1700 USERS MANUAL

REVISED: 5/1/94

Omega Engineering
One Omega Drive
P O Box 4047
Stamford, CT 06907
Phone: 1-800-DAS-IEEE
Fax: 203-359-7990
e-mail: das@omega.com
www.omega.com

The 1700 series are not intrinsically safe devices and should not be used in an explosive environment unless enclosed in approved explosion-proof housings

TABLE OF CONTENTS

Chapter 1
Getting Started

Introduction

The 1700 Series of Digital I/O to Computer Interfaces provide computer monitoring and control of devices through solid state relays or TTL signals. The status of inputs and outputs is communicated to the host in ASCII format using RS-232C or RS-485 serial communications.

With the 1700 series the user can control digital inputs and outputs individually or all at once. Any channel may be designated as an input or output by the user. Many industrial applications require a safe start-up condition to prevent accidents at critical points in the process.The onboard nonvolatile EEPROM memory stores the user-specified initial condition (input or output) of each channel; thereby eliminating the need for software initialization routines when power is applied or restored.

The 1700 series may be setup in special modes which allow them to communicate without being polled by a host computer. Collectively these modes are called Continuous Input/Output Modes. In many applications the burden on the host may be greatly simplified and in some cases the host may be eliminated altogether.

The 1700 series include:

1711/1712	15 channel I/O modules.
H1750	24 channel I/O board.
H1770	64 channel I/O board.

Getting Started

The instructions in this chapter cover all 1700 models; however, for simplicity we use the 1711 & 1712 in the figures. If you have an H1700 board see the appropriate appendix for instructions on getting started.

Default Mode

All models contain an EEPROM (Electrically Erasable Programmable Read Only Memory) to store setup information. The EEPROM replaces the usual array of switches necessary to specify baud rate, address, parity, etc. The memory is nonvolatile which means that the information is retained even if power is removed. No batteries are used so it is never necessary to open the module case.

The EEPROM provides tremendous system flexibility since all of the module's setup parameters may be configured remotely through the communications port without having to physically change switch settings. There

is one minor drawback in using EEPROM instead of switches; there is no visual indication of the setup information in the module. It is impossible to tell just by looking at the module what the baud rate, address, parity and other settings are. It is difficult to establish communications with a module whose address and baud rate are unknown. To overcome this, each module has an input pin labelled DEFAULT*. By connecting this pin to Ground, the module is put in a known communications setup called Default Mode.

The Default Mode setup is: 300 baud, one start bit, eight data bits, one stop bit, no parity, any address is recognized.

Grounding the DEFAULT* pin does not change any of the setups stored in EEPROM. The setup may be read back with the Read Setup (RS) command to determine all of the setups stored in the module. In Default Mode, all commands are available.

A module in Default Mode will respond to any address except the four identified illegal values (NULL, CR, $, #). A dummy address must be included in every command for proper responses. The ASCII value of the module address may be read back with the RS command. An easy way to determine the address character is to deliberately generate an error message. The error message outputs the module's address directly after the "?" prompt.

Setup information in a module may be changed at will with the SetUp (SU) command. Baud rate and parity setups may be changed without affecting the Default values of 300 baud and no parity. When the DEFAULT* pin is released, the module automatically performs a program reset and configures itself to the baud rate and parity stored in the setup information.

The Default Mode is intended to be used with a single module connected to a terminal or computer for the purpose of identifying and modifying setup values. In most cases, a module in Default Mode may not be used in a string with other modules.

RS-232 & RS-485 Quick Hook-Up

Software is not required to begin using your 1700 module. We recommend that you begin to get familiar with the module by setting it up on the bench. Start by using a dumb terminal or a computer that acts like a dumb terminal. Make the connections shown in the quick hook-up drawings, Figures 1.1 or 1.2. Put the module in the Default Mode by grounding the Default* terminal. Initialize the terminal communications package on your computer to put it into the "terminal" mode. Since this step varies from computer to computer, refer to your computer manual for instructions.

Begin by typing $1DI and pressing the Enter or Return key. The module will

respond with an * followed by the data reading at the input, typically 8000. Once you have a response from the module you can turn to the Chapter 4 and get familiar with the command set.

All modules are shipped from the factory with a setup that includes a channel address of 1, 300 baud rate, no linefeeds, no parity, no echo and two-character delay. Refer to the Chapter 5 to configure the module to your application.

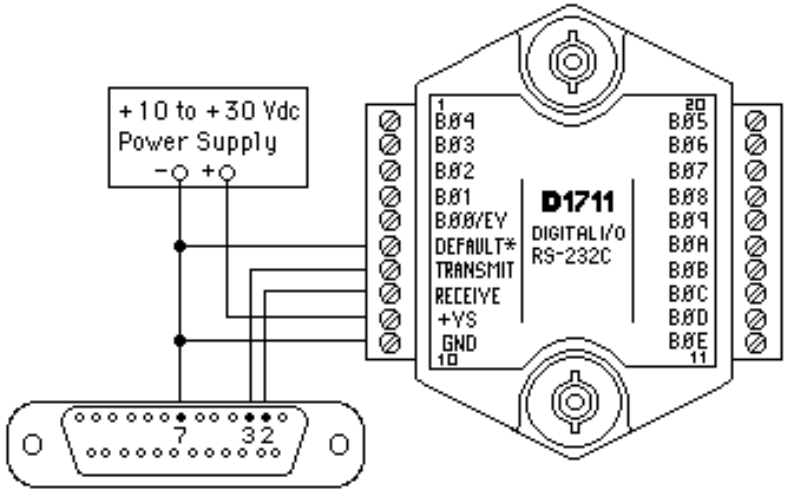


Figure 1.1 RS-232 Quick Hook-Up.

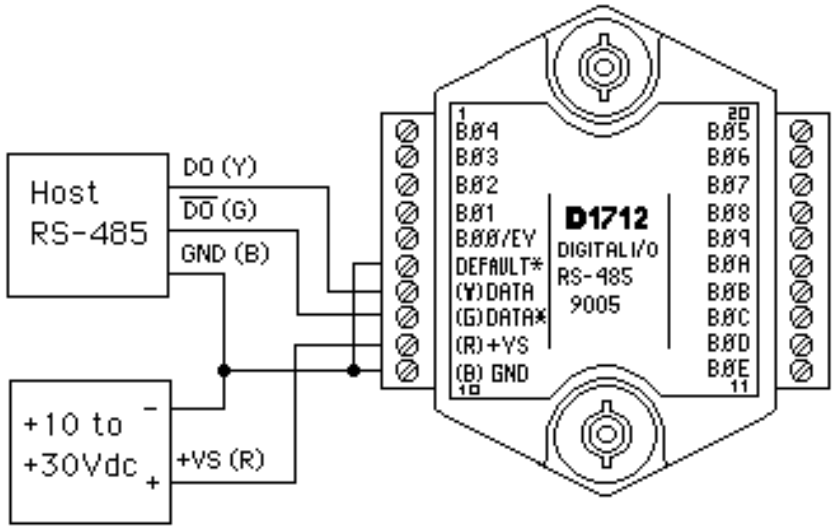


Figure 1.2 RS-485 Quick Hook-Up.

RS-485 Quick Hook-up to a RS-232 port

An RS-485 module may be easily interfaced to an RS-232C terminal for evaluation purposes. This connection is only suitable for benchtop operation and should never be used for a permanent installation. Figure 1.3 shows the hook-up. This connection will work provided the RS-232C transmit output is current limited to less than 50mA and the RS-232C receive threshold is greater than 0V. All terminals that use 1488 and 1489 style interface IC's will satisfy this requirement. With this connection, characters generated by the terminal will be echoed back. To avoid double characters, the local echo on the terminal should be turned off.

If the current limiting capability of the RS-232C output is uncertain, insert a 100Ω to 1kΩ resistor in series with the RS-232 output.

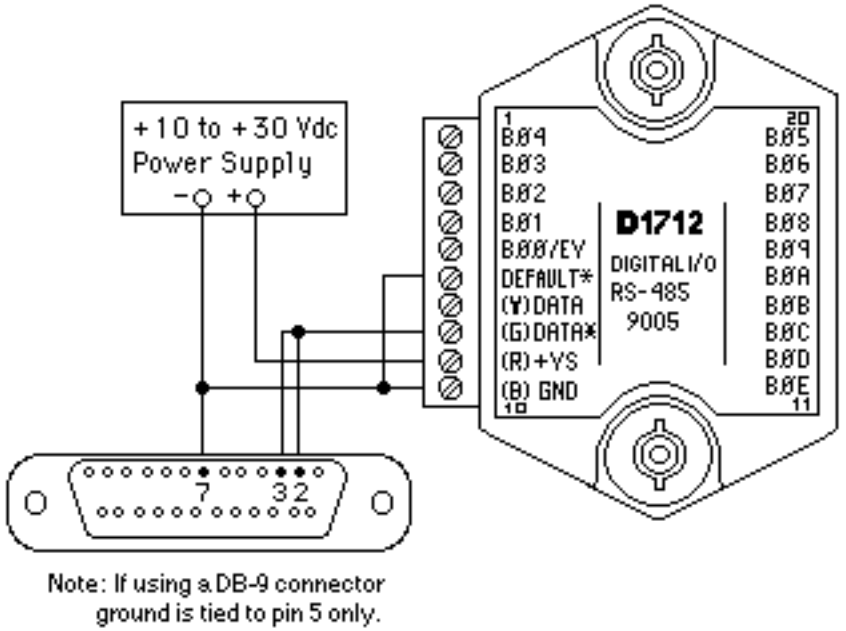


Figure 1.3 RS-485 Quick Hook-Up with an RS-232 Port.

S1000 Software
Software is available to assist the user in setting up the 1700 modules. The S1000 software runs on the IBM compatible PC's and is available free of charge.

Chapter 2
Functional Description

The D1700 Digital I/O modules provide remote control and monitoring of on-off signals in response to simple commands from a host computer. Digital commands are transmitted to the D1700 units using standard RS-232 or RS-485 communications links. Commands and responses are in the form of simple English ASCII character strings for ease of use. The ASCII protocol allows the units to be interfaced with dumb terminals and modems as well as intelligent controllers and computers.

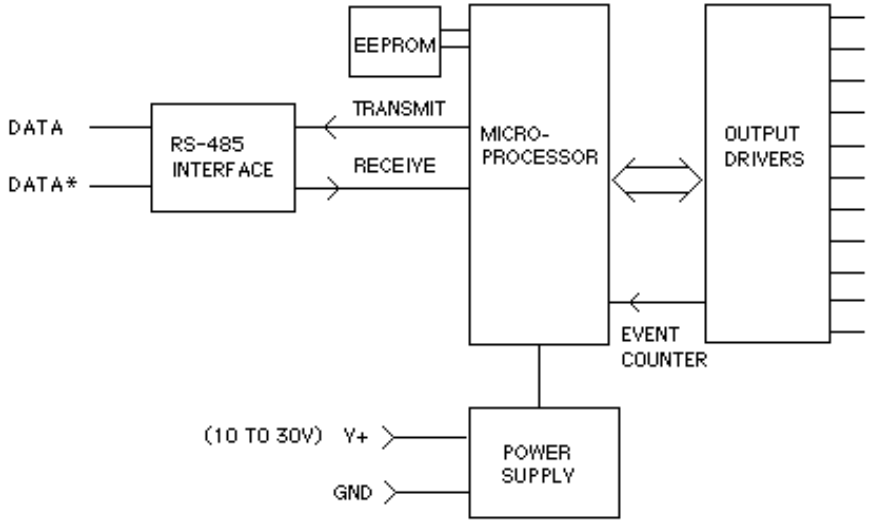


Figure 2.1 Digital I/O Functional Block Diagram.

Figure 2.1 shows a functional block diagram of a D1712. An 8-bit CMOS microprocessor is used to provide an intelligent interface between the host and the bi-directional I/O lines. The microprocessor receives commands and data from the host computer through a serial communications port. Specialized communications components are used to interface the microprocessor to the RS-485 communications standard. Commands received by the microprocessor are thoroughly checked for syntax and data errors. Valid commands are then processed to complete the desired function. A wide variety of commands are available to configure and control the digital I/O

lines. Responses to the host commands are then produced by the microprocessor and transmitted back to the host over the RS-485 serial link.

An Electrically Erasable Programmable Read-Only Memory (EEPROM) is used to retain important data even if the module is powered down. The EEPROM contains setup information such as the address, baud rate, and parity as well as I/O configuration data.

Each digital line on the D1712 is bidirectional and may be individually configured by the user to be an input or an output. The direction assignments of all the lines are stored in EEPROM so that the lines are automatically configured each time the D1712 is powered up.

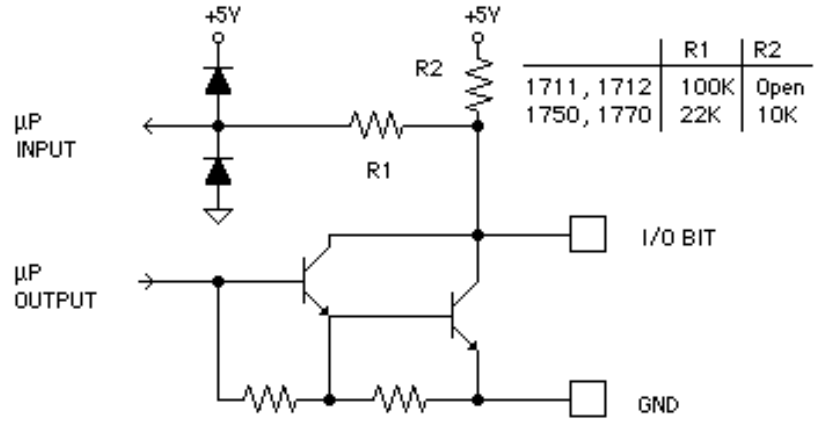


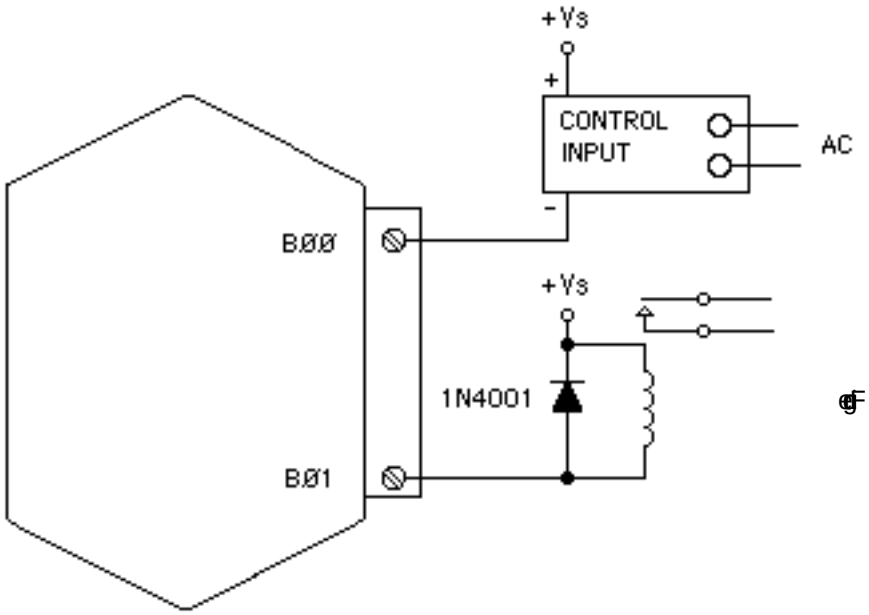
Figure 2.2 Digital I/O Circuit.

Figure 2.2 is a detail diagram of a single I/O line circuit. The output driver is a darlington circuit capable of sinking 100mA with a maximum output voltage of 30V. The maximum total current that may be handled by the D1711 or D1712 package is 1A. The output saturation voltage at 100mA is 1.2V max. Pullup resistors are not provided in the modules.

When the I/O pin is configured as an input, the output driver is turned off. The input state is read by the microprocessor through an input protection circuit consisting of a 100K resistor and diodes. This allows the input values

to range from 0 to 30V without damaging the microprocessor. Note that with the output driver off, the 100K resistor produces a leakage current if the I/O line is greater than +5V.

When a read function is performed on an I/O pin, the actual logical state of the pin is read back even if the pin is configured as an output. this provides a means to verify the state of the output.



