets the 'scan finished' bit (bit 4) in the status byte. To stop the scan prematurely, the command ES (end scan) may be used. The number of triggers received since the scan started is returned by the ?N command which may be sent any time during the scan. Other commands may be sent while a scan is in progress, but, since processing these commands takes time, this practice is not recommended, especially at scan rates close to the maximum. The maximum scan rates, as well as the maximum number of points in a scan is shown in the table below.

Maximum Scan Parameters (SC Command)

# of ports	Max trig rate	Max # of triggers
1	2.1 KHz	3711
2	1.3 KHz	1855
3	910 Hz	1237
4	740 Hz	927
5	600 Hz	742
6	510 Hz	618
7	440 Hz	530
8	390 Hz	463

Once a scan has been completed the user may read the stored values with the N (next point) command. Each time the N command is sent, the **CIM** returns the next value in the scan. The value of the ports are sent in the same order as they were specified in the scan command. Thus if SC4,6,1:50 were sent, the scan points would be sent in the following order:

Command List

Input/Output Commands

I<n> n=0-8 Designates the first n analog ports as inputs, the remainder become outputs. Default is I8.

?<n> n=1-8 [?3;?5] Returns the value of the designated analog port.

?B<n> n=1,2 [?B1] Returns the value (0 or 1) of the designated digital bit port.

?D Returns the value of the internal 8 bit digital input port.

?S Returns the value of the status byte, and clears the status byte.

C Configures B2 as an input and resets the B2 counter.

?C Returns the number of pulses occurring at B2 since the previous ?C.

S<n>=<x> n=1-8; x=-10.237 to +10.237 [S8=6 or S2=-41.5E-2] Sets the analog port n (which must be designated as an output) to the value x Volts. x may also be expressed in exponential format with a two digit exponent preceded by an E.

SB<n>=<m> n=1,2; m=0,1 [SB1=1] Designates digital bit n as output and sets its value to TTL low if m=0 or high if m=1 .

SB<n>=I n=1,2 [SB2=I] Designates the selected bit as an input.

SD=<n> n=0-255 [SD=128] Sets the 8-bit digital output port to the value n.

SM=<n> n=0-255 [SM=16] Sets the GPIB SRQ mask to the value n (See GPIB discussion).

Trigger Commands

MS Sets the synchronous mode. Responses to ? commands are returned immediately after the next trigger.

MA Sets the asynchronous mode (default). Responses to ? commands are returned right after command is received.

T<n> n=1 to 32,767 [T3] Designates every nth pulse at B1 as a trigger.

DT Masks the trigger input so that no triggers are recognized.

ET Unmasks the trigger input.

PB<n> n=1,2 [PB2] Outputs a 10µsec TTL pulse at bit port n.

P/<n> n=1-255 [P/3] Outputs a 10µsec TTL pulse at B2 each nth trigger.

Scan Commands

SC<i>,<k>:<n> i..k=1-8,D [SC1,3,D:500] Scans the list i..k of analog ports or digital port for n triggers. Total # of samples may not exceed 3711.

ES End the current scan immediately and reset the point sending counter.

N Send the next point of stored scan.

?N Returns # of points scanned.

A<n>,<i> n=1-255; i=1-255 [A16,2] Adds n x 2.5mV to the value of analog port 8 (must be a positive output) on every ith trigger.

SS<i>,<k>:<n> i..k=1-8,D [SS1:10] Scans the list i..k of analog ports or digital port for n triggers. Data is sent in a 2 byte binary format while the scan is in progress.

X Sends the data of a stored scan in 2 byte binary format.

Miscellaneous Commands

MR Master Reset. Returns the **CIM** command settings to their default values.

W<n> n=0-255 Introduces a delay of approximately n x 400µsec before sending each character over the RS232. Default is W255. Allows slow peripherals to keep up.

Z<i>,<k> [Z13,13] Changes the end-of-record characters sent by the **CIM** to those specified by the ASCII codes, i...k.

Figure 2 - The CIM Command List

port 4 at 1st trigger
port 6 at 1st trigger
port 1 at 1st trigger
port 4 at 2nd trigger
.
.
.
port 1 at 50th trigger

The N command may not be sent while a scan is in progress. If N is sent after the entire scan has already been read, a 'command parameter out of range' error is generated setting bit 2 in the status byte and blinking the red LED, and no value is returned. The 'next point' pointer can be made to point to the beginning of the scan at any time by using the ES command. The pointer is automatically reset to the beginning of the scan following completion of a scan.

The CIM has a limited capacity to ramp one of its analog output ports during a scan. The command A<n>,<m> causes the unit to add n x 2.5 mV to the value of port 8 (port 8 must be configured as output) at each mth trigger. Using the A command reduces the maximum trigger rate by about 20%. The starting value of port 8 should be set using the S8= command and port 8 must have been set as an output with the I command. Both m and n may range from 1 to 255. When the output reaches its maximum value of +10.237 volts it 'wraps around' and begins ramping at 0. The A command may not be sent when the value of port 8 is negative.

RECEIVING DATA WHILE SCANNING: THE SS COMMAND

It may sometimes be desirable to receive scan data from the CIM while a scan is in progress. This can be accomplished with the SS command;

SS<port 1>,<port 2>,...,<port n>:<num of triggers>

When this command is received the CIM begins scanning the designated ports as in a normal scan. Instead of storing the data so that it may be read back later the CIM sends each scan point over the bus as it is read. The values of the ports are not sent as ASCII characters in this case. Analog ports are sent as 2 binary bytes which represent a sign bit and 12 data bits. The first byte sent contains the sign bit (1=negative, 0=positive) in the bit 4 position and the high 4 data bits in bits 3 through 0. The second byte sent contains data bits 7 through 0. The analog voltage may be reconstructed by multiplying the signed integer by 0.0025 volts. The digital port value is sent as a single byte preceded by a dummy identifier byte with a value of 0FFH (11111111 binary). At the conclusion of the scan, two dummy bytes with the value 0FFH are sent when in the RS232 mode. In the GPIB mode, the last byte is sent with the End-or Identify message (EOI).

When the SS command is given, the user should be prepared to read the data which is being sent by the CIM as fast as it is being accumulated. Untransmitted data bytes are temporarily buffered up to a maximum number of 7420. If the number of unread data bytes exceeds 7420, the scan is stopped and a 'missed data' error is generated. Note that the normal restrictions on the number of points scanned do not hold for the SS type of scan. The only restriction in this case is that the total number of bytes sent during the scan must be less than 65535. The number of bytes may be calculated simply by multiplying the number of points in the scan by the number of ports scanned at each point times 2 bytes per port. Thus, SS1:25000 is a legitimate command since it only requires sending 50,000 bytes.

Because the SS command may require that the user be able to acquire unformatted binary data at high data rates, it will not be usable with all languages or with all computers. Special I/O drivers may be necessary to take advantage of this feature. Examples of how this is done are provided in Appendix A. If the additional power provided by the SS command is not required, the user may acquire data in a much simpler manner using the SC and N commands.

FAST TRANSFER OF SCAN DATA: THE X COMMAND

When many data points are acquired during a scan, the amount of time necessary to transfer the scan data to the user's computer using the N command may become significant. To allow for faster transfer of scan data, the X command can be used to cause the CIM to send the entire scan over the

RS232 or GPIB lines in binary format. When the X command is received, the CIM first waits for approximately 37*W mS to allow the user's computer time to prepare for the incoming data. W here is the 'wait' value that is set using the W command. After waiting for this time period, the CIM begins sending the stored scan data, byte by byte, in the same format used in the SS command; i.e. the first byte contains the sign bit and the 4 high data bits, and the second byte contains the low 8 data bits. The digital port value is sent as a single byte preceded by an 'identifier byte' equal to 0FFH (11111111 binary).

If, at any point during the scan dump, the CIM has to wait for more than 9 mS for the 'clear to send' signal in the RS232 mode or for 'ready for data' (RFD) from all listeners in the GPIB mode, the scan dump is abandoned and the remainder of the data is not sent. Thus, using the X command requires the user's computer to become a dedicated listener while the data is being transferred. When all the data has been sent, the CIM finishes the dump by sending two terminating 'FF (hex)' bytes in RS232 or a single 'FF (hex) byte with the EOI message in GPIB. Note that these terminating characters are always sent when the X command is used and are not affected by the choice of terminating characters made with the Z command.

MASTER RESET

The MR (master reset) command resets the unit to its default state. All ports are configured as inputs. Any data which is waiting to be sent is lost. Therefore it is bad practice to place a MR command on the same line following a ? command as the unit will send an indeterminate number of characters in response to the ? command before processing the MR command and flush the remainder of the response.

WAIT COMMAND

The CIM normally waits until the 'Clear to Send' line on the RS232 bus is asserted before sending characters over the bus. However, as some peripherals may not bother to set and reset the CTS line, it is possible for the CIM to send data out at a rate too fast for some computers and terminals to handle. The CIM may be 'slowed down' using the W command. Sending W<n><cr> (0<n<255) causes the unit to wait about n x 400 usec before sending each character over the RS232 bus. Note that the default setting for the wait parameter is W255. Thus, to operate at the fastest data rate over the RS232, it is necessary to set the wait parameter to some lower value each time the unit is powered on.

SETTING THE TERMINATION SEQUENCE: THE Z COMMAND

Although the default terminating characters described above will suffice for interfacing the CIM with a wide variety of hardware and software, it will occasionally be necessary to send special terminating sequences to fit the requirements of some computers. This can be done with the Z command. The format for the command is:

Z<n1>,<n2>,<n3>,<n4>

where n1, n2, n3, and n4 are decimal values between 0 and 255 corresponding the the ASCII codes of the desired termination characters. For instance, if we wanted to send each value followed by the "*" character (ASCII 42), two carriage returns, (ASCII 13) and a line feed (ASCII 10) we would send:

Z42,13,13,10

If a ?1 command was received after this command had been sent and the voltage at port 1 was 2 volts, the string of characters sent back by the CIM would be 2.000*<cr><cr><lf>. Up to 4 terminating characters may be sent with the command. Note that the Z command only affects the characters sent by the CIM. Command lines received by the CIM must always be terminated by the carriage return character regardless of the Z command.

For GPIB, the 'E' character (ASCII 69) has special significance. Requesting 'E' as part of a terminating sequence causes the character preceding the 'E' to be sent with the end or

identify (EOI) message (an ASCII 'E' will not be sent.) The 'E' character should always be the last character in the terminating sequence when using the GPIB; when not using the GPIB-RS232 echo mode, the 'E' character must be the last character in the terminating string.

STATUS BYTE

The CIM contains an 8-bit status register which the user may read to obtain information on the unit's status. The status byte may be read in two ways: by sending the ?S command, which returns the value of the byte in ASCII coded decimal, or when using the GPIB, by performing a serial poll. The status byte returned by the unit reflects all of the status conditions which have occurred since the last time the byte was read After the status byte has been read, it is cleared by the CIM . Thus, in situations where it is critical to monitor a status condition, the status byte should be read initially to clear all previous conditions. The status bits are described in detail in Figure 3.

