

KEITHLEY

M1000/M2000

User Guide

A GREATER MEASURE OF CONFIDENCE

WARRANTY

Hardware

Keithley Instruments, Inc. warrants that, for a period of one (1) year from the date of shipment (3 years for Models 2000, 2001, 2002, 2010 and 2700), the Keithley Hardware product will be free from defects in materials or workmanship. This warranty will be honored provided the defect has not been caused by use of the Keithley Hardware not in accordance with the instructions for the product. This warranty shall be null and void upon: (1) any modification of Keithley Hardware that is made by other than Keithley and not approved in writing by Keithley or (2) operation of the Keithley Hardware outside of the environmental specifications therefore.

Upon receiving notification of a defect in the Keithley Hardware during the warranty period, Keithley will, at its option, either repair or replace such Keithley Hardware. During the first ninety days of the warranty period, Keithley will, at its option, supply the necessary on site labor to return the product to the condition prior to the notification of a defect. Failure to notify Keithley of a defect during the warranty shall relieve Keithley of its obligations and liabilities under this warranty.

Other Hardware

The portion of the product that is not manufactured by Keithley (Other Hardware) shall not be covered by this warranty, and Keithley shall have no duty of obligation to enforce any manufacturers' warranties on behalf of the customer. On those other manufacturers' products that Keithley purchases for resale, Keithley shall have no duty of obligation to enforce any manufacturers' warranties on behalf of the customer.

Software

Keithley warrants that for a period of one (1) year from date of shipment, the Keithley produced portion of the software or firmware (Keithley Software) will conform in all material respects with the published specifications provided such Keithley Software is used on the product for which it is intended and otherwise in accordance with the instructions therefore. Keithley does not warrant that operation of the Keithley Software will be uninterrupted or error-free and/or that the Keithley Software will be adequate for the customer's intended application and/or use. This warranty shall be null and void upon any modification of the Keithley Software that is made by other than Keithley and not approved in writing by Keithley.

If Keithley receives notification of a Keithley Software nonconformity that is covered by this warranty during the warranty period, Keithley will review the conditions described in such notice. Such notice must state the published specification(s) to which the Keithley Software fails to conform and the manner in which the Keithley Software fails to conform to such published specification(s) with sufficient specificity to permit Keithley to correct such nonconformity. If Keithley determines that the Keithley Software does not conform with the published specifications, Keithley will, at its option, provide either the programming services necessary to correct such nonconformity or develop a program change to bypass such nonconformity in the Keithley Software. Failure to notify Keithley of a nonconformity during the warranty shall relieve Keithley of its obligations and liabilities under this warranty.

Other Software

OEM software that is not produced by Keithley (Other Software) shall not be covered by this warranty, and Keithley shall have no duty or obligation to enforce any OEM's warranties on behalf of the customer.

Other Items

Keithley warrants the following items for 90 days from the date of shipment: probes, cables, rechargeable batteries, diskettes, and documentation.

Items not Covered under Warranty

This warranty does not apply to fuses, non-rechargeable batteries, damage from battery leakage, or problems arising from normal wear or failure to follow instructions.

Limitation of Warranty

This warranty does not apply to defects resulting from product modification made by Purchaser without Keithley's express written consent, or by misuse of any product or part.

Disclaimer of Warranties

EXCEPT FOR THE EXPRESS WARRANTIES ABOVE KEITHLEY DISCLAIMS ALL OTHER WARRANTIES, EXPRESS OR IMPLIED, INCLUDING WITHOUT LIMITATION, ALL IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. KEITHLEY DISCLAIMS ALL WARRANTIES WITH RESPECT TO THE OTHER HARDWARE AND OTHER SOFTWARE.

