SCM9B-3000/4000 SERIES USERS MANUAL

REVISED: 6/1/94

DATAFORTH CORPORATION
3331 EAST HEMISPHERE LOOP
TUCSON, AZ 85706

TELEPHONE: 520-741-1404

SCM9B-3000/4000 modules are not intrinsically safe devices and should not be used in an explosive environment unless enclosed in approved explosion-proof housings.

TABLE OF CONTENTS

WARNING
The circuits and software contained in SCM9B-3000 and SCM9B-4000 series 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 SCM9B-3000/4000 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. DATAFORTH is not responsible for any consequential damages.

Chapter 1
Getting Started

Introduction

The SCM9B-3000/4000 series are completely self-contained computer-to-analog output interfaces. They are designed to be mounted remotely from a host computer and communicate with standard RS-232 and RS-485 serial ports. Simple ASCII commands are used to control a 12-bit DAC (Digital-to-Analog Converter) which is scaled to provide commonly used current and voltage ranges. An on-board microprocessor is used to provide the communications interface and many intelligent analog output functions.

SCM9B-3000 versions provide a basic computer-to-analog output interface for cost-sensitive applications where some of the intelligent enhancements are not required. SCM9B-3000 units feature step-function outputs, fixed input scaling, and no analog readback.

SCM9B-4000 versions perform all of the SCM9B-3000 functions plus many additional intelligent enhancements:

Controlled output slew rates
True analog readback
Programmable input data scaling
Programmable starting value
Watchdog timer

This manual has been written to be a guide for both the SCM9B-3000 and SCM9B-4000 units. Basic operating characteristics of both models are identical and unless otherwise noted, the information in this manual applies to both versions. Commands and functions exclusive to the SCM9B-4000 are so noted in the text.

Terminal Designations

All SCM9B-3000 and SCM9B-4000 units have similar terminal designations, although there are slight variations between current/voltage and RS-232/RS-485 models.

Pin 1	+I OUT or +V OUT
Pin 2	-I OUT or -V OUT

Pins 1 and 2 are the connections to the analog output signal. On voltage models, the input data is scaled so that the voltage at +V OUT is positive with respect to -V OUT. Voltage outputs can source or sink current.

On current output models, the output current flows from the +I OUT terminal to the -I OUT terminal, so for a typical resistive load, the +I OUT terminal will be at a more positive voltage level than the -I OUT terminal. Current outputs can only source current.

Pins 1 and 2 are electrically isolated from the other pins.

Pin 3	DI2
Pin 4	DI1/UP*
Pin 5	DI0/DN*

Pins 3-5 are digital input pins. They may be used as general-purpose inputs or they may be set-up to provide special functions that control the analog output. The standard factory set-up configures the UP* and DN* pins to provide manual up and down control of the analog output. The * designation indicates that the labels are negative true. A full functional description of these pins may be found in Chapter 6.

Pin 6	DEFAULT*

Grounding this pin places the module in Default Mode, described in detail below.

Pin 7	TRANSMIT or DATA
Pin 8	RECEIVE or DATA*

Pins 7 and 8 are connections to the serial communications lines connecting the module to the host computer or terminal.

On RS-232 models, the TRANSMIT pin is the serial output connection from the module. The RECEIVE pin is the serial input into the module.

On RS-485 versions, DATA and DATA* are connections to the balanced RS-485 communications lines. DATA and DATA* are sometimes labeled DATA+ and DATA- respectively.

Pin 9	V+
Pin 10	GND

Pins 9 and 10 are the power connections. The SCM9B-3000/4000 modules operate on 10-30V unregulated power.

Default Mode

All SCM9B-3000/4000 modules contain an EEPROM (Electrically Erasable Programmable Read Only Memory) to store setup information and calibration constants. The EEPROM replaces the usual array of switches and pots 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 and pot settings. However, 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 labeled 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. 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 SCM9B-3000/4000 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.

Connect a suitable voltmeter or ammeter to the output connections of the module to monitor the output signal. If an ammeter is not available to measure the signals from current-output modules, a sense resistor and a voltmeter may be used as shown in Fig. 1.1. Turn power on to the module. Momentarily ground the UP* pin on the connector. The output signal should increase in value as the UP* pin is held low. Now release the UP* pin and ground the DN* (down) pin. The output signal should decrease in value as the pin is held low.

This demonstrates the “Manual Mode” method of controlling the output. It is also a quick check to see if the module is connected and working properly.

Use your terminal to type the command $1RD and terminate the command with a carriage return. The module will respond with an * followed by the output data reading. The data includes sign, seven digits and a decimal point. For example, a typical reading might be *+00015.00. This is an output status reading and it should closely correspond with the reading on your meter.

Now type the command:

$1AO+00010.00 and terminate with a carriage return.

The module should respond with ‘*’ and the output will change to 10 millivolts (or milliamps). This demonstrates the Analog Output (AO) command, which is the primary method of controlling the analog output.

If you have a voltage output module,try these commands:

$1AO+00000.00 (terminate with a carriage return)

$1AO+00100.00

$1AO+01000.00

$1AO+01234.00

For current output models these commands are more appropriate:

$1AO+00004.00

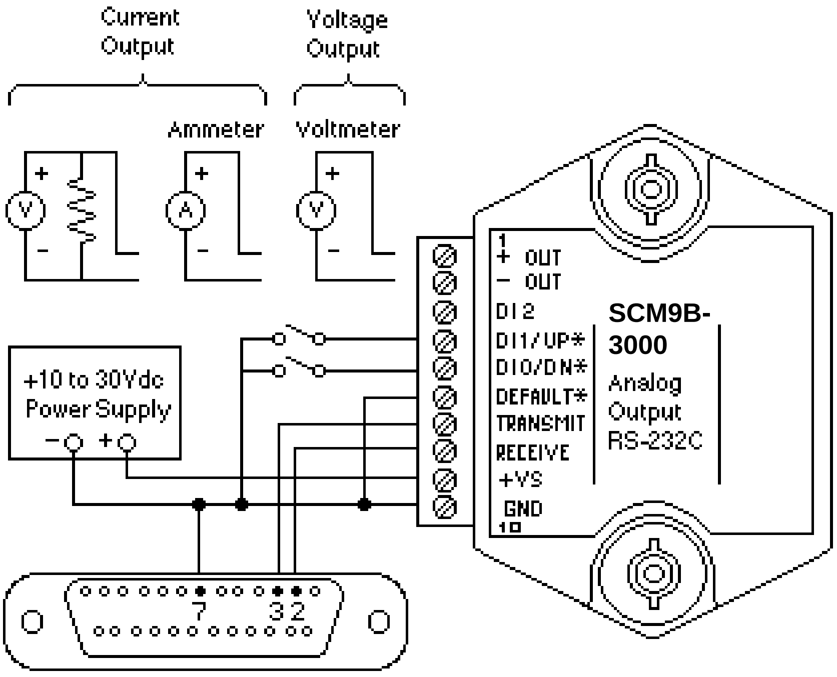
$1AO+00020.00

$1AO+00010.00

Remember to terminate each command with a carriage return.

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, limits off, no echo and two-character delay. Refer to the Chapter 5 to configure the module to your application.



Note: If using a DB-9 connector ground is tied to pin 5 only. Pin 2 is tied to TRANSMIT and pin 3 is tied to RECEIVE on the module.

Figure 1.1 SCM9B-3000/4000 RS-232C 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-232C output.

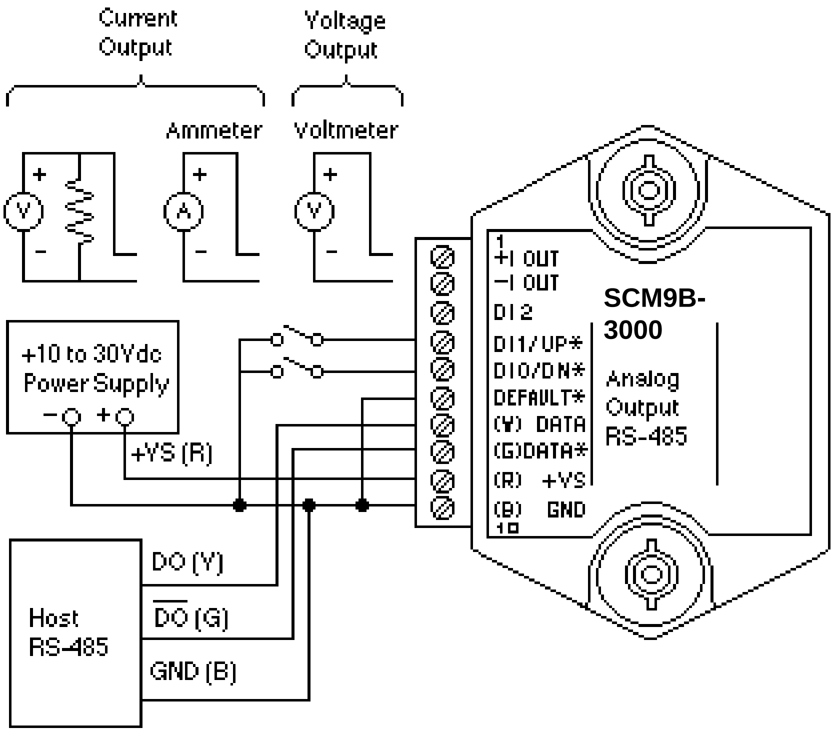
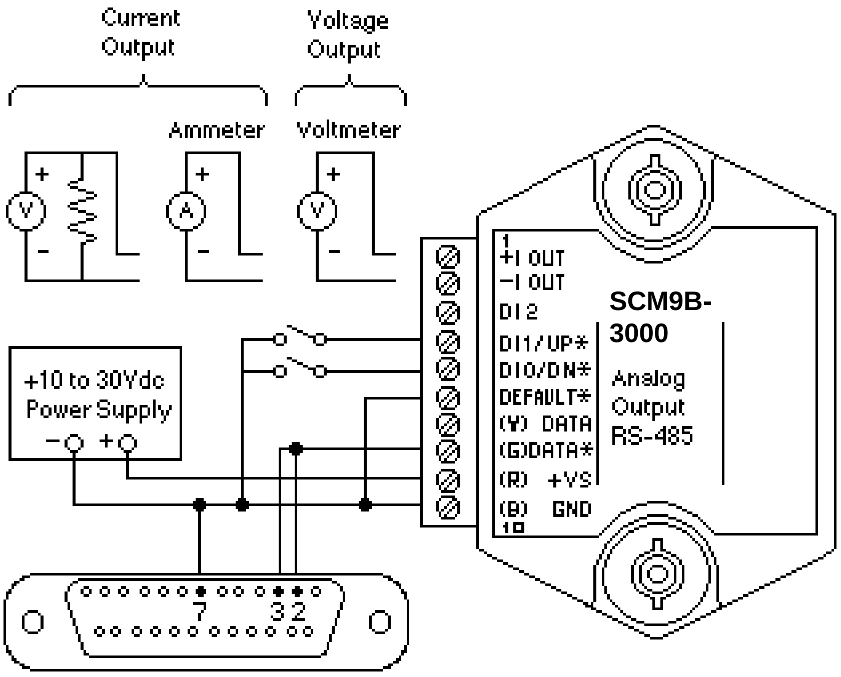


Figure 1.2 SCM9B-3000/4000 RS-485 Quick Hook-up.



Note: If using a DB-9 connector ground is tied to pin 5 only.

Figure 1.3 SCM9B-3000/4000 RS-485 Quick Hook-up with RS-232C Port.

Chapter 2
Functional Description

The SCM9B-3000/4000 Computer to Analog Output interfaces provide accurate analog process control signals in response to simple digital commands from a host computer. The SCM9B-3000/4000 units are completely self-contained and are designed to be operated remotely from the host. Digital commands are transmitted to the SCM9B-3000/4000 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 shows a functional block diagram of the SCM9B-3000/4000. The key block is the 12-bit Digital to Analog Converter (DAC). The DAC converts digital data derived from host commands into the desired analog output. All of the other components provide a supporting role for proper operation of the DAC.

An 8-bit CMOS microprocessor is used to provide an intelligent interface between the host and the DAC. 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 either RS-232 or RS-485 communications standards. 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 control the DAC, read status information, and to configure the module to fit the user's requirements. Responses to the host commands are then produced by the microprocessor and transmitted back to the host over the RS-232/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 calibration data.