ERRORS

Whenever a 'parameter out of range' or an 'unrecognized command' error occurs, the appropriate status bits are set and the red LED flashes. In addition, the command queue is reset so that any commands which were pending at the time the error occurred are not executed. Note that an 'A/D overflow' or a 'missed data' error do not reset the command queue or interrupt the processing of commands. In the former case, the overflow value is stored as 10.237 volts and in the latter, the scan is continued even though triggers are being missed. It is the user's responsibility to check for these errors.

The Status Byte

Bit	Value	Explanation
b7	128	**Busy.** When this bit is set it indicates the CIM has one or more unprocessed commands pending on its command queue. For RS232, this bit is always high as the ?S command will itself be an unprocessed command.
b6	64	**SRQ.** This bit indicates that the CIM has requested service from the GPIB controller. This occurs when one of the status bits which the user has 'unmasked' with the SM= command is set. The SRQ bit is reset when the controller performs a serial poll and reads the status condition which generated the SRQ.
b5	32	**Trigger received.** Indicates that the CIM has been triggered.
b4	16	**Scan finished.** This bit is set at the completion of a scan.
b3	8	**Missed data.** This bit is set when the trigger rate in the synchronous mode exceeds the allowed maximum (see Note 1.)
b2	4	**Command parameter out of range.** This bit is set if a parameter associated with a command is not in the allowed range. For instance, S8=45 or SC1:10000 will both generate out of range errors (see Note 2.)
b1	2	**A/D overflow.** This bit is set whenever the unit reads a value at one of the input ports which exceeds +10.237 volts.
b0	1	**Unrecognized command.** Indicates that the unit has received an illegal command string (see Note 3.)

Note 1: Missed Data

1) External trigger rate is too fast.
2) Command queue overflow.
3) Transmission buffer overflow.
4) Available buffer space exceeded in SS command.

Note 2: Parameter Out of Range

1) Scan command designates more than 8 ports to be scanned.
2) Number of scan points specified in the SC command is 0 or greater than the maximum allowed.
3) SS command requires sending greater than 64k bytes.
4) The A command is sent when port 8 is <0.
5) Either parameter in A command is >255 or second parameter =0.
6) Any parameter in Z command is >255.
7) ?C command is sent when B2 is an output.
8) S command specifies an analog port configured as an input.
9) Parameter in I command is >8 or <0.
10) Analog value in S command is >10.2375 volts or <-10.2375.
11) Parameter in SD command is >255 or <0.
12) Parameter in SM command is >255 of <0.
13) Try to set nonexistent port or bit with S command.
14) Parameter in SB command is <0 or >1.
15) Time out on X command.
16) N command received during a scan.
17) X command received during a scan.

Note 3: Unrecognized Command

1) Illegal character in a command string.
2) <cr> or ';' not detected at end of command.
3) Non numeric value detected in a numeric string.
4) '=' character missing in S command.

Figure 3 - The Status Byte

COMMUNICATIONS

INTRODUCTION TO RS232

The RS232 is a standard for bit serial asynchronous data communication. The standard defines the format for data transmission, electrical specifications for signal levels, and mechanical dimensions of connectors.

Despite the existence of a definition of a standard, there are so many permutations of control lines, data formats, and transmission speeds, that getting two RS232 devices to communicate usually requires some work. In this section, we will provide some basic information to aid you in connecting your RS232 device to the Computer Interface.

Case 1 - The Simplest Configuration

In this case, one wire is used to send data from device A to device B and another wire is used to send data from device B to device A (Figure 4) Notice that pin 2 is an output on device A and an input on device B. The RS232 defines two types of devices; DTE (Data Terminal Equipment) and DCE (Data Communications Equipment). An RS232 port on a computer may be either a DTE or DCE but nearly every terminal with an RS232 port is a DTE. The **CIM** may be configured as either a DTE or a DCE by placing the 16 pin header on the circuit board in either the DTE or DCE socket.

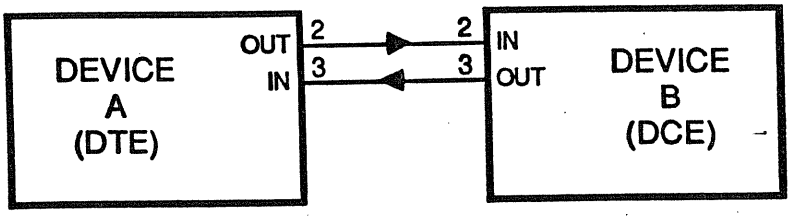
As an example, consider connecting an RS232 ASCII computer terminal to the **CIM** (specific details are contained in Example 1 of Appendix A.) The terminal will be DTE and so the **CIM** header must be in the DCE socket. To operate correctly, the **CIM** and the terminal must have the same settings for baud rate, parity, and number of stop bits. Even set correctly, it may not work as there are other lines in the RS232 Standard which are used to indicate that a device is ready to accept data. If the terminal responds to this line, it will believe that the **CIM** is not ready to accept data and will therefore not send any data.

Case 2 - RS232 With Control Lines

The data lines are the same as in Case 1. In addition, two control lines are used:

CTS (Pin 5)	'Clear To Send' is a signal asserted by the DCE to tell the DTE that the DCE is ready to receive data.
DTR (Pin 20)	'Data Terminal Ready' is a signal asserted by the DTE to tell the DCE that the DTE is ready to receive data.

Case 1
The Simplest Configuration



Case 2
RS232 With Control Lines

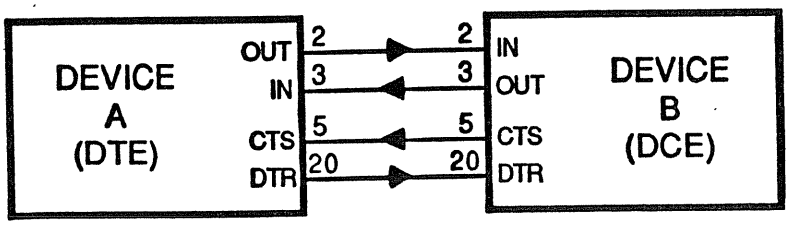


Figure 4 - RS232 Communication

The CIM responds to the control lines as follows:

1) If the lines are not connected, the CIM assumes that you are ready to receive data.

2) Data will not be transmitted from the CIM if the DTR line (pin 20) is low (CIM configured as a DCE) or the CTS line (pin 5) is low (CIM configured as a DTE). This is useful in the case when your program has asked for data but is not yet ready to receive it. If data transmission is not suspended, then data can be overwritten in the UART (it is not being retrieved by the program) and therefore lost. When this happens, the 'over-run' flag will be set in your computer's UART and it may be recognized by the operating system generating an error message such as "I/O Device Error". See the "W" command in the Command List (Figure 2)

Baud Rates

The RS232 baud rate of the CIM is switch selectable from 300 to 19.2K baud (see configuration switch setting Figure 23.) 19.2K baud means that data is transmitted at 19,200 bits/second. With one start bit, 2 stop bits, 8 data bits, and no parity bits, each ASCII character requires 573 µsec to be transmitted (11bit/19.2K baud.) A typical data string 5.127 <cr> has 6 characters, requiring 3.4 msec to be sent, resulting in a maximum data transfer rate of 290 samples/sec. If the SS or X command is used, then the transfer rate is increased to about 1000 samples/sec.

Stop Bits

Generally, selection of 2 stop bits will result in fewer data transmission errors.

Parity

The Parity bit provides a check against data loss. They are not commonly used in a local data transmission environment. If the parity option is selected, the CIM will transmit 8 data bits and a parity bit, however, no parity check of incoming data is done.

Voltage Levels

The RS232 uses bipolar voltage levels (Figure 5) The control lines use positive logic. For example, the DCE tells the DTE that it is clear to send (CTS) by placing >+3 VDC on pin 5 of the interface. Similarly, the DTE can tell the DCE that it is not

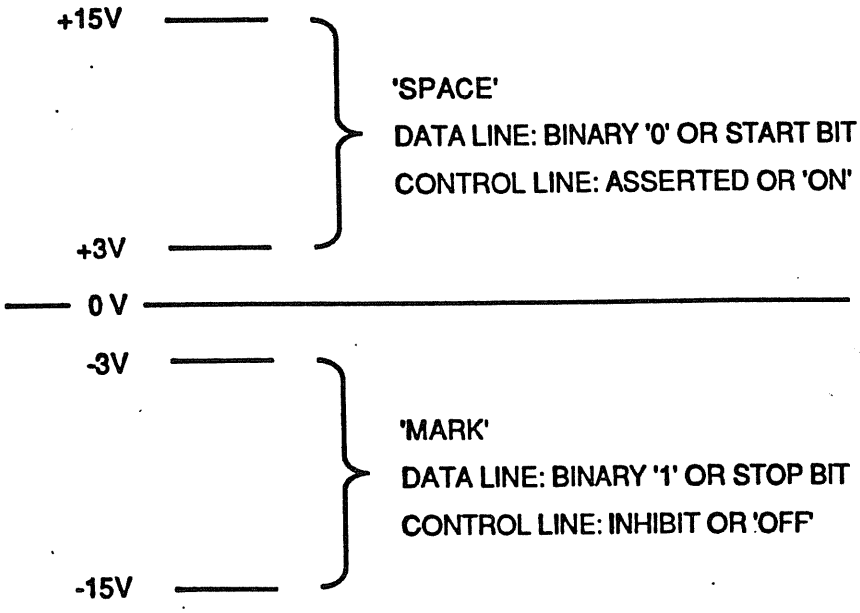


Figure 5 - RS232 Voltage Levels

ready by placing -3 VDC on pin 20 (DTR) of the interface. The data lines, pins 2 and 3, use negative logic. A 'zero' bit is represented by a positive voltage and a 'one' bit is represented by a negative voltage. A start bit is a positive voltage and a stop bit is a negative voltage. Data is transmitted with the least significant bit first. The letter 'A', which has the ASCII code 41H (0100 0001), would appear as in Figure 6. If a parity option was selected, the parity bit would be sent after the 8th data bit, but before the first stop bit.

