DGH A2400 USERS MANUAL

REVISED: 4/17/95

DGH CORPORATION
P. O. BOX 5638
MANCHESTER, NH 03108

TELEPHONE: 603-622-0452

The A2400 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

WARRANTY

DGH warrants each A2400 series module to be free from defects in materials and workmanship under normal conditions of use and service and will replace any component found to be defective, on its return to DGH, transportation charges prepaid within one year of its original purchase. DGH assumes no liability, expressed or implied, beyond its obligation to replace any component involved. Such warranty is in lieu of all other warranties expressed or implied.

WARNING

The circuits and software contained in A2400 modules are proprietary. Purchase of these products does not transfer any rights or grant any license to the circuits or software used in these products. Disassembling or decompiling of the software program is explicitly prohibited. Reproduction of the software program by any means is illegal.

As explained in the setup section, all setups are performed entirely from the outside of the A2400 module. There is no need to open the module because there are no user-serviceable parts inside. Removing the cover or tampering with, modifying, or repairing by unauthorized personnel will automatically void the warranty. DGH is not responsible for any consequential damages.

RETURNS

When returning products for any reason, contact the factory and request a Return Authorization Number and shipping instructions. Write the Return Authorization Number on the outside of the shipping box. DGH strongly recommends that you insure the product for value prior to shipping. Items should not be returned collect as they will not be accepted.

Shipping Address:

DGH Corporation
Hillhaven Industrial Park
146 Londonderry Turnpike
Hooksett, NH 03106

Chapter 1
Getting Started

Introduction

This manual describes the function and application of the Radio Modem Interface Module (A2400). The A2400 provides an intelligent interface between radio modems available from many manufacturers and devices designed to operate on a bi-directional RS-485 serial bus. Although the A2400 has been designed specifically for our family of industrial I/O modules, it may also be used with other RS-485 devices.

Figure 1.1 depicts a typical application that incorporates A2400's. In many data acquisition situations, the sensor data is inaccessible to the host computer due to large distances or the lack of telephone facilities to incorporate conventional dial-up modems. In some cases, sensor data may have to be monitored full time and the cost of telephone service can be prohibitive. For these and a multitude of other reasons the use of a radio link can be the best solution.

Unfortunately, radio modems are designed for computer-to-computer communications and require a certain amount of intelligence at each radio site in order to construct useful systems. The cost of a local computer at each radio can easily make the concept impractical. The Radio Modem Interface Module (A2400) fills the need for a low-cost intelligent interface between the radio modem and the RS-485 data acquisition devices.

In a typical system as shown in Figure 1.1, there is one host or master computer and any number of slave sites. The master radio transmitter and the slave receivers communicate on the same radio frequency, and of course, the slave transmitters and the master receiver are tuned to the same frequency. While it is common to use two frequencies for simultaneous transmitting and receiving, it is possible to use one frequency for all communications. In an idle condition all slave transmitters are turned off. Each slave site is assigned a unique address so that the master may direct commands to a particular site. To initiate a communications sequence, the master will transmit a command by radio which is received by all the slaves. The transmitted command contains an address which directs the command to a particular slave site. The slave site that matches the address will respond to the command. At this time, the addressed slave site will turn on its radio transmitter and communicate back to the master in response to the command. Once the response is complete the slave will turn off the transmitter and wait for a new command. To avoid interference, only one slave transmitter can be on at any given time. The primary function of the

A2400 is to control the slave transmitter to allow multiple slave sites.

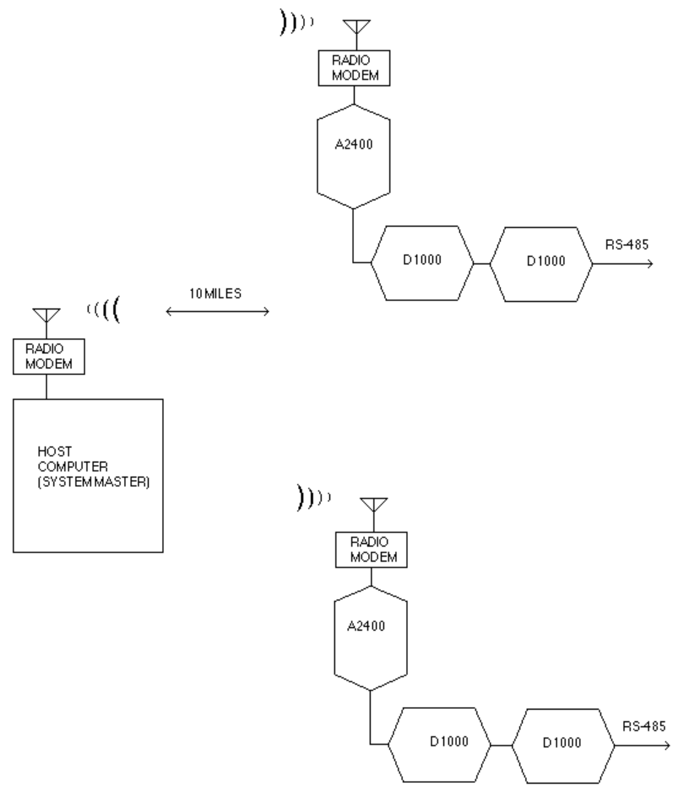

Figure 1.1 System Overview.

Leased Lines
This manual has been written with emphasis on radio modems. However, the A2400's may be used just as effectively with leased telephone lines. Typically, leased lines do not have dial-up capability and some means of addressing and multiplexing must be employed if multiple stations are used. A2400's may be used with leased line modems in an identical manner as with radio modems.

Getting Started

To get your A2400 up and running for an initial check-out, connect the unit to a power supply and terminal as shown in Figure 1.2. The power supply can be any dc source from 10 to 30 volts, capable of 1 Watt of power. The terminal can be any RS-232 dumb terminal set for 300 baud. A computer configured as a terminal can also be used. Be sure to ground the DEFAULT* pin.

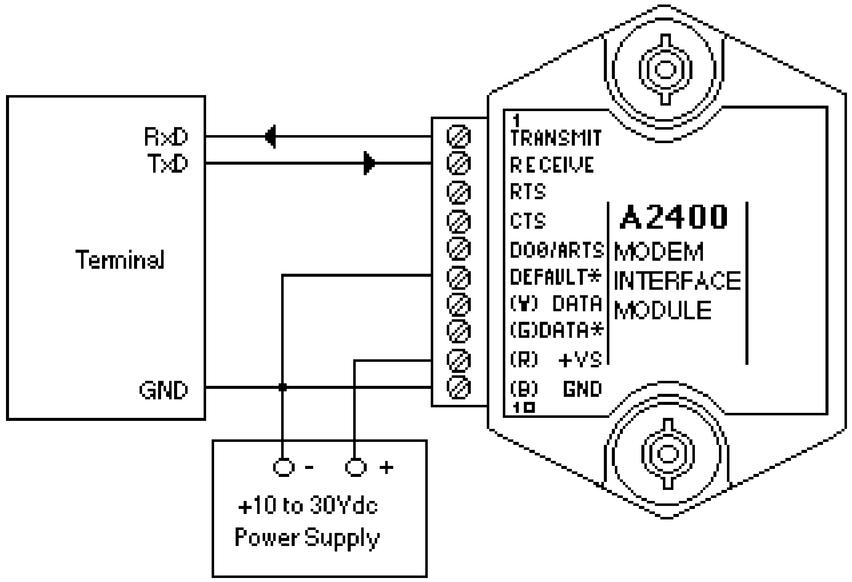

Figure 1.2 A2400 Quick Hookup.

After checking the connections, power up the A2400. Type the following command on the terminal:

$1RD

Make sure to use upper case characters for the ‘RD’ (Read Data) command and terminate the command with a carriage return. The A2400 should reply with the message:

***+99999.99**

This message is terminated with a carriage return. If the response message cannot be obtained, re-check all the wiring, making sure that the proper power is on the A2400 connector and that the DEFAULT* line is shorted to the GND pin. The terminal must be set to 300 baud.