In response to host commands, the microprocessor produces the appropriate digital data necessary to control the DAC. Digital data is transmitted to the DAC through opto-isolators which provide electrical isolation. The DAC produces a precise analog current that is directly proportional to the magnitude of the digital data. The DAC output current is then processed and amplified by signal conditioning circuits to produce the desired output voltage or current. Output protection circuits are included to protect the module from potentially damaging output faults.

SCM9B-4000 models also feature a simple Analog to Digital Converter (ADC) which is used to monitor the output signal. The ADC input is tied directly to the analog output and converts the signal level to digital data. The digital data is optically isolated and may be read by the microprocessor. This circuitry allows the SCM9B-4000 user to directly monitor the output signal and ensure its integrity.

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. It also produces ±15 volts to power the DAC and associated output circuitry.

The power supplied to the DAC and output circuitry is transformer isolated from the input power and communications connections. The transformer along with the opto-isolators provide an isolation barrier between the output section and the rest of the circuitry. The isolation barrier is extremely helpful in breaking ground loops and isolating troublesome common-mode voltages that are often found in large systems. The isolation barrier also provides damage protection for the module and the host in cases where the output lines may accidently contact AC power lines.

The combination of an accurate high-resolution DAC and a dedicated microprocessor produces a very powerful system for the generation of process control signals. The power of the microprocessor is used to provide software addressing for multidrop capability, data formatting in engineering units, limit checking, digital calibration, and a host of other features not possible with unintelligent analog output systems.

During normal operation, the microprocessor constantly updates the DAC data at a rate of 1000 times per second, even if the output is stable. The SCM9B-4000 fully utilizes this characteristic to provide controlled output slew rates. Linear output ramp signals are created by incrementally stepping the DAC every millisecond with values precisely calculated by the microprocessor. The small output steps created at millisecond intervals are used to approximate ramp outputs. Slope rates are programmable and may be changed at any time with simple commands. Linear ramps may be initialized with a single command from the host computer. No further intervention or monitoring is required from the host; the SCM9B-4000 does the rest.

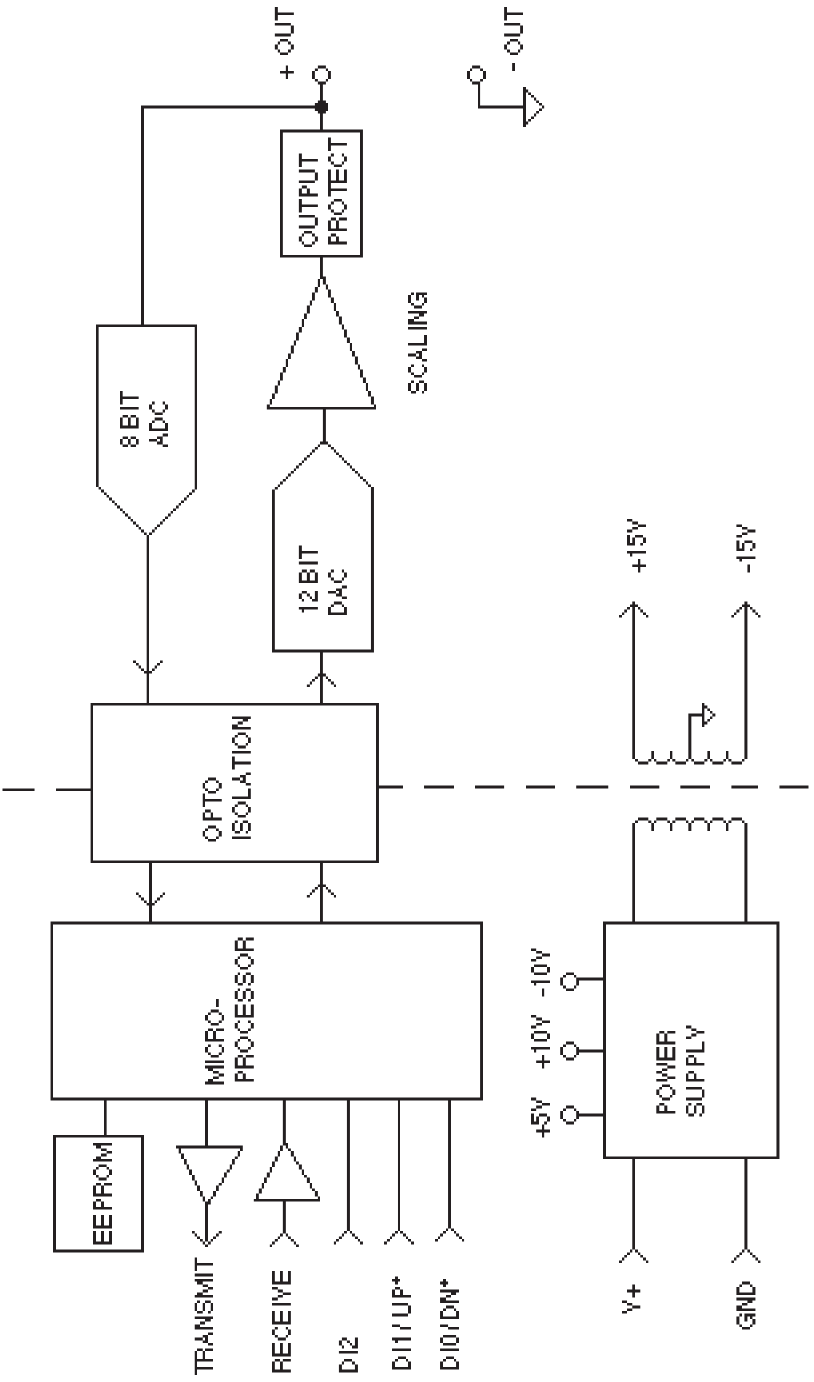


Figure 2.1 Block Diagram.

Chapter 3
Communications

Introduction

The SCM9B-3000/4000 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.

Communications to the SCM9B-3000/4000 modules are performed with two or three-character ASCII command codes such as RD to Read Data from the analog 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: ***+00123.00**

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
DI, HX, WE	3mS
ID	130mS
All other commands	35 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 SCM9B-3000/4000 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, the SCM9B-3000/4000 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–daisy-chain connection
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 multiparty system; however the SCM9B-3000/4000 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.

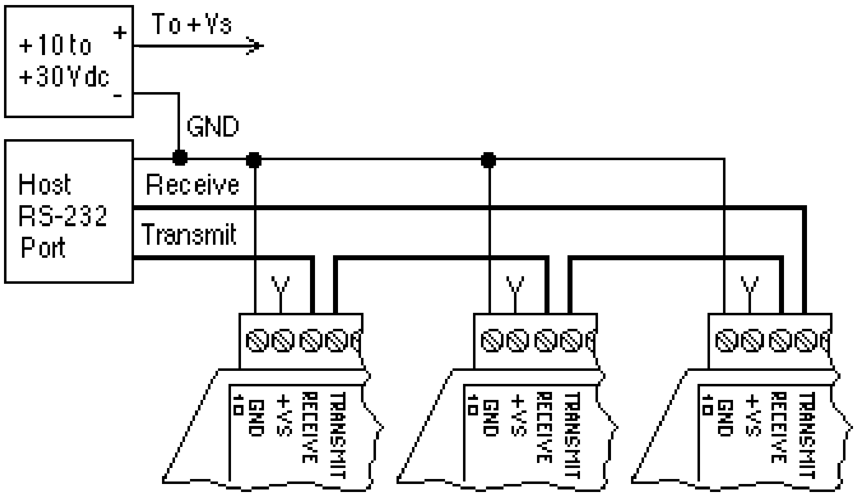


Figure 3.1 RS-232 Daisy-Chain Network.

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 breaks 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 SCM9B-3000/4000 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 retransmitting 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) can disconnect modules without losing communications
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 the A1000 and A2000 interface converters that will convert RS-232 signals to RS-485 or repeat RS-485 signals. The A1000 converters also include a +24Vdc, one amp power supply for powering SCM9B-1000 series modules. The A1000 or A2000 connected as an RS-485 repeater can be used to extend an existing RS-485 network or connect up to 124 modules

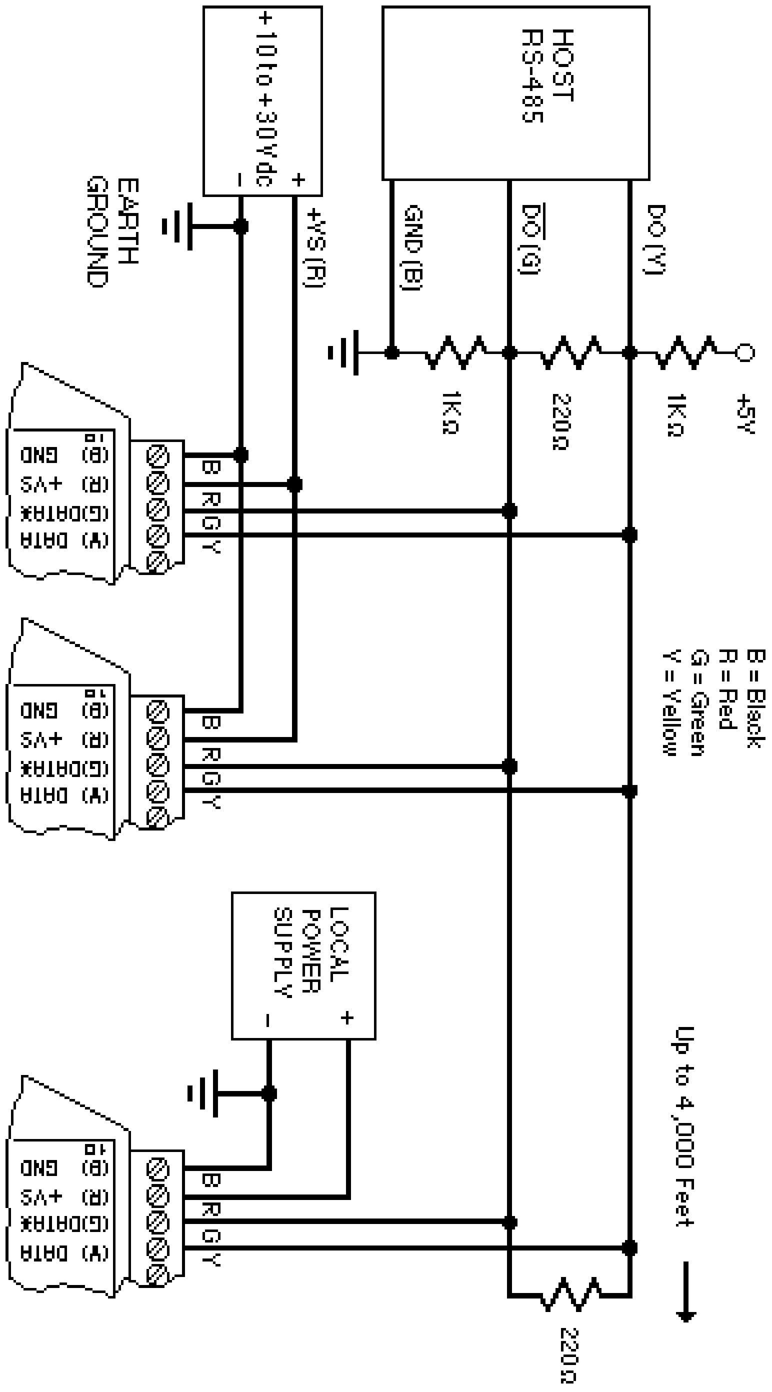


Figure 3.2 RS-485 Network.

on one serial 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. When using long transmission lines and/or high baud rates, the data lines should be terminated at each end with 200 ohm resistors. Standard values of 180 ohms or 220 ohms are acceptable.

During normal operation, there are periods of time where all RS-485 drivers are off and the communications lines are in an 'idle' high impedance condition. During this condition, the lines are susceptible to noise pickup which may be interpreted as random characters on the communications line. To prevent noise pickup, all RS-485 systems should incorporate 1K ohm bias resistors as shown in Figure 3.2. The resistors will maintain the data lines in a 'mark' condition when all drivers are off.

The A1000 series converter boxes have the 1KΩ resistors built-in.