'Eavesdropping'

When you are trying to get the RS232 to work with your computer, it is helpful to be able to 'eavesdrop' on the RS232 data lines going between the CIM and the computer. This can be done with an ASCII RS232 computer terminal and the connector shown in Figure 7 . To test the connector, place the hook clip on pin 2 of the same connector (shorting pin 2 to pin 3). Now, when you type at the terminal keyboard, data transmitted from pin 2 is received at pin 3 and displayed on the terminal screen. To use as a debugging tool, attach the hook clip to either pin 2

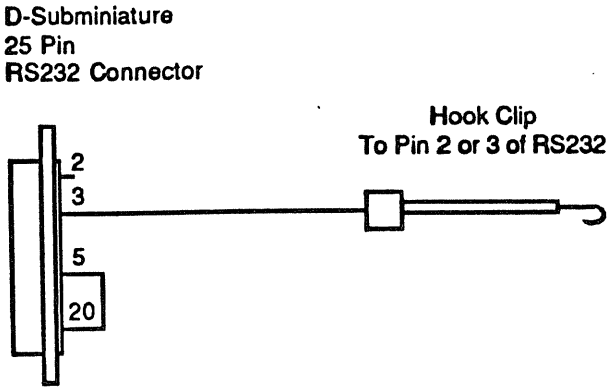


Figure 7 - Eavesdropper

or pin 3 of the RS232 data lines to show either data sent from the DTE or DCE (The hook clip may be placed on one of the top 2 lines in the DTE/DCE Header in the CIM). The baud rate, parity, and stop bits of the terminal must match those of the CIM and the computer. If your terminal has a mode which will display control characters (such as carriage returns and line feeds) it is helpful to operate in that mode.

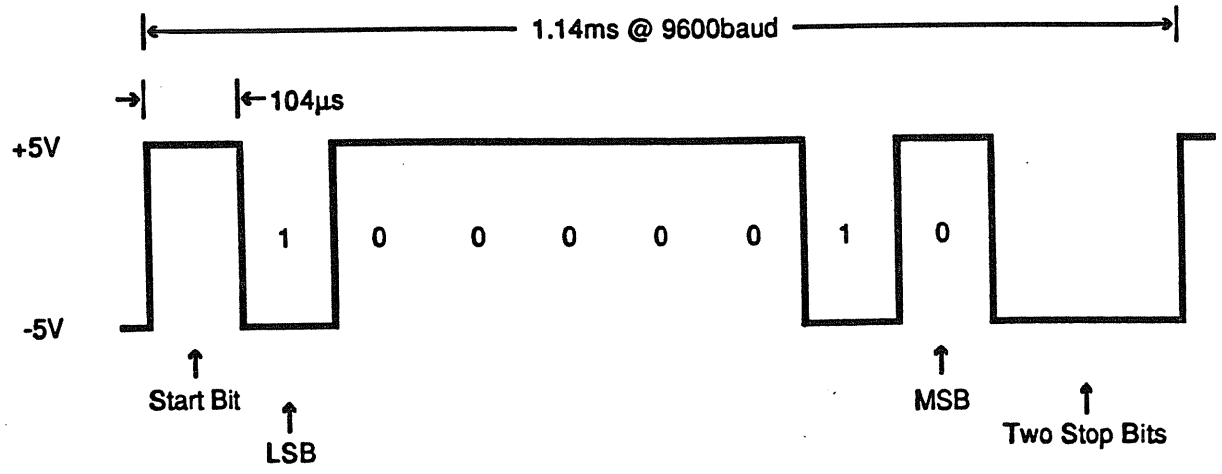


Figure 6 - The Letter 'A'

INTRODUCTION TO GPIB

The IEEE-488 Standard specifies the voltage levels, handshake requirements, timing and hardware details, including pinout and connector dimensions, for a 16 line byte serial bit parallel bus configuration. One major characteristic of this interface is that many instruments may communicate over the same cable and through the same port. Also, because the bits are passed in parallel, it offers speed advantages over the RS232 (about 20 µS per byte).

The controller (generally your computer) coordinates data transfer on the bus by designating all participating instruments (including itself) as either a talker or a listener. Listeners can receive data placed on the bus by the Talker. Devices can have the capacity to operate in either mode. The address of each device is set by switches in the device and must be between 0 and 30.

BUS DESCRIPTION

BYTE TRANSFER CONTROL GROUP. This consists of 3 negative logic lines that implement the GPIB handshaking. The NRFD (Not Ready For Data) line is held low by any designated listener who is not ready to accept data. When every listener is ready, the line goes high and the talker may release data to the bus. After data is on the bus, the talker pulls the DAV (Data Valid) line down. At this point, each listener retrieves the data. Before and during the retrieval of the data, the listener holds the NDAC (No Data Accepted) line down. When every listener has received the data, the NDAC line goes high, allowing the talker to release the DAV line high. Finally, the listener pulls down the NDAC line until another transfer is initiated.

DATA BUS: There are eight data lines which use negative logic and pass the bits of each byte in parallel.

GENERAL INTERFACE LINES: These five lines operate independently of the handshake lines and use negative logic.

1) The EOI (End or Identify) line is used by the talker to designate the end of message.
2) The SRQ (Service Request) line is used by any device to ask for service. The controller can serial poll each device (each device returns an 8 bit status byte) to determine who needs attention. It can also do a parallel poll using the EOI and ATN lines where each device is assigned a single data line.
3) The ATN (Attention) line makes both talkers and listeners accept information and passes control of the DAV line to the controller. This line is used by the controller to identify talkers and listeners through their addresses.
4) The REN (Remote Enable) line changes the status of an instrument from local to remote.
5) The IFC (Interface Clear) line clears the bus of all data and activity.

A complete description of the General Purpose Interface Bus is beyond the scope of this manual. The user should consult the manual for the particular GPIB controller he is using for specific information on how to send and receive characters over the bus, how to perform serial polls, etc. Instead, we will look at those features and commands of the CIM which relate specifically to the GPIB.

GPIB CAPABILITIES OF THE CIM

The GPIB capabilities of the CIM consistent with IEEE-488 standard (1978) are shown in Figure 8 . Note that the unit has no parallel poll capability. The responses of the CIM to some of the IEEE-488 standard commands are shown in Figure 9.

Code	Function
SH1	Source handshake capability
AH1	Acceptor handshake capability
T5	Basic Talker, Serial Poll, Unaddressed to talk if addressed to listen
L4	Basic Listener, Unaddressed to listen if addressed to talk
SR1	Service request capability
RL0	No remote-local capability
PP0	No parallel poll capability
DC1	Device Clear capability
DT1	Device Trigger capability

Figure 8 - CIM GPIB Capabilities

Mnemonic	Command	CIM response
DCL	Device Clear	Equivalent to Power On
SDC	Selected Device Clear	Equivalent to Power On
GET	Group Execute Trigger	None in Asynchronous Mode In Synchronous, same as a trigger at B1
SPE	Serial Poll Enable	Places status byte on bus and clears status byte

Figure 9 - CIM Response To Standard GPIB Commands

SETTING THE ADDRESS

The GPIB address of the CIM is set using bits 5 through 1 of the configuration switch. Bit 5 is the most significant bit, and setting a given switch 'ON' is equivalent to making it a binary 1. The unit is shipped configured for RS232 operation so the configuration switch must be set the first time the unit is used with the GPIB.

SERIAL POLLS AND SERVICE REQUESTS

The status byte sent by the unit when it is serial polled is the same status byte which is read using the ?S command. Of course, when the CIM is serial polled, it does not encode the status byte as a decimal number. The user can program the CIM to generate a service request (SRQ) to the controller every time a given status condition occurs. This is done using the SM=<n> command. The mask byte M which is set with this command is periodically logically anded with the status byte. If the result is nonzero, the CIM generates a service request and leaves the status byte unchanged until the controller performs a serial poll to determine the cause of the service request. When the unit has been serial polled, it loads a new status byte which reflects all of the status conditions which have occurred since the service request was generated. For instance, suppose we wanted to generate a service request each time a scan was finished or the CIM missed data. We would send the command SM=24 since 24 decimal is 00011000 binary which corresponds to the 'missed data' and 'scan finished' bits in the status byte. If, during the course of a scan, the unit was triggered too fast and missed data, a service request would be generated. If the unit serial polled the CIM before the scan finished, then the first serial poll would reveal only the 'missed data' condition, after which the CIM would load the new status byte reflecting the 'scan finished' condition and generate a new service request. It is particularly useful to use the SM command to unmask the 'Scan Finished' bit. This eliminates the need to use the ?N command to detect the completion of a scan, increases the maximum data rate and leaves your computer free for other tasks during the scan.

GPIB RS232 ECHO MODE

It is sometimes useful when debugging a GPIB system to have some way of monitoring exactly what is going back and forth over the bus. The CIM has the capability to echo all characters sent and received over the GPIB to its RS232 port. This mode of operation is set by turning bits 8 and 7 of the configuration switch ON. This will automatically configure the RS232 port for 9600 baud, no parity operation. The number of stop bits is still user selectable with configuration switch bit 6. Of course, since the RS232 port operates at much lower speeds than the GPIB is capable of, the GPIB cannot be operated at high data rates in this mode. It is useful, however, for determining if what is actually being sent to the CIM corresponds to what is supposed to be sent to the CIM.

TROUBLESHOOTING

1) CIM Problem. If, when the power is initially applied, both the RED and GREEN LED's come on and stay on, then the CIM memory is defective and the unit should be returned for service.

2) RS232 Eavesdropping. This is described in the communications section at the end of the RS232 description..

3) Emulating the CIM with a Terminal. When trying to debug RS232 interfaces, it can be useful to substitute an RS232 terminal for the CIM . This will allow you to both see what is being sent to the CIM by the program, and supply responses to the computer via the terminal's keyboard (responses normally sent by the CIM).
For a computer with an RS232 port configured as a DCE, simply substitute the terminal (a DTE device) for the CIM . For a computer with an RS232 port configured as a DTE (i.e. the ASYNC port on the IBM PC) a "null modem" cable is necessary to connect the computer terminal to the DTE port. This cable is made by swapping lines on the RS232 to allow two DTE devices to be connected together (Figure 10).

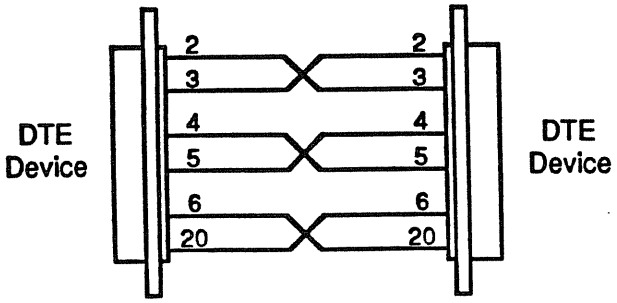


Figure 10 - Null Modem Cable

4) GPIB Eavesdropper. When debugging an IEEE-488 (GPIB) interface, an ASCII terminal (9600 baud only) may be used to view the GPIB transaction on the RS232 connector of the CIM when the Echo switch (switch 6) is on. A GPIB extender is helpful in allowing the simultaneous connection of GPIB and RS232. When using this technique, the W0 command should be sent to the CIM so that the GPIB will not be unnecessarily slowed by the terminal. The CIM must be configured as a DCE so that it may talk to the terminal, a DTE device.