If, after several attempts, the response message does not appear, refer to Chapter 8 Troubleshooting in this manual.

If you have an IBM PC or compatible computer, running the S1000 setup software will ease the task of setting up the A2400 for your application.

After establishing communications with the A2400, read the manual and feel free to experiment with the various commands and setups available. If communications is lost due to an improper setup, returning back to the hook-up of Figure 1.2 will restore communications to the A2400.

Default Mode

The A2400 contains 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 setup parameters may be configured remotely through the communications port without having to physically change switch settings. There is one 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 A2400 what the baud rate, address, parity and other settings are. It is difficult to establish communications with a A2400 whose address and baud rate are unknown. To overcome this, the A2400 has a pin labeled DEFAULT*. By connecting this pin to ground (GND) the A2400 is forced to a known communications setup called Default Mode.

The Default Mode setup is: 300 baud, 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 the communications parameters stored in the A2400.

An A2400 in Default Mode will respond to any address except the six illegal values (NULL, CR, $, #, {, }). A dummy address must be included in every command for proper responses.

Setup information in an A2400 may be changed at will with the SetUp (SU) command. Baud rates 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. During normal operation, the DEFAULT* pin should be left open.

Chapter 2
Functional Description

Block Diagram

The A2400 is an RS-232/RS-485 converter specifically designed to interface D series RS-485 modules to radio modems. To this end the A2400 provides three functions:

1) Perform the RS-232 to RS-485 electrical conversion.

2) Control the data direction of the RS-485 bus.

3) Create hand-shaking signals to control the modem.

A simplified block diagram of the A2400 is illustrated in Figure 2.1. This shows the RS-485 and RS-232 drivers connected to their respective interface pins. The A2400 contains two UARTs (Universal Asynchronous Receiver / Transmitters), one dedicated to each port. The RS-485 port is connected to a UART that is integral to the supervisory microprocessor. The RS-232 port connects to a second UART external to the microprocessor. All data communicated to and from the A2400 on either port must pass through the microprocessor. The micro controls the data flow depending on the content of the data and setup information specified by the user.

The A2400 contains an Electrically Erasable Programmable Read-Only Memory (EEPROM) which is used to store operating parameters specified by the user. The EEPROM will retain the setup data even if power is removed from the A2400. The EEPROM requires no battery and is guaranteed to retain data for at least ten years. The setup data stored in the EEPROM includes the baud rate, address, parity, timing data, etc. When power is applied to the A2400, the internal microprocessor reads the setup data from the EEPROM and automatically configures itself. The setup data may be downloaded with a terminal or computer connected to the RS-232 port.

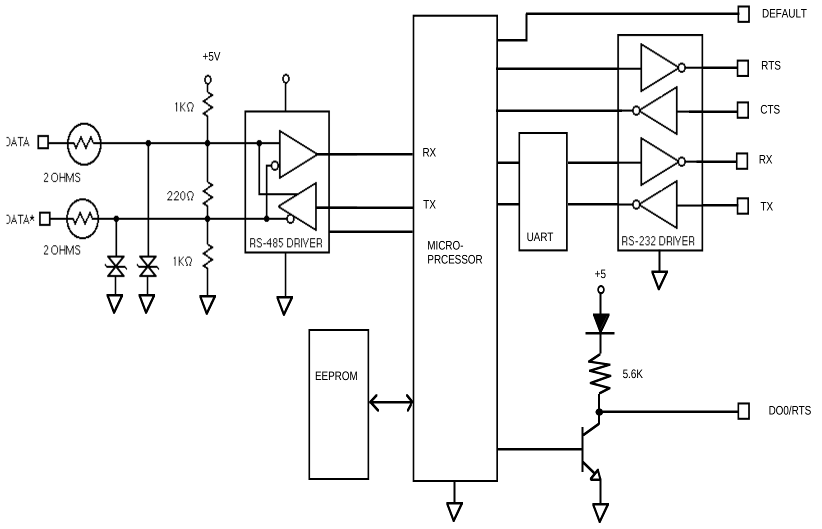

Figure 2.1 A2400 Block Diagram.

Pinout

1) TRANSMIT   This is the RS-232 Transmit output from the A2400. This pin is normally connected to the Receive input of a modem. This output is also used to connect to a terminal or computer to configure the A2400

2) RECEIVE   This is the RS-232 Receive input of the A2400. This pin is normally connected to the Data Output of a radio modem. This input is also used to connect to a terminal or computer to configure the A2400

3) RTS   RS-232 Request To Send output. This output is used to control the transmitter of the modem which allows multiple transmitters to exist on the same system. The RTS output is typically connected to the RTS input of the modem. The timing of the RTS signal is user-configurable with the T1, T2, T3 commands. The polarity of the RTS signal may be configured with the Setup (SU) command.

4) CTS   This is the RS-232 Clear To Send input. Some modems provide a signal to indicate that the transmitter is ready after the RTS line has been asserted. This ready signal may be connected to the CTS input to provide a hardware handshake to provide a fast turn-around time. If the CTS line is not used, it may be left open and delay time T2 will function as a software handshake.

5) DO0/ARTS   This is a digital output that can be configured to perform two functions. The function configuration is set by a bit in the Setup (SU) command. Normally this pin is configured as a general-purpose digital output. It may be turned on and off by commands from the host computer. The exact circuit schematic of the output is shown in Fig. 2.1. This circuit provides an output that is TTL and CMOS compatible. The diode in the circuit allows the collector to be pulled up to 30 volts to interface to relays and other higher-voltage devices. This output may also be configured as an Alternate Request To Send signal. With this setup, the ARTS signal exactly mimics the RTS output. This output may be used when a TTL signal is desired or a high-current open-collector signal is necessary.

6) DEFAULT*   By grounding this pin, the A2400 is placed in a known communication setup. This is essential if the baud rate and address of the A2400 are not known. The default communications setup is: 300 baud, no parity, any address is recognized. The DEFAULT* pin should be grounded only if the A2400 is being setup or configured. In normal operation this pin is left open.

7) (Y)DATA+   This is the positive polarity signal of the differential RS-485

bus. This bus connects to multidrop RS-485 devices such as D series modules.

8) (G)DATA- This is the negative polarity of the differential RS-485 bus.

9) (R)V+ A2400 power connection. The A2400 operates on 10 to 30 volts dc.

10) (B)GND This is the ground connection common to all circuits. The A2400 does not have isolation between power and the two communications ports.

Note that pins 7 through 10 are designated Y, G, R, and B respectively. This corresponds to the Yellow, Green, Red, and Black colors normally found in common telephone cable. All D series RS-485 devices carry this nomenclature. If all yellow connections are connected together, and green to green, etc., the RS-485 wiring will be correct.

RS-485 Termination

The RS-485 port lines are terminated as shown in Fig. 2.1. The 220 ohm resistor is used as a transmission line terminator to improve signal fidelity. A similar 220 ohm resistor should be installed at the far end of the RS-485 bus. The 1 K ohm resistors are used to bias the signal lines to a Mark condition. This is necessary when all transmitters of the RS-485 bus are off, which is the most prevalent condition. The resistor bias helps to prevent noise pickup.

The RS-485 port is protected from potentially destructive voltages with positive temperature coefficient thermistors and transient suppressors.

A2400 Operation

A typical installation is shown in Figure 3.1(see Chapter 3). The A2400 is used to provide an interface between a modem and a string of standard modules. Each module and the A2400 has a unique address. In a typical communications sequence the host computer and radio (not shown) will transmit a module command over the air. The modem receiver picks up the message data and presents it to the A2400 on the RS-232 port.

Character Filter

Due to the nature of radio data transmission, noisy data is almost always present at the output of the modem. This is due to inadequate squelch or noise generated when radio transmitters are turned on and off. The A2400 uses noise reduction techniques to reduce the possibility of bad characters reaching the RS-485 bus.