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.

Chapter 4
Command Set

The SCM9B-3000/4000 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. 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 (will be covered 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. 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 that 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. Commands must be transmitted as upper-case characters.

A two-character checksum may be appended to any command message (except the ID command) as a user option. See ‘Checksum’ later in this chapter .

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 most common type of data used in commands and responses is analog data. Analog data is always represented in the same format for all models in the SCM9B-3000/4000 series. Analog data is represented as a nine-character string consisting of a sign, five digits, decimal point, and two additional digits. The string represents a decimal value in engineering units. Examples:

```
+12345.68
+00100.00
-00072.10
-00000.00
```

When using commands that require analog data as an argument, the full nine-character string must be used, even if some digits are not significant. Failure to do this results in a SYNTAX ERROR.

Analog data responses from the module will always be transmitted in the nine-character format. This greatly simplifies software parsing routines since all analog data is in the same format for all module types.

In many cases, some of the digits in the analog data may not be significant. For instance, in the SCM9B-3151 0 to 20mA output module, the data is scaled in milliamps. The full scale output is +00020.00mA. The left three digits have no significance. However, the data format is always adhered to in order to maintain compatibility with other module types.

The maximum computational resolution of the module is 16 bits, which is less than the resolution that may be represented by an analog data variable. This may lead to round-off errors in some cases. For example, a limit value may be stored in a SCM9B-3000/4000 module using the 'HI' command:

```
Command:    $1HI+12345.67
Response:   *
```

The limit value is read back with the Read HIgh (RHI) command:

```
Command:    $1RHI
Response:   *+12345.60
```

It appears that the data read back does not match the value that was originally saved. The error is caused by the fact that the value saved exceeds the computational resolution of the module. This type of round-off error only appears when large data values saved in the module's EEPROM are read back. In most practical applications, the problem is non-existent.

The Digital Input, Hex Output and Setup commands use hexadecimal representations of data. The data structures for these commands are detailed in the command descriptions.

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. They 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 be 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 20 printable characters. If a properly addressed module receives a command message of more than 20 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 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 ' * ' 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 20 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 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 responds 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 when 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:	**$1RD**	**(short form)**
Response:	***+00072.10**	
Command:	**#1RD**	**(long form)**
Response:	***1RD+00072.10A4**	**(A4=checksum)**

Checksum
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 (except 'ID') to the 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 calculates the checksum for the message. If the calculated checksum does not agree with the transmitted checksum, the module responds with a ‘BAD CHECKSUM’ error message and the command is aborted. If the checksums agree, the

command is executed. If the module receives a single extra character, it responds with 'SYNTAX ERROR' and the command is aborted. For example:

Command:	**$1RD**	**(no checksum)**
Response:	***+00072.10**	
Command:	**$1RDEB**	**(with checksum)**
Response:	***+00072.10**	
Command:	**$1RDAB**	**(incorrect checksum)**
Response:	**?1 BAD CHECKSUM**	
Command:	**$1RDE**	**(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:	***+00072.10**	
Command:	**#1RD**	**(long form)**
Response:	***1RD+00072.10A4**	**(A4=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 #1HX07FF

Characters:	#	1	H	X	0	7	F	F
ASCII hex values:	23	31	48	58	30	37	46	46

Sum (hex addition) 23 + 31 + 48 + 58 + 30 + 37 + 46 + 46 = 1E7

The checksum is E7 (hex). Append the characters E and 7 to the end of the message: #1HX07FFE7.

Example: Verify the checksum of a module response *1RD+00072.10A4

The checksum is the two characters preceding the CR: A4

Add the remaining character values:

```
 *    1    R    D    +    0    0    0    7    2    .    1    0
2A + 31 + 52 + 44 + 2B + 30 + 30 + 30 + 37 + 32 + 2E + 31 + 30 = 2A4
```

The two lowest-order hex digits of the sum are A4 which agrees with the transmitted checksum.

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.

SCM9B-3000/4000 User Commands

Table 4.1 shows all the SCM9B-3000/4000 commands. For each case, a typical command and response is shown. Note that some commands only respond with an * as an acknowledgment. For clarity, Table 4.1 separates D4000 commands from the commands that are common to both the SCM9B-3000 and SCM9B-4000. Table 4.1 also separates write protected commands from commands that are not write protected.

Each SCM9B-3000/4000 user command is described in detail following Table 4.1. All of the commands are listed in alphabetical order according to command nomenclature. Commands that are exclusive to the SCM9B-4000 are noted near the right hand margin. For example:

Manual Slope (MS) **(SCM9B-4000)**

Table 4.1 SCM9B-3000/4000 Command Set

Command	Definition	Typical Command Message	Typical Response Message
SCM9B-3000/4000 Commands			
ACK	Acknowledge	$1ACK	*
AO	Analog Output	$1AO+00020.00	*
DI	Digital Input	$1DI	*0007
HX	Hex Output	$1HX0FFF	*
RAO	Read Analog Output	$1RAO	*+00017.50
RD	Read Data	$1RD	*+00012.34
RHI	Read High Limit	$1RHI	*+00020.00
RID	Read Identification	$1RID	* B O I L E R
RLO	Read Low Limit	$1RLO	*+00000.00
RMS	Read Manual Slope	$1RMS	*+00004.00
RMX	Read Maximum	$1RMX	*+00020.00
RMN	Read Minimum	$1RMN	*+00000.00
RS	Read Setup	$1RS	*31070140
RSU	Read Setup	$1RSU	*31070140
WE	Write Enable	$1WE	*
The following SCM9B-3000/4000 commands are Write Protected			
HI	High Limit	$1HI+00015.00	*
ID	Identification	$1IDBOILER	*
LO	Low Limit	$1LO+00004.00	*
RR	Remote Reset	$1RR	*
SU	Setup	$1SU310701C0	*
TMX	Trim Maximum	$1TMX+00020.17	*
TMN	Trim Minimum	$1TMN+00000.95	*
SCM9B-4000 Commands			
RAD	Read Analog Data	$1RAD	*+00012.34
RPS	Read Present Slope	$1RPS	*+00001.00
RSL	Read Slope	$1RSL	*+00001.00
RSV	Read Starting Value	$1RSV	*+00005.00
RWT	Read Watchdog Timer	$1RWT	*+00010.00
The following SCM9B-4000 commands are Write Protected			
MS	Manual Slope	$1MS+00001.00	*
MX	Maximum	$1MX+00100.00	*
MN	Minimum	$1MN-00025.00	*
SL	Slope	$1SL+00001.00	*
SV	Starting Value	$1SV+00004.00	*
TRX	Trim Readback Maximum	$1TRX	*
TRN	Trim Readback Minimum	$1TRN	*
WT	Watchdog Timer	$1WT+00010.00	*

WSL	Write Slope To EEPROM	$1WSL+00100.00	*

Acknowledge (ACK)

The ACKnowledge command is a hand-shaking command used in conjunction with the Analog Output (AO) command. It is used to confirm the data sent to a module. See the Analog Output (AO) command for examples of ACK usage.

```
Command:     $1ACK
Response:    *

Command:     #1ACK
Response:    *1ACK2A
```

Analog Output (AO)

The Analog Output (AO) command is the primary command used to control the analog output, whether it is current or voltage. The AO command can function in two different ways, depending on whether the '$' or the '#' command prompt is used. In either case the analog output is specified in the standard 7-digit data format:

```
Command:     $1AO+00010.00
Response:    *
```

If the analog output is scaled in milliamps, this particular command will direct the SCM9B-3000 to produce 10mA. In this example, the '$' command prompt is used to obtain an analog output immediately after the command is received by the module. The module performs the output function and responds with a '*' to provide a simple acknowledgement that the command has been executed.

The '#' form of the AO command requires the host to verify and acknowledge the command data before the module will execute the command. The data is acknowledged by the host with the ACKnowledge (ACK) command. Here is a typical command sequence:

```
Command:      #1AO+00010.00
Response:    *1AO+00010.0095
```

The host command is echoed back along with a checksum as is true with any command when used with the '#' command prompt. At this point the module has not performed the AO command. It is waiting for the host to acknowledge the command by sending an ACK command. This allows the host to examine the command as received by the module and verify that the data is correct. If the host is satisfied that the command data and the checksum are correct, it directs the module to go ahead and perform the AO by sending the ACK command. To complete the sequence:

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

At this point the AO command will be performed by the module.

If the host determines that the data is not correct, it may abort the handshaking sequence by sending any valid command to the module (except for the ACK command of course). Example:

Command:	**#1AO+00010.00**
Response:	***1AO+00030.0097**

In this case, the host examines the response data and determines that a communications error must have occurred since the response data does not match the command data. The command sequence may be aborted by simply sending a new AO command:

Command:	**#1AO+00010.00**
Response:	***1AO+00010.0095**

This time the host verifies that the data is correct and commands the module to complete the task:

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

Only at this point will a change occur on the analog output.

The output data specified in the AO command must lie within the input range of the module or else the command is aborted and the module will respond with a LIMIT ERROR message. The input range may be checked using the Read MiNimum (RMN) and Read MaXimum (RMX) commands. This is a typical command/response sequence that may be generated with a SCM9B-3252 0-20mA module:

Command:	**$1RMN**	
Response:	***+00000.00**	**(this is the lower range limit)**
Command:	**$1RMX**	
Response:	***+00020.00**	**(this is the upper range limit)**
Command:	**$1AO+00025.00**	
Response:	**?1 LIMIT ERROR**	**(the input range has been exceeded)**
Command:	**$1AO+00015.00**	

Response: ***** **(data is within range)**

The data in the AO command is also checked against user-defined limits specified by the LO and HI commands. Exceeding the user-defined limits will generate a LIMIT ERROR. (See LO and HI commands).

Any of the Manual Modes has priority over the AO command, and in some cases a MANUAL MODE error may be generated. See Manual Mode section for details.

Digital Input (DI)

The DI command reads the status of the digital inputs and the status of the analog output. The response to the DI command is four hex characters representing two bytes of data. The first byte contains the analog output status. The second byte contains the digital input data.

Command:	**$1DI**
Response:	***0003**

Command:	**#1DI**
Response:	***1DI0003AB**

The first response byte gives the status of the analog output on SCM9B-4000 units with controlled-rate outputs:

00	The output is steady-state.
01	Indicates the output is still slewing

The second byte displays the hex value of the digital input data.

Digital Inputs	DI2	DI1/UP*	DI0/DN*
Data Bits	2	1	0

All other bits read back as ‘0’

For example: A typical response from a $1DI command could be: *0107. This response indicates that the output is still slewing and all digital inputs are = 1.

The DI command will return the state of the digital inputs even if one of the Manual Modes is in effect.

When reading digital inputs with a checksum, be sure not to confuse the checksum with the data.

Hex Output (HX)

The HeX Output (HX) command controls the analog output by sending hexadecimal data directly to the Digital to Analog Converter (DAC). The SCM9B-3000/4000 uses a 12-bit DAC with inputs ranging from $0000 (- full scale) to $0FFF (+ full scale) . The HX command uses this data to control the DAC:

Command:	**$1HX07FF**
Response:	*
Command:	**#1HX07FF**
Response:	***1HX07FFEE**

This command will set the DAC to half scale. The leading zero is included to allow for future enhancements.

The HX command controls the DAC directly without checking limits, scaling, or trims. It is used by the factory for test purposes. However, it may be used in control situations where the absolute output value is relatively unimportant. The primary attribute of the HX command is speed, since it is not encumbered by the computation necessary for the AO command.

High Limit (HI)

The HIgh Limit (HI) command sets a maximum limit to the analog output data. The data specified by the HI command is stored in nonvolatile memory and it is compared to the data specified by any subsequent Analog Output (AO) commands. If the AO data exceeds the HI limit, the AO command is aborted and the module will generate a LIMIT ERROR message.

Command:	**$1HI+00015.00**
Response:	*****
Command:	**#1HI+00015.00**
Response:	***1HI+00015.009B**

In each of the two command examples, the HI limit has been set to 15 (milliamps, millivolts, or other units). If an attempt is made to exceed this limit with an Analog Output (AO) command, a LIMIT ERROR will result and the AO command is aborted.