5) Program 'hanging' due to hardware problems. When your computer and the CIM are talking, it is very easy for the system to "hang." This may occur because the computer is waiting for a response from the CIM which does not come because:
a) The RS232 or GPIB cable was not attached or has come loose.
b) The configuration switches are not set properly (Baud rate, parity, stop bits, GPIB address conflict). Also, the switches can be set correctly, but the power must be turned off and then on again before the CIM will read them.
c) The 16 pin Header to select DCE or DTE is not in the correct socket.
d) Your computer requires a control line of the RS232 to be asserted, but you are only using two wires for your cable.

6) Program 'hanging' due to software problems:
a) The CIM is in the "Mode Synchronous" and is waiting for a trigger input before sending data.
b) The command from the program asking for data was invalid, or a transmission error occurred (ex. we have observed Microsoft's interpreted BASIC on the IBM PC occasionally send a curly bracket (ASCII 0FDH) instead of a carriage return (ASCII 0DH). When this happens, the CIM sends no response as it is waiting for the carriage return to end the command.
c) The initial command was invalid because of a "garbage character" which was sent by the computer when it was turned on. It is good practice to send several carriage returns and a ?S when the program starts to clear out bad characters and any resulting errors.
d) The CIM is not sending the correct 'end-of-record' marker for your computer. For example, it appears that Microsoft's Rev 3.2 FORTRAN on the IBM PC under DOS 2.0 requires two carriage returns for an end-of-record marker. The Z command can be used to set the CIM end-of-record marker to 2 carriage returns. [The end-of-record marker is that sequence which indicates that the response is complete. From the keyboard, a single carriage return is the end-of-record marker.]
e) Answers are coming back from the CIM too fast, overwriting the end-of-record markers, and causing the computer to hang waiting for a complete response. In this case, the W command can be used to slow down the response time of the CIM preventing overwriting.

f) The command, though formatted correctly, was invalid due to incorrect initializing of the CIM relative to the command: for example, S8=2 is an invalid command if port 8 has not been designated an output port. Note that a previous program or user can leave the CIM initialized in a variety of configurations incompatible with your program. Including a MR in your program is good protection against this situation.
g) The command, though formatted correctly, was invalid due to parameter limitations such as: SC1:4000 overruns the memory; "S8=",A where the program has set A = 11 and is out-of-range; and A6,1 when port 8 was left at -1 volt.
h) A command was sent too early, as in asking for N prior to completing the scan.
i) The command was invalid due to syntax errors. These errors can occur simply, as in using "," instead of ";" to separate commands, or more obscurely, such as "S",A,"=5" where A is a variable which produces an error because it is sent with leading blanks. Leading blanks are allowed for variable settings, but not in command codes.

7) Incorrect Data. Finally, your program may successfully run but produce invalid data. This can occur through A/D overflows or missed data due to the trigger rate being to fast. In these cases, the red 'error' LED will flash but program execution will not be interrupted.

CALIBRATION

There are only two pots which are used to calibrate the CIM . These pots control the gain of the unit's DAC for negative and positive outputs. To calibrate the unit you will need a 4 1/2 digit voltmeter with a dc accuracy equal to or better than 0.02%, and an ASCII RS232 computer terminal.

Before starting, record the switch settings and the DCE/DTE jumper location so that the unit may be easily returned to service following calibration. To start, establish communications between the CIM and the computer terminal by selecting the same baud rates, parity bits and stop bits. For a 9600 baud terminal with no parity bit, 8 bit data word and 2 stop bits, set the CIM switches as in the RS232 example of Figure 2. Also, make sure that the jumper header is in the DCE position (bottom socket, near the back panel RS232 connector). Connect the terminal to the CIM 's RS232 connector with a cable that has (at least) pins 2, 3, 5, 6 and 7. When the unit is plugged in and the power turned on, the sign on message should appear on the terminal screen.

The object of the calibration is to set the gain of the DAC output for negative and then positive outputs. The calibration must be done in that order--negative first, then positive.

Connect the 4-1/2 digit voltmeter to analog output #1. Select the 20 VDC full scale range. Enter the commands as shown below (use the return key at the end of each line). In this procedure, the minus signs are important. [Comments are in square brackets]

```
OK->  I0          [No inputs, all outputs]
OK->  S1 = -0.1   [Set output #1 to -0.1 VDC]
```

Now, place the voltmeter in the relative position to null any offset which may be present (if your DVM does not have a relative button, you must remember the offset voltage: the object is to make the difference in the two readings equal to 10.000 VDC). Then send the command to set the output to -10.100V.

```
OK->  S1 = -10.1
```

Now, adjust P1, the top pot near the channel 3 input, so that the difference between the outputs reads -10.000 +/- .002 VDC.
Then send the command to set the output to +0.100V.

```
OK->  S1 = 0.1
```

Again, toggle the DVM's 'relative' button in order to null any offset. Then send the command to set the output to +10.100V.

```
OK->  S1 = 10.1
```

Now, adjust P2, the lower pot near channel #6 input BNC, so that the difference between the two output values is 10.000 +/- .002 VDC.

When the calibration is completed, return the switch settings and the DCE/DTE jumper to their original positions.

CIRCUIT DESCRIPTION

The CIM is a microprocessor based instrument which may be used to input and output analog and digital voltages. The analog I/O is via rear panel BNC connectors (8 channels); digital I/O is via the rear panel digital I/O connector (2 definable bits
8 bits of input and 8 bits of output with strobes). The CIM may be interfaced to computers by either the IEEE-488 (also known as the GPIB or HPIB) or by the RS232. The communication interface is defined by setting the configuration switch on the circuit board.

Analog voltages may be read or set over ±10.2375 VDC with a resolution of 2.5 mV, and an accuracy of +0.05%. Digital bits may be set, reset, tested and counted. One front panel bit is also used to synchronize data acquisition when the unit is programmed to acquire scans.

MICROPROCESSOR AND DIGITAL I/O
CIM Schematic Page 1

The instrument uses a Z80-A CPU, U1. An 8K byte ROM, U2, is used for program storage and an 8K byte RAM, U3, is used for buffer memory. A power-on-reset circuit, C3 and U36, will restart the program whenever the unit is plugged into the NIM bin, or when the bin is turned on. The CPU is clocked at 4 MHz by a crystal control oscillator, U17. This clock is also divided down to generate other frequencies that are used by the communications circuits.

All of the I/O ports and VLSI devices are I/O mapped. U11, U18 and U19 serve as an I/O port decoder to select one device during I/O operations by the processor.

The 8 configuration switches are read through the octal buffer U12. (There are other hardware consequences of reading this port: the 12 bit DAC is cleared and the 8 bit offset DAC is loaded. More about this in the analog description.)

The 8253-5 programmable timer, U5, may be read, set and programmed by the Z80-A. This IC has three programmable counters: two are used to count digital pulses at the front panel BNC inputs, and the third is used to generate the x16 baud rate clock for the RS232 UART. This baud rate is derived from the 4MHz clock input.

The 2 definable digital bits may be read through part of U13, an octal tristate buffer. U13 also reads the "compare" bit (used in A/D conversion) and the DCE/DTE bit (from the RS232 interface).

The 8 bit digital I/O port, whose connector may be accessed from the rear of the CIM, is read through U14, an octal buffer. The separate 8 bit output port is set by an output to U16, an octal latch and driver. An input from U14 will generate a 600 nS, active low strobe signal which indicates that the input port is being read. This signal, "RD I/O", appears on pin 2 of the 20 pin connector. An output to U16 will generate a 600 nS, active low strobe signal, "WR I/O", which appears on pin 19 of the 20 pin connector. This signal indicates that the new data has been written to the 8 bit output port.

An 8 bit output latch, U15, is used to set the 2 definable digital bits high, low or off. When off, the bit may be used as an input. To see how this works, notice that when both bits Q0 and Q1 of U15 are low, transistors Q3 and Q4 are both off, so digital input #1 is a high impedance input which may be read by U13 or counted by U5. If bit Q0 is set high, transistor Q4 will be forced into saturation, grounding the digital output #1 through the current limiting 50 ohm resistor, R7. If bit Q0 is set low, and bit Q1 set high, then digital output #1 is pulled high by the emitter of Q3. The output current is limited by R6 to provide short circuit protection.

Other output bits of U15 are used to control the green ACTIVITY LED, D2, the red ERROR LED, D3, and to control certain aspects of A/D and D/A conversion (the 'POLARITY' and 'SAMPLE' bits).

ANALOG I/O
CIM Schematic Page 2

A single 12 bit digital-to-analog converter, U20, is used in all of the analog input and output operations. A precision 10.240 VDC reference, U38, is used as a voltage reference for the 12 bit DAC. The DAC is loaded by the Z80, 4 bits at a time. A final write operation is used to transfer the 12 bit input to the DAC's internal control register.

An op amp (the first 1/4 of U26) is used to convert the DAC's output current to a voltage. In addition to the current from the 12 bit DAC, current from an eight bit DAC, U32, is summed at U26. This eight bit DAC is used to correct for offset errors which can accumulate as analog voltages pass through buffers, inverters, S/H amps and comparators. These offsets are carefully measured after the unit is manufactured, and values to compensate for these offsets are placed in the unit's ROM. At the start of any A/D or D/A conversion, the 12 bit DAC is cleared and the appropriate offset byte is loaded into U32 to eliminate the unwanted offset.

The DAC voltage may be inverted or not inverted by the 2/4 of U26 under the control of the POLARITY bit. If the polarity bit is low, the 2/4 of U26 is a precision inverter; if the polarity bit is high, it is a precision unity follower.

The DAC voltage may be multiplexed to one of eight sample and hold output amplifiers [U27, U28] which can provide analog outputs to the front panel. The processor refreshes these S/H amplifiers every few mS.

For a channel to become an output, the appropriate switch in U24 or U25 must be closed. When the switch is closed, current from the particular S/H amp will pull the front panel output to the programmed level. The analog switch's channel resistance (a few hundred Ω) is effectively eliminated by the high open loop gain of the S/H amp. Analog outputs have a measured resistance of less than 1Ω. The processor can select zero to eight of the channels as outputs.

A/D CONVERSION

The processor uses the offset-corrected 12 bit DAC to do analog-to-digital conversions on the inputs by programmed successive approximations. The processor selects 1 of 8 channels by the analog multiplexer, U23. The selected voltage is buffered by the 3/4 of U26 and sampled by the S/H amplifier (U21 and 4/4 U26.) This analog sample is passed to the comparator, U31, to be compared with outputs from the DAC. The processor will cause the DAC output to converge to the analog sample with 13 tests of the COMPARE bit (sign plus 12 bits) by the method of successive approximations. The A/D conversion process takes about 300μS.

The ADC may be done on a channel whether it has been set as an input or an output. As inputs, the channel impedance is 1Mμ (RN4). The input buffer amplifiers are protected from excessive inputs by the 10Kμ resistors of RN3.