2.3 Digital Outputs Used With Relays.

Figure 2.3 shows typical connections to solid-state relays and electromechanical relays. When electromechanical relays are used, always include a flyback diode to avoid damage to the output driver.

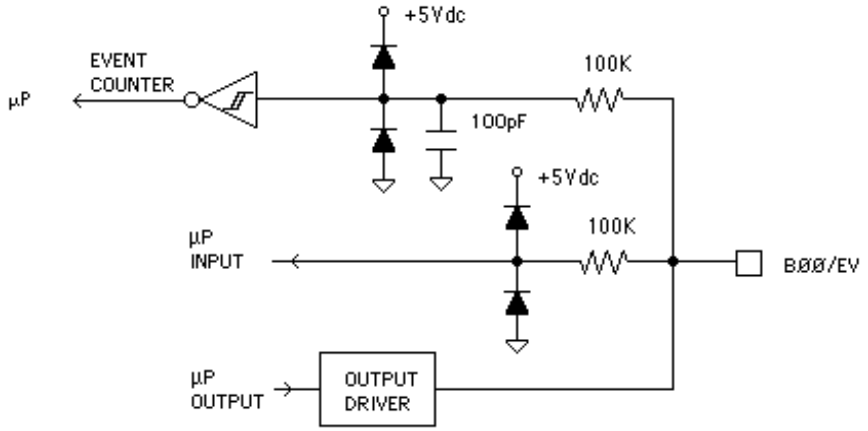


Figure 2.4 D1711, D1712 Events Counter Circuit.

Figure 2.4 is a detail schematic of the B00/EV pin. This pin is identical to all other pins but it has the event counter circuitry added on. The event counter circuitry consists of input protection components and a capacitor to provide some noise filtering. The event data is buffered by a Schmitt-trigger gate which outputs the event signal to the microprocessor.

The microprocessor contains a user-programmable filter to debounce the event counter input. The filter is necessary when the event signal is derived from mechanical contacts such as switches or relays. The filter constant is user-selectable for 0,5,20 or 50ms. Figure 2.5 shows the filter action for the 5ms setting.

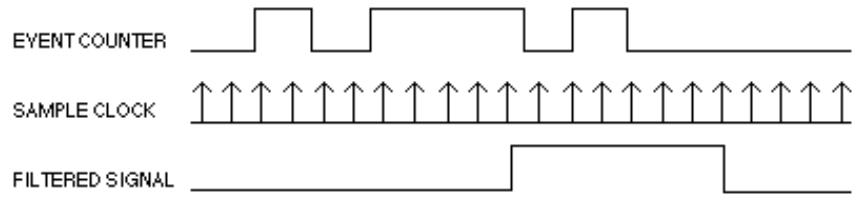


Figure 2.5 Event Counter Debounce Filter.

The microprocessor samples the event input at 1ms intervals. The input signal must be high for at least five consecutive samples before it will be

counted as a high transition. Similarly, the input must be low for five sample periods before it is counted as a low signal. If the filter is set for 20ms, the input must be stable for 20 consecutive samples, etc.

The last major block in the diagram is the power supply. The power supply converts the raw 10 to 30 volts supplied by the user into regulated voltages used in the module. It produces +5V necessary to operate the microprocessor and EEPROM. On RS-232 units, the power supply produces ±10V necessary for the RS-232 communications standard.

Chapter 3
Communications

Introduction

The D1700 modules have been carefully designed to be easy to interface to all popular computers and terminals. All communications to and from the modules are performed with printable ASCII characters. This allows the information to be processed with string functions common to most high-level languages such as BASIC. For computers that support RS-232C, no special machine language software drivers are necessary for operation. The modules can be connected to auto-answer modems for long-distance operation without the need for a supervisory computer. The ASCII format makes system debugging easy with a dumb terminal.

This system allows multiple modules to be connected to a communications port with a single 4-wire cable. Up to 32 RS-485 modules may be strung together on one cable; 124 with repeaters. A practical limit for RS-232C units is about ten, although a string of 124 units is possible. The modules communicate with the host on a polling system; that is, each module responds to its own unique address and must be interrogated by the host. A module can never initiate a communications sequence. A simple command/response protocol must be strictly observed to avoid communications collisions and data errors.

Communication to the D1700 modules is performed with two- or three-character ASCII command codes such as DO for Digital Output. A complete description of all commands is given in the Chapter 4. A typical command/response sequence would look like this:

Command: **$1RD**
Response: ***+99999.99**

A command/response sequence is not complete until a valid response is received. The host may not initiate a new command until the response from a previous command is complete. Failure to observe this rule will result in communications collisions. A valid response can be in one of three forms:

1) a normal response indicated by a ‘ * ‘ prompt
2) an error message indicated by a ‘ ? ‘ prompt
3) a communications time-out error

When a module receives a valid command, it must interpret the command, perform the desired function, and then communicate the response back to the host. Each command has an associated delay time in which the module is busy calculating the response. If the host does not receive a response in an appropriate amount of time specified in Table 3.1, a communications time-out error has occurred. After the communications time-out it is assumed that no response data is forthcoming. This error usually results when an improper command prompt or address is transmitted. The table below lists the timeout specification for each command:

Mnemonic	Timeout
ACK, CB, CE, CP, DI, DO, RA, RAB, RAP, RB, RD,RP, RS, RSU, SB, SP, RIA, RCM, RR, WE	≤ 5.0 ms
EC, RE, RWT, RID, RIV, RCT, AIB, AIP, AOB, AOP, CIA, CMC, CMD, CME, CMT	≤ 15.0 ms
WT, CT, SU, AIO, ID, IV	≤ 100 ms

Table 3.1 Response Timeout Specifications.

The timeout specification is the turn-around time from the receipt of a command to when the module starts to transmit a response.

Data Format

All modules communicate in standard NRZ asynchronous data format. This format provides one start bit, seven data bits, one parity bit and one stop bit for each character.

RS-232C

RS-232C is the most widely used communications standard for information transfer between computing equipment. RS-232C versions of the 1700 series will interface to virtually all popular computers without any additional hardware. Although the RS-232C standard is designed to connect a single piece of equipment to a computer, this system allows for several modules to be connected in a daisy-chain network structure.The advantages offered by the RS-232C standard are:

1) widely used by all computing equipment
2) no additional interface hardware in most cases
3) separate transmit and receive lines ease debugging
4) compatible with dumb terminals

However, RS-232C suffers from several disadvantages:

1) low noise immunity
2) short usable distance - 50 to 200 feet
3) maximum baud rate - 19200
4) greater communications delay in multiple-module systems
5) less reliable–loss of one module breaks chain
6) wiring is slightly more complex than RS-485
7) host software must handle echo characters

Single Module Connection

Figure 1.1 shows the connections necessary to attach one module to a host. Use the Default Mode to enter the desired address, baud rate, and other setups (see Setups). The use of echo is not necessary when using a single module on the communications line.

Multi-party Connection

RS-232C is not designed to be used in a multi-party system; however the D1700 modules can be daisy-chained to allow many modules to be connected to a single communications port. The wiring necessary to create the daisy-chain is shown in Figure 3.1. Notice that starting with the host, each Transmit output is wired to the Receive input of the next module in the daisy chain. This wiring sequence must be followed until the output of the last module in the chain is wired to the Receive input of the host. All modules in the chain must be setup to the same baud rate and must echo all received data (see Setups). Each module must be setup with its own unique address to avoid communications collisions (see Setups). In this network, any characters transmitted by the host are received by each module in the chain and passed on to the next station until the information is echoed back to the Receive input of the host. In this manner all the commands given by the host are examined by every module. If a module in the chain is correctly addressed and receives a valid command, it will respond by transmitting the response on the daisy chain network. The response data will be ripple through any other modules in the chain until it reaches its final destination, the Receive input of the host.

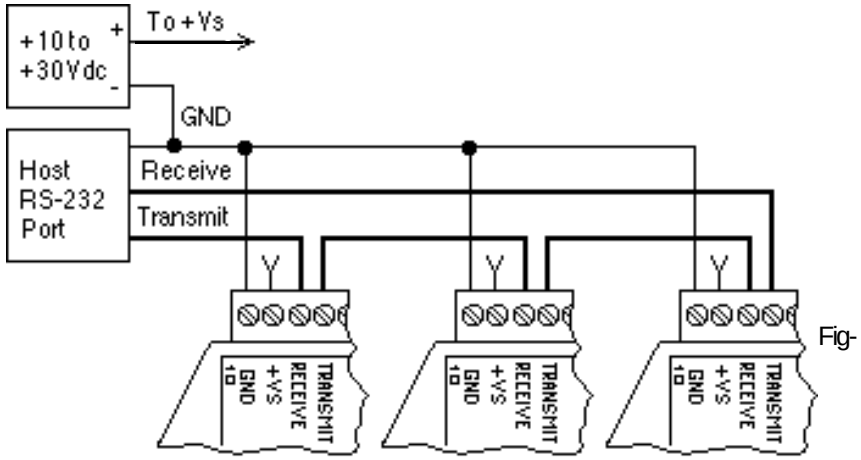


Figure 3.1 RS-232 Daisy Chain.

The daisy chain network must be carefully implemented to avoid the pitfalls inherent in its structure. The daisy-chain is a series-connected structure and any break in the communications link will bring down the whole system. Several rules must be observed to create a working chain:

1. All wiring connections must be secure; any break in the wiring, power, ground or communications will break the chain.
2. All modules must be plugged into their connectors.
3. All modules must be setup for the same baud rate.
4. All modules must be setup for echo.

Software Considerations

If the host device is a computer, it must be able to handle the echoed command messages on its Receive input along with the responses from the module. This can be handled by software string functions by observing that a module response always begins with a ‘ * ‘ or ‘ ? ‘ character and ends with a carriage return.

A properly addressed D1700 module in a daisy chain will echo all of the characters in the command including the terminating carriage return. Upon receiving the carriage return, the module will immediately calculate and transmit the response to the command. During this time, the module will not echo any characters that appear on its receive input. However, if a character is received during this computation period, it will be stored in the

module's internal receive buffer. This character will be echoed after the response string is transmitted by the module. This situation will occur if the host computer appends a linefeed character on the command carriage return. In this case the linefeed character will be echoed after the response string has been transmitted.

The daisy chain also affects the command timeout specifications. When a module in the chain receives a character it is echoed by re-transmitting the character through the module's internal UART. This method is used to provide more reliable communications since the UART eliminates any slewing errors caused by the transmission lines. However, this method creates a delay in propagating the character through the chain. The delay is equal to the time necessary to retransmit one character using the baud rate setup in the module:

Baud Rate	Delay
300	33.30ms
600	16.70ms
1200	8.33ms
2400	4.17ms
4800	2.08ms
9600	1.04ms
19200	520µs
38400	260µs

One delay time is accumulated for each module in the chain. For example, if four modules are used in a chain operating at 1200 baud, the accumulated delay time is 4 X 8.33 ms = 33.3 ms This time must be added to the times listed in Table 3.1 to calculate the correct communications time-out error.

For modules with RS-232C outputs, the programmed communications delay specified in the setup data (see Chapter 5) is implemented by sending a NULL character (00) followed by an idle line condition for one character time. This results in a delay of two character periods. For longer delay times specified in the setup data, this sequence is repeated. Programmed communications delay is seldom necessary in an RS-232C daisy chain since each module in the chain adds one character of communications delay.

Changing Baud Rate

It is possible to change the baud rate of an RS-232C daisy chain on-line. This process must be done carefully to avoid breaking the communications link.

1. Use the SetUp (SU) command to change the baud rate setup on each

module in the chain. Be careful not to generate a reset during this process. A reset can be caused by the Remote Reset (RR) command or power interruptions.

2. Verify that all the modules in the chain contain the new baud rate setup using the Read Setup (RS) command. Every module in the chain must be setup for the same baud rate.

3. Remove power from all the modules for at least 10 seconds. Restore power to the modules. This generates a power-up reset in each module and loads in the new baud rate.

4. Change the host baud rate to the new value and check communications.

5. Be sure to compensate for a different communications delay as a result of the new baud rate.

Using A Daisy-Chain With A Dumb Terminal

A dumb terminal can be used to communicate to a daisy-chained system. The terminal is connected in the same manner as a computer used as a host. Any commands typed into the dumb terminal will be echoed by the daisy chain. To avoid double characters when typing commands, set the terminal to full duplex mode or turn off the local echo. The daisy chain will provide the input command echo.

RS-485

RS-485 is a recently developed communications standard to satisfy the need for multidropped systems that can communicate at high data rates over long distances. RS-485 is similar to RS-422 in that it uses a balanced differential pair of wires switching from 0 to 5V to communicate data. RS-485 receivers can handle common mode voltages from -7V to +12V without loss of data, making them ideal for transmission over great distances. RS-485 differs from RS-422 by using one balanced pair of wires for both transmitting and receiving. Since an RS-485 system cannot transmit and receive at the same time it is inherently a half-duplex system. RS-485 offers many advantages over RS-232C:

1) balanced line gives excellent noise immunity
2) can communicate with modules at 38400 baud
3) communications distances up to 4,000 feet.
4) true multidrop; modules are connected in parallel
5) individual modules may be disconnected without affecting other modules
6) up to 32 modules on one line; 124 with repeaters
7) no communications delay due to multiple modules
8) simplified wiring using standard telephone cable

RS-485 does have disadvantages. Very few computers or terminals have built-in support for this new standard. Interface boards are available for the IBM PC and compatibles and other RS-485 equipment will become available as the standard gains popularity. An RS-485 system usually requires an interface.

We offer interface converters to convert RS-232C to RS-485. These converters also include power supplies to power up to 32 modules. To expand an RS-485 system even further, repeater boxes are available from us to string up to 124 modules on one communications port.

RS-485 Multidrop System

Figure 3.2 illustrates the wiring required for multiple-module RS-485 system. Notice that every module has a direct connection to the host system. Any number of modules may be unplugged without affecting the remaining modules. Each module must be setup with a unique address and the addresses can be in any order. All RS-485 modules must be setup for no echo to avoid bus conflicts (see Setup). Also note that the connector pins on each module are labelled with notations (B), (R), (G), and (Y). This designates the colors used on standard 4-wire telephone cable:

Label	Color
(B) GND	Black
(R) V+	Red
(G) DATA*	Green
(Y) DATA	Yellow

This color convention is used to simplify installation. If standard 4-wire telephone cable is used, it is only necessary to match the labeled pins with the wire color to guarantee correct installation.

DATA* on the label is the complement of DATA (negative true).

To minimize unwanted reflections on the transmission line, the bus should be arranged as a line going from one module to the next. ‘Tree’ or random structures of the transmission line should be avoided. For wire runs greater than 500 feet total, each end of the line should be terminated with a 220Ω resistor connected between DATA and DATA*.

When using a bi-directional RS-485 system, there are unavoidable periods

of time when all stations on the line are in receive mode. During this time, the communications lines are left floating and are very susceptible to noise. To prevent the generation of random characters, the lines should be biased in a MARK condition as shown in Figure 3.2. The 1K resistors are used to keep the DATA line more positive than the DATA* line when none of the RS-485 transmitters are on. When enabled, the low impedance of an RS-485 driver easily overcomes the load presented by the resistors.

Special care must be taken with very long busses (greater than 1000 feet) to ensure error-free operation. Long busses must be terminated as described above. The use of twisted cable for the DATA and DATA* lines will greatly enhance signal fidelity. Use parity and checksums along with the '#' form of all commands to detect transmission errors. In situations where many modules are used on a long line, voltage drops in the power leads becomes an important consideration. The GND wire is used both as a power connection and the common reference for the transmission line receivers in the modules. Voltage drops in the GND leads appear as a common-mode voltage to the receivers. The receivers are rated for a maximum of -7V. of common-mode voltage. For reliable operation, the common mode voltage should be kept below -5V.

To avoid problems with voltage drops, modules may be powered locally rather than transmitting the power from the host. Inexpensive 'calculator' type power supplies are useful in remote locations. When local supplies are used, be sure to provide a ground reference with a third wire to the host or through a good earth ground. With local supplies and an earth ground, only two wires for the data connections are necessary.

Communications Delay

All modules with RS-485 outputs are setup at the factory to provide two units of communications delay after a command has been received (see Chapter 5). This delay is necessary when using host computers that transmit a carriage return as a carriage return-linefeed string. Without the delay, the linefeed character may collide with the first transmitted character from the module, resulting in garbled data. If the host computer transmits a carriage return as a single character, the delay may be set to zero to improve communications response time.