The first operation performed on the modem data is to check for noise and framing errors. If either condition exists, the bad character is re-formatted as a null character (ASCII $00). Since the null is not a legal character for use as an address in the modules, transmitting a null is preferable to aborting the character when an error is detected. This cuts down on the possibility of a module being incorrectly addressed.

After noise and framing errors are checked, the data must be qualified with a character filter. This filter will flush any data until a valid prompt character is detected. These characters are: '$', '#', '{', and '}'. Parity is ignored. Once a prompt character is received, The A2400 assumes that a valid command is being transmitted. The A2400 will transfer the data to the RS-485 bus or hold it in an internal buffer depending on the address data. At this time the character filter checks for a carriage return character which terminates the command. If a carriage return is detected, the character filter is reset to flush characters until another valid prompt is found. If 32 characters are received after a prompt without a carriage return found, the data is considered to be noisy and the character filter is reset for prompt detection.

The character filter may be disabled for non-D series systems.

The qualified data may take one of two paths depending on the address data. Commands addressed to the A2400 itself are not transmitted on the RS-485 bus. Therefore the prompt character is saved in buffer memory until the address character can be examined. If the A2400 detects its own address, the subsequent command data is processed internally to the A2400. No data will appear on the RS-485 bus.

If the address does not match the A2400, the prompt and address characters are transmitted to the RS-485 port, along with any subsequent data until a carriage return or character over-run occurs.

The A2400 ignores parity on all data except for commands addressed to itself.

The RS-485 port is normally in receive mode, and when the A2400 places data on the bus it enables the RS-485 transmitter on a per-character basis. This means that the port is returned to receive mode immediately after a command has been transmitted.

Assume that a module on the RS-485 bus has received a correctly addressed command and it responds back with information on the bus. The A2400 receives this information and places it in a buffer that can hold up to 96 characters. The parity of received characters is ignored. As soon as a character is received, the A2400 starts a timing sequence to control the modem transmitter. Three user-programmable timers, T1, T2, and T3 control the data flow. See Figure 2.2

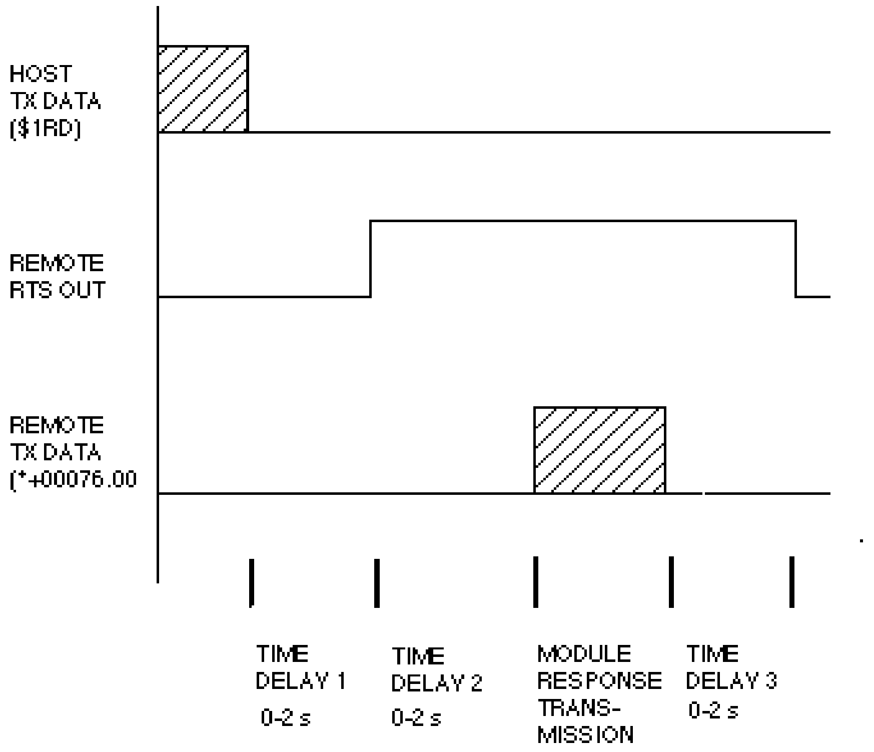

Figure 2.2 Programmable Delay Times.

T1

As soon as the A2400 detects a character in the RS-485 receive buffer, time delay T1 is activated. This is a dead time to allow the host to prepare for the receipt of a message. This is particularly important when a simplex connection is used, where the send and receive data is transmitted on the same frequency. During this time the A2400 creates no control output, but any data received on the RS-485 port is stored in the receive buffer. At the end of time T1, the A2400 asserts the RTS control signal to ‘key’ or turn on the transmitter of the radio.

T2

Once the RTS signal has been asserted, the T2 delay is activated. This is a delay time to allow the transmitter to power up and settle in anticipation of a transmission back to the host. The settling time required is specified by the modem manufacturer. When the T2 time period is over, the A2400 will start to transmit the data held in the receive buffer, and will continue to transmit until the buffer is empty.

Some radio modems provide a CTS (Clear To Send) signal that indicates that the transmitter has settled and is ready for data. This signal may be connected to the CTS input of the A2400 to provide hardware handshaking.

The delay period T2 ends when either the CTS signal is detected or the T2 timer ends, whichever comes first.

T3

The A2400 will transmit data to the modem until the receive buffer is empty. When the receive buffer is empty, and the last character has been transmitted, time delay T3 is activated. T3 provides two functions: it provides a clean break between the transmitted data and the turn-off of the radio transmitter, and it allows the host to poll more data without keying the transmitter on and off. During time T3, the host may transmit another command which would pass through the A2400 to the RS-485 bus. Typically, this command would generate a response from a device on the RS-485 bus. If the response data is received by the A2400 before T3 is complete, T3 is canceled and the received data is immediately transferred to the modem. When the receive buffer is empty, T3 is activated again, and the cycle repeats itself. This allows the host to establish communications with the remote radio and talk back and forth without wasting time re-keying the transmitter for each response.

If T3 times out and the receive buffer is empty, the radio connection is terminated by turning off the RTS signal.

Chapter 3
Communications

Introduction

The A2400 modules have been carefully designed to be easy to interface to all radio modems and many leased-line modems. 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. The ASCII format makes system debugging easy with a dumb terminal.

This system allows multiple modules to be connected through the A2400 to a modem with a single 4-wire cable. Up to 32 RS-485 modules may be strung together on one cable; 121 with repeaters. 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 A2400 modules is performed with two- or three-character ASCII command codes such as RD for Read Data. 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 assuming that delay times T1, T2, T3 = 0:

Table 3.1 Response Timeout Specifications.

Mnemonic	Timeout
DO, OC, CC, RD, REA, RID, RLP, RS, RSP, RSU, RT1, RT2, RT3, WE	≤ 10 ms
EA, ID, LP, RID, RR, SP, SU, T1, T2, T3	≤ 100 ms

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 A2400 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-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-232:

1) balanced line gives excellent noise immunity
2) can communicate with modules at 38400 baud
3) communications distances up to 10,000 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

An RS-485 system usually requires an interface such as the A2400 to convert RS-232 to RS-485.