Limitation of Liability

KEITHLEY INSTRUMENTS SHALL IN NO EVENT, REGARDLESS OF CAUSE, ASSUME RESPONSIBILITY FOR OR BE LIABLE FOR: (1) ECONOMICAL, INCIDENTAL, CONSEQUENTIAL, INDIRECT, SPECIAL, PUNITIVE OR EXEMPLARY DAMAGES, WHETHER CLAIMED UNDER CONTRACT, TORT OR ANY OTHER LEGAL THEORY, (2) LOSS OF OR DAMAGE TO THE CUSTOMER'S DATA OR PROGRAMMING, OR (3) PENALTIES OR PENALTY CLAUSES OF ANY DESCRIPTION OR INDEMNIFICATION OF THE CUSTOMER OR OTHERS FOR COSTS, DAMAGES, OR EXPENSES RELATED TO THE GOODS OR SERVICES PROVIDED UNDER THIS WARRANTY.



Keithley Instruments, Inc.

28775 Aurora Road • Cleveland, Ohio 44139 • 440-248-0400 • Fax: 440-248-6168

1-888-KEITHLEY (534-8453) • www.keithley.com

Sales Offices: BELGIUM:

Bergensesteenweg 709 • B-1600 Sint-Pieters-Leeuw • 02-363 00 40 • Fax: 02/363 00 64

CHINA:

Yuan Chen Xin Building, Room 705 • 12 Yumin Road, Dewai, Madian • Beijing 100029 • 8610-6202-2886 • Fax: 8610-6202-2892

FINLAND:

Tietäjäsentie 2 • 02130 Espoo • Phone: 09-54 75 08 10 • Fax: 09-25 10 51 00

FRANCE:

3, allée des Garays • 91127 Palaiseau Cédex • 01-64 53 20 20 • Fax: 01-60 11 77 26

GERMANY:

Landsberger Strasse 65 • 82110 Germering • 089/84 93 07-40 • Fax: 089/84 93 07-34

GREAT BRITAIN:

Unit 2 Commerce Park, Brunel Road • Theale • Berkshire RG7 4AB • 0118 929 7500 • Fax: 0118 929 7519

INDIA:

Flat 2B, Willocrissa • 14, Rest House Crescent • Bangalore 560 001 • 91-80-509-1320/21 • Fax: 91-80-509-1322

ITALY:

Viale San Gimignano, 38 • 20146 Milano • 02-48 39 16 01 • Fax: 02-48 30 22 74

JAPAN:

New Pier Takeshiba North Tower 13F • 11-1, Kaigan 1-chome • Minato-ku, Tokyo 105-0022 • 81-3-5733-7555 • Fax: 81-3-5733-7556

KOREA:

2FL., URI Building • 2-14 Yangjae-Dong • Seocho-Gu, Seoul 137-888 • 82-2-574-7778 • Fax: 82-2-574-7838

NETHERLANDS:

Postbus 559 • 4200 AN Gorinchem • 0183-635333 • Fax: 0183-630821

SWEDEN:

c/o Regus Business Centre • Frosundaviks Allé 15, 4tr • 169 70 Solna • 08-509 04 679 • Fax: 08-655 26 10

SWITZERLAND:

Kriesbachstrasse 4 • 8600 Dübendorf • 01-821 94 44 • Fax: 01-820 30 81

TAIWAN:

1FL., 85 Po Ai Street • Hsinchu, Taiwan, R.O.C. • 886-3-572-9077 • Fax: 886-3-572-9031

The
M1000/M2000
User Guide

Revision A, - May 1988
Copyright © Keithley MetraByte Corp. 1988
Part Number: 24744

Warranty Information

All products manufactured by Keithley MetraByte are warranted against defective materials and workmanship for a period of one year from the date of delivery to the original purchaser. Any product that is found to be defective within the warranty period will, at the option of Keithley MetraByte, be repaired or replaced. This warranty does not apply to products damaged by improper use.

Warning

Keithley MetraByte assumes no liability for damages consequent to the use of this product. This product is not designed with components of a level of reliability suitable for use in life support or critical applications.

Disclaimer

Information furnished by Keithley MetraByte is believed to be accurate and reliable. However, the Keithley MetraByte Corporation assumes no responsibility for the use of such information nor for any infringements of patents or other rights of third parties that may result from its use. No license is granted by implication or otherwise under any patent rights of Keithley MetraByte Corporation.