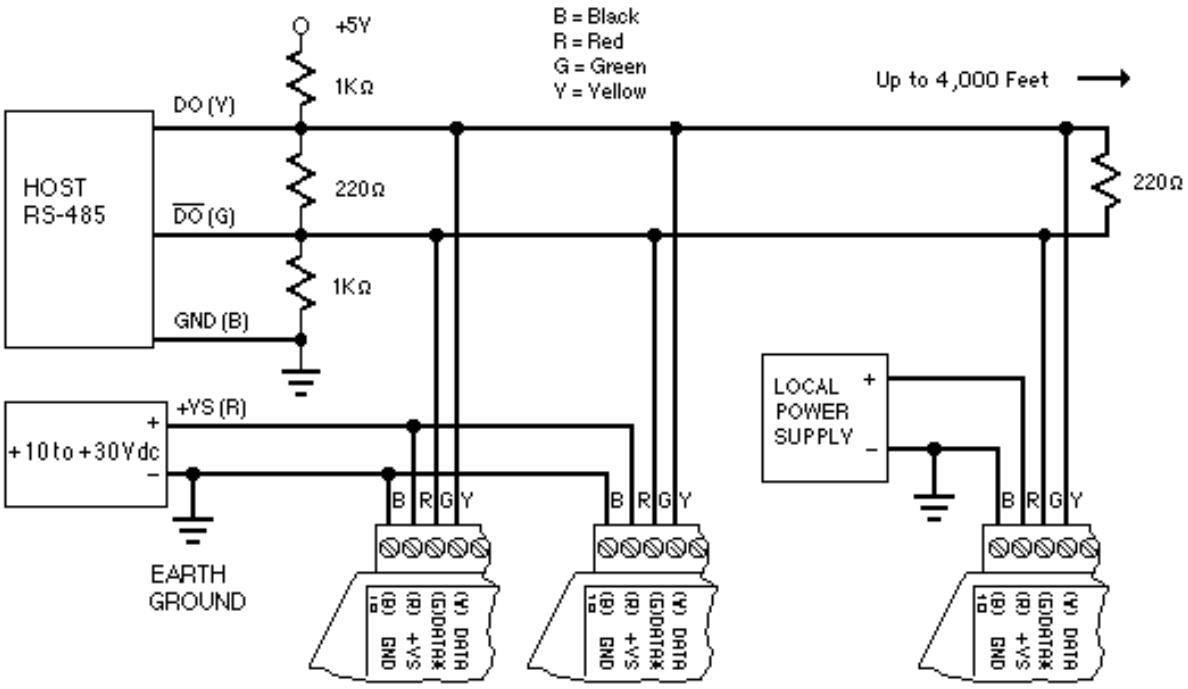


Figure 3.2 RS-485 Network.

Chapter 4
1700 Command Set

The D1700 modules operate with a simple command/response protocol to control all module functions. A command must be transmitted to the module by the host computer or terminal before the module will respond with useful data. A module can never initiate a communications sequence (unless it is setup for Continuous Output Mode (see Chapter 6). A variety of commands exists to exploit the full functionality of the modules. A list of available commands and a sample format for each command is listed in Table 4.1.

Command Structure

Each command message from the host must begin with a command prompt character to signal to the modules that a command message is to follow. There are two valid prompt characters; a dollar sign character ($) is used to generate a short response message from the module. A short response is the minimum amount of data necessary to complete the command. The second prompt character is the pound sign character (#) which generates long responses (the long response format will be covered a little later).

The prompt character must be followed by a single address character identifying the module to which the command is directed. Each module attached to a common communications port must be setup with its own unique address so that commands may be directed to the proper unit. Module addresses are assigned by the user with the SetUp (SU) command. For ease in debugging, printable ASCII characters such as ‘1’ (ASCII $31) or ‘A’ (ASCII $41) are the best choices for address characters.

The address character is followed by a two or three character command which identifies the function to be performed by the module. All of the available commands are listed in Table 4.1 along with a short function definition. All commands are described in full later in this section. Commands must be transmitted as upper-case characters.

A two-character checksum may be appended to any command message as a user option. See ‘Checksum’ section below.

All commands must be terminated by a Carriage Return character (ASCII $0D). (In all command examples in this text the Carriage Return is either implied or denoted by the symbol ‘CR’.)

Data Structure
Many commands require additional data values to complete the command definition as shown in the example commands in Table 4.1. The particular data necessary for these commands is described in full in the complete command descriptions.

The majority of data values used with the D1700 series is in the form of hexadecimal (base 16) numbers representing digital data. Each hexadecimal ASCII digit represents four bits of digital data. For example: E5 (hex) = 1110 0101 (binary)

An example command may look like this:

Command: $1DOFFFF

This is an example of the Digital Output (DO) command. This particular command would be used to turn on 16 bits of data represented by 'FFFF'.

Data read back from the Event Counter with the Read Events (RE) command is in the form of a seven-digit decimal number. For example:

Command: $1RE
Response: *0000123

Analog data is represented in a form of sign, five digits, decimal point and two additional digits:

Command: $1RWT
Response: *+00010.00

The analog data format is used with the WT and CT commands.

Bit Addresses
There are several commands that are used to manipulate a single bit. These commands require a bit address so that the desired action will be directed to the correct I/O line. Bit addresses may be specified in two different formats, the Bit format and the Position format.

The Bit format specifies the desired I/O line using a two-character hexadecimal number preceded by the letter 'B'. For example:

Command: $1SB0F

This is an example of the Set Bit (SB) command. The command action is directed to the address 0F (hexadecimal).

The Position format uses a decimal address preceded by the letter ‘P’. For example:

Command: $1SP15

This is an example of the Set Position (SP) command. The command action is directed to the I/O line address 15 (decimal). Note that the last two command examples produce the same results. The choice of the Bit notation or Position notation is strictly a matter of user preference.

Logic Convention

Most devices in the D1700 family feature open-collector transistor outputs to interface directly with solid-state relays. The control input of the relay is generally connected between the output line and a source of power. With conventional relays, the output transistor is turned on to sink current through the relay, turning the relay on. The logic convention used in the D1700 series requires a logical ‘1’ to turn on the relay. This means that the output voltage measured at the I/O line will be near ground potential (low). This is an example of negative logic.

The logic convention used to read input data is positive logic. This means that a ‘high’ voltage potential at the I/O line will be read back as a logical ‘1’. A low potential will be read back as a logical ‘0’.

Write Protection

Many of the commands listed in Table 4.1 are under the heading of ‘Write Protected Commands’. These commands are used to alter setup data in the module’s EEPROM. These commands are write protected to guard against accidental loss of setup data. All write-protected commands must be preceded by a Write Enable (WE) command before the protected command may be executed.

Miscellaneous Protocol Notes

The address character must transmitted immediately after the command prompt character. After the address character the module will ignore any character below ASCII $23 (except, of course, CR). This allows the use of spaces (ASCII $20) within the command message for better readability if desired.

The length of a command message is limited to 25 printable characters. If a properly addressed module receives a command message of more than 25 characters the module will abort the whole command sequence and no response will result.

If a properly addressed module receives a second command prompt before it receives a CR, the command will be aborted and no response will result.

Response Structure

Response messages from the D1700 module begin with either an asterisk '*' (ASCII $2A) or a question mark '?' (ASCII $3F) prompt. The '*' prompt indicates acknowledgment of a valid command. The '?' prompt precedes an error message. All response messages are terminated with a CR. Many commands simply return a single '*' character to acknowledge that the command has been executed by the module. Other commands send data information following the '*' prompt. The response format of all commands may be found in the detailed command description.

The maximum response message length is 25 characters.

A command/response sequence is not complete until a valid response is received. The host may not initiate a new command until the response from a previous command is complete. Failure to observe this rule will result in communications collisions. A valid response can be in one of three forms:

1) a normal response indicated by a ' * ' prompt
2) an error message indicated by a ' ? ' prompt
3) a communications time-out error

When a module receives a valid command, it must interpret the command, perform the desired function, and the communicate the response back to the host. Each command has an associated delay time in which the module is busy calculating the response. If the host does not receive a response in an appropriate amount of time specified in Table 3.1, a communications time-out error has occurred. After the communications time-out it is assumed that no response data is forthcoming. This error usually results when an improper command prompt or address is transmitted.

Long Form Responses

When the pound sign ' # ' command prompt is used, the module will respond with a 'long form' response. This type of response will echo the command message, supply the necessary response data, and will add a two-

character checksum to the end of the message. Long form responses are used in cases where the host wishes to verify the command received by the module. The checksum is included to verify the integrity of the response data. The ' # ' command prompt may be used with any command. For example:

Command:	**$1DI**	**(short form)**
Response:	***8000**	
Command:	**#1DI**	**(long form)**
Response:	***1DI8000B0**	**(B0=checksum)**

For the D1700 commands that affect the digital outputs, the '#' form of a command starts a handshaking sequence that must be terminated with an Acknowledge (ACK) command. (See ACK command)

Checksum

The checksum is a two character hexadecimal value appended to the end of a message. It verifies that the message received is exactly the same as the message sent. The checksum ensures the integrity of the information communicated.

Command Checksum

A two-character checksum may be appended to any command to the D1700 module as a user option. When a module interprets a command, it looks for the two extra characters and assumes that it is a checksum. If the checksum is not present, the module will perform the command normally. If the two extra characters are present, the module will calculate the checksum for the message. If the calculated checksum does not agree with the transmitted checksum, the module will respond with a 'BAD CHECKSUM' error message and the command will be aborted. If the checksums agree, the command will be executed. If the module receives a single extra character, it will respond with a 'SYNTAX ERROR' and the command will be aborted. For example:

Command:	**$1DI**	**(no checksum)**
Response:	***8000**	
Command:	**$1DIE2**	**(with checksum)**
Response:	***8000**	
Command:	**$1DIAB**	**(incorrect checksum)**
Response:	**?1 BAD CHECKSUM**	

Command: $1DIE (one extra character)
Response: ?1 SYNTAX ERROR

Response Checksums
If the long form ' # ' version of a command is transmitted to a module, a checksum will be appended to the end of the response. For example:

Command: $1DI (short form)
Response: *8000

Command: #1DI (long form)
Response: *1DI8000B0 (B0=checksum)

Checksum Calculation
The checksum is calculated by summing the hexadecimal values of all the ASCII characters in the message. The lowest order two hex digits of the sum are used as the checksum. These two digits are then converted to their ASCII character equivalents and appended to the message. This ensures that the checksum is in the form of printable characters.

Example: Append a checksum to the command #1DOFF00

Characters:	#	1	D	O	F	F	0	0
ASCII hex values:	23	31	44	4F	46	46	30	30

Sum (hex addition) 23 + 31 + 44 + 4F + 46 + 46 + 30 + 30 = 1D3

The checksum is D3 (hex). Append the characters D and 3 to the end of the message: #1DOFF00D3

Example: Verify the checksum of a module response *1DI8000B0

The checksum is the two characters preceding the CR: B0

Add the remaining character values:

*	1	D	I	8	0	0	0	
2A+	31+	44+	49+	38+	30+	30+	30=	1B0

The two lowest-order hex digits of the sum are B0 which agrees with the transmitted checksum.

Note that the transmitted checksum is the character string equivalent to the calculated hex integer. The variables must be converted to like types in the host software to determine equivalency.

If checksums do not agree, a communications error has occurred.

If a module is setup to provide linefeeds, the linefeed characters are not included in the checksum calculation.

Parity bits are never included in the checksum calculation.

Table 4.1. 1700 Command Set

Command	Definition	Typical Command Message	Typical Response Message
ACK	Acknowledge	$1ACK	*
CB	Clear Bit	$1CB0C	*
CP	Clear Position	$1CP12	*
DI	Digital Input	$1DI	*8007
DO	Digital Output	$1DO1234	*
RA	Read Assignments	$1RA	*0F0F
RAB	Read Assignment Bit	$1RAB01	*O
RAP	Read Assignment Pos.	$1RAP01	*I
RB	Read Bit	$1RB0F	*1
RD	Read Data	$1RD	*+99999.99
RE	Read Event Counter	$1RE	*0001234
RID	Read Identification	$1RID	*BOILER
RIV	Read Initial Value	$1RIV	*0F0F
RP	Read Position	$1RP15	*0
RS	Read Setup	$1RS	*31070102
RSU	Read Setup	$1RSU	*31070102
RWT	Read Watchdog Timer	$1RWT	*+00010.00
SB	Set Bit	$1SB0C	*
SP	Set Position	$1SP12	*
WE	Write Enable	$1WE	*
The following 1700 commands are Write Protected			
AIB	Assign Input Bit	$1AIB0F	*
AIO	Assign Input/Output	$1AIO0F0F	*
AIP	Assign Input Position	$1AIP15	*
AOB	Assign Output Bit	$1AOB0F	*
AOP	Assign Output Pos.	$1AOP15	*
CE	Clear Event Counter	$1CE	*

EC	Event Read & Clear	$1EC	*0001234
ID	Identification	$1IDBOILER	*
IV	Initial Value	$1IV0F0F	*
RR	Remote Reset	$1RR	*
SU	Setup	$1SU31070102	*
WT	Watchdog Timer	$1WT+00010.00	*

The following 1700 commands are used with the special Continuous Input/Output Modes:

CIA	Continuous Input Address	$1CIA31	*
CMC	Continuous Mode-Change	$1CMC	*
CMD	Continuous Mode Disable	$1CMD	*
CME	Continuous Mode-Edge	$1CME	*
CMI	Continuous Mode-Input	$1CMI	*
CMT	Continuous Mode-Timer	$1CMT	*
CT	Continuous Timer	$1CT+00010.00	*
RCM	Read Continuous Mode	$1RCM	*D
RCT	Read Continuous Timer	$1RCT	*+00010.00
RIA	Read Input Address	$1RIA	*31

1700 Command Set

ACKnowledge (ACK)

The ACKnowledge command is a hand-shaking command that may be used with any command that will affect the the digital outputs such as the Digital Output (DO) command. It is used to confirm the data sent to a module and adds another level of data security to guard against transmission errors when performing output functions.

Command: $1ACK
Response: *

Command: #1ACK
Response: *1ACK2A

The ACK command is used in conjunction with the '#' form of an output command. For example:

Command: #1DOFFFF
Response: *1DOFFFF06

Note that the command is echoed back with a checksum (06) which is the case any time the '#' prompt is used. However, in the case of the DO command, the output data has not been changed at this point. The command data is echoed back so that the host may verify that the correct

message has been received by the module. If the command data is confirmed to be correct, the host may then activate the command by issuing an ACK command:

Command: **$1ACK**
Response: ***

Only at this point will the outputs be affected by the DO command.

If the host detects an error in the response data, it may recover by simply repeating the original command. For example:

Command: **#1DOFFFF**
Response: ***1DOFFFE05**

In ths case the response data does not match the original command, indicating that the module may have received the command incorrectly due to noise on the transmission line. However, the erroneous data does not reach the output since the module must receive an ACK to complete the command. To correct the error, the host may re-issue the original command:

Command: **#1DOFFFF**
Response: ***1DOFFFF06**

This time the response data is correct, and the DO command may be completed by sending the acknowledgement:

Command: **$1ACK**
Response: ***

Commands that require ACK handshaking are: AIB, AIO, AIP, AOB, AOP, CB, CP, DO, SB, and SP.

An ACK command used without an associated output command will generate a COMMAND ERROR.

Assign Input Bit (AIB)
Assign Input Position (AIP)
Assign Output Bit (AOB)
Assign Output Position (AOP)
The Assign Input and Assign Output commands are used to specify the data direction of an individual I/O line. The Assign Input commands configure an

individual bit to be used as an input to read external signals. The Assign Output commands configure data bits to be outputs to control external equipment.

This command configures Bit 05 to be an output:

Command: **$1AOB05**
Response: *****

This command configures Bit 0C to be an input:

Command: **$1AIB0C**
Response: *****

When used with the '#' prompt, the AI and AO commands require an ACK command from the host to complete the bit assignment:

Command: **#1AIB0C**
Response: ***1AIB0C9A**

Command: **$1ACK**
Response: *****

(See Acknowledge (ACK) command for more detail)

The Assign Input Position (AIP) and the Assign Output Position commands operate in the same manner as the AIB and AOB command except that the bit positions are specified in decimal (base 10) notation.