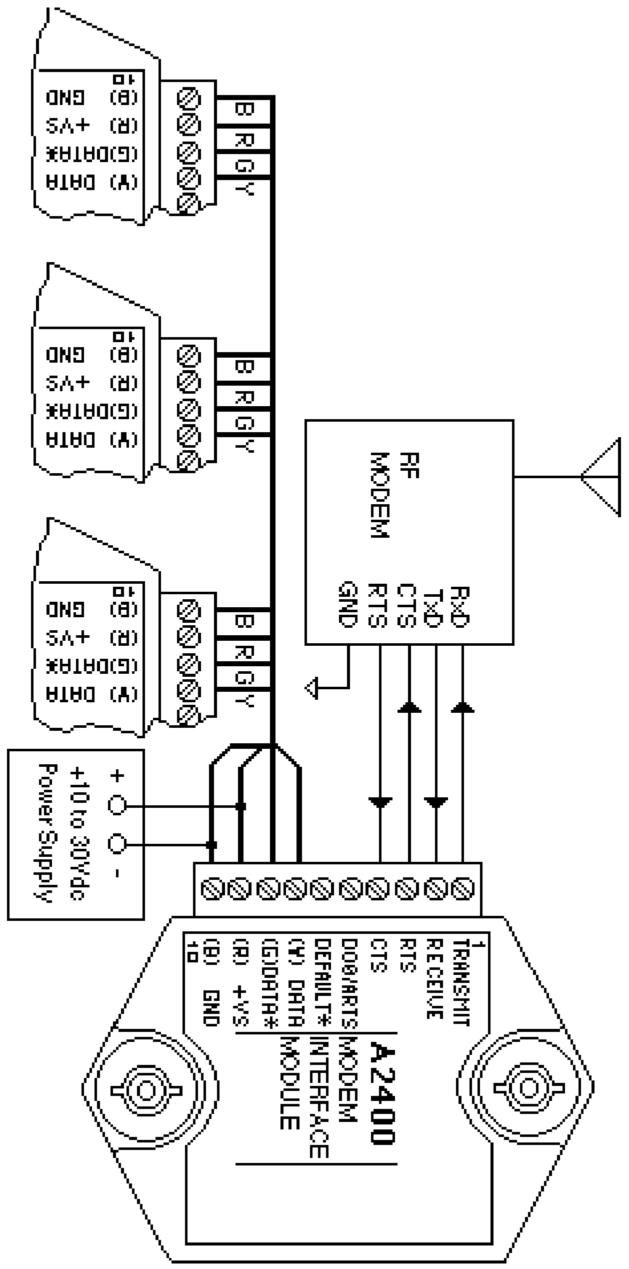

Figure 3.1 RS-485 System.

RS-485 Multidrop System

Figure 3.1 illustrates the wiring required for multiple-module RS-485 system. Notice that every module has a direct connection to the A2400. 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.

To minimize unwanted reflections on the transmission line, the bus should be arranged as a line going from one module to the next, starting with the A2400. ‘Tree’ or random structures of the transmission line should be avoided. For wire runs greater than 500 feet total, the end of the bus should be terminated with a 220Ω resistor connected between DATA+ and DATA-. The A2400 has a resistor built in to terminate the start of the bus.

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 are biased in a MARK condition as shown in Figure 2.1. 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.

Chapter 4
A2400 Command Set

The A2400 operates with a simple command/response protocol to control all module functions. A command must be transmitted to the A2400 by the host computer or terminal before the A2400 will respond with useful data. A module can never initiate a communications sequence. A list of available commands and a sample format for each command is listed in Table 4.1.

The following text describes the protocol normally used with the A2400. In addition to the normal operation there are two other modes of operation called the Extended Address mode and the Transparent mode. For larger systems, the Extended Address protocol described in Chapter 7 may be more appropriate. The Transparent mode is for systems that use non-D series protocols. Transparent mode is described in Chapter 8.

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. For standard addressing 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 form responses (the long response format will be covered later).

The prompt characters must be followed by a single ASCII character address 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 discussed in full detail later in this chapter. All 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.

Write Protection

Some 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 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

A communications time-out error can occur if the A2400 has not received a command correctly. This may be due to noise on the communications channel, incorrect address, hardware failures, etc. There is certain amount of time after which it can be assumed by the host that a response message will never occur. With the A2400, the response time is variable due to the programmable delays built into the unit. Delay times T1 and T2 must be added to the message times listed in Table 2.0 to calculate the maximum time necessary to respond to a command.

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 A2400. The checksum is included to verify the integrity of the response data. The ‘#’ command prompt may be used with any command. For example:

Command:	**$1RD**	**(short form)**
Response:	***+99999.99**	
Command:	**#1RD**	**(long form)**
Response:	***1RD+99999.99D9**	

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 transmitted to an addressable converter 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:    $1WE               (no checksum)
Response:   *

Command:    #1WEF0             (with checksum)
Response:   *

Command:    $1WEF1             (incorrect checksum)
Response:   ?1 BAD CHECKSUM

Command:    $1WEF              (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:    $1RD                (short form)
Response:   *+99999.99

Command:    #1RD                (long form)
Response:   *1RD+99999.99D9
```

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 $1WE

Characters:	$	1	W	E
ASCII hex values:	24	31	57	45
Sum (hex addition)	24 +	31 +	57 +	45 = F1

The checksum is F1 (hex). Append the characters F and 1 to the end of the message: $1WEF1

Example: Verify the checksum of a module response *1WEF7

The checksum is the two characters preceding the CR: F7

Add the remaining character values:

*	1	W	E		
2A +	31+	57+	45	=	F7

The two lowest-order hex digits of the sum are F7 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.

A2400 User Commands

Table 4.1 shows all the A2400 commands. For each case, a typical command and response is shown. Note that some commands only respond with an * as an acknowledgment. Table 4.1 also separates write protected commands from commands that are not write protected.

For clarity, Table 4.1 separates A2400 Extended Address mode commands from other commands. Note that the Extended Address commands use a different prompt and a two character address.

Each A2400 user command is described in detail following Table 4.1. All of the commands are listed in alphabetical order according to command nomenclature.

Chapter 5
Setup Information/SetUp Command

The A2400 features a wide choice of user configurable options which gives them the flexibility to operate on virtually any radio or leased-line modem. 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 (122 values)
Linefeeds
Parity (odd, even, none)
Baud rate (300 to 115200)
Addressing (Normal, Extended)
Character Filter enable/disable
Alternate RTS
Digital output enable
CTS active low/high
RTS active low/high

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[byte 1][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 examine the existing setup data before proceeding with the SU command.

Byte 1

Byte 1 contains the A2400 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 six ASCII codes that are illegal for use as an address. These codes are $00, $0D, $24, $23, $7B, $7D 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 122 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	7C	|
37	7	4E	N	65	e	7E	~
38	8	4F	O	66	f		
39	9	50	P	67	g		

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 0-3 specify the communications baud rate. The baud rate can be selected from ten values between 300 and 115200 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 accidentally 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 '31070000'. 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:    *31070000
```

Let's say we wish to change the baud rate to 9600 baud. The code for 9600 baud is '0010' (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:     $1SU31020000
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.

Bit 4
Bit 4 of byte 2 is 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**				
57600 BAUD					**1**	**0**	**0**	**1**
115200 BAUD					**1**	**0**	**0**	**0**
38400 BAUD					**0**	**0**	**0**	**0**
19200 BAUD					**0**	**0**	**0**	**1**
9600 BAUD					**0**	**0**	**1**	**0**
4800 BAUD					**0**	**0**	**1**	**1**
2400 BAUD					**0**	**1**	**0**	**0**
1200 BAUD					**0**	**1**	**0**	**1**
600 BAUD					**0**	**1**	**1**	**0**
300 BAUD					**0**	**1**	**1**	**1**

Byte 3
This byte contains determines which addressing mode will be used. The default value for this byte is ‘00’.

Normal addressing
The normal addressing mode refers to the D series protocol of using a single ASCII character for a channel address. There are up to 124 possible addresses in this mode.

Extended addressing
The extended addressing mode expands the number of usable channel addresses to matrix of 124 X 124 for modules by using the curly brace prompt with a single ASCII character address.

Chapter 6
Delay Time Programming

Each A2400 contains user-programmable delays to properly sequence the transmission of data from a remote radio modem to a host computer. The delays are required to sequence an external transmit enable signal required by most radio modems. The external transmit enable signal most often used is the RS-232 Request to Send (RTS) signal.