Notes

Keithley MetraByte/Asyst/DAC is also referred to here-in as *Keithley MetraByte*.

Basic[™] is a trademark of Dartmouth College.

IBM[®] is a registered trademark of International Business Machines Corporation.

PC, XT, AT, PS/2, and **Micro Channel Architecture**[®] (MCA) are trademarks of International Business Machines Corporation.

Microsoft[®] is a registered trademark of Microsoft Corporation.

Turbo C[®] is a registered trademark of Borland International.

New Contact Information

Keithley Instruments, Inc.
28775 Aurora Road
Cleveland, OH 44139

Technical Support: 1-888-KEITHLEY
Monday – Friday 8:00 a.m. to 5:00 p.m (EST)
Fax: (440) 248-6168

Visit our website at <http://www.keithley.com>

The following safety precautions should be observed before using this product and any associated instrumentation. Although some instruments and accessories would normally be used with non-hazardous voltages, there are situations where hazardous conditions may be present.

This product is intended for use by qualified personnel who recognize shock hazards and are familiar with the safety precautions required to avoid possible injury. Read and follow all installation, operation, and maintenance information carefully before using the product. Refer to the manual for complete product specifications.

If the product is used in a manner not specified, the protection provided by the product may be impaired.

The types of product users are:

Responsible body is the individual or group responsible for the use and maintenance of equipment, for ensuring that the equipment is operated within its specifications and operating limits, and for ensuring that operators are adequately trained.

Operators use the product for its intended function. They must be trained in electrical safety procedures and proper use of the instrument. They must be protected from electric shock and contact with hazardous live circuits.

Maintenance personnel perform routine procedures on the product to keep it operating properly, for example, setting the line voltage or replacing consumable materials. Maintenance procedures are described in the manual. The procedures explicitly state if the operator may perform them. Otherwise, they should be performed only by service personnel.

Service personnel are trained to work on live circuits, and perform safe installations and repairs of products. Only properly trained service personnel may perform installation and service procedures.

Keithley products are designed for use with electrical signals that are rated Installation Category I and Installation Category II, as described in the International Electrotechnical Commission (IEC) Standard IEC 60664. Most measurement, control, and data I/O signals are Installation Category I and must not be directly connected to mains voltage or to voltage sources with high transient over-voltages. Installation Category II connections require protection for high transient over-voltages often associated with local AC mains connections. Assume all measurement, control, and data I/O connections are for connection to Category I sources unless otherwise marked or described in the Manual.

Exercise extreme caution when a shock hazard is present. Lethal voltage may be present on cable connector jacks or test fixtures. The American National Standards Institute (ANSI) states that a shock hazard exists when voltage levels greater than 30V RMS, 42.4V peak, or 60VDC are present. **A good safety practice is to expect that hazardous voltage is present in any unknown circuit before measuring.**

Operators of this product must be protected from electric shock at all times. The responsible body must ensure that operators are prevented access and/or insulated from every connection point. In some cases, connections must be exposed to potential human contact. Product operators in these circumstances must be trained to protect themselves from the risk of electric shock. If the circuit is capable of operating at or above 1000 volts, **no conductive part of the circuit may be exposed.**

Do not connect switching cards directly to unlimited power circuits. They are intended to be used with impedance limited sources. NEVER connect switching cards directly to AC mains. When connecting sources to switching cards, install protective devices to limit fault current and voltage to the card.

Before operating an instrument, make sure the line cord is connected to a properly grounded power receptacle. Inspect the connecting cables, test leads, and jumpers for possible wear, cracks, or breaks before each use.

When installing equipment where access to the main power cord is restricted, such as rack mounting, a separate main input power disconnect device must be provided, in close proximity to the equipment and within easy reach of the operator.

For maximum safety, do not touch the product, test cables, or any other instruments while power is applied to the circuit under test. ALWAYS remove power from the entire test system and discharge any capacitors before: connecting or disconnecting cables or jumpers, installing or removing switching cards, or making internal changes, such as installing or removing jumpers.