All of the Assign commands alter the contents of the EEPROM and therefore must be preceded by a Write Enable (WE) command.

The I/O direction assignments altered by the Assign commands are saved in EEPROM so that all pin directions are automatically configured when the device is powered up.

Assign Input/Output (AIO)

The Assign Input/Output (AIO) command is used to configure the data direction of all data lines at once. The direction data is represented in hexadecimal notation. A logical '1' indicates that an I/O line will be configured as an output. A logical '0' specifies a data input. The length of the hex

data argument will vary according to the number of I/O lines available and the word length that is setup in the device (see Setup section).

This command configures 16 bits of I/O lines to be outputs:

Command: **$1AIOFFFF**
Response: ***

This command configures 23 lines to be inputs and the LSB as an output:

Command: **$1AIO000001**
Response: ***

Up to 64 I/O lines may be configured at once,:

Command: **$1AIOF01234AA5500FF88**
Response: ***

The '#' form of the AIO command requires an 'ACK' to complete the direction assignments (see ACK command).

The AIO command stores the data direction assignments in EEPROM so that the I/O lines are configured automatically when the device is powered up.

The AIO must be preceded by a Write Enable (WE) command.

Clear Bit (CB)
Clear Position (CP)
Set Bit (SB)
Set Position (SP)

The Clear Bit command is used to turn off a single output bit. The CB command uses hexadecimal notation to address the desired bit:

Command: **$1CB0A**
Response: ***

In this case the hexadecimal bit number 0A is turned off. No other bits are affected.

If the CB command is used with the '#' prompt, an ACK command is required to complete the command. For example:

Command: **#1CB1F**
Response: ***1CB1F57**

In this case the module has echoed the command along with the response checksum '57'. At this point no output action has taken place. The purpose of the response message is to allow the host to examine the command received by the module. Thus, the host may verify that the command was received without error. Once the host is satisfied with the response data, it may activate the command by responding with an Acknowledge (ACK) command:

Command: **$1ACK**
Response: ***

At this point the output bit (B1F in this case) will be turned off.

The CB command will be executed only if the addressed bit has been previously assigned to be an output. An attempt to clear an input bit will result in an OUTPUT ERROR message and the command will be aborted. The bit direction may be assigned with the AI, AO, and AIO commands.

An attempt to clear a bit which does not exist will result in a VALUE ERROR, indicating an incorrect bit address.

To verify the results of a CB command the output bit value may be read back with the Read Bit (RB) command.

The Set Bit (SB) command operates exactly like the CB command except that the addressed bit is turned on.

The Clear Position (CP) and Set Position (SP) commands are similar to the CB and SB commands except the desired bit is specified with a decimal address. The following two commands perform exactly the same function:

Command: **$1SB0F**
Response: ***

Command: **$1SP15**
Response: ***

Clear Events (CE)
The Clear Events command clears the event counter to 00000000.

Command: **$1CE**
Response: ***

Note: When the events Counter reaches 9999999, it stops counting. A CE or EC command must be sent to resume counting.

See also the Events Read & Clear (EC) command.

Continuous Input Address (CIA)
The CIA command is used to specify the input address of a Continuous Input module. The address is specified as a two-character code indicating the ASCII equivalent of the address character:

Command: **$1CIA41**
Response: ***

In this example,the input address is specified as ASCII '41',which is the code for character 'A'. If the module is set to Continuous Input mode, it will respond to data strings containing address 'A'.

The Continuous Input Address should not be confused with the polled address as specified with the Setup (SU) Command. Refer to Chapter 6 for specific uses of the CIA command. An attempt to set the CIA address with the same value of the polled address will result in an ADDRESS ERROR response.

The Continuous Input Address is stored in non-volatile memory. The CIA command must be preceded with a WE command. The address value may be read back with the Read Input Address (RIA) Command.

Continuous Mode-Change (CMC)
Continuous Mode Disable (CMD)
Continuous Mode-Edge (CME)
Continuous Mode-Input (CMI)
Continuous Mode-Timer (CMT)
The Continuous Mode Commands are used to select and enable Continuous Modes as described in Chapter 6. Only one mode may be selected at any time.

CMC - This output mode produces a data stream each time the input data lines change.

CMD - Disable all Continuous Modes. This is the normal condition when D1700 modules are used in a polled system.

CME - Produce an output data stream when the edge-trigger input receives a positive transition.

CMI - Enable Continuous Input mode which will allow the module to accept data from a continuous Output module.

CMT - This command enables the Timer Continuous Output Mode. In this mode a module will output data periodically at a rate specified by the Continuous Timer (CT) command.

All five Continuous Mode commands require no argument and return no data:

Command: $1CMD
Response: *

The Continuous Mode selection is saved in non-volatile memory and is immediately active when power is applied to the module. With the exception of the CMD command, all of the Continuous Mode commands are write-protected and must be preceded with a WE command. Although the Disable command is stored in non-volatile memory it is not write-protected in order to disable a continuous - output module quickly.

The Continuous Mode setup may be read back with the Read Continuous Mode (RCM) command.

Digital Input (DI)

The Digital input command is used to read the logical state of all of the I/O lines in parallel. The DI command reads the state of both input and output lines.

Command: $1DI
Response: *1234

The number of data bits read back is a function of the unit's word length setup (see Setup chapter). It is possible to read up to 64 channels:

Command: **$1DI**
Response: ***00FF00EE00CC1234**

The rightmost hex digit always represents the least-significant bits, BØØ-BØ3.

If the '#' version of the command is used, do not confuse the checksum with the digital data.

Digital Output (DO)
The Digital Output command is used to specify the output data to all outputs at once:

Command: **$1DO00FF**
Response: *****

In this example, 16 bits of output data are specified in parallel. The 'FF' data commands the least significant eight bits (B00 to B07) to turn on. The '00' data turns off the next eight bits (B08 to B0F). This command is appropriate for devices setup for two words of data.

The hex data length specified in the DO command must match the word length setup in the D1700 or else the device will send a SYNTAX ERROR. The following command example may be used with a device set up for eight words:

Command: **$1DO1234567890ABCDEF**
Response: *****

See the Setup chapter for word length description.

If the DO command is used with the '#' command prompt, an ACK command is required to complete the output function (see ACK command).

I/O lines assigned to be inputs will ignore data sent by the DO command. No error message will be generated by outputting data to input channels using the DO command.

Events Read & Clear (EC)
The EC command is used to read the value of the Events Counter and automatically clears the count to zero:

Command: **$1EC**
Response: ***0000123**

The EC command eliminates a problem that may occur with a Read Events (RE) and Clear Events (CE) command sequence. Any counts that may occur between the RE-CE sequence will be lost. The EC command guarantees that the Event Counter is read and cleared without missing any counts.

Identification (ID)
The IDentification (ID) command allows the user to write a message into the nonvolatile memory which may be read back at a later time with the Read IDentification (RID) command. It serves only as a convenience to the user and has no other affect on module operation. Any message up to 16 characters long may be stored in memory. Useful information such as the module location, calibration data, or model number may be stored for later retrieval.

Message examples:

Command: **$1IDBOILER ROOM** (module location)
Response: ***

Command: **#1IDBOILER ROOM** (module location)
Response: ***1IDBOILER ROOM02**

Command: **$1ID 12/3/88** (calibration date)
Response: ***

Command: **$1ID 3125** (model number)
Response: *

The ID command is write-protected.
Since the ID command has a variable length syntax, command checksums cannot be appended to the message.

Initial Value (IV)
The Initial Value command allows the user to preset the startup condition of the digital outputs. When the D1700 unit is powered up, it reads data from the non-volatile memory to set up the initial output conditions. First it reads

the I/O direction data previously specified with the assignment commands. Then it reads out the Initial Value and performs an internal Digital Output command. Therefore the digital outputs are set to a known value upon startup.

The Initial Value is specified with hex data:

Command: **$1IV00FF**
Response: ***

Read Assignment (RA)
The Read Assignment command is used to read back the data direction configuration of all the I/O lines. The assignments are represented in hexadecimal notation, with a '1' signifying an output assignment and a '0' indicating an input assignment. The length of the hex data string will vary according to the number of I/O lines available and the number of words setup in the device (see Setup section). The LSB is always to the right:

Command: **$1RA**
Response: ***00FF**

This response indicates that the most significant eight I/O lines are configured as inputs and the least significant eight lines are configured as outputs.

For a device that is setup with a word length of '8', the RA command will read back the data direction of 64 I/O lines:

Command: **$1RA**
Response: ***00FF00FF88110044**

Read Assignment Bit (RAB)
Read Assignment Position (RAP)
The RAB and RAP commands are used to read back the input or output assignment of a single I/O line.. These commands use Bit or Position bit addressing to identify the desired bit. The D1700 will return an 'I' character if the bit is assigned to be an input , or an 'O' character if the bit is assigned as an output:

Command: **$1RAB0E**
Response: ***O**

Command: **$1RAP15**
Response: ***I**

Read Bit (RB)
Read Position (RP)

The Read Bit command is used to read the logical state of any individual I/O line, input or output. The desired bit is specified with the Bit notation:

Command: **$1RB0F**
Response: ***1**

The response data is a '1' or '0' character indicating the state of the I/O line in positive logic.

Attempting to read a non-existing I/O line will result in a VALUE ERROR.

The Read Position command performs the same function except the data bit is addressed in Position (decimal) notation:

Command: **$1RP15**
Response: ***1**

Note that the last two command examples perform the same function.

Read Continuous Mode (RCM)

The RCM command is used to read back the Continuous Input/Output Mode:

Command: **$1RCM**
Response: ***D**

The response is a single character indicating the Continuous Mode:

C- Continuous Output Change Mode
D- Continuous Output Disabled
E- Continuous Output Edge Trigger
I- Continuous Input
T- Continuous Output Timer

Read Continuous Timer (RCT)
The RCT command is used to read back the time value set by the Continuous Timer (CT) command:

Command: **$1RCT**
Response: ***+00005.00**

The Continuous Timer data is scaled in units of seconds.

The Continuous Timer function is detailed in Chapter 6.

Read Input Address (RIA)
The RIA command reads back the Continuous Input Address stored in non-volatile memory. This command in useful only for modules that are to be used in Continuous Input Mode (Chapter 6).

Command: **$1RIA**
Response: ***41**

The response to the RIA command is the ASCII code of the Continuous Input Address character. In this example, '41' is the ASCII code for character 'A'.

Read Data (RD)
The Read Data (RD) command is used to read analog data from the devices. Since the Digital I/O products do not acquire analog data, this command will always result in a fixed response:

Command: **$1RD**
Response: ***+99999.99**

Command: **#1RD**
Response: ***1RD+99999.99D9**

The RD command is included in the 1700 series to be compatible with other our products. In many systems that include analog input modules, the host will acquire data with a software loop containing the RD command. The RD command is included in digital I/O products so they may be included in the scanning loop. A proper response from an RD command is a good indication that the digital I/O device is powered up and running. It also serves to reset the Watchdog Timer (see the WT command).

Since the RD command is the most frequently used command in a system, a special truncated form of the command is available to speed up scanning rates. If a module is addressed without a command, the RD command is assumed by default:

Command: **$1**
Response: ***+99999.99**

Read Event Counter (RE)
The RE command reads the number of events that have been accumulated in the Events Counter. The output is a seven-digit decimal number:

Command: **$1RE**
Response: ***0000107**

The maximum accumulated count is 9999999. When this count is reached, the Events counter stops counting. The counter may be cleared at any time with the Events Read & Clear command (EC) or the Clear Events command (CE).

The Event Count is cleared to zero upon power-up.

The Remote Reset (RR) does not affect the Event Count.

When reading the Event Counter with a checksum, be sure not to confuse the checksum with the data.

Read Identification
The Read IDentification (RID) command reads out the user data stored by the IDentification (ID) command. The ID and RID commands are included as a convenience to the user to store information in the D1700's nonvolatile memory.

Command: **$1RID**
Response: ***BOILER ROOM (example)**

Command: **#1RID**
Response: ***1RIDBOILER ROOM54 (example)**

In this case the RID command has read back the message "BOILER ROOM"

previously stored by the ID command. See ID command.

Read Initial Value (RIV)

The Read Initial Value command is used to read back the Initial Value stored in the EEPROM. The Initial Value is the output data used to initialize the D1700 upon power-up. The Initial Value is set with the Initial Value (IV) command.

Command: **$1RIV**
Response: ***0F0F**

The length of the hex data returned is dependant on the specific D1700 model and the number of words in the setup (see Setup chapter)

Read Input Bit (RIB)
Read Input Position (RIP)

The Read Input commands are used to read the logical state of individual I/O lines. The desired line is specified with either Bit or Position addressing:

Command: **$1RIB0F**
Response: ***1**

The module will respond with a '1' or a '0' indicating the state of the specified I/O line in positive logic.

Command: **$1RIP15**
Response: ***1**

Note that in the two command examples the same I/O line is addressed.

The RIB and RIP will read the state of any line whether it is configured as an input or an output. Therefore it is useful in monitoring the true state of output data lines.

Read Setup

The Read SetUp (RSU) command reads back the setup information loaded into the module's nonvolatile memory with the SetUp (SU) command. The response to the RSU command is four bytes of information formatted as

eight hex characters.

The response contains the module's channel address, baud rate and other parameters. Refer to the setup command (SU), and Chapter 5 for a list of parameters in the setup information.

When reading the setup with a checksum, be sure not to confuse the checksum with the setup information.

Command: **$1RSU**
Response: ***31070102**

Command: **#1RSU**
Response: ***1RSU31070102E3**

The Read Setup (RS) command performs the same function, and is included to be compatible with the D1000/2000 series.

Command: **$1RS**
Response: ***31070102**

Command: **#1RS**
Response: ***1RS310701028E**

Remote Reset

The Remote Reset (RR) command allows the host to perform a program reset on the module's microcomputer. This may be necessary if the module's internal program is disrupted by static or other electrical disturbances.

Command: **$1RR**
Response: ***

Command: **#1RR**
Response: ***1RRFF**

The RR command is required to modify the baud rate of a module (see Setup section).

The RR command will not affect the output data or the Event Counter.

The RR command is write-protected.

Read Watchdog Timer
The Read Watchdog Timer (RWT) command reads the time interval necessary to activate the watchdog timer. The data is scaled in minutes.

Command: **$1RWT**
Response: ***+00010.00 (10 minutes)**

Command: **#1RWT**
Response: ***1RWT+00010.0002 (10 minutes)**

In each of the two example commands, the response data indicates that the watchdog timer period is 10 minutes. The watchdog timer value may be set with the Watchdog Timer (WT) command.

Setup Command (SU)
Each module contains an EEPROM (Electrically Erasable Programmable Read Only Memory) which is used to store module setup information such as address, baud rate, parity, etc. The EEPROM is a special type of memory that will retain information even if power is removed from the module. The EEPROM is used to replace the usual array of DIP switches normally used to configure electronic equipment.

The SetUp command is used to modify the user-specified parameters contained in the EEPROM to tailor the module to your application. Since the SetUp command is so important to the proper operation of a module, a whole section of this manual has been devoted to its description. See Chapter 5.

The SU command requires an argument of eight hexadecimal digits to describe four bytes of setup information:

Command: **$1SU31070102**
Response: *****

Command: **#1SU31070102**
Response: ***1SU3107010291**

Watchdog Timer (WT)

The Watchdog Timer (WT) command stores a data value in EEPROM specifying the time-out value of the watchdog timer. The time data is scaled in minutes:

Command: **$1WT+00010.00**
Response: *****

Command: **#1WT+00010.00**
Response: ***1WT+00010.00B0**

These two command examples set the watchdog time value to 10 minutes. In this example, if the module does not receive a valid command for a period of 10 minutes, the digital outputs will automatically be forced to the Initial Value. The purpose of the Watchdog Timer is to force the digital outputs to a known 'safe' value in the event of a host or communications link failure.

The Initial Value is set with the IV command.

The watchdog timer may be disabled by setting the timer value to +99999.99.

WT command data less than .16 minutes will result in a VALUE ERROR.

The WT command is write protected.