COMMUNICATIONS INTERFACES

CIM Schematic Page 3

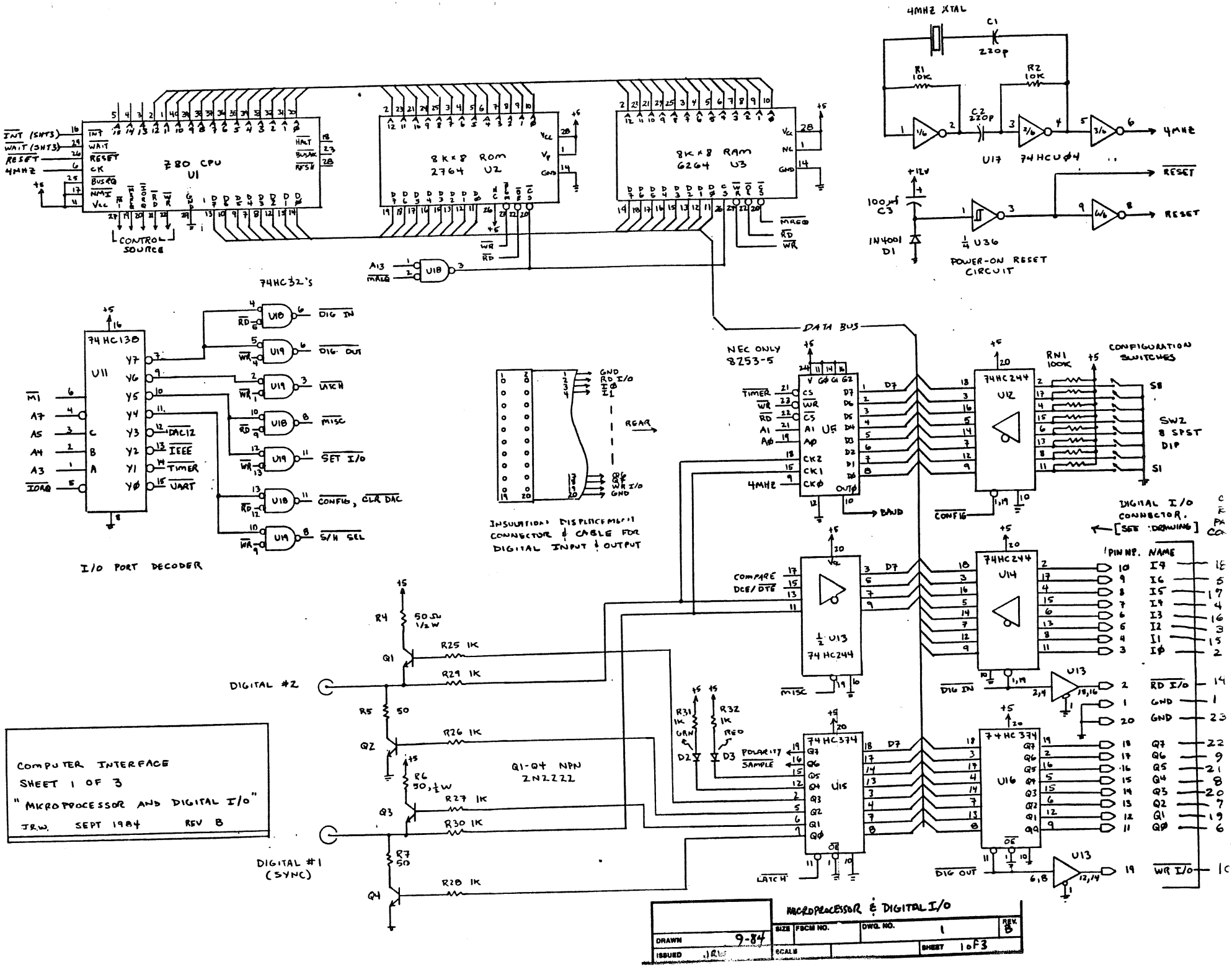
RS232 INTERFACE

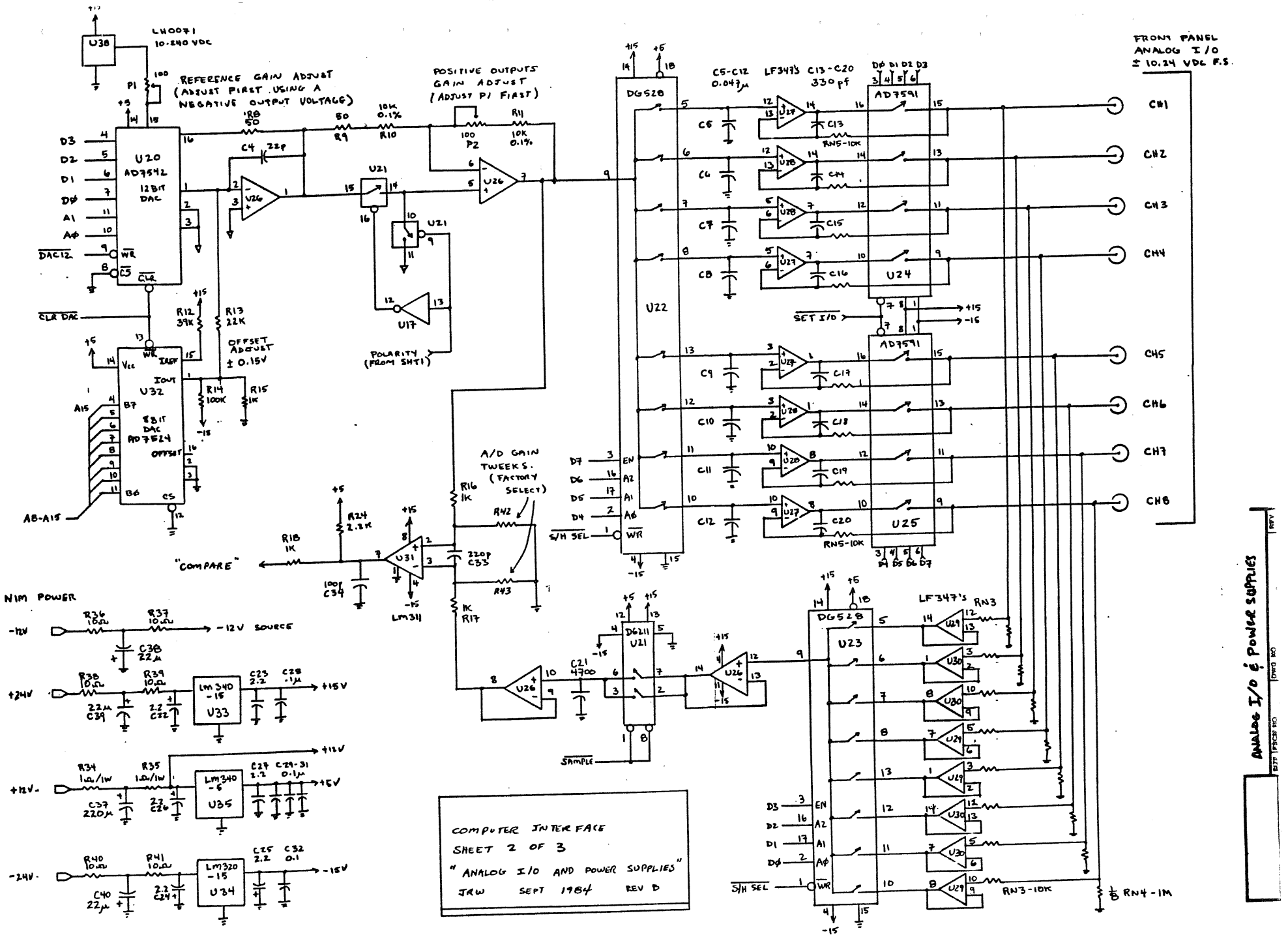
The RS232 interface uses an 8251 UART (U4) to send and receive bytes in a bit serial fashion. Any standard rate between 300 and 19,200 baud may be selected with the configuration switches. The x16 baud rate clock to the UART comes from the 8253 programmable counter/timer. A 2 MHz clock to the UART is derived from the 4 MHz CPU clock by U9. The number of stop bits and parity format can also be specified by the configuration switches. The RS232 interface can be configured as either DTE (Data Terminal Equipment) or DCE (Data Communications Equipment) by moving a 16 pin jumper assembly between two sockets on the circuit board. When a data byte is received by the UART, the RxRDY output is set high, which interrupts the Z80A in order to remove the character from the UART's receiver data register.

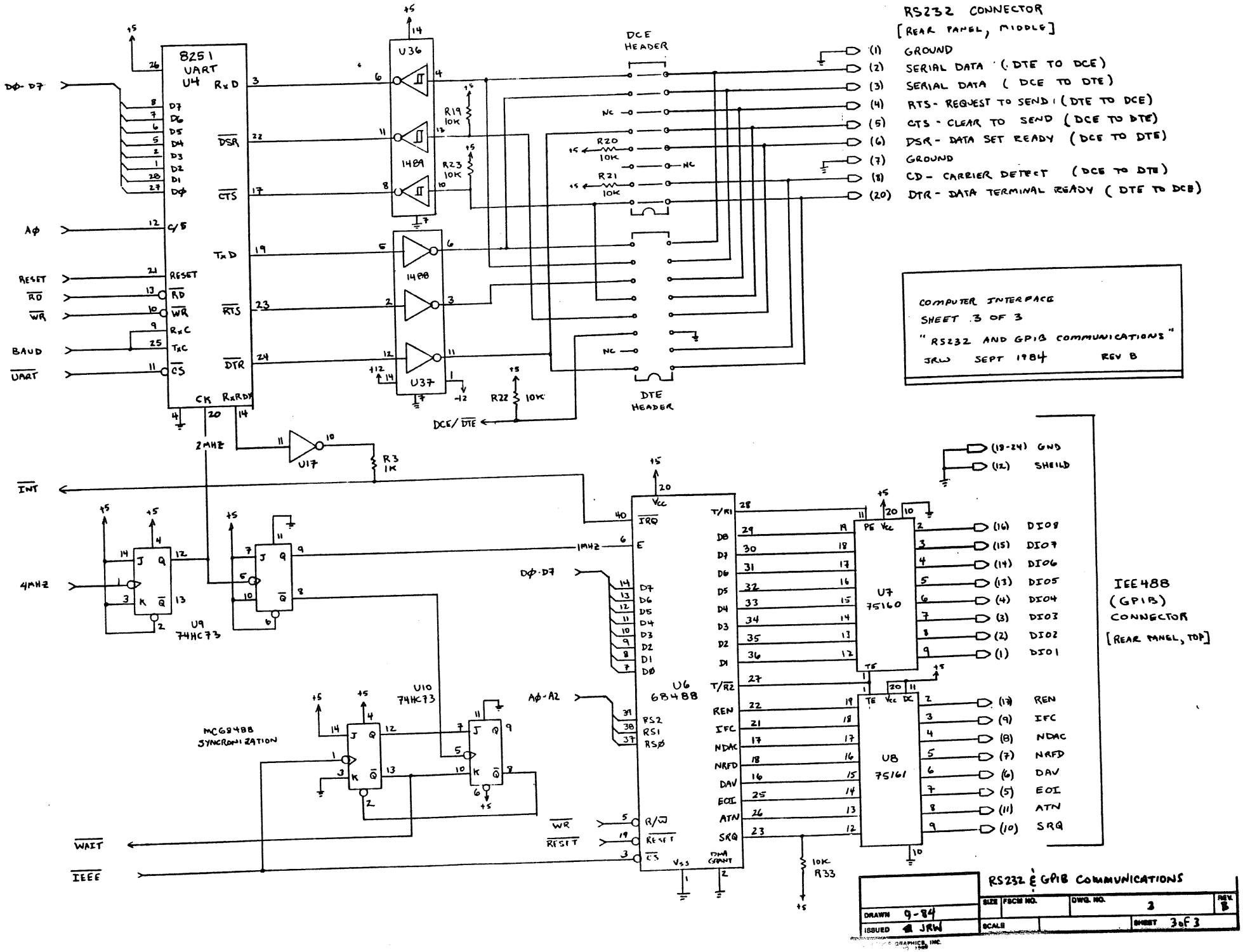
GPIB INTERFACE

The interface between the **CIM** and the GPIB is provided by the GPIB controller chip, U6, an MC68488. GPIB data and control lines are buffered by the bus drivers U7 and U8. The controller chip uses a 1MHz enable clock which is derived from the 4MHz CPU clock by U9. I/O transaction between the CPU and the controller chip must be synchronized to this 1MHz clock: U10 causes this synchronization to occur by making the CPU wait for up to 1μS whenever the controller chip is selected by the CPU.