Do not touch any object that could provide a current path to the common side of the circuit under test or power line (earth) ground. Always make measurements with dry hands while standing on a dry, insulated surface capable of withstanding the voltage being measured.



The instrument and accessories must be used in accordance with its specifications and operating instructions or the safety of the equipment may be impaired.


Do not exceed the maximum signal levels of the instruments and accessories, as defined in the specifications and operating information, and as shown on the instrument or test fixture panels, or switching card.


When fuses are used in a product, replace with same type and rating for continued protection against fire hazard.

Chassis connections must only be used as shield connections for measuring circuits, NOT as safety earth ground connections.

If you are using a test fixture, keep the lid closed while power is applied to the device under test. Safe operation requires the use of a lid interlock.

If  or  is present, connect it to safety earth ground using the wire recommended in the user documentation.

The  symbol on an instrument indicates that the user should refer to the operating instructions located in the manual.

The  symbol on an instrument shows that it can source or measure 1000 volts or more, including the combined effect of normal and common mode voltages. Use standard safety precautions to avoid personal contact with these voltages.

The **WARNING** heading in a manual explains dangers that might result in personal injury or death. Always read the associated information very carefully before performing the indicated procedure.

The **CAUTION** heading in a manual explains hazards that could damage the instrument. Such damage may invalidate the warranty.

Instrumentation and accessories shall not be connected to humans.

Before performing any maintenance, disconnect the line cord and all test cables.

To maintain protection from electric shock and fire, replacement components in mains circuits, including the power transformer, test leads, and input jacks, must be purchased from Keithley Instruments. Standard fuses, with applicable national safety approvals, may be used if the rating and type are the same. Other components that are not safety related may be purchased from other suppliers as long as they are equivalent to the original component. (Note that selected parts should be purchased only through Keithley Instruments to maintain accuracy and functionality of the product.) If you are unsure about the applicability of a replacement component, call a Keithley Instruments office for information.

To clean an instrument, use a damp cloth or mild, water based cleaner. Clean the exterior of the instrument only. Do not apply cleaner directly to the instrument or allow liquids to enter or spill on the instrument. Products that consist of a circuit board with no case or chassis (e.g., data acquisition board for installation into a computer) should never require cleaning if handled according to instructions. If the board becomes contaminated and operation is affected, the board should be returned to the factory for proper cleaning/servicing.

M1000 SERIES

USERS MANUAL

REVISION: 3/30/87

The information in this publication has been carefully checked and is believed to be accurate; however, no responsibility is assumed for possible inaccuracies or omissions. Applications information in this manual is intended as suggestions for possible use of the products and not as explicit performance in a specific application. Specifications may be subject to change without notice.

M1000 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

Warranty 5

CHAPTER 1 Getting Started

Default Mode 1-1

Quick Hook-Up 1-2

CHAPTER 2 Functional Description

Block Diagram 2-4

CHAPTER 3 Communications

RS-232C 3-2

Single Module and Multi-party Connection 3-3

Software Considerations 3-4

Changing Baud Rate 3-5

Using a Daisy-Chain With a Dumb Terminal 3-6

RS-485 3-6

RS-485 Multidrop System 3-7

CHAPTER 4 Command Set

Table of Commands 4-7

User Commands 4-8

Error Messages 4-16

CHAPTER 5 Setup Information and Command

Command Syntax 5-2

Setup Hints 5-13

CHAPTER 6 Digital I/O Function

Digital Outputs 6-1

Digital Inputs 6-3

Events Counter 6-4

Alarm Outputs 6-5

On-Off Controller 6-5

Setpoint 6-9

CHAPTER 7 Power Supply

CHAPTER 8 Troubleshooting

CHAPTER 9 Calibration

Appendix A ASCII TABLE

Appendix B M1400 Data Sheet

Appendix C M1500 Data Sheet

Appendix D M1600 Data Sheet

WARRANTY

MetraByte Corp. warrants your M1000 series module to be free from defects in parts, materials and workmanship under normal use and service for a period of one year from the date of delivery, and will repair or replace, at its sole option, any defective unit brought to its attention during that period.