Command:	**$1AO+00016.00**
Response:	**?1 LIMIT ERROR**

The HI command and its complement, the LOw Limit (LO) command restrict the range of analog outputs that may be obtained with the Analog Output (AO) command. This is useful in applications where unrestricted outputs may cause damage or improper operation of other equipment or processes.

The HI limit may be effectively disabled by setting it to it's highest value:

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

The HI data may be read back with the Read HI (RHI) command.

The HI command is write protected and must be preceded with a Write Enable (WE) command.

The HI limit will not restrict outputs produced by the HeX Output (HX) command or the Manual Mode inputs.

In SCM9B-4000 applications,the HI data is not affected by the MiNimum (MN) and MaXimum (MX) commands. If the input range is rescaled, the HI data must be changed to an appropriate value.

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 3251              (model number)
Response:   *
```

The ID command is write-protected.

Caution: Command checksums are not supported by the ID command. Messages longer than 16 characters will abort the command.

LOw Limit (LO)
The LOw Limit (LO) command sets a minimum limit to the analog output data. The data specified by the LO command is stored in nonvolatile memory and it is compared to the data specified by any subsequent Analog Output (AO) commands. If the AO data is less than the LO limit, the AO command is aborted and the module will generate a LIMIT ERROR message.

Command:	**$1LO+00004.00**
Response:	*****

Command:	**#1LO+00004.00**
Response:	***1LO+00004.00A3**

In each of the two command examples, the LO limit has been set to 4 (milliamps, millivolts, or other units). If an attempt is made to exceed this limit with an Analog Output (AO) command, a LIMIT ERROR will result and the AO command is aborted.

Command:	**$1AO+00002.00**
Response:	**?1 LIMIT ERROR**

The LO command and its complement, the HIgh Limit (HI) command restrict the range of analog outputs that may be obtained with the Analog Output (AO) command. This is useful in applications where unrestricted outputs may cause damage or improper operation of other equipment or processes.

The LO limit may be effectively disabled by setting it to it’s lowest value:

Command:	**$1LO-99999.99**
Response:	*****

The LO data may be read back with the Read LO (RLO) command.

The LO command is write-protected and must be preceded with a Write Enable (WE) command.

The LO limit will not restrict outputs produced by the HeX Output (HX) command or the Manual Mode inputs.

In SCM9B-4000 applications, the LO data is not affected by the MiNimum

(MN) or MaXimum (MX) scaling commands. If the input range is rescaled, the LO data must be changed to an appropriate value.

Manual Slope (MS) (SCM9B-4000)

The Manual Slope (MS) command sets the output slew rate for manual control using the UP* and DN* (down) input pins. The slope data is scaled in either mA/S or V/S:

```
Command:    $1MS+00004.00
Response:   *

Command:    #1MS+00004.00
Response:   *1MS+00004.00A8
```

These command examples set the manual slew rate to 4mA/S or 4V/S.

The manual slope value only controls the output slew rate when using the manual UP* and DN* inputs. Output changes caused by the Analog Output (AO) command are controlled with slew rates specified by the SLope (SL) or Write SLope (WSL) commands. Therefore, manual and computer-controlled outputs have separate slew rate controls.

The manual slope value may be read back with the Read Manual Slope (RMS) command.

The MS command is write-protected.

Maximum (MX) (SCM9B-4000)
Minimum (MN) (SCM9B-4000)

The MaXimum (MX) and MiNimum (MN) commands are used to rescale the input ranges of SCM9B-4000 modules to units that may be more appropriate to a particular application.

```
Command:    $1MX+00020.00
Response:   *

Command:    #1MX+00020.00
Response:   *1MX+00020.00AB

Command:    $1MN+00000.00
Response:   *

Command:    #1MN+00000.00
Response:   *1MN+00000.009F
```

The MiNimum (MN) command assigns an input data value corresponding to the -full scale analog output value.

The MaXimum (MX) command assigns an input data value corresponding to the +full scale analog output value.

The MN and MX commands are covered thoroughly in chapter 10.

The MN and MX values are saved in nonvolatile memory and may be read back with the Read MiNimum (RMN) and Read MaXimum (RMX) commands.

The MN and MX commands are write-protected.

Read Analog Data (RAD) (SCM9B-4000)

All SCM9B-4000 modules contain an Analog-to-Digital Converter (ADC) which may be used to directly monitor the analog output signal. The ADC data is obtained with the Read Analog Data (RAD) command. The data is scaled in the same units as used with the Analog Output (AO) command. The ADC data obtained with the RAD command provides a check to assure the user that the module is working properly and no output fault conditions exist. Refer to the SCM9B-4000 section for more information.

```
Command:    $1RAD
Response:   *+00012.30

Command:    #1RAD
Response:   *1RAD+00012.30E1
```

Read Analog Output (RAO)

The Read Analog Output (RAO) command is used to read back the data sent by the most recent Analog Output (AO) command. It is particularly useful when the SCM9B-4000 is used with very low output slope values. The RAO gives the eventual final output of the analog output.

The RAO simply reads back the argument of the most recent AO command and does not necessarily correlate with the actual analog output. See the RD command.

```
Command:    $1RAO
Response:   *+00017.50

Command:    #1RAO
```

Response: ***1RAO+00017.50F3**

Read Data (RD)

The Read Data (RD) command reads back the digital data being sent to the DAC at the time the RD command is performed. It is used to obtain the status of the output signal at any time. The data obtained is scaled in the same units as used with the Analog Output (AO) command.

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

Command: **#1RD**
Response: ***1RD+00010.009B**

The RD command will read back instantaneous DAC data even if the output is being changed with the Manual Mode inputs or with the controlled output slew rates that may be obtained in SCM9B-4000 units.

Since the RD command is the primary means of monitoring output data, a special short form of the command is available for faster response. If a SCM9B-3000/4000 unit is addressed without a command, the RD command is assumed by default:

Command: **$1**
Response: ***+00012.34**

Command: **#1**
Response: ***1RD+00010.009B**

Note that the RD command returns the digital data that the microprocessor is currently sending to the DAC. It provides no guarantee that the analog output signal is being generated properly and that no output fault conditions exist. However, for a module that has been installed and verified for proper operation, the RD command is a reliable indicator of the output signal.

Read HIgh Limit (RHI)

The Read HIgh Limit (RHI) command reads back the HI Limit value stored in the nonvolatile memory. The HI limit may be changed by the HI command.

Command: **$1RHI**
Response: ***+00020.00**

```
Command:    #1RHI
Response:   *1RHI+00020.00E9
```

Read IDentification (RID)

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 D3000'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 LOw Limit (RLO)

The Read LOw limit (RLO) command reads back the LO limit data stored in the nonvolatile memory. The LO limit may be changed by the LO command.

```
Command:    $1RLO
Response:   *+00004.00

Command:    #1RLO
Response:   *1RLO+00004.00F5
```

Read Manual Slope (RMS)

The Read Manual Slope (RMS) command is used to read back the slope constant used in manual mode. This slope constant is implemented only when the analog output is controlled using the Up and Down pins on the terminal connector. The scaling is in units of mA/S or V/S for current and voltage outputs respectively. In SCM9B-4000 units, the Manual Slope value may be modified by the MS command.

```
Command:    $1RMS
Response:   *+00004.00

Command:    #1RMS
Response:   *1RMS+00004.00FA
```

Read MaXimum (RMX)

The Read MaXimum (RMX) command reads out the scaling data corresponding to + full scale at the analog output. The MaXimum data may be changed by using the MX command (SCM9B-4000 only).

```
Command:    $1RMX
Response:   *+00020.00

Command:    #1RMX
Response:   *1RMX+00020.00FD
```

Read MiNimum (RMN)

The Read MiNimum (RMN) command reads out the scaling data corresponding to - full scale at the analog output. The MiNimum data may be changed with the MN command (SCM9B-4000 only).

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

Command:    #1RMN
Response:   *1RMN+00000.00F1
```

Read Present Slope (RPS) (SCM9B-4000)

The Read Present Slope (RPS) reads back the output slope rate value currently active in SCM9B-4000 modules. The slope data is scaled in either mA/S or volts/S, depending on the output type:

```
Command:    $1RPS
Response:   *+00010.00

Command:    #1RPS
Response:   *1RPS+00010.00FA
```

The response data returned by these two example commands indicates that the present slope rate is either 10V/S or 10mA/S. The present slope may differ from the rate stored in EEPROM. See SCM9B-4000 section for details.

Read SetUp (RS or RSU)

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:   *310701C0
```

```
Command:     #1RSU
Response:    *1RSU310701C0F4
```

The Read Setup (RS) command performs the same function, and is included to be compatible with the SCM9B-1000/2000 series.

```
Command:     $1RS
Response:    *310701C0

Command:     #1RS
Response:    *1RS310701C09F
```

Read SLope (RSL)

The Read SLope (RSL) command reads back the output slew rate constant stored in EEPROM. The slope data is scaled in V/S or mA/S. depending on the type of output.

```
Command:     $1RSL
Response:    *+00010.00

Command:     #1RSL
Response:    *1RSL+00010.00F6
```

The data returned by these two command examples indicate that the slew rate stored in EEPROM is either 10V/S. or 10mA/S. This rate is not necessarily the rate currently used by the analog output. See SCM9B-4000 section.

Read Starting Value (RSV) (SCM9B-4000)

The Read Starting Value command reads the value of the desired start-up analog output which has been programmed by the user.

```
Command:     $1RSV
Response:    *+00005.00

Command:     #1RSV
Response:    *1RSV+00005.0004
```

Read Watchdog Timer (RWT) (SCM9B-4000)

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. See SCM9B-4000 section for watchdog timer information.

Remote Reset (RR)
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 will halt any analog output to it's present value.

The RR command is write-protected.

The RR command is required for a baud rate change.

Setup (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:	**$1SU31070182**
Response:	*****
Command:	**#1SU31070182**
Response:	***1SU3107018299**

SLope (SL) **(SCM9B-4000)**

The SLope (SL) command is used to set the output slew rate for analog outputs performed by the Analog Output (AO) command. The slope data is scaled in either V/S or mA/S:

```
Command:    $1SL+00100.00
Response:   *

Command:    #1SL+00100.00
Response:   *1SL+00100.00A4
```

These two sample commands will set the output slope rate to 100 V/S or 100 mA/S.

The SLope (SL) command data is saved only in Random Access Memory (RAM) and is not stored in EEPROM. The SL command is not write protected. The SL command is used in applications where frequent changes in the output slope rate is desired. See SCM9B-4000 section for further details.

Starting Value (SV) (SCM9B-4000)

The Starting Value (SV) command is used to program the desired analog output value when the unit is powered up. The output will automatically go to the programmed value with the slew rate stored in EEPROM.

```
Command:    $1SV+00005.00
Response:   *

Command:    #1SV+00005.00
Response:   *1SV+00005.00B2
```

Each of the two example commands sets the starting value to +00005.00. This value is stored in EEPROM. When the SCM9B-4000 unit is powered up, it automatically performs an internal Analog Output (AO) command with the stored data. If the AO command would have resulted in an error (LIMIT ERROR, MANUAL MODE) the start-up command is aborted and the SCM9B-4000 will start up at - Full Scale. The scaling of the start-up data is determined by the input scaling range fixed by the MiNimum (MN) and MaXimum (MX) limits.

The Starting Value is the ‘safe’ output value used when the watchdog timer times out. See SCM9B-4000 section.

The SV command is write-protected.

Trim MaXimum (TMX)
Trim MiNimum (TMN)

The TMX and TMN commands are used to calibrate the analog output circuitry of the module. These commands are used to communicate actual measured output data to the modules so that a trim calculation may be performed:

```
Command:    $1TMN+00000.12
Response:   *

Command:    #1TMX+00019.98
Response:   *1TMX+00019.9818
```

Refer to the Calibration section for details on output trims and the use of the TMN and TMX commands.

Caution: Unwarranted use of the TMN and TMX commands will destroy the calibration of the unit. These commands must be used with a calibrated voltmeter or ammeter to assure output accuracy.

Trim Readback MaXimum (TRX) (SCM9B-4000)
Trim Readback MiNimum (TRN) (SCM9B-4000)

The TRX and TRN commands are used on SCM9B-4000 modules to trim the Analog-to-Digital Converter (ADC) which provides the analog readback of the output signal. Refer to the Calibration section.