The controller chip will interrupt the CPU whenever a transaction occurs on the GPIB which requires the CPU's intervention (such as the GPIB requesting data from the **CIM**). Most GPIB transactions, including transactions with other instruments, do not require the CPU's intervention. The **CIM** 's address on the GPIB is set by the configuration switch when power is applied to the **CIM** .


4MHZ XTAL
C1
220P
R1 10K
R2 10K
C2 220P
U17 74HCU04
4MHZ
RESET
+12V
100µF C3
1N4001 D1
1/4 U36
POWER-ON RESET CIRCUIT
Z80 CPU U1
8K x 8 ROM 2764 U2
8K x 8 RAM 6264 U3
CONTROL SOURCE
74HC32's
U18
74HC138 U11
I/O PORT DECODER
DIG IN
DIG OUT
LATCH
MISC
SET I/O
CONFIG, CLR DAC
S/H SEL
DAC12
IEEE
TIMER
UART
INSULATION DISPLACEMENT CONNECTOR & CABLE FOR DIGITAL INPUT & OUTPUT
REAR
DATA BUS
NEC ONLY 8253-5
U15
74HC244 U12
RN1 100K
CONFIGURATION SWITCHES
SW2 8 SPST DIP
BAUD
CONFIG
COMPARE DCE/DTE
1/2 U13 74HC244
74HC244 U14
DIGITAL I/O CONNECTOR. [SEE DRAWING]
PIN NO. NAME
I9 I6 I5 I4 I3 I2 I1 I0
RD I/O
GND
WR I/O
R4 50 1/2W
Q1
R25 1K
R29 1K
DIGITAL #2
R5 50
Q2
R26 1K
R6 50, 1/4W
Q3
R27 1K
R30 1K
DIGITAL #1 (SYNC)
R7 50
Q4
R28 1K
Q1-Q4 NPN 2N2222
R31 1K GRN
R32 1K RED
D2
D3
POLARITY SAMPLE
74HC374 U15
74HC374 U16
LATCH
DIG OUT
Computer Interface
Sheet 1 of 3
"Microprocessor and Digital I/O"
JRW SEPT 1984 REV B
MICROPROCESSOR & DIGITAL I/O
DRAWN 9-84
ISSUED
SIZE FSCM NO. DWG. NO. 1 REV. B
SCALE
SHEET 1 OF 3


U38
LH0071
10-240 VDC
REFERENCE GAIN ADJUST
(ADJUST FIRST USING A
NEGATIVE OUTPUT VOLTAGE)
POSITIVE OUTPUTS
GAIN ADJUST
(ADJUST P1 FIRST)
U20
AD7542
12 BIT
DAC
DAC12
CLR DAC
U32
8 BIT
DAC
AD7524
OFFSET
ADJUST
± 0.15V
U21
U17
POLARITY
(FROM SHT1)
U22
DG528
C6-C12
0.047µ
LF347's
C13-C20
330 pF
AD7591
SET I/O
U24
U25
CH1
CH2
CH3
CH4
CH5
CH6
CH7
CH8
FRONT PANEL
ANALOG I/O
± 10.24 VDC F.S.
A/D GAIN
TWEEKS.
(FACTORY
SELECT)
"COMPARE"
LM311
S/H SEL
SAMPLE
NIM POWER
-12V SOURCE
LM340 -15
U33
LM340 -6
U35
LM320 -15
U34
U23
RN3-10K
RN4-1M
RN5-10K
COMPUTER INTERFACE
SHEET 2 OF 3
"ANALOG I/O AND POWER SUPPLIES"
JRW SEPT 1984 REV D
ANALOG I/O & POWER SUPPLIES

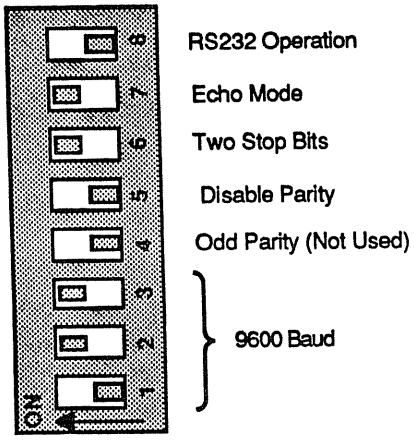

RS232 CONNECTOR
[REAR PANEL, MIDDLE]
(1) GROUND
(2) SERIAL DATA (DTE TO DCE)
(3) SERIAL DATA (DCE TO DTE)
(4) RTS - REQUEST TO SEND (DTE TO DCE)
(5) CTS - CLEAR TO SEND (DCE TO DTE)
(6) DSR - DATA SET READY (DCE TO DTE)
(7) GROUND
(8) CD - CARRIER DETECT (DCE TO DTE)
(20) DTR - DATA TERMINAL READY (DTE TO DCE)
DCE HEADER
DTE HEADER
8251 UART U4
U36
1489
1488
U37
R19 10K
R20 10K
R21 10K
R22 10K
R23 10K
NC
DCE/DTE
COMPUTER INTERFACE
SHEET 3 OF 3
" RS232 AND GPIB COMMUNICATIONS "
JRW SEPT 1984 REV B
2MHZ
U17
R3 1K
INT
4MHZ
U9 74HC73
U10 74HC73
MC68488 SYNCRONIZATION
U6 68488
U7 75160
U8 75161
IEEE 488 (GPIB) CONNECTOR
[REAR PANEL, TOP]
(13-24) GND
(12) SHEILD
(16) DIO8
(15) DIO7
(14) DIO6
(13) DIO5
(4) DIO4
(3) DIO3
(2) DIO2
(1) DIO1
(17) REN
(9) IFC
(8) NDAC
(7) NRFD
(6) DAV
(5) EOI
(11) ATN
(10) SRQ
10K R33
WAIT
IEEE
RS232 & GPIB COMMUNICATIONS
DRAWN 9-84
ISSUED JRW
SIZE
FSCM NO.
DWG. NO. 3
REV B
SCALE
SHEET 3 of 3

EXAMPLE 1 - RS232 COMPUTER TERMINAL

This example will allow you to communicate with the **CIM** from a standard RS232 computer terminal.

1. Configure an ASCII terminal for 9600 baud, two stop bits, no parity bits, and 8 data bits.

2. Set the **CIM** configuration switch as follows:



3. Place the 16 pin header in the socket marked 'DCE' (the lower socket of two, located near the RS232 connector on the inside of the **CIM** .)

4. Turn the CIM bin off, then on, to reset the **CIM** . The copyright sign-on message should appear on the screen along with the prompt 'OK->'.

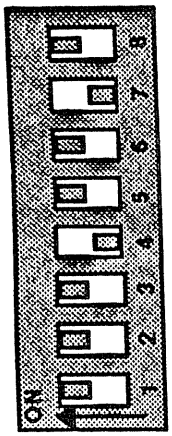
5. Set voltages at the outputs and check them with a voltmeter as follows:

```
OK-> I0
OK-> S1=8
OK-> S2=7
OK-> S3=6
OK-> S4=5
OK-> S5=4
OK-> S6=3
OK-> S7=2
OK-> S8=1
OK-> ?1;?2;?3;?4;?5;?6;?7;?8
8.000
7.000
6.000
5.000
4.000
3.000
2.000
1.000
OK-> W0;?1;?2;?3;?4;?5;?6;?7;?8
```

EXAMPLE 2 - IBM PC VIA GPIB (INTERPRETED BASIC)

This program requires the Capital Equipment Corporation GPIB card which is a short card for the IBM PC or XT and has firmware in ROM to interface high level languages to the GPIB. In this program, the CEC card's ROM starts at OC0000H, the system controller's address is 21 and the CIM has been assigned as GPIB address 23.

The configuration switch on the CIM should be set as follows:



GPIB Operation

No Echo

Two Stop Bits (Not Used)

GPIB Address = 23

Calls in Microsoft's BASIC for the PC are done to memory locations specified by the name of the subroutine. The address is relative to the segment address specified by the DEF SEG statement preceding CALL.

To monitor the GPIB activity with an RS232 terminal, switch 7 should be set to ON, the CIM configured as a DCE, and the ASCII terminal attached to the appropriate pin of the RS232. The terminal should be set for 9600 Baud, 8 data bits, 2 stop bits, and no parity.

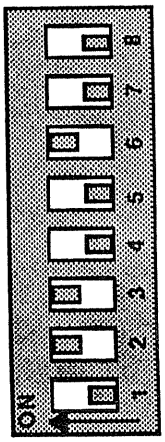
Program:

```
10   'GPIB TEST ROUTINE
20   'SENDS ?1 CONTINUALLY AND PRINTS RESPONSE
30   '
40   CLS
50   DEF SEG =&HC000                'ADDRESS OF CONTROLLER CARD
60   S$=SPACE$(80)                  'USED TO ERASE INPUT LINE
70   INIT=0: TRANSMIT=3: RECV=6: ADDR%=21: SYS.CONT%=0
80   CALL INIT(ADDR%, SYS.CONT%)
90   A$= "IFC MTA LISTEN 23 DATA '?1' 13"
100  B$= "UNL MLA TALK 23"
110  ANS$=SPACE$(10)
120  CALL TRANSMIT(A$, STATUS%)
130  IF STATUS%<>0 GOTO 200
140  CALL TRANSMIT(B$, STATUS%)
150  IF STATUS%<>0 GOTO 200
160  CALL RECV(ANS$, LENGTH%, STAUS%)
170  IF STATUS%<>0 GOTO 200
180  PRINT USING "##.###"; VAL(ANS$)
190  GOTO 120
200  PRINT "STATUS CODE ="; STATUS%
210  STOP
220  END
```

EXAMPLE 3 - IBM PC VIA RS232 (BASIC)

In this example, the IBM PC's "ASYNC" port (known as COM1: or AUX: to DOS users) will be used to communicate with the **CIM** . Only two wires between the IBM PC ASYNC port and the **CIM** are required (pins #2 and #3 of RS232), but pins # 5,6,8,and 20 should be connected together on the connector at the IBM end.