MetraByte Corp makes no implied warranty that the M1000 series modules will be suitable to your purpose.

Some states do not allow the exclusion of implied warranties or limited warranties, so the above may not apply to you.

In no event will MetraByte Corp. be liable to you for any damages, including lost profits, lost savings, or other incidental or consequential damages arising out of the use or inability to use this product, even if MetraByte or an authorized MetraByte dealer has been advised of the possibility of such damages, or for any claim by any other party.

MetraByte Corp. cannot assume responsibility for infringement of present or future patents or other third party rights resulting from the use of these products.

Some states do not allow the limitation or exclusion of liability for incidental or consequential damages so the above limitation or exclusion may not apply to you.

This warranty is void if the product has been repaired or altered except by MetraByte or has been subjected to misuse, negligence, or accident. In no case shall MetraByte's liability exceed the original purchase price. The aforementioned provisions do not extend the original warranty period of any product which has been repaired or replaced by MetraByte.

WARNING

The circuits and software contained in the M1000 Series modules are proprietary to MetraByte Corporation. 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 M1000 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. MetraByte is not responsible for any consequential damages.

RETURNS

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

CHAPTER 1

GETTING STARTED

Default Mode

All M1000 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. 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 could be very difficult to establish communications with a module whose address and baud rate are unknown. To overcome this difficulty, each module has an input pin labeled DEFAULT *. By connecting this pin to Ground, the module is placed in a known communications setup called Default Mode.

The Default Mode setup is: 300 baud, no parity, any address is recognized.

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

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

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

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

RS-232 & RS-485 Quick Hook-Up

Software is not required to begin using your M1000 module. We recommend that you begin to get familiar with the module by setting it up on the bench. Start by using a dumb terminal or a computer that acts like a dumb terminal. Make the connections shown in the quick hook-up drawings, Figures 1.1 or 1.2. Put the module in the default mode by grounding the Default* terminal. Initialize the terminal communications package on your computer to put it into the "terminal" mode. Since this step varies from computer to computer, refer to your computer manual for instructions.

Begin by typing \$1RD and pressing the Enter or Return key. The module will respond with an * followed by the data reading at the input. The data includes sign, seven digits and a decimal point. For example, if you are using a thermocouple module and measuring room temperature your reading might be *+00025.00. The temperature reading will initially be in °C which has been preset at the factory. Once you have a response from the module you can turn to the section on commands 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, alarms off, no echo and 2 character delay. Refer to the setup section to configure the module to your application. .