Write Enable

The Write Enable (WE) command must precede commands that are write-protected. This is to guard against accidentally writing over valuable data in the EEPROM. To change any write protected parameter, the WE command must precede the write-protected command. The response to the WE command is an asterisk indicating that the module is ready to accept a write-protected command. After the write-protected command is successfully completed, the module becomes automatically write disabled. Each write-protected command must be preceded individually with a WE command. For example:

Command: **$1WE**
Response: *****

Command: **#1WE**
Response: ***1WEF7**

If a module is write enabled and the execution of a command results in an error message other than WRITE PROTECTED, the module will remain write enabled until a command is successfully completed resulting in an ' * ' prompt. This allows the user to correct the command error without having to execute another WE command.

ERROR MESSAGES

All modules feature extensive error checking on input commands to avoid erroneous operation. Any errors detected will result in an error message and the command will be aborted.

All error messages begin with **"?"**, followed by the channel address, a space and error description. The error messages have the same format for either the ' $ ' or ' # ' prompts. For example:

?1 SYNTAX ERROR

There are eight error messages, and each error message begins with a different character. Host computer software can identify an error by the first character; it is not necessary to read the whole string.

ADDRESS ERROR

There are four ASCII values that are illegal for use as a module address: NULL ($00), CR ($0D), $ ($24), and # ($23). The ADDRESS ERROR will occur when an attempt is made to load an illegal address into a module with the SetUp (SU) command. An attempt to load an address greater than $7F will also produce an error.

An attempt to use the Continuous Input Address (CIA) command to specify an illegal address or an address identical to the polling address will create an error.

BAD CHECKSUM

This error is caused by an incorrect checksum included in the command string. The module recognizes any two hex characters appended to a command string as a checksum. Usually a BAD CHECKSUM error is due to noise or interference on the communications line. Often, repeating the command solves the problem. If the error persists, either the checksum is calculated incorrectly or there is a problem with the communications

channel. More reliable transmissions might be obtained by using a lower baud rate.

COMMAND ERROR
This error occurs when a command is not recognized by the module. Often this error results when the command is sent with lower-case letters. All valid commands are upper-case.

OUTPUT ERROR
An attempt to use a CB, CP, SB, or SP command to set or clear a digital I/O line that has been assigned as an input will generate an OUTPUT ERROR.
The Digital Output (DO) command will not generate an OUTPUT ERROR.

PARITY ERROR
A parity error can only occur if the module is setup with parity on (see Setup). Usually a parity error results from a bit error caused by interference on the communications line. Random parity errors are usually overcome by simply repeating the command. If too many errors occur, the communications channel may have to be improved or a slower baud rate may be used.

A consistent parity error will result if the host parity does not match the module parity. In this situation, the easiest solution may be to change the parity in the host to obtain communication. At this point the parity in the module may be changed to the desired value with the SetUp (SU) command.

The parity may be changed or turned off by using Default Mode.

SYNTAX ERROR
A SYNTAX ERROR will result if the structure of the command is not correct. This is caused by having too few or too many characters, signs or decimal points missing or in the wrong place. Table 4.1 lists the correct syntax for all the commands.

VALUE ERROR
This error results when an incorrect character is used as a numerical value. Data values can only contain decimal digits 0-9. Hex values can range from 0-F.

WRITE PROTECTED
All commands that write data into nonvolatile memory are write-protected to prevent accidental erasures. These commands must be preceded with a Write Enable (WE) command or else a WRITE PROTECTED error will result.

Chapter 5
Setup Information/SetUp Command

The modules feature a wide choice of user configurable options which gives them the flexibility to operate on virtually any computer or terminal based system. The user options include a choice of baud rate, parity, address, and many other parameters. The particular choice of options for a module is referred to as the setup information.

The setup information is loaded into the module using the SetUp (SU) command. The SU command stores 4 bytes (32 bits) of setup information into a nonvolatile memory contained in the module. Once the information is stored, the module can be powered down indefinitely (10 years minimum) without losing the setup data. The nonvolatile memory is implemented with EEPROM so there are no batteries to replace.

The EEPROM has many advantages over DIP switches or jumpers normally used for option selection. The module never has to be opened because all of the options are selected through the communications port. This allows the setup to be changed at any time even though the module may be located thousands of feet away from the host computer or terminal. The setup information stored in a module may be read back at any time using the Read Setup command (RSU).

The following options can be specified by the SetUp command:

Channel address (124 values)
Linefeeds
Parity (odd, even, none)
Baud rate (300 to 38,400)
Echo
Communication delay (0-6 characters)
Word length

Each of these options will be described in detail below. For a quick look-up chart on all options, refer to Tables 5.1-4.

Command Syntax
The general format for the SetUp (SU) command is:

$1SU[byte1][byte 2][byte 3][byte 4]

A typical SetUp command would look like: $1SU31070102

Notice that each byte is represented by its two-character ASCII equivalent. In this example, byte 1 is described by the ASCII characters '31' which is the equivalent of binary 0011 0001 (31 hex). The operand of a SU command must contain exactly 8 hex (0-F) characters. Any deviation from this format will result in a SYNTAX ERROR.

For the purposes of describing the SetUp command, 'bit 7' refers to the highest-order bit of a byte of data. 'Bit 0' refers to lowest-order bit

'bit number':	7	6	5	4	3	2	1	0	
binary data:	0	0	1	1	0	0	0	1	= $31 (hex)

The SU command is write protected to guard against erroneous changes in the setup data; therefore each SU command must be preceded by a Write Enable (WE) command. To abort an SU command in progress, simply send a non-hex character (an 'X' for example) to generate a SYNTAX ERROR, and try again.

Caution: Care must be exercised in using the SU command. Improper use may result in changing communications parameters (address, baud rate, parity) which will result in a loss of communications between the host and the module. In some cases the user may have to resort to using Default Mode to restore the proper setups. The recommended procedure is to first use the Read Setup (RS) command to to examine the existing setup data before proceeding with the SU command.

Byte 1

Byte 1 contains the module (channel) address. The address is stored as the ASCII code for the string character used to address the module. In our example command $1SU31070102 , the first byte '31' is the ASCII code for the character '1'. If our sample command is sent to a module, the EEPROM will be loaded with the address '1', which in this particular case remains unchanged. To change the module address to '2' , byte 1 of the SetUp command becomes '32', which is the ASCII code for the character '2'. Now the command will look like this: $1SU32070102. When this command is sent, the module address is changed from '1' to '2'.

The module will no longer respond to address '1'.

When using the SU command to change the address of a module, be sure to record the new address in a place that is easily retrievable. The only way

to communicate with a module with an unknown address is with the Default Mode.

The most significant bit of byte 1 (bit 7) must be set to '0'. In addition, there are four ASCII codes that are illegal for use as an address. These codes are $00, $0D, $24, $23 which are ASCII codes for the characters NUL, CR, $, and #. Using these codes for an address will cause an ADDRESS ERROR and the setup data will remain unchanged. This leaves a total of 124 possible addresses that can be loaded with the SU command. It is highly recommended that only ASCII codes for printable characters be used ($21 to $7E) which greatly simplifies system debugging with a dumb terminal. Refer to Appendix A for a list of ASCII codes. Table 5.1 lists the printable ASCII codes that may be used as addresses.

Table 5.1 Byte 1 ASCII Printable Characters.

HEX	ASCII	HEX	ASCII	HEX	ASCII	HEX	ASCII
21	!	3A	:	51	Q	68	h
22	"	3B	;	52	R	69	i
25	%	3C	<	53	S	6A	j
26	&	3D	=	54	T	6B	k
27	'	3E	>	55	U	6C	l
28	(	3F	?	56	V	6D	m
29	)	40	@	57	W	6E	n
2A	*	41	A	58	X	6F	o
2B	+	42	B	59	Y	70	p
2C	,	43	C	5A	Z	71	q
2D	-	44	D	5B	[	72	r
2E	.	45	E	5C	\	73	s
2F	/	46	F	5D	]	74	t
30	0	47	G	5E	^	75	u
31	1	48	H	5F	_	76	v
32	2	49	I	60	`	77	w
33	3	4A	J	61	a	78	x
34	4	4B	K	62	b	79	y
35	5	4C	L	63	c	7A	z
36	6	4D	M	64	d	7B	{
37	7	4E	N	65	e	7C	|
38	8	4F	O	66	f	7D	}
39	9	50	P	67	g	7E	~

Byte 2
Byte 2 is used to configure some of the characteristics of the communications channel; linefeeds, parity, and baud rate.

Linefeeds
The most significant bit of byte 2 (bit 7) controls linefeed generation by the module. This option can be useful when using the module with a dumb terminal. All responses from the modules are terminated with a carriage return (ASCII $0D). Most terminals will generate a automatic linefeed when a carriage return is detected. However, for terminals that do not have this capability, the modules can generate the linefeed if desired. By setting bit 7 to '1' the module will send a linefeed (ASCII $0A) before and after each response. If bit 7 is cleared (0), no linefeeds are transmitted.

When using the '#' command prompt, the linefeed characters are not included in the checksum calculation.

Parity
Bits 5 and 6 select the parity to be used by the module. Bit 5 turns the parity on and off. If bit 5 is '0', the parity of the command string is ignored and the parity bit of characters transmitted by the module is set to '0'.

If bit 5 is '1', the parity of command strings is checked and the parity of characters output by the module is calculated as specified by bit 6.

If bit 6 is '0', parity is even; if bit 6 is '1', parity is odd.

If a parity error is detected by the module, it will respond with a PARITY ERROR message. This is usually caused by noise on the communications line.

If parity setup values are changed with the SU command, the response to the SU command will be transmitted with the old parity setup. The new parity setup becomes effective immediately after the response message from the SU command.

Baud Rate
Bits 0-2 specify the communications baud rate. The baud rate can be selected from eight values between 300 and 38400 baud. Refer to Table 5.2 for the desired code.

The baud rate selection is the only setup data that is not implemented directly after an SU command. In order for the baud rate to be actually

changed, a module reset must occur. A reset is performed by sending a Remote Reset (RR) command or powering down. This extra level of write protection is necessary to ensure that communications to the module is not accidently lost. This is very important when changing the baud rate of an RS-232C string. For more information on changing baud rate, refer to Chapter 3.

Let's run through an example of changing the baud rate. Assume our sample module contains the setup data value of '31070102'. Byte 2 is '07'. By referring to the SU command chart we can determine that the module is set for no linefeeds, no parity, and baud rate 300. If we perform the Read Setup command with this module we would get:

Command: **$1RS**
Response: ***31070102**

Let's say we wish to change the baud rate to 9600 baud. The code for 9600 baud is '010' (from Table 5.2). This would change byte 2 to '02'. To perform the SU command we must first send a Write Enable command because SU is write protected:

Command: **$1WE**
Response: ***
Command: **$1SU31020180**
Response: ***

This sequence of messages is done in 300 baud because that was the original baud rate of the module. The module remains in 300 baud after this sequence. We can use the Read Setup (RS) command to check the setup data:

Command: **$1RS**
Response: ***31020102**

Notice that although the module is communicating in 300 baud, the setup data indicates a baud rate of 9600 (byte 2 = '02'). To actually change the baud rate to 9600, send a Remote Reset (RR) command (RR is write protected):

Command: **$1WE**
Response: ***
Command: **$1RR**
Response: ***

Up to this point all communications have been sent at 300 baud. The module will not respond to any further communications at 300 baud because it is now running at 9600 baud. At this point the host computer or terminal must be set to 9600 baud to continue operation.

If the module does not respond to the new baud rate, most likely the setup data is incorrect. Try various baud rates from the host until the module responds. The last resort is to set the module to Default Mode where the baud rate is always 300.

Setting a string of RS-232C modules to a new baud rate requires special consideration. Refer to Chapter 3 for instructions.

Bits 3 and 4
These two bits of byte 2 are not used and should be set to '0'.

Table 5.2 Byte 2: Linefeed, Parity and Baud Rate.

BYTE 2 FUNCTION	DATA BIT 7	6	5	4	3	2	1	0
LINEFEED	1							
NO LINEFEED	0							
NO PARITY		0	0					
NO PARITY		1	0					
EVEN PARITY		0	1					
ODD PARITY		1	1					
NOT USED				X	X			
38400 BAUD						0	0	0
19200 BAUD						0	0	1
9600 BAUD						0	1	0
4800 BAUD						0	1	1
2400 BAUD						1	0	0
1200 BAUD						1	0	1
600 BAUD						1	1	0
300 BAUD						1	1	1

Byte 3
This byte contains the setup information for additional communications options. The default value for this byte is '01'.

Echo
When bit 2 is set to '1', the module will retransmit any characters it has received on the communications line. This option is necessary to 'daisy-chain' multiple RS-232C modules. Echo is optional for systems with a single RS-232C module. Bit 2 must be cleared to '0' on RS-485 models. See Chapter 3 for a more complete description.

Delay
Bits 0 and 1 specify a minimum turn-around delay between a command and the module response. This delay time is useful on host systems that are not fast enough to capture data from quick-responding commands such as RD. This is particularly true for systems that use software UART's. The specified delay is added to the typical command delays listed in the Software Considerations section of Chapter 3. Each unit of delay specified by bits 0 and 1 is equal to the amount of time required to transmit one character with the baud rate specified in byte 2. For example, one unit of delay at 300 baud is 33.3 mS; for 38.4 kilobaud the delay is 0.26 mS. The number of delay units is selectable from 0 to 6 as shown in Table 5.3.

In some systems, such as IBM BASIC, a carriage return (CR) is always followed by a linefeed (LF). The modules will respond immediately after a command terminated by a CR and will ignore the linefeed. To avoid a communications collision between the linefeed and the module response, the module should be setup to delay by 2 units.

Table 5.3 Byte 3 Options.

BYTE 3 FUNCTION	DATA BIT 7	6	5	4	3	2	1	0
NOT USED	X	X	X	X	X			
NO ECHO						0		
ECHO						1		
NO DELAYS							0	0
2 BYTE TIME DELAYS							0	1
4 BYTE TIME DELAYS							1	0
6 BYTE TIME DELAYS							1	1

Byte 4

Event Counter Filter
The D1711/1712 contains a programmable digital filter to control the bandwidth of the Event Counter. The filter is particularly useful when the Event Counter is used to count transitions from switches or other electro-mechanical contacts. The filter will debounce noisy signals to provide error-free transition counting.

The filter constant is controlled by bits 4 and 5 of byte 4. The selections are shown in Table 5.4. If no filter is selected, the Event Counter bandwidth is 20kHz. This setting is ideal for electronic signals with clean transitions. To debounce noisy signals, filter constants of 5, 20, and 50ms are available. The operation of the digital filter is described in Chapter 2.

Word Length
The D1700 command set is used by many digital I/O devices, ranging from 1 to 64 I/O lines. Many of the commands such as the DO, DI, and RA commands operate on all of the data lines in parallel. The word length setup is used to adjust the amount of hex bit data transmitted to and from the D1700 device. One word of data is defined to be eight bits, represented by two hexadecimal digits. The number of words required may be adjusted to a value most appropriate for a specific device.

The word length can vary from 1 to 8 words, specified by bits 0-2. A word length of 0 is not allowed.

As an example, a D1712 has 15 I/O lines, and a typical setup for this device would be: 31070102

The '02' indicates that the device is setup for two words, or 16 bits of data. A typical DO command to this unit would be:

Command: **$1DO1234**
Response: *****

Since the D1712 is setup for two words, it will accept the four digits of hex data. If the data length is incorrect, an error will be generated:

Command: **$1DO12345**
Response: **?1 SYNTAX ERROR**

The word length setup also affects the parallel readback commands such as:

Command: **$1DI**
Response: ***ABCD**

Notice that with a word setup of '2', the DI command returns 2 words of data. The same effect occurs with the RA and RIV commands.

It is possible to setup a module with a word length that does not correspond with the physical I/O data width. For example, the D1712 may be setup with word length = '1'. Setup data = 31070101

With this setup, the DI command returns eight bits of data:

Command: **$1DI**
Response: ***CD**

The correct DO argument is now two hex digits:

Command: **$1DO34**
Response: ***

A word length of '1' may be appropriate for the D1712 if only eight bits of the device are used or if maximum communications speed is desired.

Regardless of the word setup, the rightmost hex digit of the bit data is always the least significant I/O data. The most significant data is appended or truncated as necessary corresponding to the word length setup.

It is also possible for the word length to be greater than the physical data width of the device. The D1712 may be setup with a word length of '3': 31070103

In this case, all parallel data values must be 24 bits or six hex digits wide:

Command: **$1DO123456**
Response: ***
Command: **$1DI**
Response: ***003456**

The D1712 contains 15 I/O lines. For the DO command, the most significant nine of the 24 bits will be ignored. The DI command will return with '1' data for the nine most significant bits.