The RS-232 RTS signal normally becomes active just before data transmission begins, remains active while all data is transmitted and then is immediately turned off. This sequence is essentially the same when using radio modems. However, a certain amount of time is required after a transmit enable signal (RTS) is activated for the radio transmitter to turn on. Transmission of data cannot begin until after the transmitter is settled and ready. Since this period of time may vary from vendor to vendor, user-programmable delays are provided in the A2400. These delays can be set long enough to ensure that the transmitter is on. The delay times are specified using commands from the A2400 command set and are stored in nonvolatile memory.

The A2400 uses three programmable delay times to properly sequence the data applied to the radio modem. These delay times are called T1, T2, and T3.

T1 is used to allow the host transmitter to turn off. T1 may be set to guarantee a specified amount of dead time between the radio modem receive data and the leading edge of the RTS signal which is used to key the transmitter.

T2 is used to allow the radio modem transmitter time to stabilize. T2 is the time between the leading edge of the RTS signal and the beginning of data transfer.

T3 is the delay time between the last data character to be transmitted and the trailing edge of RTS which turns off the transmitter.

The three delay times each have a programmable range of 0 to 2000 ms and are specified using the standard D series data format '+#####.##'. The delay time values may be set to 1ms resolution. Use commands T1, T2 and T3 to specify the delay values. These commands are write protected and must be preceded by a write enable (WE) command.

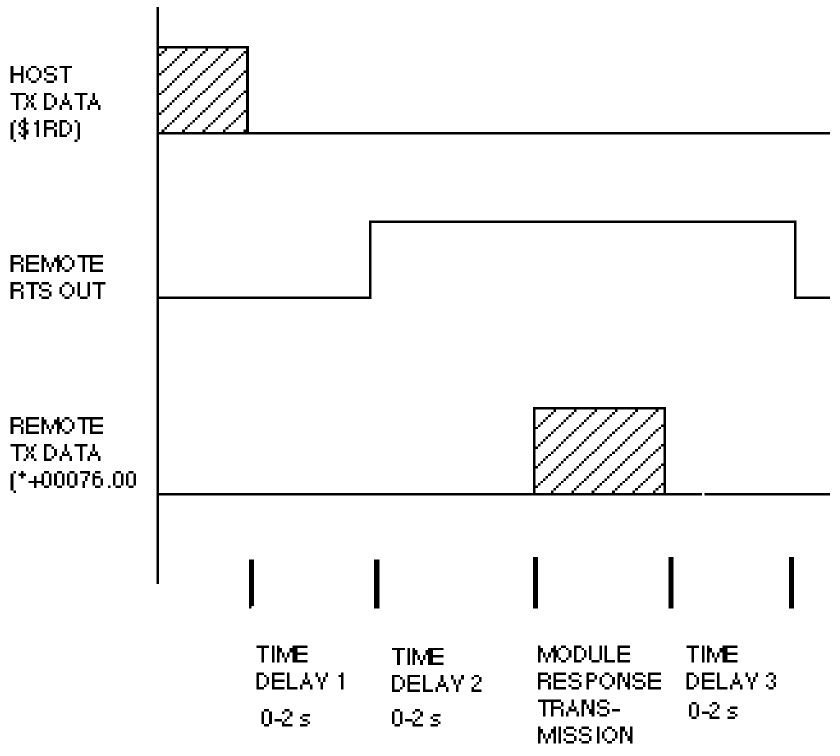

Figure 6.1 Programmable delay times.

As described in figure 6.1 the communications sequence assumes a host computer communicating with a module on the RS-485 bus through the A2400.

In an idle condition, when no data is present on the communications lines, the A2400 turns its RS-485 transceiver to receive mode and monitors activity on the RS-485 bus. When data is detected on the RS-232 input, the A2400 bus arbitrator immediately turns on the RS-485 driver and transmits the data over the RS-485 bus. This data is normally command data being sent to a module on the bus. When the RS-232 command data is complete, the A2400 immediately turns its RS-485 transceiver back to receive mode and monitors the RS-485 bus.

In normal operation the A2400 looks for a D series response prompt character, either a '*' or '?'. When a response prompt is received, the A2400 begins the first delay time , T1. Meanwhile, the A2400 stores the response data from the module until a carriage return character is detected, signifying the end of the response string. The A2400 checks to see if delay time T1 has expired. If not, it waits until the delay is complete. After T1 is done, the A2400 activates the RTS signal to key the radio transmitter. After the RTS signal is activated, delay time T2 is started to allow the transmitter to settle. After the completion of delay time T2, the A2400 then checks for an active (high) CTS line from the radio, then outputs the buffered response data to the RS-232 Transmit line. When the transmit data is complete, the A2400 starts delay time T3. After the completion of delay time T3, the A2400 turns off the RTS signal and is now ready for the next command - response sequence.

Note that T1 only guarantees a minimum turn-around delay time. The actual time is a function of the response time of the module and the amount of time necessary for the radio's transmitter to turn off after the RTS signal becomes inactive.

If for some reason there is no response to a command, no RTS signal is generated and the transmitter will not be keyed on. This allows the module addressing to be spread among several radio modems on the same frequency. For example, radio modem #1 could be connected to modules with addresses of 1, 2, 3, and radio modem #2 could be connected to modules with addresses of 4, 5, 6. The modems can operate on the same frequency without interference since only the correctly addressed modules will return response data.

The commands used to specify the delay times are similar to other D series commands, using the standard format for analog data with the time values scaled in milliseconds:

```
Command:     $1T1+00100.00          (set T1 to 100 ms.)
Response:    *

Command:     $1T2+00350.00          (set T2 to 350 ms.)
Response:    *
```

Command:	**$1T3+00050.00**	**(set T3 to 50 ms.)**
Response:	*****	

Time may be set to 1 ms. resolution.

The T1, T2, T3 commands are write-protected and must be preceded by a Write Enable (WE) command.

The delay times are stored in nonvolatile memory.

The delay times are inactive in Default Mode.

The delay times stored in the A2400 may be read back with RT1, RT2, and RT3 commands:

Command:	**$1RT1**
Response:	***+00100.00**

Chapter 7
Power Supply

A2400 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 A2400 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 A2400 consume a maximum of .5 watts and this figure should be used in determining the power supply current requirement.

In some cases, the A2400 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.

A2400 modules may be locally powered. 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 A2400 modules are protected against power supply reversals.

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.

Chapter 9
Extended Addressing

The A2400 may be configured to a special command format called Extended Addressing. This mode uses a different prompt, either '{' or '}' to distinguish it from the regular command syntax. The major difference in syntax for the Extended Addressing mode is that it uses a two-character address. A typical command in Extended Address mode would look like this:

Command:	**{01WE**
Response:	*****

Both the command and response are terminated with carriage returns. Note that the command uses a two-character address, '01.'

There are two benefits to using Extended Addressing with the A2400:

1) Greatly expanded addressing capability.

2) Allow for a more structured addressing method in large systems.

With single-byte addressing of the normal command structure, address space is limited to 122 points. Extended addressing allows an addressing range of more than 1.8 million points. More realistically, if only printable characters are used for addresses, the total address range is more than 700,000 points.

Open/Close Channel Commands
From an operational standpoint, A2400's with Extended Addressing are used differently than A2400's with normal addressing. An A2400 configured to Extended Addressing acts as a 'data gate' that controls the data that may appear on the RS-485 bus. This 'data gate' may be turned on and off with the Open Channel (OC) and the Close Channel (CC) commands.

Refer to Figure 7.1. This illustrates a small system of two remote radio sites, each with an A2400 and two modules. Note that the two modules have identical addresses at each site. The radio modems are effectively addressed by the A2400's, one with an address of '01' and the other address '02'.