```
Command:    $1TRN
Response:   *

Command:    #1TRN
Response:   *1TRN4F
```

Watchdog Timer (WT) (SCM9B-4000)

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 analog output will automatically be forced to the Starting Value. See SCM9B-4000 section.

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

WT command data less than 0.16 minutes will result in a VALUE ERROR. The WT command is write protected.

Write Enable (WE)

The Write Enable (WE) command must precede commands that are write-protected. This is to guard against accidentally writing over valuable data in 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.

Write SLope To EEPROM (SCM9B-4000)

The Write SLope (WSL) command is used to set the output slew rate for analog outputs performed by the Analog Output (AO) command. The slope data is scaled in either V/S or mA/S:

Command:	**$1WSL+00100.00**
Response:	*****
Command:	**#1WSL+00100.00**

Response: *1WSL+00100.00FB

These two sample commands will set the output slope rate to 100V/S or 100mA/S.

The Write SLope (WSL) command stores the rate data in Random Access Memory (RAM) and in nonvolatile EEPROM.

The WSL command is write protected.

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 nine 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.

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.

LIMIT ERROR
A LIMIT ERROR may occur when using the Analog Output (AO) command if:

a) the AO data exceeds the input span range defined by the MiNimum (MN) and MaXimum (MX) values
b) the AO data exceeds a limit set by the LOw Limit (LO) or HIgh Limit (HI) commands
c) the output is inhibited by a limit switch (see Manual Modes)

MANUAL MODE
This error may occur when using the Analog Output (AO) or HeX Output (HX) commands while the output is being controlled by either the Up/Down Manual Mode or the Controller Input Manual Mode. The Manual Modes have priority over the host-generated commands.

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 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 used in the SetUp (SU) and HeX Output (HX) commands can range from 0-F.

A VALUE ERROR will be generated by a SCM9B-4000 module if a TMN, TMX, TRN, or TRX command is attempted while the output is slewing.

A VALUE ERROR is generated by TMN and TMX commands when an attempt is made to calibrate a module beyond the allowed trim range.

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 (RS).

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)
Number of displayed digits
Limits enable/disable
Continuous Input enable/disable (D4000)
Manual Mode enable/disable
Manual Up/Down Inputs
Controller Inputs
Limit Switches: Normally Open/Normally Closed

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: $1SU31070180.

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. Appendix A contains a convenient hex-to-binary conversion chart.

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 $1SU31070180 , 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: $1SU32070180. 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 ‘1’.

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.

Let’s run through an example of changing the baud rate. Assume our sample module contains the setup data value of ‘31070180’. 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:   *31070180
```

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:   *31020180
```

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
NO LINEFEED	0							
LINEFEED	1							
NO PARITY		0	0					
EVEN PARITY		0	1					
NO PARITY		1	0					
ODD PARITY		1	1					
NOT USED				0	0			
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’.

Continuous Input **(SCM9B-4000)**
Bit 5 enables the continuous input option available on SCM9B-4000 units and it is normally set to ‘0’. Setting Bit 5 to ‘1’ enables the continuous input. Refer to the SCM9B-4000 section for more information on the continuous input option.

Limit Disable
Bit 4 may be used to disable any limit checking on limits set by the LO and HI limit commands. Bit 4 is normally set to ‘0’; Bit 4 is set to ‘1’ to inhibit limit checking.

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 DI. 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	**0**	**0**						
CONTINUOUS DISABLED			**0**					
CONTINUOUS ENABLED			**1**					
LIMITS ENABLED				**0**				
LIMITS DISABLED				**1**				
NOT USED					**0**			
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

This setup byte specifies the number of displayed digits and the Manual Mode configuration.

Number of displayed digits

For ease of use, the data format of all modules is standardized to a common 7-digit configuration consisting of sign, 5 digits, decimal point, and two more digits. Typical data looks like: +00100.00. However, best-case resolution of the DAC (digital-to-analog converter) is 12 bits or about 3 1/2 digits. In some cases, the resolution of the output format is much greater than the resolution of the DAC. In such cases, the low-order digits would display meaningless information. Bits 6 and 7 are used to insert trailing zeros into the data format to limit the output resolution and mask off meaningless digits.

Bit 7	**Bit 6**		
0	**0**	**XXXX0.00**	**(4 displayed digits)**
0	**1**	**XXXXX.00**	**(5 displayed digits)**
1	**0**	**XXXXX.X0**	**(6 displayed digits)**
1	**1**	**XXXXX.XX**	**(7 displayed digits)**

Manual Mode Disable

The number of displayed digits only affects data read back by the RD and RAD commands.

Bit 2 is normally set to ‘0’ which allows the Manual Mode inputs to affect the analog output. Setting bit 2 to ‘1’ disables the Manual Modes.

Manual Mode Select
Bits 0 and 1 allow the user to select among four different Manual Modes. Manual Modes allow the analog output to be controlled by the UP* and DN* pins on the module connector. Details on Manual Modes are given in the Manual Mode section.

Table 5.4 Byte 4 Displayed Digits and Manual Mode Select

BYTE 4

FUNCTION	DATA BIT							
	7	6	5	4	3	2	1	0
+XXXX0.00 DISPLAYED DIGITS	0	0						
+XXXXX.00 DISPLAYED DIGITS	0	1						
+XXXXX.X0 DISPLAYED DIGITS	1	0						
+XXXXX.XX DISPLAYED DIGITS	1	1						
NOT USED			0	0	0			
MANUAL MODES ENABLED						0		
MANUAL MODES DISABLED						1		
UP/DOWN MODE							0	0
CONTROLLER INPUT							0	1
LIMIT SWITCHES N. O.							1	0
LIMIT SWITCHES N. C.							1	1

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 SCM9B-3000 unit and you wish to setup 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:   *310701C0
```

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:    $1SU310705C0
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
SCM9B-312X, 316X, 412X, 416X	31070180
SCM9B-313X, 314X, 317X, 318X	31070140
SCM9B-413X, 414X, 417X, 418X	31070140
SCM9B-325X, 326X, 425X, 426X	310701C0

Setup Software
S3000 setup software for the IBM PC and compatibles is available to facilitate module setup. Contact factory for details.

Chapter 6
Digital I/O Functions and Manual Mode

MANUAL MODES/DIGITAL INPUTS

Each SCM9B-3000/4000 module has three digital input connections designated as DI0/DN*, DI1 /UP*, and DI2. These inputs have a dual function; they may be used as control inputs which influence the analog output or they may be used as general-purpose digital inputs. The function of the input pins is programmable with the SetUp (SU) command.

The inputs are protected to voltages up to ±30V and are normally pulled up to the logic "1" condition (see Figure 6.1). Digital inputs can be read with the Digital Input (DI) command. Voltage inputs less than 1V are read back as '0'. Signals greater than 3.5V are read as '1'.

Switch closures can be read by the digital input by simply connecting the switch between GND terminal and a digital input. Internal pull-ups are used so additional parts are unnecessary.

The pull-ups supply only 0.5mA ; therefore, self-wiping switches designed for low current operation should be used. For other types of switches, it may be necessary to provide extra pull-up current with an external resistor. The resistor should be tied between the switch and +V.

Digital inputs may be used to sense AC voltages by using isolated sensing modules offered by many manufacturers.

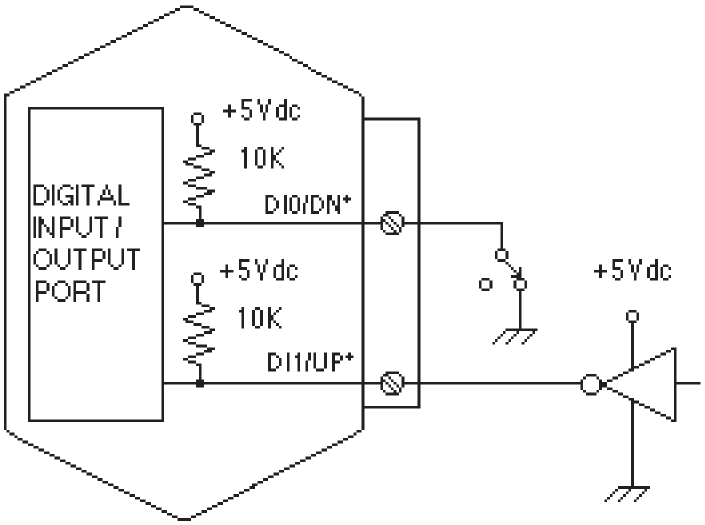


Figure 6.1 Digital Inputs.

MANUAL MODES

The SCM9B-3000/4000 modules may be configured to use the digital inputs to control the analog output. These functions are called Manual Modes. Four different Manual Modes may be specified:

Up/Down
Controller Input
Limit Switch NO
Limit Switch NC

These modes are selected by Bits 0 and 1 of Byte 4 in the Setup data. (See Setup section). Also, the Manual Mode Disable bit (Bit 2, Byte 4) must be cleared to enable Manual Modes.

UP/DOWN

Manual Up/Down control is the standard configuration when the module is shipped from the factory. This configuration provides a local operator interface to control the analog output value independent of the host computer. The analog output may be moved up or down by manipulating the UP* and DN* inputs. The control inputs may come from simple switches or may be logic signals originating from other equipment. Figure 6.2 shows the simplest connection. With the two switches, four different input combinations are possible:

UP*	DN*	
0	0	Hold
0	1	Slope Up
1	0	Slope Down
1	1	No Action

Since the digital inputs are pulled up internally, no connection or an open pushbutton generates a logic ‘1’. Shorting the input line to ground or closing the pushbutton generates a logic ‘0’. The ‘*’ in the terminal labels indicate that the inputs are negative true.

If both switches are open, a logic 1, 1 is generated and no manual action is performed.

If the UP* signal input is grounded by closing the UP pushbutton, the analog output will slope up to + Full Scale. A smooth slope in the output is generated by incrementing the DAC approximately 1000 times a second. If only the DN* input is grounded, the analog output will slope towards - Full Scale. The analog output will stop moving when the switches are released.

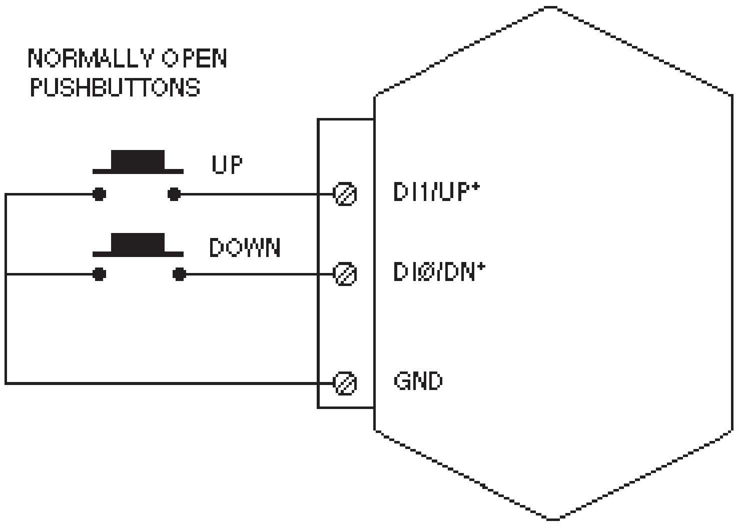


Figure 6.2 Manual Up/Down Control.

The slope rate on SCM9B-3000 modules is fixed and cannot be changed. The manual slope on SCM9B-3000 units is scaled so that a full-scale output change requires 5 seconds to complete. The manual slope rate on SCM9B-4000 units may be programmed to any desired value with the Manual Slope (MS) command.

The Manual Modes have priority over host-generated output commands. If either or both of the UP* and DN* inputs is held low, an Analog Output (AO) or HeX output (HX) command generated by the host will result in a MANUAL MODE error message and the host command is aborted. This brings us to the fourth switch combination, when both input switches are on. If both UP* and DN* signals are held at logic ‘0’, the analog output will hold its present value. Any attempts by the host computer to change the output will result in a MANUAL MODE error.

Another useful switch configuration is shown in Figure 6.3. This circuit is useful when the module is on-line with a host which is actively sending output commands to the module. This circuit will lock out the host while manual operations are being performed. Under normal host control, the Manual/Host switch is left open. For manual operation, the toggle switch is closed, grounding both UP* and DN* inputs. This will prevent the host from controlling the analog output. The output may be controlled manually by depressing the normally-closed pushbuttons. Note that the ‘UP’ button is connected to the DN* input.