The deliberate use of dummy data may seem wasteful, but it can be useful for streamlining host software. For example, in a system with a mix of 24 and 15 bit devices, the host software may be simplified by standardizing to word length '3' for all devices.

The word length setup has no affect on commands using single-bit 'Bit' or 'Position' addressing.

BYTE 4

FUNCTION	**DATA BIT**							
	7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
NOT USED	**X**	**X**						
NO FILTER			**0**	**0**				
5ms			**0**	**1**				
20ms			**1**	**0**				
50ms			**1**	**1**				
1 WORD					**0**	**0**	**0**	**1**
2 WORDS					**0**	**0**	**1**	**0**
3 WORDS					**0**	**0**	**1**	**1**
4 WORDS					**0**	**1**	**0**	**0**
5 WORDS					**0**	**1**	**0**	**1**
6 WORDS					**0**	**1**	**1**	**0**
7 WORDS					**0**	**1**	**1**	**1**
8 WORDS					**1**	**0**	**0**	**0**

Setup Hints

Until you become completely familiar with the SetUp command, the best method of changing setups is to change one parameter at a time and to verify that the change has been made correctly. Attempting to modify all the setups at once can often lead to confusion. If you reach a state of total confusion, the best recourse is to reload the factory setup as shown in Table 5.5 and try again, changing one parameter at a time. Use the Read Setup (RS) command to examine the setup information currently in the module as a basis for creating a new setup. For example:

Assume you have a D1711 unit and you wish to set the unit to echo so that it may be used in a daisy-chain (See Communications). Read out the current setup with the Read Setup command:

Command: **$1RS**
Response: ***31070102**

By referring to Table 5.3, we find that the echo is controlled by bit 2 of byte 3. From the RS command we see that byte 3 is currently set to 01. This is the hexadecimal representation of binary 0000 0001. To set echo, bit 2 must be set to '1'. This results in binary 0000 0101. The new hexadecimal value of byte 3 is 05. To perform the SU command, use the data read out with the RS command, changing only byte 3:

Command: $1WE (SU is write-protected)
Response: *
Command: $1SU31070502
Response: *

Verify that the module is echoing characters and the setup is correct.

By using the RS command and changing one setup parameter at a time, any problems associated with incorrect setups may be identified immediately. Once a satisfactory setup has been developed, record the setup value and use it to configure similar modules.

If you commit an error in using the SetUp command, it is possible to lose communications with the module. In this case, it may be necessary to use the Default Mode to re-establish communications.

Table 5.5 Factory Setups by Model.
(All modules from the factory are set for address '1', 300 baud, no parity)

Model	Setup Message
D1711	31070102
D1712	31070102
H1750	31070103
H1770	31070108

S1000 Software
Setting up your D1000 module may be greatly simplified by using the setup program provided in the S1000 software package. The S1000 software runs on IBM PC's or compatibles and is free of charge. The setup program provides a menu-driven operator interface which greatly simplifies the setup process and decreases the chances of setup errors.

Chapter 6
Continuous Input/Output

The D1711/1712 modules may be setup in special modes which allow them to communicate without being polled by a host computer. Collectively these modes are called Continuous Input/Output Modes. In many applications the burden on the host may be greatly simplified and in some cases the host may be eliminated altogether.

Continuous Output
A D1711/1712 in continuous mode will produce an output string in the same format as the response to a #1DI command:

***1DI8000B0**

Note that the output message contains the response prompt (*), the module address (1), the status of the digital I/O lines (8000) and a checksum (B0). In continuous mode, a D1711/1712 module produces a response to a #1DI command without actually receiving the command. The output data string may be triggered in one of three ways:

Timer Mode: In this mode, a software timer is activated in the module with the Continuous Timer (CT) command. The CT command specifies a time period that repeats indefinitely. After each timeout, the module will output the status data. The module will periodically output the digital input data until the continuous mode is disabled.

Edge-Trigger Mode: In this mode the D1711/1712 will output a data string when it receives a trigger signal on the B00/EV I/O Pin. The edge trigger mode will produce an output in response to an external event. It also provides a means of daisy-chaining several continuous output modules together.

Change Mode: In this mode the D1711/1712 continuously monitors the status of the I/O lines. If a change is detected in status, an output data message is initiated.

Continuous Input
A module setup for continuous input will respond to data produced by a continuous-output module. The data string from a continuous output module is interpreted as an output command by a continuous input module. This allows data to be read at one module and replicated at the outputs of

another module without a host computer.

Continuous Input/Output Commands

The D1711/1712 modules contain several commands to setup the continuous modes. They are listed here for quick reference. A more complete description of each command may be found in the Chapter 4.

$1CMD	Continuous Mode Disable
$1CMT	Enable Timer-Triggered Continuous Output Mode
$1CME	Enable Edge-Triggered Continuous Output Mode
$1CMC	Enable Change-Triggered Continuous Output Mode
$1CMI	Enable Continuous Input Mode
$1RCM	Read Continuous Mode Type.

Response is D, T, E, C or I for Disabled, Timer, Edge, Change or Input respectively.

$1CT	Specify Continuous Timer value in seconds.
$1RCT	Read Continuous Timer
$1CIA	Specify Continuous Address
$1RIA	Read Continuous Input Address

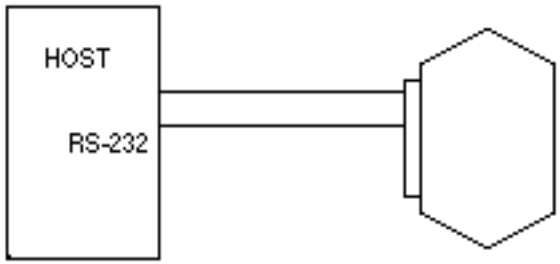
Continuous Output Trigger Signal

In order to facilitate daisy-chaining of continuous output modules, each module will produce an output trigger signal each time it completes an output data string. The output trigger is a 5 millisecond low pulse which appears on the Default * pin. The Default * pin is normally an input pin used to place the module in a known communications setup. This is also true when a module is set for Continuous Mode. However, when a module produces a continuous output, the Default * pin momentarily becomes an output and produces a low-going trigger pulse. This trigger pulse may be used to trigger another module set in Edge trigger mode. In this manner, many modules may be daisy-chained together in continuous mode.

Applications

There is a wide variety of system configurations which may be implemented with the continuous mode modules. It would be impossible to detail every possible connection. However, a variety of examples will be given to demonstrate typical usage.

A) Timer Mode (Figure 1)



In this configuration, a D1712 module is set to continuously output data to a host computer or display device. It is not necessary for the host to poll the D1712 to obtain data. The host computer must have an interrupt-driven serial input for proper operation.

For this example, we will setup the D1712 to output data every 10 seconds. (WE commands are not shown but necessary for write-protected commands).

1) Setup the D1712 as usual with the setup (SU) command for correct communications to the host.

Command: **$1SU31070102**
Response: ***

2) Assign the I/O lines of the D1712 to be inputs:

Command: **$1AIO0000**
Response: ***

3) Set the Continuous Timer (CT) for a 10 second interval:

Command: **$1CT+00010.00**
Response: ***

This tells the D1712 to output data continuously in 10 second intervals.

4) Activate the continuous output with the Continuous Mode Timer (CMT) command. This will activate the continuous output data.

5) Every 10 seconds, the D1712 will read the status of its I/O lines and output the status of those lines as if it was responding to a #1D1 Command:

Response: *1DI1234B2

6) The Continuous Mode may be disabled by the host by sending a Continuous Mode Disable command:

Command: $1CMD
Response: *

The CMD is not write-protected and a Write Enable (WE) command is not required.

To avoid communications collisions, the host should wait for a continuous output response, and then immediately issue the CMD command. In our current example, the host has 10 seconds to issue the CMD command, so the likelihood of a collision is remote. It is possible for the host to disable the continuous mode even if the Continuous Timer is set for 0 seconds. The host must issue the CMD command immediately after the carriage return from the D1712 is received. When the D1712 reads a '$'or '#' character on the communications line, it will temporarily halt the continuous mode output and look for an address character. If the D1712 detects its own address, it will read and process the rest of the command. Otherwise it will resume the continuous mode output.

B) Timer Mode With Outputs

This configuration is shown in Figure 1. However, this time the D1712 is setup with digital outputs. For example, the high-order 7 bits could be configured as outputs:

Command: $1AIOFF00
Response: *

Setup the Continuous Mode just like example A:

Command: $1CT +00010.00
Response: *

Command: **$1CMT**
Response: ***

The D1712 will continuously output the status of the I/O lines, including the outputs, every 10 seconds.

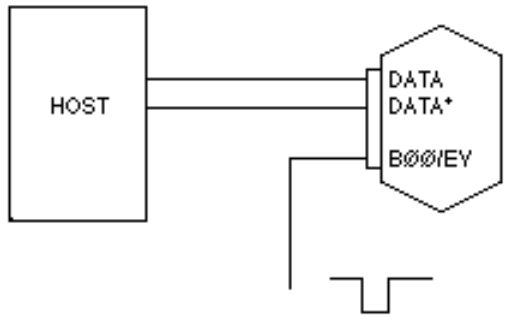
However, with this setup, the host may respond with DO, SB, CB, or other output commands to control the digital outputs in response to the input data or some other control decision.

It is not necessary to disable the continuous mode before issuing the output command. However, to avoid communications collisions the host command should be timed to avoid the continuous response from the D1712. The easiest way to do this is to wait for a continuous output string from the module and then immediately issue the output command.

Another method of performing output functions is to disable the Continuous Mode by issuing a Continuous Mode Disable (CMD) command. The D1712 now acts normally and any of the I/O commands may be performed. The Continuous Mode may be resumed with a Continuous Mode Timer (CMT) command.

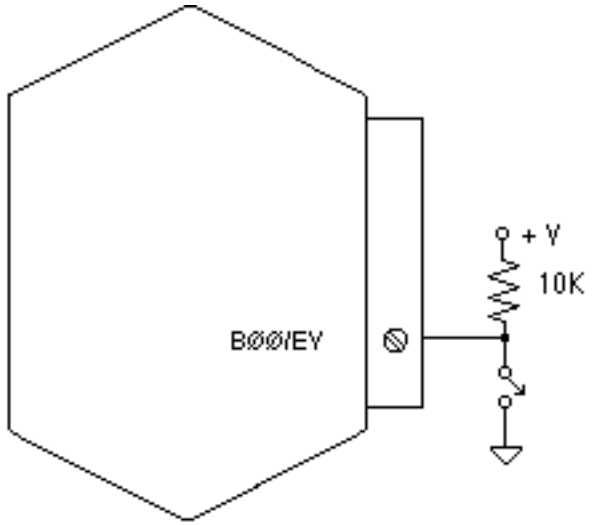
C) Edge-Trigger Mode With Host

The D1712 may be triggered by an external digital signal which will command the D1712 to read the status of the I/O lines and report the data (Figure 2).



The external trigger signal is connected to the B00/EV pin of the module. Since the B00 pin is used for the trigger, it is not available for general-

purpose I/O in this application. The trigger input is designed to accept a TTL-level signal, although it will withstand a 0-30V input without damage. The input may be triggered with a switch by adding a pull-up resistor. (Figure 3)



The module is triggered on a positive-going edge to the B00/EV pin.To setup the D1712 for edge-trigger mode:

1) Use the setup (SU) command to set the desired communications parameters. In this case the D1712 will be setup for address '2':

Command: **$1SU32070102**
Response: ***

2) The B00 I/O line must be assigned as an input to accept the trigger signal. Other I/O lines may be assigned as inputs or outputs depending on the application:

Command: **$2AIOFF00**
Response: ***

This command assigns B00 through B07 as inputs, and B08 through B0E as outputs.

3) The Continuous Timer (CT) command may be used to specify a delay time between the Trigger signal and the output data string. This feature is useful in some applications when multiple modules are tied together which will be illustrated in other examples. For this and most edge-trigger applications, set the Continuous timer to 0 seconds:

Command: **$2CT+00000.00**
Response: *****

4) Enable the edge trigger mode with the Continuous Mode Edge-trigger (CME) command.

5) When the D1712 senses a positive-going trigger on the B00/EV line, it will perform the equivalent of a #2DI command and output the data:

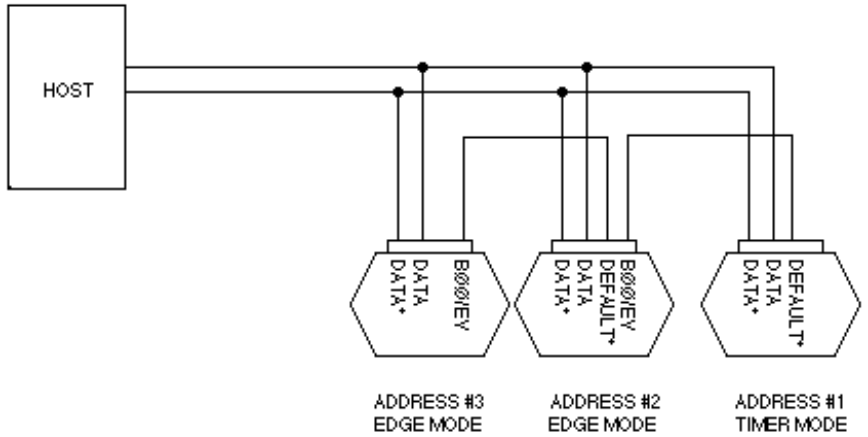
Response: ***2DI1235B3**

6) The host may terminate the edge-trigger mode with a Continuous Mode Disable (CMD) command. Precautions must be used to avoid communications collisions between the host command and responses from the module. The best method of disabling the continuous mode is to issue the 'long form' version of the CMD command:

Command: **#2CMD**
Response: ***2CMD30**

The command is issued by the host and then the host looks for the correct response string to be returned by the D1712. If the correct response string is detected, then the host knows that the continuous mode has been disabled. If the correct response string is not received, it may be assumed that the CMD command collided with response data from the module. The host simply repeats the CMD command until the correct response is obtained. Communications collisions are not harmful to RS485 hardware. However, the host serial input must be able to accept framing errors and 'noise' characters gracefully when collisions occur.

D) Continuous Output Daisy-Chain With Host (Figure 4)



This configuration uses one module (address 1) in Timer mode which produces a trigger signal on the Default * line to trigger another module (address 2) which is set for Edge trigger mode. The third module (address 3) is set-up for edge-trigger mode and receives its trigger signal from the Default * pin of module #2. Additional edge-triggered modules may be implemented by connecting the trigger output (Default *) of a module #3 to the trigger input (B00/EV) of the next module in the series, and this connection may be repeated for any additional modules. A typical application will have 1 timer module with any number (up to 123) of edge-triggered modules.

The net result of this connection is a periodic burst of data from all the modules without the need for polling by the host. The data stream from this system would typically look like this:

*1DI1234B2
*2DI0001AA
*3DIFFFF02

Each data string is terminated by a carriage return. Note that the module address is transmitted with the data to easily determine the origin of the data.

The easiest way of setting up a system like this is to install the modules and operate them as a polled system first. Once the wiring and the operation of all the modules is established, the string may be set for Continuous Mode. First, set up all the Edge Triggered modules as described in Example C. The last step is to setup the Timer module as described in Example A. The CT time specified in the Timer module must be long enough to allow all the modules to respond. If the CT time is too short, module #1 will start to output data before module #3 has finished, resulting in a communications collision.

In some cases, especially if a large number of modules are connected in a string, the amount of data transmitted may overload the serial port buffer of the host. In this case, the data may be slowed down by specifying a finite amount of time in the Continuous Timer (CT) of each edge-triggered module. A module in Edge trigger mode will delay the output data after it is triggered by the amount specified in CT.