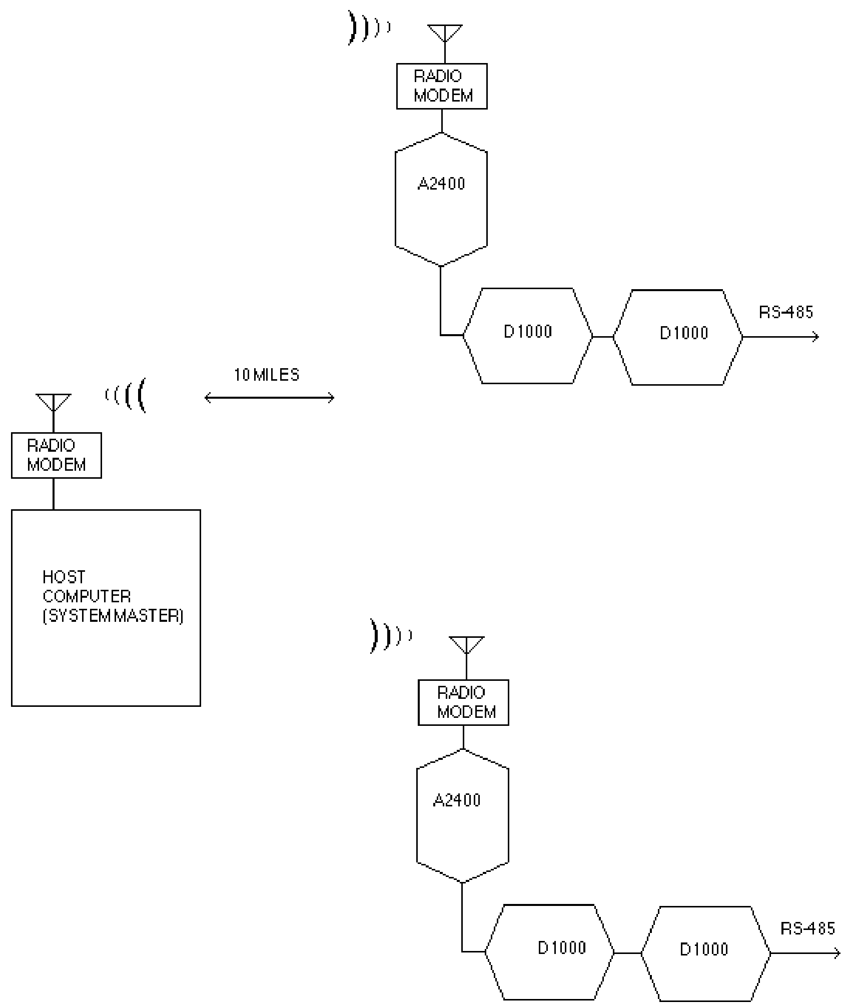

Figure 7.1 Typical system overview.

When the system is initially powered up, the A2400's are set to the Close Channel condition. This means that no data received by the radio modems will pass to the RS-485 bus at either site. In order to communicate to the modules, one of the A2400's must be set to the Open Channel condition:

```
Command:    {01OC
Response:   *
```

This command orders the A2400 address '01' to open the 'data gate' between the radio modem and the two-module string on the RS-485 bus. Now the host computer may communicate to the modules in a normal fashion:

```
Command:    $1RD
Response:   *+00100.00                 (typical)


Command:    $2RD
Response:   *+00123.45                 (typical)
```

Note that A2400 address '02' is still in a Close Channel state. No data received by this A2400 will be allowed to pass to the RS-485 bus. Therefore there is no conflict between the two module strings containing identical addresses. Only one A2400 is allowed to be in an Open Channel state at any time.

To access the module string located at A2400 address '02', first close the channel at A2400 #01:

```
Command:    {01CC
Response:   *
```

Now open the 'data gate' at A2400 #02:

```
Command:    {02OC
Response:   *
```

Any commands from the host are now directed through A2400 #02:

```
Command:    $1RD
Response:   *+00000.00

Command:    $2RD
Response:   *00005.00
```

Figure 7.1 shows a very simple system but the same addressing methodmay be used to construct very large systems. Each RS-485 string may handle up to 122 addresses, and up to 14884 A2400's may have unique addresses.

Structured Addressing
Even for a relatively small system, it can be advantageous to employ a hierarchical addressing system as used in Fig. 7.1. This is particularly true in systems that consist of many sites that are identical. From a host software standpoint, each site can be treated identically with the same module addresses, with each site having a different A2400 address.

Extended Address Syntax
The command syntax used with Extended Addressing is quite similar to the normal protocol. The Extended Address commands are initiated with a '{' character (left curly brace, ASCII $7B), or a '}' character (right curly brace, ASCII $7E). The '{' prompt is analogous to the '$' prompt in that it returns the shortest possible response to complete the command. The '}' prompt is similar to the '#' prompt in that the command is echoed and a checksum is generated along with the other data necessary to complete the response. The '*' response prompt is used in all command forms.

The Extended Address commands use a two-character ASCII address, each character may be one of 122 legal possibilities. Illegal characters are: NULL ($00), CR ($0D), $ ($24), # ($23), { ($7B), and } ($7E).

Command examples with Extended Address '01':

Command:	**{01WE**	
Response:	*****	
Command:	**}01WE**	
Response:	***01WE27**	
Command:	**{01RS**	
Response:	***31070000**	**(typical)**
Command:	**}01RS**	
Response:	***01RS31070000BB**	**(typical)**

Checksums may be appended to commands:

Command:	**{01WE78**
Response:	*****

All commands that are available with single-byte addressing may be accessed with Extended Addressing, and vice-versa; the only exceptions being the OC and CC commands, which can be used only with Extended Addressing.

OC and CC Command Formats

Once an A2400 has been configured correctly in Extended Address mode, the only commands necessary for normal operation are the Open Channel (OC) and Close Channel (CC) commands. Since these commands are used most often, there are several variations available, both implied and explicit, which trade-off speed with data security.

Implied CC Command

In Extended Address mode, the A2400 is programmed to close the ‘data gate’ each time it detects a ‘{‘ or ‘}’ character. This is to safeguard against the possibility of two or more A2400’s being on at the same time, which could result in data collisions. This is done automatically whether the A2400 is addressed or not. No response is generated by an implied CC command.

Implied OC Command

Since the OC command is the most commonly used command in Extended Address mode, it has been assigned to be the default command if only the address is sent as a command:

Command:	**{01**	**({01OC is implied)**
Response:	*****	**(A2400 #01 data path is open)**

Command:	**}01**	**(}01OC is implied)**
Response:	***01OC1D**	

With the implied OC and CC commands, different A2400 channels may be opened and closed with a minimum of command overhead:

Command:	**{01**
Response:	*****

This command opens the data channel of A2400 #01. At the same time, all other A2400’s on the same system detect the ‘{‘ character and interpret it as an implied Close Channel command. All other A2400’s will automatically be set to Close Channel.

Command:	**{02**
Response:	*****

This command will close the channel at A2400 #01 and open the channel at A2400 #02. This is the quickest method of opening and closing A2400 channels. It also offers the least amount of data security. A2400 #01 was closed solely upon detecting the '{' character. There is no confirmation that A2400 #01 is closed. If A2400 #01 missed the '{' character due to noise, it would remain open, with the possibility of two A2400's in the open condition, an undesirable state of affairs. This method has merit in systems with a known clean communications channel and when speed is important.

Another variation of the implied OC command is the long form:

Command:	**}01**
Response:	***01OC1D**

Single Command Module Address
Any module located on an RS-485 string may be individually addressed with one command by appending the module command to an implied OC command:

Command:	**{01$1RD**
Response:	***+00100.00**

In this command form, the {01 is an implied Open Channel command to A2400 #01. The '{' is also an implied Close Channel command to all other A2400's. It is followed immediately by a $1RD command, which is the Read Data command for module #1 connected to A2400 #01. No response results from the {01 portion of the command, but the $1RD data is transmitted on the RS-485 bus. Module #1 responds back with the data *+00100.00.

Again, this command form is a trade-off between speed and data security. It is a convenient command form to communicate with an individual module without explicitly opening or closing an A2400 channel. This could be a good command format to be used with noise-free data channels and where speed is very important.

Explicit Open and Close Channel Commands
A greater level of data integrity may be obtained by using explicit forms of the Open Channel (OC) and Close Channel (CC) commands. This simply means that the 'data gate' is controlled by command sequences that require a confirmation from the addressed A2400.