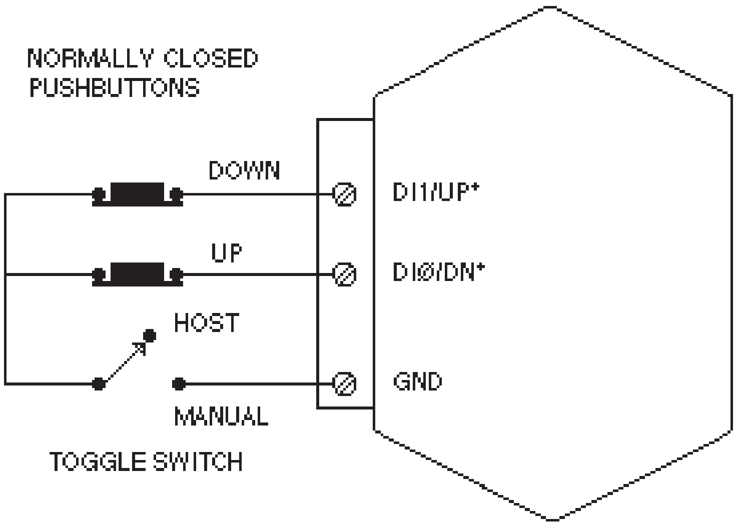


Figure 6.3 Manual Up/Down Control With Host Lock-Out.

CONTROLLER INPUT
This Manual Mode is a variation of the manual up/down control specifically setup for operation with ON-OFF controllers. With this mode, a D3000/4000 unit may be used to add an analog control output to an ON-OFF or time-proportional controller. The truth table for this mode is:

UP*	DN*	
0	0	Slope Up
0	1	No Action
1	0	Slope Down
1	1	No Action

With this setup, the DN* input acts as an enable signal. If the DN* input is high or open, no Manual Mode action takes place. If the signal is grounded, the controller input is enabled and the analog output will slope up or down. The slope direction is controlled by the UP* input. In this mode of operation, the analog output value is the integral of the UP* signal input. In order to keep the analog output in the linear region, an external signal must be used to manipulate the UP* input. This is usually done through feedback.

The slope rate used at the analog output in controller mode is the value specified for manual slope. The slope rate is fixed on SCM9B-3000 units; on SCM9B-4000 units the slope is programmable with the Manual Slope (MS) command.

LIMIT SWITCHES

Two of the Manual Modes allow the use of limit switches or other external digital signals to limit the analog output that may be obtained with the Analog Output (AO) command. The limit switch mode may be programmed to accommodate either normally-open (NO) switches or normally-closed (NC) switches. See the Setup section for mode selection details.

Figure 6.4 shows a typical module with normally-open limit switches. If the switches remain open, module operation is not affected. If the Down Limit switch is closed, an attempt to decrease the analog output signal with the AO command will result in a LIMIT ERROR message and the command will be aborted. An AO command to increase the analog output will be performed normally. As long as the Down Limit switch is closed, the analog output cannot be decreased from its present value with an AO command.

Conversely, if the Down Limit switch is open and the UP Limit switch is closed, an attempt to increase the analog output with the AO command will result in a LIMIT ERROR. The output may be decreased with no error.

If both limit switches are closed, any attempt to use the AO command will result in a LIMIT ERROR.

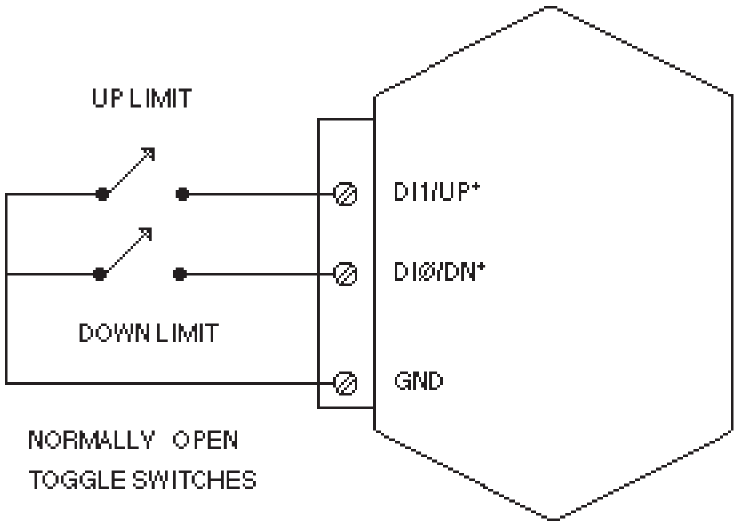


Figure 6.4 Using Switches to Limit Analog Output.

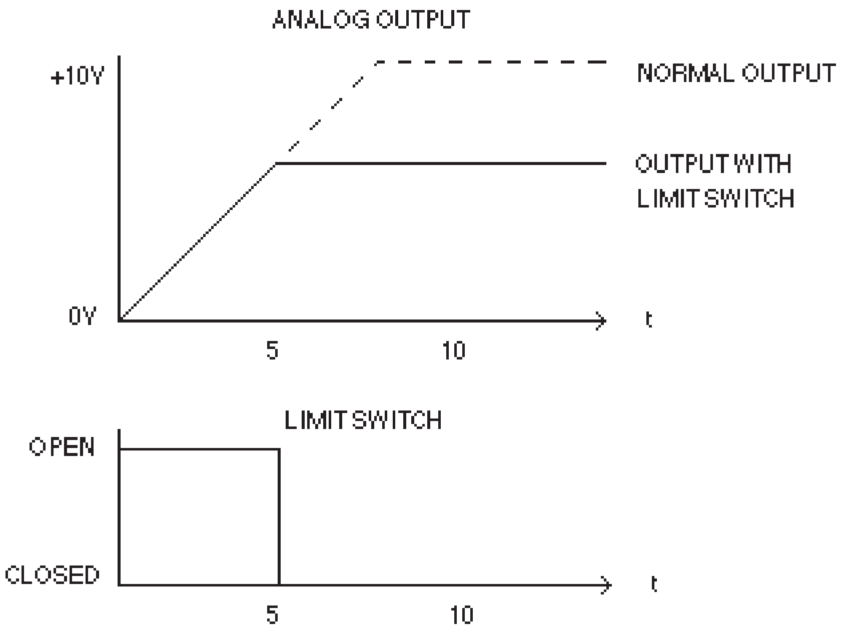


Figure 6.5 Using Limit Switches To Stop An Analog Output Ramp.

On SCM9B-4000 units with controlled output ramps, the limit switches will stop the output even after a successful AO command. Figure 6.5 illustrates this action. In this example a SCM9B-4181 voltage-output module is programmed with an output slope of 1V/S. Assume that the output voltage value is initially 0V. The command: $1AO+10000.00 will ramp the output to +10V. However, if the UP* limit switch is activated before the output reaches 10V, the output will stop and the AO command is terminated. The limit switch condition may be read by the DI command and the analog output may be read with the RD or RAD commands.

The Manual Mode setup may be configured to allow either Normally Open (NO) or Normally Closed (NC) switches. The truth table for either mode is:

Inputs		Action	
DN*	UP*	NO	NC
0	0	Hold	No Limit
0	1	Down Limit	Up Limit
1	0	Up Limit	Down Limit
1	1	No Limit	Hold

1= Open Switch

Chapter 7
Power Supply

SCM9B-3000/4000 modules may be powered with an unregulated +10 to +30Vdc supply. 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.

All SCM9B-3000/4000 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. SCM9B-3000/4000 voltage output modules consume a maximum of 0.75 watts and SCM9B-3000/4000 current output models consume 1.0 watts maximum. The power consumption figures should be used in determining the power supply current requirement. For example, assume a 24 volt power supply will be used to power four voltage output modules. The total power requirement is 4 X 0.75 = 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.

The low voltage detection cicuit shuts down the module at approximately 9.5Vdc. If the module is interogated while in a low power supply condition, the module will not respond. Random NOT READY error messages could indicate that the power supply voltage is periodically drooping below the 10V minimun.

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 communications 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.

All SCM9B-3000/4000 modules are protected against power supply reversals.

Chapter 8
Troubleshooting

Symptom:
RS-232 Module is not responding to commands.
RS-485 Module is not responding to commands.
Module responds with ?1 COMMAND ERROR TO every command.
Characters in each response message appear as graphics characters.

• 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.

• 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.

• Module responds with ?1 COMMAND ERROR TO every command
Ensure that characters in the command message are uppercase characters. All commands consist of uppercase characters only.

• Characters in each response message appear as graphics characters
1 Set the communications software parity setting to "M" for 'MARK' parity type and 7 data bits. Or, utilize any parity type in both the module and software other than "NO" parity.

2 In custom written software routines, mask off the most significant bit of each received character to logic "0". Thus forcing the received character to 7-bit ASCII value.

Chapter 9
Calibration

SCM9B-3000/4000 units feature state-of-the art digital trimming techniques to eliminate the need for calibration pots or other hardware trims. Calibration is performed with trim commands through the communications port. The on-board microprocessor is used to calculate calibration constants which are then stored in the nonvolatile EEPROM. Field calibration of the units may be performed without the need to physically access the device.

Digital calibration is made possible by reserving a small portion of the DAC scale for trim purposes. The DAC hardware is capable of producing outputs in excess of the full scale range normally accessed by the Analog Output (AO) command. This may be demonstrated by using the Hex Output (HX) command, which controls the DAC directly without trims. For example, a SCM9B-3181 module has a nominal output of 0 to 10V. The absolute maximum output of the DAC may be obtained with the HX command:

```
Command:    $1HX0FFF
Response:   *
```

A measurement of the output signal would typically read about +10.2 volts. This shows that 0.2 volts is the excess DAC range available for trimming. Typically about 2% of the DAC range is reserved for trim purposes.

The only equipment necessary for calibration is a suitable voltmeter or ammeter (0.02% accurate) to monitor the output signal and a terminal or computer to communicate to the device.

Calibration is performed by comparing the ideal desired output to the actual measured output. The ideal output is set by using the Analog Output (AO) command. After the actual output value is measured with a calibrated meter, the actual value is communicated to the module with the Trim MiNimum (TMN) or Trim MaXimum (TMX) commands. The TMN command trims the - full scale output; TMX trims + full scale. After receiving a TMN or TMX command, the module compares the AO data to the actual output value and computes a new calibration factor to reduce the error to zero.

The actual data specified by the TMN and TMX command must be presented in standard 7-digit format, in units of millivolts or milliamps:

```
$1TMX+10012.00   (10.012V)

$1TMX+00020.05   (20.05mA)

$1TMN-04990.00   (-4.99V)

$1TMN+00000.02   (+.02mA)
```

Trim resolution is 1LSB of the DAC which is full scale ÷ 4096.

Calibration Procedure-Voltage units

1) Connect voltmeter to analog output

2) Set the output to -full scale with Analog Output (AO) command.

3) Measure the output voltage.

4) Report the actual output value to the module with the Trim MiNimum (TMN) command. The module will adjust the output to a new value.

5) Check the output value with the meter. If the output is not within 1LSB, repeat step 4.

6) Set the output to + full scale with the Analog Output (AO) command.

7) Measure the output voltage.

8) Report the actual output voltage to the module with the Trim MaXimum (TMX) command. The module will adjust the output value.

9) Check the output value with the meter. If the output is not within 1LSB, repeat step 8.

To further illustrate the calibration procedure, here is a typical sequence used to calibration a SCM9B-3181 which has an output of 0 to 10V:

Set the output to - full scale:

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

Measure the output voltage. In this case, the measured output is -12 millivolts. Report the actual value to the module with the TMN command:

```
Command:    $1 WE
Response:   *

Command:    $1TMN-00012.00
Response:   *
```

Measure the output value with the meter. The output measures +1mV, which is within 1LSB (2.5mV).

Now set the output to + full scale:

```
Command:    $1AO+10000.00
Response:   *
```

The measured output is +10.123V. Report the measured output to the module with the Trim Maximum (TMX) command:

```
Command:    $1WE
Response:   *

Command:    $1TMX+10123.00
Response:   *
```

The output now measures +10.005V, which is still not within specification. Repeat the TMX command with the new value:

```
Command:    $1WE
Response:   *

Command:    $1TMX+10005.00
Response:   *
```

The output now measures 9.999V, which is within 1LSB (2.5mV).