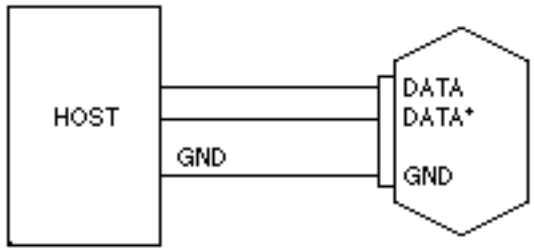
The host may disable the continuous output data by sending a Continuous Mode Disable command to the timer module:

Command: **#1CMD**
Response: ***1CMD2F**

The host should repeat the CMD until the proper response is obtained. After the timer mode is disabled, the string of modules may be polled by the host. To return to continuous output operation, enable the continuous mode of the Timer module:

Command: **$1CMT**
Response: *****

E) Change Mode (Figure 5)



A D1712 module set up for change mode will output a data string if one of its digital input lines has changed state. The module will output the data string reporting the new state of the inputs.

To setup the D1712 for change mode:

Assign the desired I/O lines to inputs:

Command: **$1AIO00FF**
Response: *****

Note that not all lines are required to be inputs. In this example, digital I/O lines B00-B07 are set to outputs and B08-B0E are set to inputs.

For this example, set the Continuous Timer to zero:

Command: **$1CT+00000.00**
Response: *****

Set the module to Continuous Mode:

Command: **$1CMC**
Response: *****

The D1712 will continually scan the Digital I/O lines to detect any changes of state. If a change is found, the new state of the I/O lines is reported to the host:

***1DI01FFD5**

After the response is transmitted, the D1712 will resume scanning the I/O lines.

The host may disable the continuous output mode by sending a CMD command:

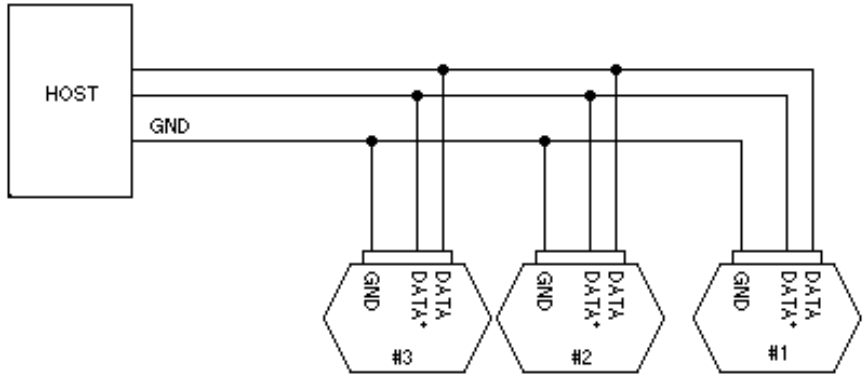
Command: **#1CMD**
Response: ***1CMD2F**

The module may now be interrogated with the normal command -response sequence. This method is useful when the host is required to produce an output response to a change in the input status. The host may control the digital output lines (in this case B00-B07) with normal I/O commands. After

the control function is completed, the D1712 may be returned to Continuous Mode with the CMC command.

The change mode is ideal in applications where the state of the digital inputs is expected to change infrequently. Inputs such as security switches, fire detectors, alarm switches, etc. are not expected to change but must be detected by the host computer. By using a D1712 in change mode, the host may be alerted to a change in input status on an interrupt basis thereby saving computer time scanning inputs that are static.

F) Change Mode With Multiple Modules: (Figure 6)



It is possible to configure two or mode modules to Continuous Output Change mode on the same serial port.

This configuration may be used to extend the number of inputs monitored. The one drawback to this connection is that there is no means of avoiding a communications collision if two modules attempt to output data messages at exactly the same time. This will result in communications errors. Theoretical considerations aside, this type of connection may be very useful if the following guidelines are adhered to:

1) The inputs being scanned are primarily static. This is usually the case when monitoring security and alarm type of inputs where the change of an input indicates an extraordinary event. This cuts down the likelihood that two events would occur at the same time.

2) Checksums and parity must be used to detect communications errors caused by data collisions.

3) The host input port should be setup so that any activity on the input lines is evidence that a change in input status has occurred. This will cover the unlikely possibility that two modules are responding at exactly the same time. In this case the host may disable the Continuous Mode and poll the modules directly to read the input lines.

4) Use the highest baud rate possible to reduce the likelihood of collisions.

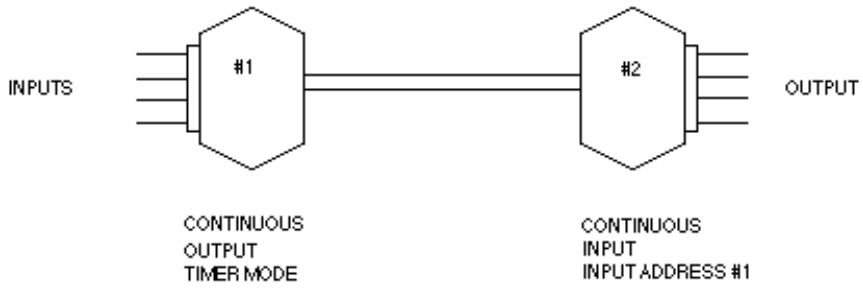
5) The Continuous Timer may be used to limit responses from a module. This particularly useful if an input is likely to turn on and off quickly and constantly, causing a continuous stream of data from one module. The CT command may be used to set a 'dead time' after a module has produced an output response:

Command: **$1CT+00005.00**
Response: ***

With the Continuous Timer set to 5 seconds, module #1 will pause for 5 seconds after each response before resuming scanning the digital I/O lines. This prevents the module from hogging the communications bus in response to continuously changing input lines.

G) Continuous Input Mode

The D1711/1712 modules may be set to a special mode called Continuous Input Mode which allows the module to respond to data transmitted by another module. A module in Continuous Input Mode may be paired with a module in Continuous Output Mode to provide digital data transfer without a supervisory host. Figure 6 shows the simplest connection.
(Figure 7)



Module #1 is setup in Continuous Output Timer Mode as described in Example A. Module #1 will read the state of the digital inputs and produce data messages on the communications bus. In this application, setting the Continuous Timer to zero will produce the fastest response to input changes.

Module #2 is setup for Continuous Input Mode. The digital I/O lines of module #2 are assigned as outputs. In continuous input mode, module #2 will use the data from module #1 as a command to control the digital outputs. The net effect is that the outputs of module #2 are controlled directly by the inputs of module #1.

For example, an output message from module #1 might look like:

***1DIA0A059 (59 is checksum)**

Module #2 in Continuous Input mode will interpret this data as a Digital Output command. Internally, the continuous input module will translate this data and perform the same function as:

$1DO5F5FAD (AD is checksum)

Note that the original data ‘A0A0’ is complemented to ‘5F5F’. This is necessary so that a high input at module #1 appears as a high output at module #2. As a result, the state of the digital inputs on module #1 is recreated at the digital outputs of module #2.

Since module #1 is continually outputting data on the communications lines, any changes in the state of the digital inputs on module #1 will be transmitted to module #2 and the output lines will change to reflect the new state.

To setup module #2 for Continuous Input Mode:

1) Setup the module for an address different from the Continuous Output module. In this example, the Continuous Output module is setup for address ‘1’. The Continuous Input module will be setup for address ‘2’:

Command: $1SU32070102
Response: *

Any address may be used for the Continuous Input module as long as it is different from module #1.

The communications setups for both modules must match. They must be setup with identical baud rate and parity settings. Also, the word length setup must be identical.

2) Assign the digital I/O lines to be outputs:

Command: $2AIOFFFF
Response: *

3) The Continuous Input module must be assigned a Continuous Input Address (CIA). This address is different from the normal communications address. This address is necessary so that the continuous input module may selectively read data on the communications bus. The full purpose of the CIA will be demonstrated in the next few examples. In this case, module #2 is setup to respond to data from module #1. Character '1' or ASCII '31' is the Continuous Input Address:

Command: $2CIA31
Response: *

The input address '31' is stored in nonvolatile memory. It can be read back with the Read Input Address command:

Command: $2RIA
Response: *31

4) Enable the Continuous Input Mode with the Continuous Mode Input command:

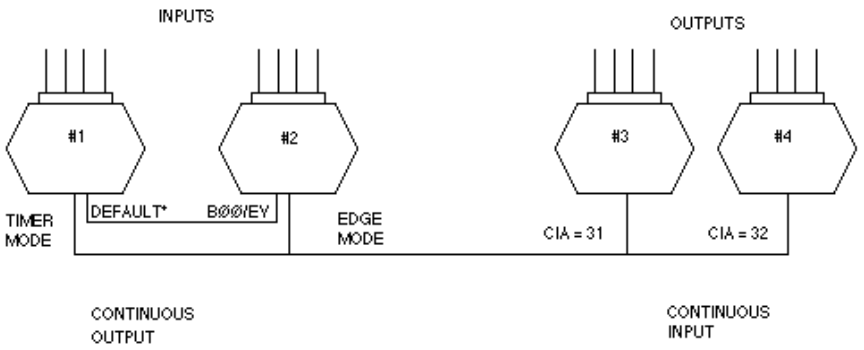
Command: $2CMI
Response: *

The Continuous mode is saved in nonvolatile memory.

After the Continuous Input module has been setup, it may be connected to the continuous output module as a stand-alone pair. Since all setup data is stored in nonvolatile memory, the input-output pair will initialize automatically upon power-up. No host is necessary for the continuous input-output function.

H) Multiple Continuous Input/Output

Figure 8 shows a system, with two modules set for continuous output mode and two modules set for Continuous Input Mode: (Figure 8)



This system is similar to example E except that 2 input-output module pairs share the same communications line. Modules #1 and #2 are setup for continuous output as detailed in example B. This pair of D1712's constantly read the state of their respective digital I/O lines and output the data on the communications bus. A typical output data stream would be:

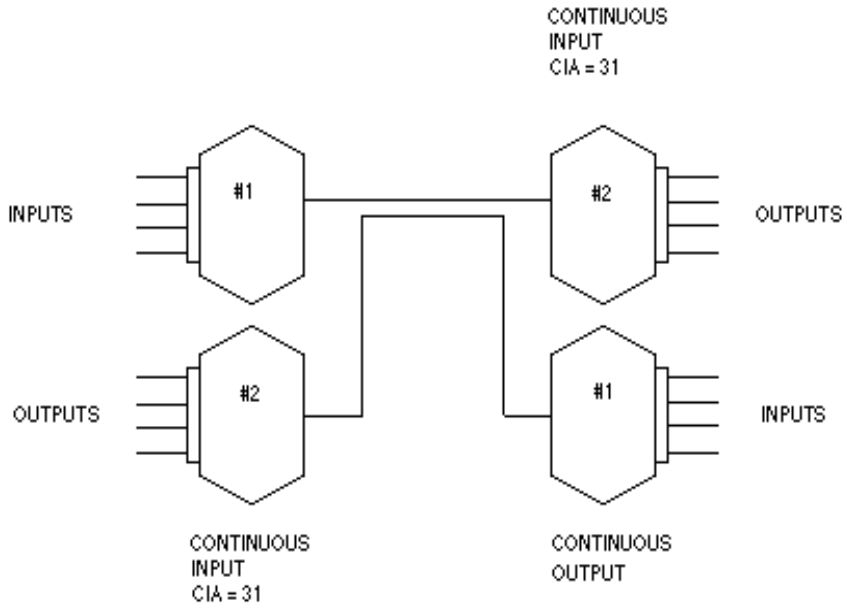
***1DI8123B6 (B6 is checksum)**
***2DIA0A0CB (CB is checksum)**

Module #3 is set for Continuous Input mode with the Continuous Input Address (CIA) equal to ASCII '31' or character '1'. This module will pick off the output data from module #1 and use the data as a command to set its output lines. Module #3 will ignore the data from module #2.

Module #4 is set for Continuous Input mode and its Continuous Input Address (CIA) is equal to ASCII '32' or character '2'. It examines the data on the communications bus and responds only to data containing the address '2'. Therefore the outputs of module #4 will follow the inputs of module #2. In theory, up to 124 pairs of modules may be linked together on a single communications bus.

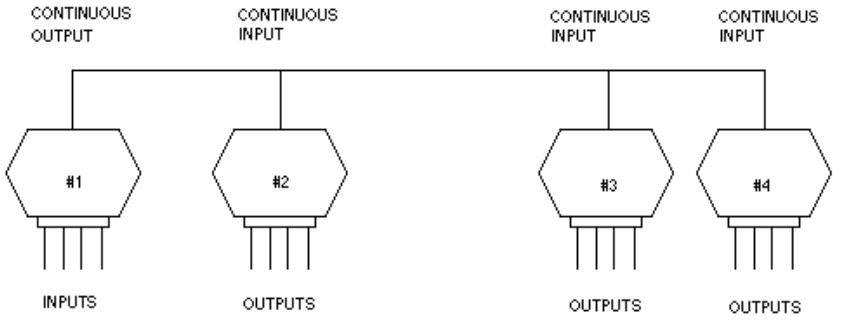
I) Bidirectional Continuous Input/Output

To provide bidirectional data transfer from one location to another, simply use two pairs of modules and two communications links: (Figure 9)



J) Multiple Outputs

The output data from a Continuous Output module may be used to control more than one continuous input module by assigning the correct Continuous Input Addresses (CIA): (Figure 10)



In this system, module #1 is set for Continuous Output mode. Modules 2, 3 & 4 are setup for Continuous Input. The three Continuous Input modules are all setup with a Continuous Input Address (CIA) of ASCII 31 or character '1'. This means that each of these three modules will accept the data from module #1 as an output command. The outputs of modules 2, 3 and 4 will replicate the input data of module #1.

Continuous Input Protocol Notes
A module in continuous input mode will respond to data in the form of:

***1DI80F096**

This is typical of a data string that may be produced by a continuous output module. The '1' denotes the address of the module producing the data stream. The Continuous Input Module will respond to this data only if it is programmed to read data from module #1. The Continuous Input Address (CIA) is used to specify which data strings will be examined by the Continuous Input module. The CIA command is used to specify the ASCII code for the address character. In this case, to allow the module to respond to data with the address tag '1', use the command:

Command: **$2CIA31**
Response: *****

The number '31' is the ASCII code for character '1'.

An important consideration in constructing a continuous input/output system is to make sure that all modules tied to the communications bus have unique address as specified by the SU command. This allows an intelligent host to use the modules in a normal polled manner. This greatly simplifies setup and debugging. The Continuous Input Addresses (CIA) may be set to any value independent of the polled address.

Chapter 7
Power Supply

D1711/D1712 modules may be powered with an unregulated +10 to +30Vdc. Power-supply ripple must be limited to 5V peak-to-peak, and the instantaneous ripple voltage must be maintained between the 10 and 30 volt limits at all times. All power supply specifications are referred to the module connector; the effects of line voltage drops must be considered when the module is powered remotely.

The D1711/D1712 modules employ an on-board switching regulator to maintain good efficiency over the 10 to 30 volt input range; therefore the actual current draw is inversely proportional to the line voltage. The D1711/D1712 consume a maximum of .75 watts and this figure should be used in determining the power supply current requirement. For example, assume a 24 volt power supply will be used to power four modules. The total power requirement is 4 X .75 = 3 watts. The power supply must be able to provide 3 / 24 = 0.125 amps.

In some cases, a small number of modules may be operated by “stealing” power from a host computer or terminal. Many computers provide a +15 volt output on the RS-232C DB-25 connector.

Small systems may be powered by using wall-mounted calculator-type modular power supplies. These units are inexpensive and may be obtained from many retail electronics outlets.

For best reliability, modules operated on long com-

munications lines (>500 feet) should be powered locally using small calculator-type power units. This eliminates the voltage drops on the Ground lead which may interfere with communications signals. In this case the V+ terminal is connected only to the local power supply. The Ground terminal must be connected back to the host to provide a ground return for the communications loop.

The D1711/D1712 modules are protected against power supply reversals.

The H1750/H1770 boards are powered by +5Vdc ±0.25V @ 30mA max. (not including any I/O module requirements.

Chapter 8
Troubleshooting

Symptom: RS-232 Module is not responding to commands

1 Using a voltmeter, measure the power supply voltage at the +Vs and GND terminals to verify the power supply voltage is between +10 and +30Vdc.

2 Verify using an ohmmeter that there are no breaks in the communications data lines.

3 Connect the module to the host computer and power-up each device (module and computer) then using a voltmeter measure the voltage between RECEIVE and GND. This voltage should be approximately - 10Vdc. Repeat the measurement between TRANSMIT and GND terminals and confirm the voltage value to be approximately -10Vdc. If either of the two readings is approximately 0.0Vdc then the communications data lines are wired backwards. Proper communications levels on both TRANSMIT and RECEIVE terminals should idle at -10Vdc.