To explicitly close the channel of A2400 #01:

Command:	**{01CC**	
Response:	*****	

The response message is a confirmation that the channel has been closed. A higher level of confirmation can be obtained with the long form:

Command:	**}01CC**	
Response:	***01CC11**	**('11' is the checksum)**

The response confirms that channel '01' has been closed.

The same level of confirmation can be used with the Open Channel command:

Command:	**{02OC**	
Response:	*****	

Command:	**}02OC**	
Response:	***02OC1E**	**('1E' is the checksum)**

The explicit forms of the OC and CC commands are particularly useful in systems where noise is present in the communications channel and data accuracy is very important. The highest level of data security may be achieved by using parity and command checksums:

Command:	**}02CC65**	**('65' is the checksum)**
Response:	***02CC12**	**('12' is the checksum)**

Chapter 10
Transparent Mode

The A2400 is an RS-232/RS-485 converter designed to provide interface virtually any product to a radio and leased telephone line modems. In normal operation the A2400 is configured to work on the D series protocol, but it may be configured to a special communications mode called the transparent mode. When used in the transparent mode the A2400 module provides an effective radio or leased-line modem interface for equipment that does not use the D series protocol.

In normal operation the A2400 communicates with the host on a polling system; that is, each module responds to its own unique address and must be interrogated by the host. The microprocessor in the A2400 examines any character transmitted to it by the host computer. When the microprocessor detects a valid prompt character ($ or #), it traps the prompt and examines the next characters for a valid address and a valid command. This simple command/response protocol must be strictly observed to avoid communications collisions and data errors.

In the transparent mode the module acts like an open gate for information. Any data presented at the input to the module will be passed on regardless of protocol.

A device that is half duplex and uses a 10-bit NRZ asynchronous data format can be used with the A2400 in the transparent mode. The 10-bit format may consist of 1 start bit, 7 data bits, 1 parity bit, 1 stop bit or 8 data bits without the parity bit.

Transparent Mode Functions
In the transparent mode the A2400 provides the user with the following functions:

1. Ready To Send (RTS) and Clear To Send (CTS) handshaking signals for RS-232 devices to interface to equipment that require those handshaking signals such as radio or lease-line modems.

2. Networking capability for RS-232 devices by providing unique channel addressing capability.

3. RS-485/RS-232 conversion for RS-485 devices to interface to equipment with an RS-232 port such as modems or a host computer.

The following examples 1 through 4 illustrate the application of the A2400 in the extended mode of operation. The A2400 in transparent mode allows equipment from various sources to be added to a network of D series modules. However the following rules must apply:

1. half duplex communications.

2. 10-bit data format.

3. The equipment must work on a command and response protocol and that protocol must not interfere with valid D series prompts ($, #) or the customer programmed prompt of another A2400 module that may be on the same line.

4. In examples where more than several devices are to be on the same line the devices must have their own channel address.

Example 1. A2400 interfacing a device to a radio or leased-line modem

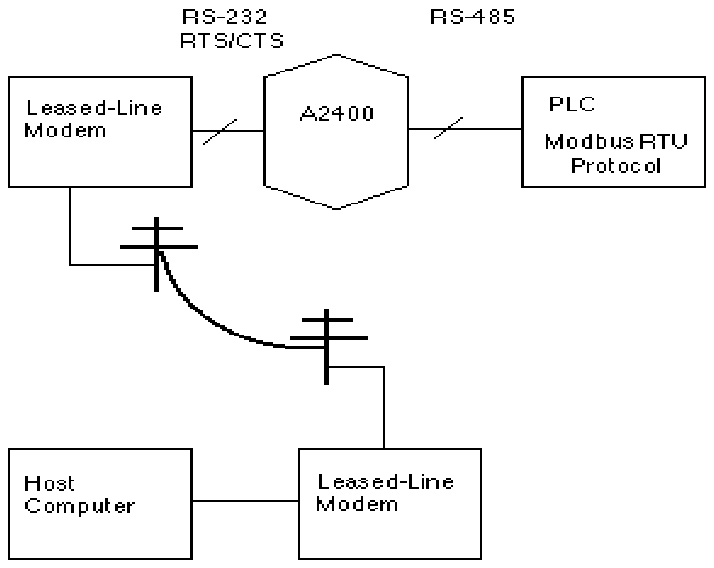

Figure 10.1 A2400 interfacing a PLC to a leased-line modem.

Example 2. A2400 networking several RS-232 devices to a host computer.

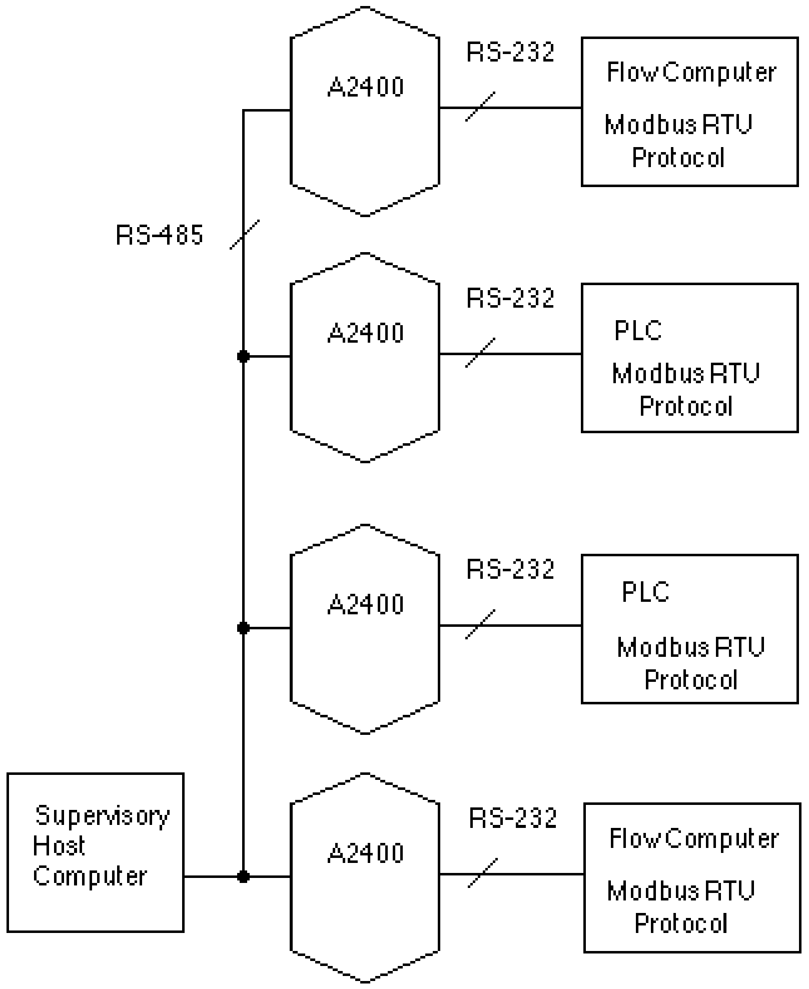

Figure 10.2 A2400 networking several RS-232 devices to a host computer.

Example 3. A2400 interfacing a device with a string of modules.