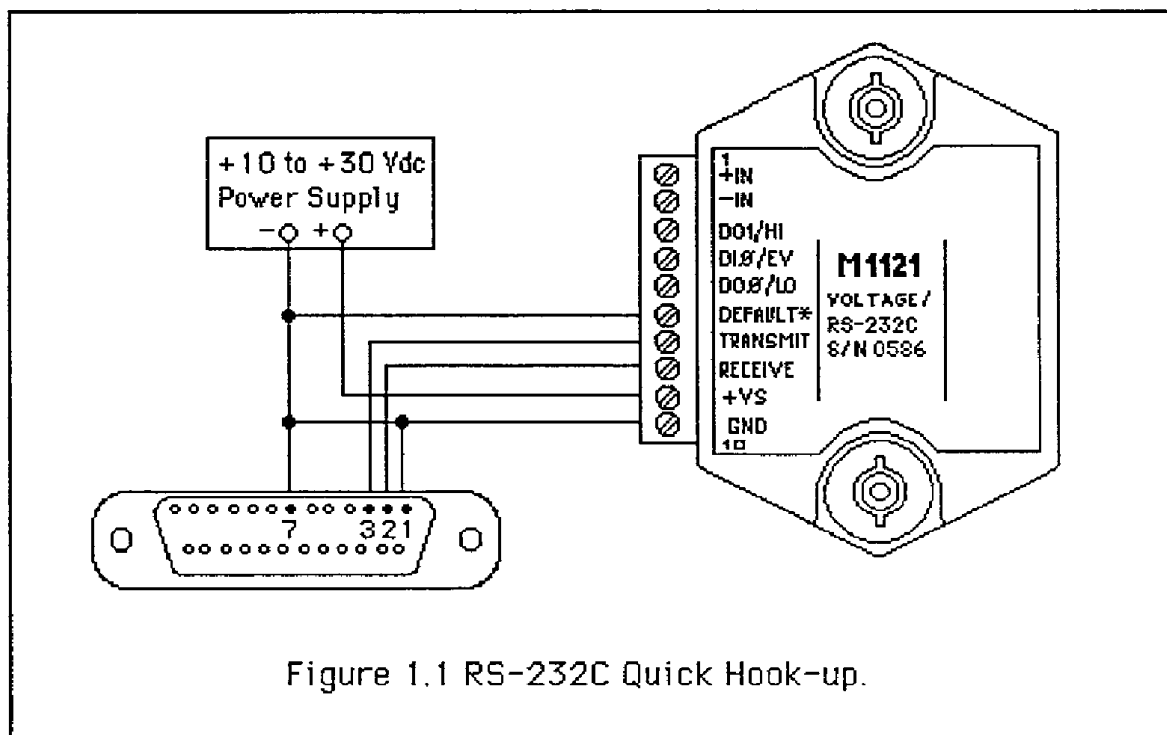
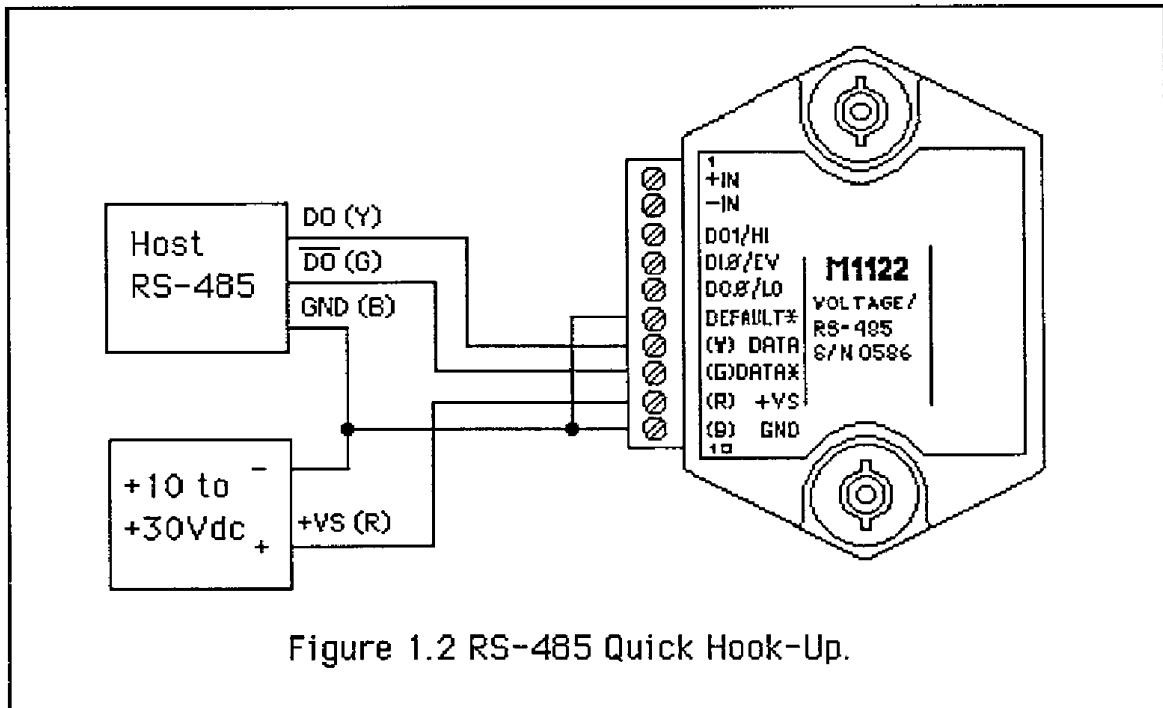


Figure 1.1 RS-232C Quick Hook-up.