Current Output Calibration
Modules with current outputs are trimmed in exactly the same manner as voltage outputs with one exception. Since the current outputs are unipolar and cannot sink current, errors will result if an attempt is made to calibrate the - full scale output at 0mA. On 0-20mA units, - full scale trim should be performed at some small positive value such as 0.5mA. This is done by simply using the AO command to output the desired trim point:

```
Command:    $1AO+00000.50    (0.5mA)
Response:   *
```

Assume that in this case the actual output is measured to be +0.63mA. Use the TMN command to report the actual output to the module:

```
Command:    $1WE
Response:   *

Command:    $1TMN+00000.63   (actual=0.63mA)
Response:   *
```

The microprocessor will calculate a trim value to force the output to the ideal output of 0.5mA.

SCM9B-4000 CALIBRATION

SCM9B-4000 modules offer the ability to re-scale the input data to any desired engineering units. This may cause problems in calibration since it may be difficult to correlate the input scaling to the output signals. In this case it may be easier to re-scale the SCM9B-4000 to standard voltage and current ranges that may be compared directly with measured output values. Calibration is then performed with the same procedure as described above. After calibration, the module may be re-scaled back to any desired engineering units with the MN and MX commands.

Analog Readback Calibration

The analog-to-digital converter (ADC) used for readback is trimmed independently of the DAC. The trim commands used to calibrate the ADC are Trim Readback MiNimum (TRN) and Trim Readback MaXimum (TRX).

To Trim the ADC, be sure the DAC output has been calibrated. Set the output to - full scale with the Analog Output (AO) command. Then perform the Trim Readback MiNimum command:

Command:	**$1WE**	**(TRN is write protected)**
Response:	*	
Command:	**$1TRN**	
Response:	*	

The microprocessor will calculate calibration values to fix that point on the ADC scale. The calibration factors are automatically stored in EEPROM.

Now set the output to + full scale with the Analog Output (AO) command. Perform the Trim Readback Maximum command:

Command:	**$1WE**	**(TRX is write protected)**
Response:	*	
Command:	**$1TRX**	
Response:	*	

This command fixes the maximum point on the ADC scale and stores the data in EEPROM.

For proper ADC calibration, the analog output must be set exactly to the -full scale and +full scale values before using the TRN and TRX commands. If these values are unknown, they may be read back using the RMN and RMX commands.

D4000 Calibration Addendum

Due to a microprocessor 'bug', early production D4000 units will not execute the TRN command if the -full scale value is negative. An attempt to execute the TRN command will result in a VALUE ERROR.

However, the ADC on these units may be trimmed by using the following procedure:

Read the -full scale value by using the Read MiNimum (RMN) command:

```
Command:    $1RMN
Response:   *-10000.00        (typical data)
```

Record the -full scale data for later use.

Rescale the -full scale data to '0' using the MiNimum (MN) command:

```
Command:    $1WE
Response:   *

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

Set the analog output data to '0':

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

Perform the TRN command:

```
Command:    $1WE
Response:   *

Command:    $1TRN
Response:   *
```

Now rescale the -full scale value back to the original data previously accessed by the RMN command:

```
Command:    $1WE
Response:   *

Command:    $1MN-10000.00          (typical data)
Response:   *
```

The ADC is now trimmed at -full scale.

Trimming the ADC at +full scale is done in a normal fashion.

Chapter 10
D4000 Features

SCM9B-4000 Features

The SCM9B-4000 series of computer-to-analog output modules contain many intelligent enhancements not found in the SCM9B-3000. The SCM9B-4000 accepts all of the SCM9B-3000 commands and contains several additional commands which take full advantage of the computational power of the on-board microprocessor. These additional features are:

Programmable output slew rates
Programmable data scaling
Programmable start-up values
Watchdog timer
True analog readback
Continuous Input

Slope Control

The operation of most digital to analog converters (including the SCM9B-3000) provides only for a step function when a new output value is desired. That is, the analog output change is instantaneous subject only to the settling time of the device. In many applications this characteristic is undesirable and a gradual controlled output slew rate is more appropriate. In a typical system where controlled output rates are desired, precious host computer time must be used to continually monitor and step the digital to analog converter (DAC) until the desired output is obtained.

The SCM9B-4000 allows the system designer to obtain controlled output slew rates automatically without host computer intervention. Programmable output slope rates may be specified by the user and stored in nonvolatile memory. If a command is given to the SCM9B-4000 to change the output value, the output will automatically slope to the new value at the specified rate. The slope value is nonvolatile and will be restored each time the module is powered up. The slope rate is specified with write-protected slope commands in units of volts per second on voltage output models and milliamps per second on current models. Slopes may be specified from a range of +99999.99 (step output) down to +00000.01 volts or mA per second in .01 increments. A slope of .01mA/second requires more than 33 minutes to perform an output change of 20mA!

The microprocessor in the SCM9B-4000 controls the output slew rate by updating the DAC of a rate of 1000 conversions per second at precise 1ms intervals. Slope data is represented in the microprocessor with high-resolution floating point numbers. Before each D-to-A conversion the slope increment is added to the present output value. The new output data is then rounded off to the nearest value that can be represented by the 12-bit D-to-A converter and a new output is obtained. In this manner the DAC is smoothly stepped until the final output value is reached as specified by the Analog Output (AO) command. The incremental steps obtainable from the 12-bit converter and the 1ms conversion rate combine to make the output change appear to be a linear ramp.

Slope Commands

The SCM9B-4000 commands that are directly related to the slope functions are:

RPS	Read Present Slope
RSL	Read Slope
SL	Slope (RAM)
WSL	Write Slope (EEPROM)

These commands are described in detail in the commands section (Chapter 4) of this manual. However, a few clarifications are necessary to fully understand the function of these commands. In the SCM9B-4000 , slope data is held in two memory areas: EEPROM and RAM. The RAM (Random Access Memory) contains the working copy of the slope data used by the microprocessor. It is in RAM that the microprocessor obtains the slope value used to modify the output data. RAM data may be read or written any number of times; however, RAM data is lost when the SCM9B-4000 is powered down.

The EEPROM (Electrically Erasable Programmable Read Only Memory) also stores the slope value. The EEPROM is nonvolatile and is used to store the slope value (as well as other data) when power to the module is turned off. When power is applied to the unit, or a Remote Reset (RR) command is performed, the slope data is transferred from the EEPROM to RAM where it is used by the microprocessor. The EEPROM data may be read an unlimited number of times; however the EEPROM is limited to 10,000 write cycles, after which data reliability is not guaranteed.

The Write Slope (WSL) command writes the desired slope rate into both EEPROM and RAM. This command will satisfy most applications where the desired slope is fixed and does not change during normal operation.

The Slope (SL) command writes the slope data into RAM only and does not affect data in EEPROM. The SL command is used in special situations where it is desirable to change the slope frequently under control of the host computer. By writing the slope data into RAM only, the 10,000 cycle write limit of the EEPROM is circumvented. For example, a voltage output SCM9B-4000 may be used to provide the input to a motor speed controller. Under control of the host computer, the motor speed is cycled up and down once every minute. Using the slope commands, the host computer may specify different rates to accelerate and decelerate the motor. If the Write Slope (WSL) command is used twice a minute to control acceleration and deceleration, the 10,000 cycle write limit of the EEPROM will be exceeded within 4 days of operation. To overcome this limitation, the Slope (SL) command may be used to dynamically change the slew rate. Since the SL command writes only to RAM, the slope data may be changed an unlimited number of times.

The Read Present Slope (RPS) command reads the slope data contained in RAM. This data might not match the data held in EEPROM if the slope has been altered by the SL command. The Read Slope (RSL) command reads the slope data contained in EEPROM. It may differ from the value held in RAM.

Additional Commands

The SCM9B-4000 controls the DAC output as a background function; most commands may be performed without affecting an output which is ramping to a new value. Exceptions are:

Analog Output (AO) command. The AO command may be performed even if the output has not reached the final value specified by a previous AO command. The AO command may be performed “on the fly" at any time. The output will simply ramp to the new value specified. In some cases the slope direction may change to reach the new value.

Write Slope (WSL) and Slope (SL) commands. The WSL and SL commands may be performed “on the fly” even if the output is changing. The output will simply ramp to the final value with the new slope specified by the WSL or SL command.

Remote Reset (RR). The RR command will immediately freeze the output to its current condition.

Watchdog Timer (WT). If a watchdog timeout occurs, the timer will take control of the output even if the output is still slewing as the result of an AO command. See description of Watchdog Timer below.

Digital Inputs: The digital UP/DN and limit switch inputs have priority over the AO command and will affect the outputs if activated. See section on digital inputs.

Status Commands

The RD command may be used at any time to read the data being fed to the DAC. This is useful when using slow slew rates to monitor the present output data.

The Digital Input (DI) command may be used as a quick status check to see if the output signal has reached its final value. Refer to DI command in chapter 4.

Manual Slopes

The SCM9B-4000 allows the user to specify the output slew rate when the output is controlled by the manual UP/DN inputs. The Manual Slope (MS) command is used to write the rate data in EEPROM. The manual slope rate is totally independent of the slew rates used with the computer controlled output (AO command). The manual slope rate may be read back with the Read Manual Slope (RMS) command.

INPUT DATA SCALING

All SCM9B-3000 and SCM9B-4000 modules are factory set with data values set in units of millivolts or milliamps. For example, the command

$1AO+00020.00

sent to a current output module tells the module to output 20mA. This command sent to a voltage output module will tell the output to go to 20mV.

The SCM9B-4000 allows the user to scale the input data to any desired units. In many applications a change in input scaling may make the data easier to read and interpret. For example, a SCM9B-4000 used to control a valve actuator may be easier to use if the data is scaled with a range of 0-100% rather than 4-20mA.

The input scaling may be changed by using the Maximum (MX) and Minimum (MN) commands. These commands are used to assign input data values to correspond with the maximum and minimum output values obtainable from the module.

The MiNimum (MN) command assigns an input data value to the -full scale output of the module. The actual -full scale analog output signal is not affected; the MN command only changes the ASCII data value that represents the -full scale output. For example, a SCM9B-4141 has a -full scale output of -10V. The module is scaled at the factory in units of millivolts so that a data value of -10000.00 represents -10V. The MN command may be used to change the data value corresponding to -10V:

Command:	**$1WE**
Response:	*****
Command:	**$1MN+00000.00**
Response:	*****

Now, the output command:

Command:	**$1AO+00000.00**

Response: *

will produce an output of -10V.

The MaXimum (MX) command assigns the data value corresponding to +full scale.

Example: A SCM9B-4181 voltage output module is used to supply the control signal to a motor speed controller. The full scale range of the SCM9B-4181 is 0 to +10V. With this voltage input the motor speed varies from 100 to 3000 RPM. To command the motor to turn at a specified RPM requires some computation to obtain the correct command data. For instance, to command the motor to run at 1500 RPM requires the command:

```
Command:    $1AO+04666.00
Response:   *
```

The data is difficult to read and interpret.

A solution to this problem is to scale the input data directly in units of RPM.

The -full scale output of 0V is assigned the value 100 RPM with the command:

```
Command:    $1MN+000100.00
Response:   *
```

The + full scale output of +10V is assigned the value of 3000 RPM with the MaXimum (MX) command:

```
Command:    $1MX+03000.00
Response:   *
```

Once the endpoint values are assigned, all other data values are interpolated linearly. Now to set the motor to 1500 RPM requires the command:

```
Command:    $1AO+01500.00
Response:   *
```

The data is much easier to interpret since the scaling is directly in units of RPM. The actual output voltage resulting from this command is +4.666 volts.

Example: A valve actuator accepts a 4-20mA signal: at 4mA the valve is fully closed and at 20mA the valve is fully open. We wish to rescale a SCM9B-4251 0-20mA module to accept data of 0% open to 100% open.

The Minimum (MN) command assigns an input data value to the - full scale output of the module, which in this case is 0mA.

Using the two scaling points (4mA, 0%) and (20mA, 100%) and a bit of computation, we find that 0mA interpolates to a value of -25%. This value is used in the argument of the MN command:

```
Command:    $1MN-00025.00
Response:   *
```

The maximum scaling point of 20mA is straight forward and is assigned the input value of 100%:

```
Command:    $1MX+00100.00
Response:   *
```

The module is now scaled in percentage of valve opening. To set the valve to 50% opening:

```
Command:    $1AO+00050.00
Response:   *
```

In this case the SCM9B-4000 module produces an output of 12mA, opening the valve halfway.

If a SCM9B-4000 module has been rescaled, the readback data obtained from the Read Data (RD) and Read Analog Data (RAD) commands are automatically rescaled to the new units.

The HI and LO limit values (if used) are not affected by rescaling the D4000, and must be individually reassigned to values appropriate to the new scaling.

The starting value not affected.

Slope rate data is not affected by changes in scaling and are always in units of volts per second or milliamps per second.

The MaXimum and MiNimum scaling points may be assigned to any values within the limitations of the standard data format. Negative scalings and inverse scalings are acceptable.

It is important to understand that rescaling only modifies the way data is represented in the module. The output range is not affected. It is not possible for the user to alter the output range or resolution.

In some applications, it may be necessary to adjust the 'number of displayed digits' as described in the Setup Chapter. The number of digits should be chosen to allow the full resolution of the DAC (4096 counts) to be represented while suppressing unwanted lower-order digits. The number of digits displayed affects only the RD and RAD commands.

STARTING VALUE

When a SCM9B-4000 module is powered up from a cold start, the analog output is automatically forced to a pre-determined starting value. This value may be specified with the Starting Value (SV) command. This feature is useful for cold-starting systems in a controlled manner. Usually the starting value is specified as a “safe” condition to protect equipment and material from damage.

The Starting Value may be read back with the Read Starting Value (RSV) command.

The SV and RSV commands are detailed in the command section of chapter 4.

WATCHDOG TIMER

SCM9B-4000 units contain a programmable software timer to provide an orderly shutdown of the output signal in the event of host computer or communications failure. The timer is preset using the Watchdog Timer (WT) command to specify a timer interval in minutes. The timer is continually incremented in software. Each time the SCM9B-4000 module receives a valid command, the timer is cleared to zero and restarted again. If the timer count reaches the preset value, the output will automatically be forced to the starting value (see SV command). The output will slew to the starting value using the present output slope rate.

The purpose of the Watchdog Timer is to safeguard against host or communications failure. The Starting Value should be programmed to provide a ‘safe’ output value to minimize damage and disruption to the system under control.

During normal operation, the host system should periodically read the status or update the module to prevent the watchdog from reaching the timeout value. Under these conditions, the watchdog has no effect on the module output. However, if the timer reaches the preset value, the SCM9B-4000 assumes that the host system or communications channel is inoperative and will force the output to a safe value.

The preset value may be read back with the Read Watchdog Timer (RWT) command.

The WT and RWT commands are detailed in the command section.

ANALOG READBACK

The Read Data (RD) command is contained in all SCM9B-3000 and SCM9B-4000 units to provide a status report of the output of a module. For a properly functioning and calibrated module, the RD data returns a very accurate readback of the output signal. However, the data obtained with the RD command only indicates the digital data that is being transferred from the on-board microprocessor to the DAC. It provides no indication as to whether the analog signal being produced by the DAC is correct. Fault conditions such as shorts on voltage output modules or open circuits on current output models cannot be detected by the RD command.

To provide a true readback of the analog output signal, the SCM9B-4000 models contain a simple analog to digital converter (ADC) which is totally independent of the DAC. The ADC is tied directly to the analog output signal and provides readback data to the microprocessor. The analog data may be read back with the Read Analog Data (RAD) command. The RAD data provides true analog readback scaled in the same units as provided with the RD command.

The ADC is not intended to be a highly accurate measurement of the output signal. Typical accuracy is about 1% of full scale (see specifications). However, when used properly, the RAD command can greatly enhance the user's confidence level that the analog output is being produced as intended. Output fault conditions from improper wiring or loads can be easily detected. The A DC also provides a form of redundancy to ensure that all the circuits in the SCM9B-4000 module are working properly.

To utilize the ADC most effectively, the output should be in a steady-state condition. Slewing outputs decrease accuracy. First obtain a status reading with the RD command. This reading gives the data value being fed to the DAC. Now obtain a readback with the RAD command. This data represents the actual analog output signal. The two data readings are scaled identically. Compare the two readings; they should differ by less than 1% of full scale (refer to specifications). If the error is within specifications, it is a very positive

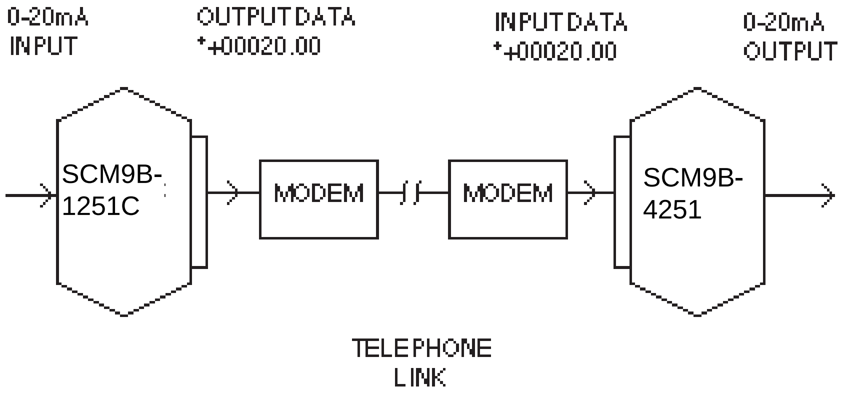


Figure 10.1 SCM9B-4000 Continuous Input Mode Application.

indication that the output is correct. Large errors indicate improper output loading or module failure.

Continuous Input

SCM9B-4000 units may be configured to operate in a special mode called Continuous Input Mode. This mode allows the SCM9B-4000 to be slaved directly to a SCM9B-1000 or SCM9B-2000 sensor interface unit. Figure 10.1 shows a typical application.

In this example, a SCM9B-1251C sensor module is used to convert a 0-20mA process signal to ASCII data. The SCM9B-1251C is operated in continuous output mode to produce data without a host. The SCM9B-1251C will produce an ASCII output data string after every analog-to-digital conversion, approximately eight times a second. Typical output data shown in Figure 10.1 is *+00020.00. The data output is connected to a modem which allows the data to be transmitted to another modem which may be located thousands of miles away. The receive modem reconstructs the ASCII data and feeds it to the SCM9B-4251 module. The SCM9B-4251 is configured in continuous input mode which allows it to accept the data string of *+00020.00 as an analog output command. The SCM9B-4251 will respond by producing an output of 20mA.

The net result of this connection is that the process variable sensed by the SCM9B-1251C may be accurately reproduced by the SCM9B-4251 anywhere a telephone connection is available. The SCM9B-4251 output will follow the input signal applied to the SCM9B-1251C. No host is necessary on either end to provide a continuous signal.

The continuous input configuration may also be used to convert analog data from one type to another. Figure 10.2 is an example.

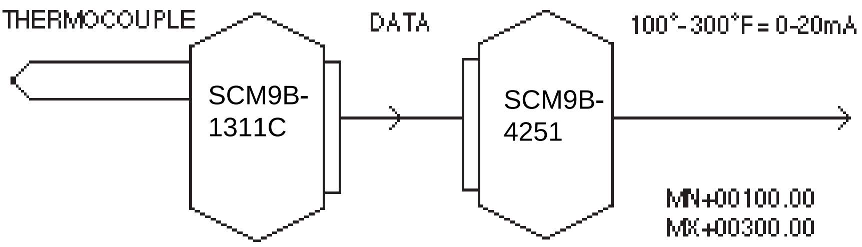


Figure 10.2 Rescaling a Temperature Input to a 0-20mA Analog Output.

In this case, a thermocouple signal may be converted to a 0-20mA analog output. Since the SCM9B-4251 may be rescaled to any input range, it may be setup to accept the thermocouple data directly. In this application, we would like the SCM9B-4251 to produce 0-20mA corresponding to 100°-300°F at the thermocouple. This is accomplished by rescaling the SCM9B-4251:

```
Command:    $1MN+00100.00
Response:   *

Command:    $1MX+00300.00
Response:   *
```

Continuous Input Setup

The Continuous Input Mode may be specified by setting bit 5,byte 3 of the setup data (see setup chapter). In addition, the DI2 terminal pin must be grounded before the module will accept continuous input commands. The DI2 connection provides an external means of controlling the continuous input option.

When a SCM9B-4000 module is setup for continuous input, it will accept all commands in a normal fashion. In addition, it will accept commands in response data format:

```
Command:  *+00123.45
```

A SCM9B-4000 module in continuous input mode will interpret this command as an analog output (AO) command comparable to:

```
Command:  $1AO+00123.45
```

However, a continuous input command will never generate a response from the SCM9B-4000. If the continuous input command is valid and error-free, the SCM9B-4000 will perform the analog output but will not produce any response on the communications line. If the continuous input data is out of range or contains errors, the SCM9B-4000 will abort the command and will produce no communications response.

The SCM9B-4000 continuous input option leads to a wide variety of system possibilities which are beyond the scope of this manual. Specific applications information may be obtained by calling the factory for assistance.

Appendix A
ASCII TABLE

Table of ASCII characters (A) and their equivalent values in Decimal (D), Hexadecimal (Hex), and Binary. 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
“	34	22	00100010	162	A2	10100010

A	D	Hex	Binary	D	Hex	Binary
#	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

A	D	Hex	Binary	D	Hex	Binary
u	117	75	01110101	245	F5	11110101
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
SCM9B-3000/4000 Specifications

Specifications (@ +25°C and nominal power supply voltage).

Analog Output
- Single channel analog output.
  Voltage: 0-1V, ±1V, 0-5V, ±5V, 0-10V, ±10V.
  Current: 0-20mA, 4-20mA.
- Output isolation to 500V rms.
- 12-bit output resolution.
- Accuracy (Integral & Differential Nonlinearity): 0.1%FSR (max).
- Zero drift: ±30µV/°C (Voltage Output).
  ±1.0µA/°C (Current Output).
- Span tempco: ±50ppm/°C max.
- 1000 conversions per second.
- Settling Time to 0.1%FS 300µS typ (1mS max).
- Output Slewing Manual Mode (-FS to +FS): 5S.
- Programmable Output Slew Rate: 0.01V/S (mA/S) to
  10,000V/S (mA/S).
- Current Output Voltage Compliance: 12V.
- Voltage Output Drive Current: 5mA max.
- Output Protection: 240VAC (current output).
  ±30V (voltage outputs).

Analog Output Readback
- 8-bit Analog to Digital Converter.
- Accuracy over Temperature (-25 to +70°C): 2.0%FS max.

Digital
- 8-bit CMOS microcomputer.
- Digital scaling and calibration.
- Nonvolatile memory eliminates pots and switches.
- Programmable data scaling (SCM9B-4000).
- Programmable High/Low output limits.
- Programmable initial value (SCM9B-4000).
- Programmable watchdog timer provides orderly shut-down in the event of host failure (SCM9B-4000).

Digital Inputs
- Voltage levels: ±30V without damage.
- Switching levels: High, 3.5V min., Low, 1.0Vmax.
- Internal pull up resistors for direct switch input.

Communications

- RS-232C, RS-485.
- Up to 124 multidrop modules per host communications port.
- User selectable channel address.
- Selectable baud rates: 300, 600, 1200, 2400, 4800, 9600,19200, 38400.
- ASCII format command/response protocol.
- Can be used with “dumb” terminal.
- Parity: odd, even, none.
- All communications setups (address, baud rate, parity) stored in nonvolatile memory using EEPROM.
- Checksum can be added to any command or response.
- Communications distance up to 4,000 feet.

Power

Requirements: Unregulated +10V to +30Vdc, 0.75W max (Voltage Output), 1.0W max (Current Output).
Internal switching regulator.
Protected against power supply reversals.

Environmental

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

Mechanical & Dimensions

Case: ABS with captive mounting hardware.
Connectors: Screw terminal barrier plug (supplied).
Replace with Phoenix MSTB 1.5/10 ST 5.08 or equivalent.

NOTE: Spacing for mounting screws = 2.700". Screw threads are 6 X 32.