4 If you are using a serial communications converter (A1000) ensure that the communications Baud Rate switch is set to the proper Baud Rate value.

5 Confirm software communications settings in Host computer match those values being used by the connected module(s).

6 If the Baud Rate value being used in the application is greater than 300 Baud and the module will only communicate 300 Baud then make sure that the

DEFAULT* terminal is not connected to Ground (GND).

7 If the module(s) are being used in a RS-232 daisy-chain communications configuration then ensure that the “Echo Bit” is enabled in the setup(SU) message of each module.

8 If the problem is not corrected after completing the steps above then connect the module by itself to a Host computer as outlined in Chapter 1.0 under “Quick Hook-up”. Start the supplied Utility software and please call the factory for further assistance.

Symptom: RS-485 Module is not responding to commands

1 Perform steps 1, 2, 4, 5 and 6 listed above.

2 Ensure that module RS-485 “Data” line (module terminal pin #7) is connected to the Host RS-485 “Data+” line.

3 Ensure that module RS-485 “Data*” line (module terminal pin #8) is connected to the Host RS-485 “Data-” line.

4 If the problem is not corrected after completing the steps above then connect the module by itself to a Host computer as outlined in Chapter 1.0 under “Quick Hook-up”. Start the supplied Utility software and please call the factory for further assistance.

Symptom: D1711, D1712 events counter not counting properly.

1 Check that the frequency of the signal, being counted is less than 60Hz.

2 Ensure that the signal levels are swinging below +1.0Vdc and greater than +3.5Vdc.

Appendix A
ASCII Table

Table of ASCII characters (A) and their equivalent values in Decimal (D), Hexadecimal (Hex), andBinary.Claret (^) represents Control function.

A	D	Hex	Binary	D	Hex	Binary
^@	0	00	00000000	128	80	10000000
^A	1	01	00000001	129	81	10000001
^B	2	02	00000010	130	82	10000010
^C	3	03	00000011	131	83	10000011
^D	4	04	00000100	132	84	10000100
^E	5	05	00000101	133	85	10000101
^F	6	06	00000110	134	86	10000110
^G	7	07	00000111	135	87	10000111
^H	8	08	00001000	136	88	10001000
^I	9	09	00001001	137	89	10001001
^J	10	0A	00001010	138	8A	10001010
^K	11	0B	00001011	139	8B	10001011
^L	12	0C	00001100	140	8C	10001100
^M	13	0D	00001101	141	8D	10001101
^N	14	0E	00001110	142	8E	10001110
^O	15	0F	00001111	143	8F	10001111
^P	16	10	00010000	144	90	10010000
^Q	17	11	00010001	145	91	10010001
^R	18	12	00010010	146	92	10010010
^S	19	13	00010011	147	93	10010011
^T	20	14	00010100	148	94	10010100
^U	21	15	00010101	149	95	10010101
^V	22	16	00010110	150	96	10010110
^W	23	17	00010111	151	97	10010111
^X	24	18	00011000	152	98	10011000
^Y	25	19	00011001	153	99	10011001
^Z	26	1A	00011010	154	9A	10011010
^[	27	1B	00011011	155	9B	10011011
^\	28	1C	00011100	156	9C	10011100
^]	29	1D	00011101	157	9D	10011101
^^	30	1E	00011110	158	9E	10011110
^_	31	1F	00011111	159	9F	10011111
	32	20	00100000	160	A0	10100000
!	33	21	00100001	161	A1	10100001

A	D	Hex	Binary	D	Hex	Binary
“	34	22	00100010	162	A2	10100010
#	35	23	00100011	163	A3	10100011
$	36	24	00100100	164	A4	10100100
%	37	25	00100101	165	A5	10100101
&	38	26	00100110	166	A6	10100110
‘	39	27	00100111	167	A7	10100111
(	40	28	00101000	168	A8	10101000
)	41	29	00101001	169	A9	10101001
*	42	2A	00101010	170	AA	10101010
+	43	2B	00101011	171	AB	10101011
,	44	2C	00101100	172	AC	10101100
-	45	2D	00101101	173	AD	10101101
.	46	2E	00101110	174	AE	10101110
/	47	2F	00101111	175	AF	10101111
0	48	30	00110000	176	B0	10110000
1	49	31	00110001	177	B1	10110001
2	50	32	00110010	178	B2	10110010
3	51	33	00110011	179	B3	10110011
4	52	34	00110100	180	B4	10110100
5	53	35	00110101	181	B5	10110101
6	54	36	00110110	182	B6	10110110
7	55	37	00110111	183	B7	10110111
8	56	38	00111000	184	B8	10111000
9	57	39	00111001	185	B9	10111001
:	58	3A	00111010	186	BA	10111010
;	59	3B	00111011	187	BB	10111011
<	60	3C	00111100	188	BC	10111100
=	61	3D	00111101	189	BD	10111101
>	62	3E	00111110	190	BE	10111110
?	63	3F	00111111	191	BF	10111111
@	64	40	01000000	192	C0	11000000
A	65	41	01000001	193	C1	11000001
B	66	42	01000010	194	C2	11000010
C	67	43	01000011	195	C3	11000011
D	68	44	01000100	196	C4	11000100
E	69	45	01000101	197	C5	11000101
F	70	46	01000110	198	C6	11000110
G	71	47	01000111	199	C7	11000111
H	72	48	01001000	200	C8	11001000
I	73	49	01001001	201	C9	11001001
J	74	4A	01001010	202	CA	11001010
K	75	4B	01001011	203	CB	11001011

A	D	Hex	Binary	D	Hex	Binary
L	76	4C	01001100	204	CC	11001100
M	77	4D	01001101	205	CD	11001101
N	78	4E	01001110	206	CE	11001110
O	79	4F	01001111	207	CF	11001111
P	80	50	01010000	208	D0	11010000
Q	81	51	01010001	209	D1	11010001
R	82	52	01010010	210	D2	11010010
S	83	53	01010011	211	D3	11010011
T	84	54	01010100	212	D4	11010100
U	85	55	01010101	213	D5	11010101
V	86	56	01010110	214	D6	11010110
W	87	57	01010111	215	D7	11010111
X	88	58	01011000	216	D8	11011000
Y	89	59	01011001	217	D9	11011001
Z	90	5A	01011010	218	DA	11011010
[	91	5B	01011011	219	DB	11011011
\	92	5C	01011100	220	DC	11011100
]	93	5D	01011101	221	DD	11011101
^	94	5E	01011110	222	DE	11011110
_	95	5F	01011111	223	DF	11011111
`	96	60	01100000	224	E0	11100000
a	97	61	01100001	225	E1	11100001
b	98	62	01100010	226	E2	11100010
c	99	63	01100011	227	E3	11100011
d	100	64	01100100	228	E4	11100100
e	101	65	01100101	229	E5	11100101
f	102	66	01100110	230	E6	11100110
g	103	67	01100111	231	E7	11100111
h	104	68	01101000	232	E8	11101000
i	105	69	01101001	233	E9	11101001
j	106	6A	01101010	234	EA	11101010
k	107	6B	01101011	235	EB	11101011
l	108	6C	01101100	236	EC	11101100
m	109	6D	01101101	237	ED	11101101
n	110	6E	01101110	238	EE	11101110
o	111	6F	01101111	239	EF	11101111
p	112	70	01110000	240	F0	11110000
q	113	71	01110001	241	F1	11110001
r	114	72	01110010	242	F2	11110010
s	115	73	01110011	243	F3	11110011
t	116	74	01110100	244	F4	11110100
u	117	75	01110101	245	F5	11110101

A	D	Hex	Binary	D	Hex	Binary
}v	118	76	01110110	246	F6	11110110
w	119	77	01110111	247	F7	11110111
x	120	78	01111000	248	F8	11111000
y	121	79	01111001	249	F9	11111001
z	122	7A	01111010	250	FA	11111010
{	123	7B	01111011	251	FB	11111011
|	124	7C	01111100	252	FC	11111100
	125	7D	01111101	253	FD	11111101
~	126	7E	01111110	254	FE	11111110
	127	7F	01111111	255	FF	11111111

Appendix B
H1770 64 Channel Digital I/O Board

The H1770 Digital I/O interface is designed to provide remote I/O capability for computers, modems, and other devices with standard serial ports. Commands communicated over standard RS-232 or RS-485 links may be used to control or read up to 64 digital I/O channels.

The H1700 is designed to interface to industry-standard solid-state relay racks. Up to four 16-channel racks may be connected to the H1770 with ribbon cable connectors.

The 64 I/O channels may be configured to be inputs or outputs in any combination designated by the user. The input/output configuration may be changed at any time through the communications port. The I/O assignments are saved in nonvolatile memory and are automaticly loaded when the unit is powered up.

Getting Started

RS-232, RS-485 Selection
The H1770 contains drivers to connect to either RS-232 or RS-485 ports. The H1770 must be configured to the desired interface before it is connected to the host. The H1770 has a 12-pin header near the edge of the board marked RS-232/RS-485. To configure the board for RS-232, make sure the three jumpers are installed adjacent to the RS-232 label. To configure the board for RS-485, make sure the jumpers are adjacent to the RS-485 label.

Pin Connections
The host interface connection is wired to the six-pin terminal plug:

Pin 1: +5V This is the power supply connection for the board. The power supply must provide +5V +/- 5% at 100 mA.

Pin 2: DEFAULT* This pin is normally left open or pulled high to +5V. When grounded, the board assumes the default communications setup of 300 baud, any address, no parity (See Chapter 1).

Pin 3: DATA/TX This pin is a serial port connection. If the board is configured for RS-485, this pin is the DATA connection. For RS-232, pin 3 is the Transmit output from the H1770.

Pin 4: DATA*/RX This pin is a serial port connection. If the board is configured for RS-485, this is the DATA*, or the negative data connection. For RS-232, this is the receive input of the H1770.

Pin 5: CONT* This pin is normally left open or pulled up to +5V. When grounded, the H1770 will be in Continuous Mode.

Pin 6: GND This is the power supply ground connection. It is also the signal ground for the serial port.

Output Connections

The digital I/O connections are made through the four ribbon cable connectors. The output connections are made to be compatable with industry-standard 16-channel solid-state relay racks. The connector nearest the edge of the board (J2) is wired to the lowest-order 16 bits.

J2

Pin	Signal
17	B0F
19	B0E
21	B0D
23	B0C
25	B0B
27	B0A
29	B09
31	B08
33	B07
35	B06
37	B05
39	B04
41	B03
43	B02
45	B01
47	B00

All even pins are connected to GND
All other pins are no connection.

J3

Pin	Signal
17	B1F
19	B1E
21	B1D
23	B1C
25	B1B
27	B1A
29	B19
31	B18
33	B17
35	B16
37	B15
39	B14
41	B13
43	B12
45	B11
47	B10

All even pins are connected to GND
All other pins are no connection.

J4

Pin	Signal
17	B2F
19	B2E
21	B2D
23	B2C
25	B2B
27	B2A
29	B29
31	B28
33	B27
35	B26
37	B25
39	B24
41	B23
43	B22
45	B21
47	B20

Even pins 18-50 are connected to GND
All other pins are no connection.

J5

Pin	Signal
17	B3F
19	B3E
21	B3D
23	B3C
25	B3B
27	B3A
29	B39
31	B38
33	B37
35	B36
37	B35
39	B34
41	B33
43	B32
45	B31
47	B30

Even pins 18-50 are connected to GND
All other pins are no connection.

Appendix C
H1750 24-Channel Digital I/O Board

The H1750 is designed to interface directly to an Opto-22 16 or 24 channel solid-state relay backplane or equivalent backplanes with the standard pin-out.

Hook-Up:
Plug the H1750 into the backplane header. Be sure the board is centered over the header and latch the board to the header.

The H1750 contains line drivers for both RS-232 and RS-485. Select the desired communications link with the three shorting bars near the edge of the H1750. Move the three shunts to the 'RS-232' position to select RS-232. Move the three shunts to the 'RS-485' position to select RS-485.

If RS-232 is selected, the DATA/TX pin on the 6-pin connector is the transmit output pin. The DATA*/RX pin is the receive input.

If RS-485 is selected, connect the Data (Data +) line to the DATA/TX pin and the Data* (Data -) line to the DATA*/RX pin.

The H1750 is powered by a regulated 4.75 to 5.5V power supply connected to the GND and +5 Vdc pins. For proper operation, relays used in a backplane should be +5V types.

I/O lines are connected to the odd-numbered pins on the header connector. The odd-numbered pins are closest to the edge of the board with the square pad being #1. All even-numbered pins are connected to ground.

Pin Connections
The host interface connection is wired to the six-pin terminal plug:

Pin 1: +5V This is the power supply connection for the board. The power supply must provide +5V +/- 5% at 100 mA.

Pin 2: DEFAULT* This pin is normally left open or pulled high to +5V. When grounded, the board assumes the default communications setup of 300 baud, any address, no parity (See Chapter 1).

Pin 3: DATA/TX This pin is a serial port connection. If the board is configured for RS-485, this pin is the DATA connection. For RS-232, pin 3 is the Transmit output from the H1750.

Pin 4: DATA*/RX This pin is a serial port connection. If the board is configured for RS-485, this is the DATA*, or the negative data connection. For RS-232, this is the receive input of the H1750.

Pin 5: CONT* This pin is normally left open or pulled up to +5V. When grounded, the H1750 will be in Continuous Mode.

Pin 6: GND This is the power supply ground connection. It is also the signal ground for the serial port.

Output Connections

The digital I/O connections are made through the connector. The output connections are made to be compatable with industry-standard 16 or 24-channel solid-state relay racks. The 50-pin ribbon connector near the edge of the board is wired as follows.

Pin	Signal
1	B17
3	B16
5	B15
7	B14
9	B13
11	B12
13	B11
15	B10
17	B0F
19	B0E
21	B0D
23	B0C
25	B0B
27	B0A
29	B09
31	B08
33	B07
35	B06
37	B05
39	B04
41	B03
43	B02
45	B01
47	B00

All even pins are connected to GND
All other pins are no connection.

Appendix D
1700 Series Specifications

1700 Series Specifications (typical @ +25°C and nominal power supply unless otherwise noted)

H1750/H1770 Digital Input/Output Boards

H1750: 24 digital input/output bits with RS-232 or RS-485 output.
H1770: 64 digital input/output bits with RS-232 or RS-485 output.

- User can define any bit as an input or an output.
- Inputs/Outputs can be read/set individually or in parallel.
- Input voltage levels: 0-10V without damage.
- Input switching levels: High, 3.5V min., Low, 1.0V max.
- Outputs: 0-10V, 15mA max. load.
- Power requirements: +5Vdc ±0.25V @ 30mA max. (not including I/O modules requirements)
- User selectable RS-232/RS-485 Communications.

D1700 Digital Input/Output Modules

D1711: 15 digital input/output bits with RS-232 output.
D1712: 15 digital input/output bits with RS-485 output.

- User can define any bit as an input or an output.
- Input voltage levels: 0-30V without damage.
- Input switching levels: High, 3.5V min., Low, 1.0V max.
- Outputs: Open collector to 30V, 100mA max. load.
- Vsat: 1.0V max @ 100mA.
- Events counter: Up to 10 million positive transitions at bandwidths of 20Hz, 50Hz, 200Hz and 20KHz.
- Power requirements: Unregulated +10V to +30Vdc, 0.75W max.
- Internal switching regulator.
- Protected against power supply reversals.

Communications

- Communications in ASCII via RS-232, RS-485 ports.
- Up to 124 multidrop boards per host communications port.
- User selectable channel address.
- NRZ asynchronous data format; 1 start bit, 7 data bits, 1 parity bit and 1 stop bit.
- Selectable baud rates: 300, 600, 1200, 2400, 4800,9600, 19200, 38400.
- ASCII format command/response protocol.
- Can be used with a “dumb” terminal.
- Parity: odd, even, none.

- All communications setups (address, baud rate, parity) stored in nonvolatile memory using EEPROM.
- Transient suppression on RS-485 Communications lines.
- Communications error checking via checksum.
- Communications distance up to 4,000 feet.

Digital

- 8-bit CMOS microcomputer.
- Nonvolatile memory storage for start up values eliminates software initialization.

Environmental

Temperature Range: Operating -25°C to +70°C.
Storage -25°C to +85°C.
Relative Humidity: 0 to 95% noncondensing.