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 50 ma and the RS-232C receive threshold is greater than 0V. All terminals that use 1488 and 1489 style interface IC's will satisfy this requirement. With this connection, characters generated by the terminal will be echoed back. To avoid double characters, the local echo on the terminal should be turned off.

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

In some rare cases it may be necessary to connect the module's DATA pin to ground through a 100Ω to 1kΩ resistor.

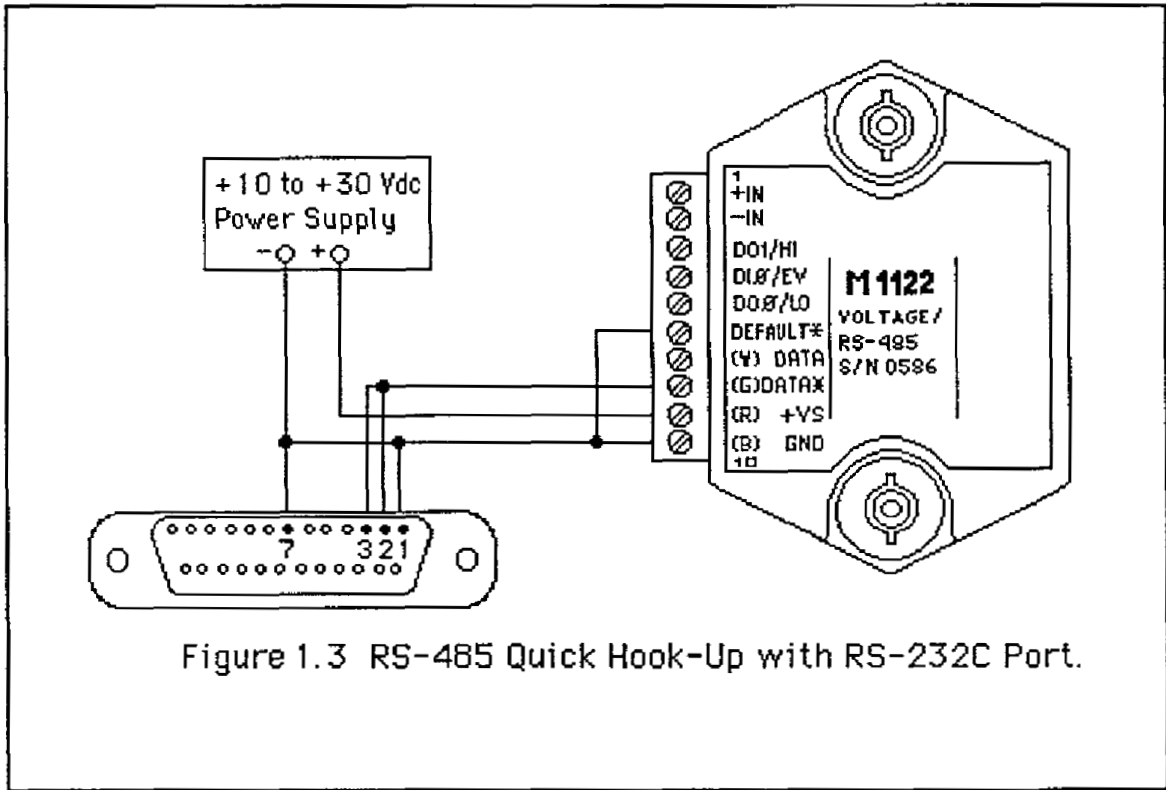


Figure 1.3 RS-485 Quick Hook-Up with RS-232C Port.

CHAPTER 2

FUNCTIONAL DESCRIPTION

A functional diagram of a typical MetraByte sensor module is shown in Figure 2.1 . It is a useful reference designed to illustrate the data path in the module and to explain the function of many of the module's commands.

The first step is to acquire the sensor signal and convert it to digital data. In Figure 2.1, all the signal conditioning circuitry has been lumped into one block, the analog-to-digital converter (A/D). Autozero and autocalibration is performed internally and is transparent to the user.