Set the switches of the **CIM** as follows:



RS232 Operation

Disable Echo

Two Stop Bits

Disable Parity

Odd Parity (Not Used)

9600 Baud

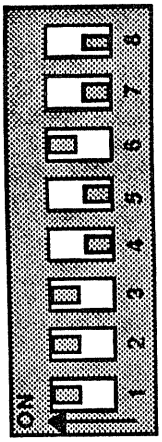
After resetting the **CIM** (toggle the power switch), run the program. The eight values displayed on the screen are values read at the eight inputs of the **CIM** . Apply voltages at the 8 inputs to test for yourself. The values will be updated at about 1 Hz (Interpreted BASIC is slow).

Program:

```
10    REM     Connect the Async port of the IBM to the CIM . You only
20    REM     need to connect pins 2 and 3 in this sample program.
30    REM
40    CLS
50    OPEN "COM1:9600,N,8,2,CS,DS,CD" AS #1
60    PRINT #1, "          "
70    PRINT #1, "MR; I8; W25"
80    LOCATE 1,1
90    FOR I = 1 TO 8
100   PRINT #1, USING "?#"; I
110   PRINT #1,
120   INPUT #1, V
130   PRINT USING "##.###";V
140   NEXT I
150   PRINT "loop count=", J
160   J = J + 1
170   GOTO 80
```

EXAMPLE 4 - IBM PC VIA RS232 (FORTRAN)

This program makes use of assembly language drivers to communicate with the **CIM** via RS232. The **CIM** internal 16 pin header should be in the socket marked DCE, and the configuration switch should be set as follows.



RS232 Operation

Disable Echo

Two Stop Bits

Disable Parity

Odd Parity (Not Used)

19.2 KBaud

The program sets the first four ports of the **CIM** as inputs. It then reads those four ports continuously, displaying its readings to a scrolling screen. The program may be paused and resumed by typing control S, and program execution is halted by typing control C. ftst.for is written in Microsoft FORTRAN v3.3. It makes calls to procedures in rs232.asm which is written in Microsoft MACRO Assembler v1.25.

To compile ftst.for:	for1 ftst; pas2
To assemble rs232.asm:	masm rs232;
To link:	link ftst rs232/NOD;
To execute:	ftst

```
c       ftst.for
c
c       A FORTRAN program that uses assembly language drivers
c       to communicate with the  CIM  via RS232.
c
c       S. Lindgren     12/11/86
c
c
        program ftst

        character       io*40
        real            v(4)

c       Initialize computer's RS232 port COM1:
        call bgn232

c       Initialize the CIM
        call snd232 ('W0;I4$')

c       The '$' is required by snd232 to mark the end of the string.
c       It is not sent to the  CIM .

100     do 200 i=1,4

c       Request an input voltage
        write (io, 1000) i
1000    format (BN, '?', I1, '$')
        call snd232 (io)

c       Receive an input voltage
```

```
        call rcv232 (io)
        read (io, 1100) v(i)
1100    format (BN, F8.3)

200     continue

c       Write the four voltages to the screen.
        write (*, 1200) v(1), v(2), v(3), v(4)
1200    format (2X, F8.3, 2X, F8.3, 2X, F8.3, 2X, F8.3)

        goto 100

        end
```

```
;rs232.asm

;rs232 interface procedures: bgn232, snd232, rcv232

                page    60,132

data            segment         public  'data'

data            ends

dgroup          group   data
code            segment         'code'
                assumecs:code,ds:dgroup,ss:dgroup

public  bgn232
bgn232          proc    far

status          equ     3fdh            ;UART status port address
modcon          equ     3fch            ;modem status reg
rxdata          equ     3f8h            ;UART data port address
intreg          equ     3f9h            ;UART interrupt register
lnctl           equ     3fbh            ;UART line control register
divreg          equ     3f9h            ;UART divisor register

keybrd          equ     16h             ;Int for rom bios call for keyboard
rs232           equ     14h             ;Int for rom bios service of rs232
cr              equ     0dh             ;carriage return
lf              equ     0ah             ;line feed


                push    bp              ;save the frame pointer
                mov     bp,sp           ;get the pointer to the function code
```

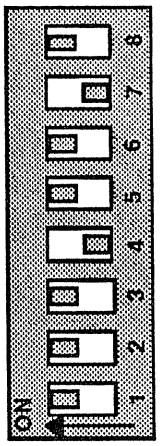
```
;               Initiallize the RS232 port.

init:           mov     ah,0            ;initialize request
                mov     dx,0            ;point to the first card
                mov     al,11100111b    ;9600 baud, no parity, 2 stop bits
                int     rs232           ;off to rom bios

;               now set 19.2 Kbaud operation

                mov     dx,lnctl        ;point to line control register
                in      al,dx           ;get it
                mov     ah,al           ;save it
                or      al,10000000b    ;turn on divisor access bit
                out     dx,al

                mov     dx,divreg       ;point to msb of divisor
                sub     al,al           ;this is 0
                out     dx,al
                mov     al,6h           ;19.2 kbaud
                dec     dx              ;point to lsb of divisor
                out     dx,al

                mov     dx,lnctl        ;point to line control reg
                mov     al,ah           ;restore divisor latch to 0
                out     dx,al

;               end of 19.2 set

                mov     dx,intreg       ;turn off UART interrupts
                sub     al,al
                out     dx,al
                mov     al,00000111b    ;set clear to send
                mov     dx,modcon       ;modem control reg
                out     dx,al
                mov     dx,status       ;point to the status port
                in      al,dx           ;clear errors
                mov     dx,rxdata       ;point to data port
                in      al,dx           ;clear pending data

                pop     bp
                ret

bgn232          endp


public          snd232
snd232          proc    far

;               Send a string via the COM1: port. Stop sending when a '$' is
;               found in the string. Do not send the '$'.

send:           push    bp
                mov     bp,sp
```

```
                push    ds
                lds     si,dword ptr [bp+6]     ;point to the first character to go
                sub     cx,cx                   ;will flag end of string when <>0
                cld

gtnxt:          lodsb                   ;get the character
                mov     ah,al           ;save it
                cmp     al,'$'          ;test for end of string
                jnz     arnd            ;jump if not at end of string

last:           mov     al,cr           ;substitute a CR
                mov     ah,al
                mov     cx,1            ;flag done

arnd:           cmp     al,' '          ;check for a superflous blank
                jz      short gtnxt

xhold1:         mov     dx,status       ;get transmitter status
                in      al,dx           ;get status
                test    al,20h
                jz      xhold1          ;if not ready wait

                mov     dx,rxdata       ;output the character
                mov     al,ah
                out     dx,al

                jcxz    short gtnxt

                pop     ds
                pop     bp
                ret     04

snd232          endp


public          rcv232
rcv232          proc    far


                push    bp
                mov     bp,sp

                les     di,dword ptr [bp+6] ;point to the start of the string

;               get the character

                mov     cx,15           ;counter for blanks
                mov     bx,0ffffh       ;counter for timout loop
                cld

statlp:         mov     dx,status       ;point to the status port
                dec     bx
                jz      timout
```

```
                in      al,dx           ;get the status byte
                test    al,1            ;check for rececived data
                jz      statlp          ;wait for data

                mov     dx,rxdata       ;point to the data port
                in      al,dx           ;get the data byte

                stosb                   ;place the character in the string
                dec     cx              ;decrement number of blanks to place

                cmp     al,0dh          ;was it a <cr>?
                jnz     statlp          ;go back for more characters

                dec     di
                mov     al,' '          ;clear remainder of buffer
rep             stosb                   ;fill with spaces

timout:         mov     sp,bp
                pop     bp
                ret     04

rcv232          endp


code            ends
                end
```

EXAMPLE 5 - IBM PC VIA GPIB USING C

This program is the same as the one in Example 4 except that it is written in C and communicates via GPIB. The CIM 's configuration switch should be set as follows.



GPIB Operation

No Echo

Two Stop Bits (Not Used)

GPIB Address = 23

The program sets the first four ports of the CIM as inputs. It then reads those four ports continuously, displaying its readings to a scrolling screen. The program may be paused and resumed by typing control S, and program execution is halted by typing control C. ctst.c is written in Microsoft C v3.0. It makes calls to procedures in GPIB-L which is included with the Capital Equipment Corp. GPIB card. Similar software is provided with other GPIB cards.

To compile ctst.c: msc ctst/Ze/AL;

To link: link ctst GPIB-L;

To execute: ctst

```
/* ****************************** ctst.c ********************************** */

/*      C program that communicates with the CIM  via GPIB.
        The functions initialize(), transmit(), and receive()
        are defined in GPIB-L.

        S. Lindgren     12/15/86
*/

#include <ms-c488.h>  /* provided with CEC gpib card */
#define CIM   23      /* gpib address */

int status, length;


main ()
        {
        int i;
        float v[4];
        char cmd[20], *recv, *get_gpib (int);

        /* Initialize the computer's GPIB card and the CIM  */
        setup();
```

```
        while (1)
                {
                for (i=1 ; i<=4 ; i++)
                        {
                        /* Request an input voltage */
                        sprintf (cmd, " '?%d' 13", i);
                        tx_gpib ( CIM , cmd);

                        /* Receive an input voltage */
                        recv = get_gpib ( CIM );
                        sscanf (recv, "%f", &v[i-1]);
                        }

                /* Write the four voltages to the screen */
                printf ("%7.3f %7.3f %7.3f %7.3f\n", v[0], v[1], v[2], v[3]);
                }
        }

/* ************************************************************************** */


setup ()
        {
        int my_address=21, system_controller=0;

        initialize (&system_controller, &my_address);
        transmit (&status, "IFC UNT UNL DCL");
        wait(2);
        transmit (&status, "IFC UNT UNL MTA REN");
        wait(2);
        tx_gpib ( CIM , " 'W0;I4' 13");
        }

/* ************************************************************************** */


tx_gpib (address,command)

int address;
char *command;
        {
        char t_string[100];

        sprintf (t_string,"UNT MTA LISTEN %d DATA %s", address, command);
        transmit (&status, t_string);
        statcheck (address);
        }
```

```
/* ********************************************************************** */


char *get_gpib (address)

 int address;
        {
        char r_string[40], recv[80];

        sprintf (r_string, "UNL MLA TALK %d", address);
        transmit (&status, r_string);
        statcheck (address);

        strcpy (recv, "                    ");
        receive (&status, &length, recv);
        statcheck (address);
        return (recv);
        }

/* ********************************************************************** */


/* check the gpib status and exit if error */

statcheck (address)
int address;
        {
        if (status != 0)
                {
                printf ("Error at device %d, status = %d\n", address, status);
                exit ();
                }
        }

/* ********************************************************************** */


 wait (n)  /* wait n seconds */
 int n;
        {
        int i,j,dum;
        for (j=n ; j>=1 ; j--)
                {
                for (i=0 ; i<=32000 ; i++)
                        dum = 10*10;
                }
        }
```

EXAMPLE 1 - IPS Power System Control

This example shows how Cryomagnetics' IPS series of superconducting magnet power supplies can be interfaced to the CIM to provide RS-232C/IEEE-488 control. The example will assume that the CIM has been connected through the RS-232 interface to a computer terminal as detailed in Appendix A - Example 1 and the section on "Trying out the CIM" at the beginning of this manual. Operation of the CIM (and IPS supplies) through computer programs is a fairly simple extension of these procedures.