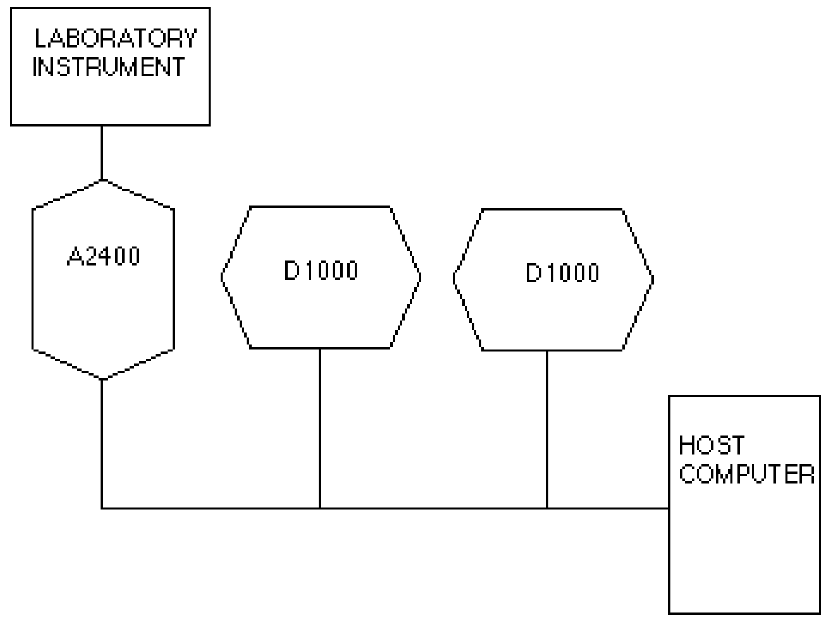

Figure 10.3 A2400 interfacing a laboratory instrument with modules.

Example 4. A2400 interfacing several instruments with a string of modules.

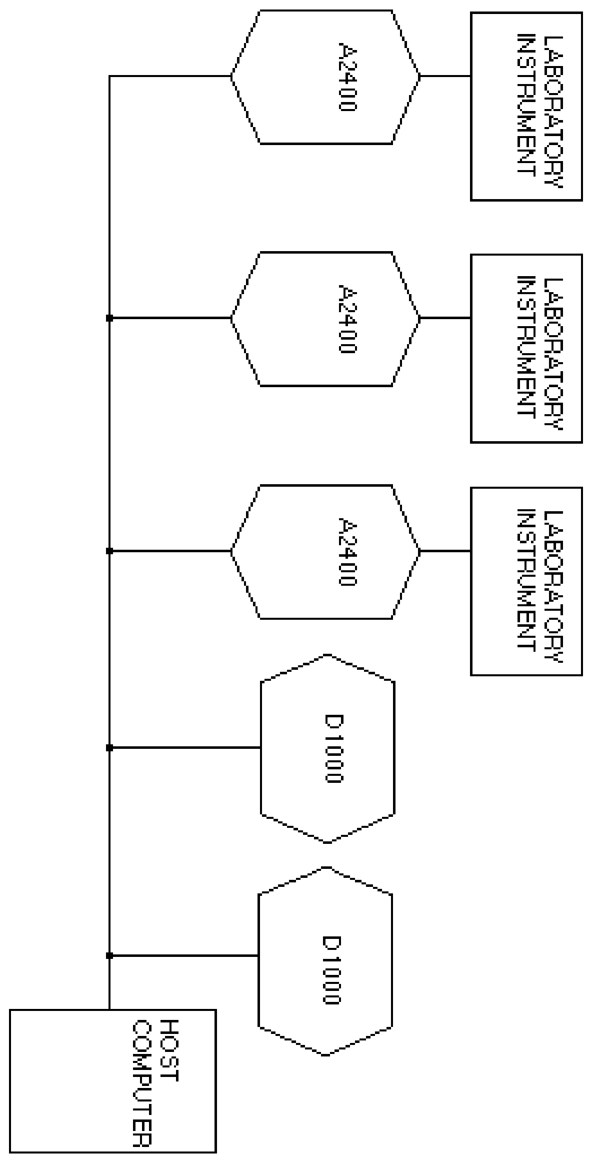

Figure 10.4 A2400 networking several laboratory instruments with modules.

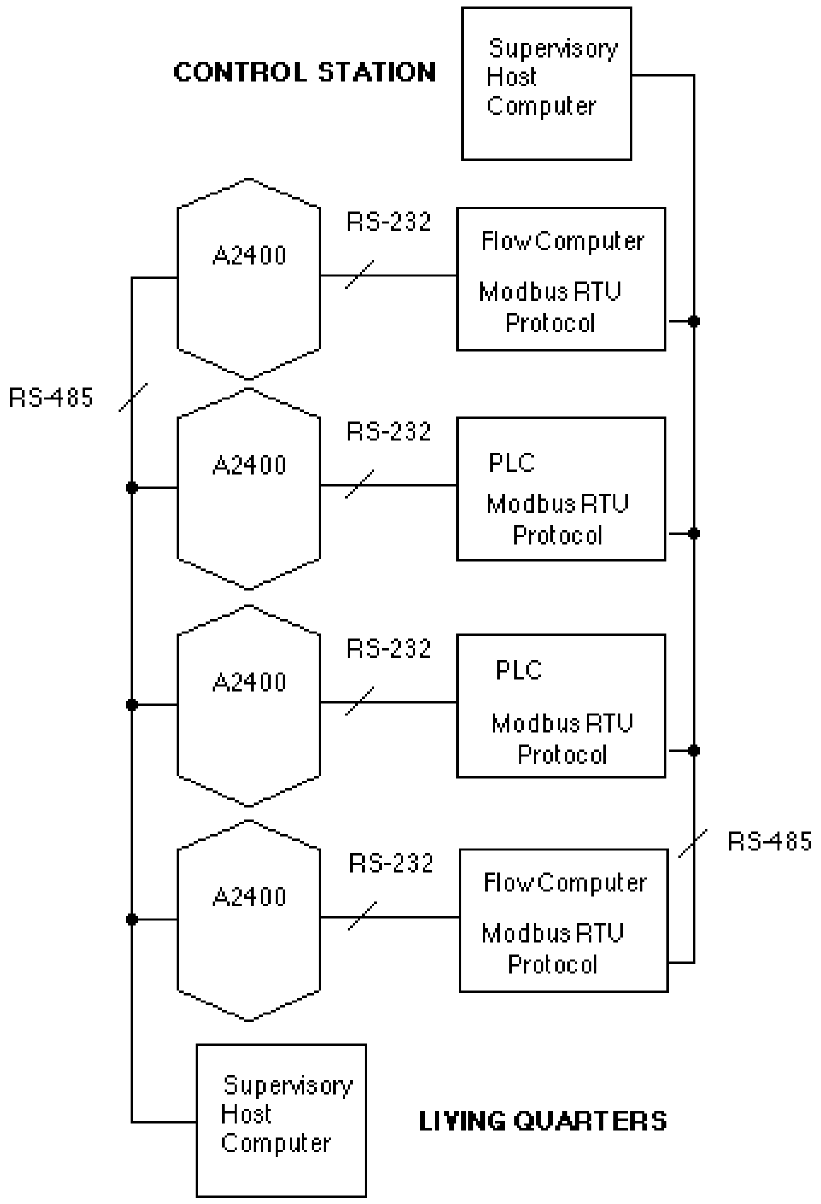

Figure 10.5 Adding secondary control using A2400 modules.

Figure 10.5 shows an application where the user was able to add a back-up or secondary control supervision using the A2400 modules. The original installation of the flow computers and PLC's were controlled by the control station using Modbus protocol with RS-485. The user was able to use A2400 modules on the RS232 ports of the devices and establish a second control loop The flow computers and PLC's had addressing capability.

TheThe transparent mode can only be enabled or disabled via the module's setup message using the SetUp command. The transparent mode can be enabled while the module is operating in either the normal or the extended addressing mode.

The TRANSPARENT mode has to be set up manually or with the DGH utility software.

Command:	**$1RS**	
Response:	***31070000**	**(Factory set-up)**
Command:	**$1WE**	
Response:	*****	
Command:	**$1SU31070030**	

Disables Character Filter and enables TRANSPARENT mode. (Bit 4 & 5 of Byte 4 set to 1)

Disabling the transparent mode requires that the A2400 module first be put into the Default Mode (connecting the Default* pin to ground). In Default Mode the module is put in a known communication setup: 300 baud, no parity, any address is recognized. Once communications is established the module's Setup Message can be changed. The transparent mode is disabled by changing bit 5 in byte 4 to a 0. (Example: $1SU31070000) or use SETUP in the utility software.