1. Configure the CIM as detailed in Appendix A - Example 1 and connect it to a computer terminal.

2. Connect the six of the CIM's analog I/O lines and ten of its digital lines (nine signal and one ground) to the IPS power supply as described in Table 1. Notice that this table describes standard connections between the CIM and an IPS power supply, a CRS-100 current reversing switch, a Model 24 sample-and-hold helium level meter, and a Model 50 liquid nitrogen level monitor. If these items (or similar items) were supplied by Cryomagnetics with the CIM, the interconnecting cable should also be included. For the purpose of this example, however, only the IPS power supply is required.

3. Short the output power leads of the IPS supply together. Also, short the output terminals of persistent switch heater A (PSH-A) together and do the same with PSH-B.

4. Power ON on the CIM and watch the computer terminal for the sign on message. Leave the power OFF on the IPS supply.

5. From the prompt, enter: OK -> I5
This tells the CIM that the first five analog ports (1-5) will be used as inputs - and that ports 6 - 8 will be outputs.

6. With the IPS supply in the "Local" mode, the ramp control switch in the DOWN position, and PSH - A & B switches OFF, power ON the supply. Set the current monitor to display CURRENT LIMIT and the Voltage Monitor to display VOLTAGE LIMIT.

7. Power ON the switches for PSH - A and B and confirm that the front panel LED's on the supply are illuminated. Note: For the persistent switch heater supplies to be controlled through the CIM, one must be sure that these front panel switches remain in the ON position.

8. On the rear panel of the IPS supply, switch to REMOTE mode. The current and voltage monitor displays on the IPS supply should read zero since no limits have been sent to the supply yet.

9. From the computer terminal, enter: OK -> S7 = 3.50
Since analog port 7 is being used to control the voltage limit, this command sets the VOLTAGE LIMIT of the IPS supply to 3.50 volts. Check to see that the voltage monitor display reads 3.50 volts.

TABLE 1

STANDARD CRYOSYSTEM INTERFACE SCHEME

CIM CONNECTION	CIM INPUT OUTPUT	CABLE TYPE	FUNCTION	CRYOSYSTEM CONNECTION
BNC (1)	I	COAX	IPS CURRENT MONITOR	IPS/BNC (1)
BNC (2)	I	COAX	IPS VOLTAGE MONITOR	IPS/BNC (2)
BNC (3)	I	COAX	MAGNET VOLTAGE MONITOR	IPS/BNC (3)
BNC (4)	I	COAX	LHE LEVEL MONITOR	MODEL 24/BNC
BNC (5)	I	COAX	LN2 LEVEL MONITOR	MODEL 50/BNC
BNC (6)	0	COAX	IPS CURRENT LIMIT SET	IPS/BNC (4)
BNC (7)	0	COAX	IPS VOLTAGE LIMIT SET	IPS/BNC (5)
BNC (8)	0	COAX	IPS CHARGE RATE SET	IPS/BNC (6)
I7 - PIN 18	I	RED/BLACK	LHE MONITOR ALARM STATUS	MODEL 24/PIN 6
I6 - PIN 5	I	WHITE/YELLOW	LHE MONITOR READING VALID	MODEL 24/PIN 4
I5 - PIN 17	I	WHITE/BLACK	NEGATIVE CURRENT DETECTED	CRS-100/TERMINAL 5
I4 - PIN 4	I	WHITE/RED	POSITIVE CURRENT DETECTED	CRS-100/TERMINAL 4
I3 - PIN 16	I	WHITE/GREEN	ZERO CURRENT DETECTED	CRS-100/TERMINAL 3
I2 - PIN 3	I	ORANGE	IPS ERROR DETECTOR STATUS	IPS/TERMINAL 9
I1 - PIN 15	I	VIOLET	IPS VOLTAGE LIMIT STATUS	IPS/TERMINAL 7
I0 - PIN 2	I	RED	IPS CURRENT LIMIT STATUS	IPS/TERMINAL 8
07 - PIN 22	0	WHITE/BLUE	LHE MONITOR UPDATE REQUEST	MODEL 24/PIN 1
06 - PIN 9	0	BLACK	CURRENT DIRECTION REQUEST	CRS-100/TERMINAL 2
05 - PIN 21	0	TAN	IPS FAST RAMP	IPS/TERMINAL 10
04 - PIN 8	0	BROWN	IPS ERROR DETECTOR ENABLE/RESET	IPS/TERMINAL 6
03 - PIN 20	0	GRAY	IPS PERSISTENT SWITCH HEATER A	IPS/TERMINAL 5
02 - PIN 7	0	YELLOW	IPS PERSISTENT SWITCH HEATER B	IPS/TERMINAL 4
01 - PIN 19	0	BLUE	IPS RAMP UP	IPS/TERMINAL 3
00 - PIN 6	0	PINK	IPS RAMP DOWN	IPS/TERMINAL 2
I/01 - PIN 24	I/0	RED/YELLOW	USER DEFINABLE I/0 BIT 1 (SYNC)	
I/02 - PIN 11	I/0	RED/GREEN	USER DEFINABLE I/0 BIT 2	
D/GND - PIN 1	-	GREEN	DIGITAL GROUND	IPS/TERMINAL 1
D/GND - PIN 23	-	WHITE	DIGITAL GROUND	CRS-100/TERMINAL 6 AND MODEL 24 PINS 8,9

10. From the computer terminal, enter: OK -> S6 = 1.430
This command sets the CURRENT LIMIT of the IPS supply to 14.30 amperes. (All IPS supplies under or equal to 100 ampere maximum current ratings have a 1 volt = 10 amperes ratio at their remote CURRENT LIMIT programming inputs). Check to see that the current monitor display reads 14.30 amperes.

11. Set the current monitor on the IPS supply to display CHARGE RATE. From the computer terminal, enter: OK -> S8 = 1.00
This command sets the CHARGE RATE of the IPS supply to 0.100 amperes per second (all IPS supplies have a 1 volt = 0.1 ampere/second ratio at their remote CHARGE RATE programming inputs). Check to see that the current monitor display reads 0.10 ampere/second.

12. Control of the IPS supply's ramp UP/PAUSE/DOWN switch, fast ramp switch, quench (error) detector switch, and persistent switch heaters are achieved through the CIM's 8-bit digital output port. Monitoring of the IPS supply's error detector, voltage and current limit status indicators are provided through the CIM's 8-bit digital input port. The command SD = n is used to control all of the output port functions while ?D is used for monitoring the input port functions (see Figure 2, the CIM Command List). The 8-bit digital ports, when configured as shown in Table 1, are easy to use. An example of the use of the ports would be:

8-BIT DIGITAL OUTPUT PORT DESIGNATIONS

VALUE	BIT #	CONTROL FUNCTION
128	07	---
64	06	---
32	05	IPS FAST RAMP (I = FAST RAMP, 0 = NORMAL RAMP)
16	04	IPS ERROR DETECTOR ENABLE/RESET (I = ENABLED, 0 = DISABLED/RESET)
8	03	IPS PSH-A (1 = ON, 0 = OFF)
4	02	IPS PSH-B (1 = ON, 0 = OFF)
2	01	IPS RAMP UP 0 0 1 1
1	00	IPS RAMP DOWN 0 1 0 1
		PAUSE RAMP DOWN RAMP UP PAUSE

8-BIT DIGITAL INPUT PORT DESIGNATIONS

VALUE	BIT #	CONTROL FUNCTION
128	I7	---
64	I6	---
32	I5	---
16	I4	---
8	I3	---
4	I2	IPS ERROR DETECTOR STATUS (1 = ERROR DETECTED, 0 = NO)
2	I1	IPS VOLTAGE LIMIT LED STATUS (1 = VOLTAGE LIMIT, 0 = NO)
1	I0	IPS CURRENT LIMIT LED STATUS (1 = CURRENT LIMIT, 0 = NO)

SAMPLE SESSION

```
OK ->    SD = 16      ; 16 = value of the error detector bit. Command
                        enables the error detector (ED).
OK ->    SD = 24      ; 16 + 8 = 24 => 8 = value of PSH-A bit. Command
                        turns on the PSH-A while keeping the ED enabled
OK ->    SD = 26      ; 16 + 8 + 2 = 26   begins ramp UP while maintaining
                        ED enabled and PSH-A ON.
```

ALLOW THE IPS SUPPLY TO RAMP UP TO THE CURRENT LIMIT ESTABLISHED IN STEP 10. WHEN THE SUPPLY HAS REACHED CURRENT LIMIT,

```
OK ->    ?1           ; Check the output current.
1.430                 ; Output current is 14.30 amperes.
OK ->    ?D           ; Check limit status.
17                    ; Indicates bits 4 (value = 16) and 0 (value = 1) are
                        high. Since only bits 0, 1, and 2 are of interest,
                        this means the IPS ED status = NO ERROR (bit 2), the
                        IPS voltage limit = NO LIMIT (bit 1) and the IPS
                        current limit = LIMITED (bit 0).
OK ->    SD = 18      ; 16 + 2 = 18 => Turn OFF PSH-A while keeping IPS supply
                        in ramp UP mode (against current limit) and with ED
                        enabled.
OK ->    SD = 49      ; 32 + 16 + 1 = 49 =>  Fast ramps output current back
                        to zero amperes.
OK ->    ?1           ; Check that the output current has returned to zero.
0.000
OK ->    ?2           ; Check IPS output voltage.
0.000
OK ->    ?3           ; Check voltage at magnet voltage taps.
0.000
```

When connected as described above, the CIM can be used to control every function on the front and rear panels of the IPS power supply other than turning on the main circuit breaker. Also, as shown in Table 1, the CIM has ports available to monitor and control other cryosystem instruments. The CIM can even be used as a data logging device without computer interaction.