The full-scale output of the A/D converter may be trimmed using the Trim Span (TS) command. The TS command adjusts calibration values stored internally in the EEPROM. The TS command should only be used to trim the accuracy of the unit with a laboratory standard reference applied to the sensor input.

The trimmed data now flows into either of two digital filters. The filter selection is performed automatically by the microprocessor after every A/D conversion. The filter selection depends on the difference of the current A/D output data and the previous data stored in the output data register. If the least significant decimal digit from the A/D differs from the old output data by more than 10 counts, the large signal filter is selected. If the change is less than 10 counts, the small signal filter is used.

The two-filter system allows for different degrees of filtering depending on the rate of the input change. For steady-state signals, the small-signal filter averages out noise and small input changes to give a stable steady-state output. The large-signal filter is activated by step changes or very noisy input signals. The time constants for the two filters can be specified independently with the SetUp (SU) command. The filter values are stored in nonvolatile memory. Typically, the small-signal filter is set to a larger time constant than the large-signal filter. This gives very good noise rejection along with fast response to step inputs.

The MetraByte modules allow for user selectable output scaling in °C or °F on temperature data. This selection is depicted in Figure 2.1 as a switch following the digital filters. The default scaling in the modules is °C, but this may be converted to °F by feeding the data through a conversion routine. The switch position is controlled by a bit in the setup data and may be changed with the SetUp (SU) command. The scaling selection is nonvolatile. For non-temperature applications, the °C position should always be selected.

The scaled data is summed with data stored in the Output Offset Register to obtain the final output value. The output offset is controlled by the user and serves many useful purposes. The data in the Output Offset Register may be

used to trim any offsets caused by the input sensor. It may be used to null out undesired signal such as a tare weight. The Trim Zero (TZ) command is used to adjust the output to any desired value by loading the appropriate data value in the offset register. The data in the offset register is nonvolatile.

The output offset may also be modified using the Set Point (SP) command. The data value specified by the SP command is multiplied by -1 before being loaded into the register. The Set Point command specifies a null value that is subtracted from the input data. The output reading becomes a deviation value from the downloaded setpoint. This feature is very useful in on-off controllers as described in the digital I/O section of this manual.

The value stored in the offset register may be read back using the Read Zero (RZ) command. Data loaded in with the SP command will be read back with the sign changed. The output register may be reset to zero with the Clear Zero (CZ) command.

The output data may be read with the Read Data (RD) command. In some cases when a computer is used as a host, it may be possible to read back the same data value several times before it is updated with a new A/D conversion. To guarantee that the same data is not read more than once, the New Data (ND) command may be used. Each time an RD or ND command is performed, the New Data Flag is cleared. The flag is set each time the output data register is loaded as the result of a new A/D conversion. The ND command will wait until the flag is set before it outputs the data reading.

The remainder of Figure 2.1 depicts several functions known collectively as the Digital I/O section. It consist of a versatile alarm function, an event counter and general-purpose digital inputs and outputs. These functions are described in detail in the Digital I/O section.

The heart of the alarm section consists of two registers that are used to store high and low alarm limit values. These registers may be down-loaded with data values by using the HI and LO alarm commands. The alarm values are loaded with the same data format that is used with the output data. The high and low alarm registers are nonvolatile so they will not be lost when the unit is powered down. The values contained in the alarm registers may be read back at any time with the Read High (RH) and Read Low (RL) commands.

The data held in the alarm registers is continually compared with the calculated output data. The result of the comparison is used to trip alarms that may be used as control outputs. The high alarm is turned on when the output data exceeds the high limit value. The low alarm is activated if the output data is less than the low alarm value. Each alarm has two user selectable modes, either Momentary (M) or Latching (L). Momentary alarms are activated only while the alarm condition is met; if the output data returns within limits, the alarm is turned off. Conversely, when latching alarms are activated, they remain on even if the

output data returns within limits. Latching alarms are turned off with the Clear Alarms (CA) command or if the opposite alarm limit is exceeded.

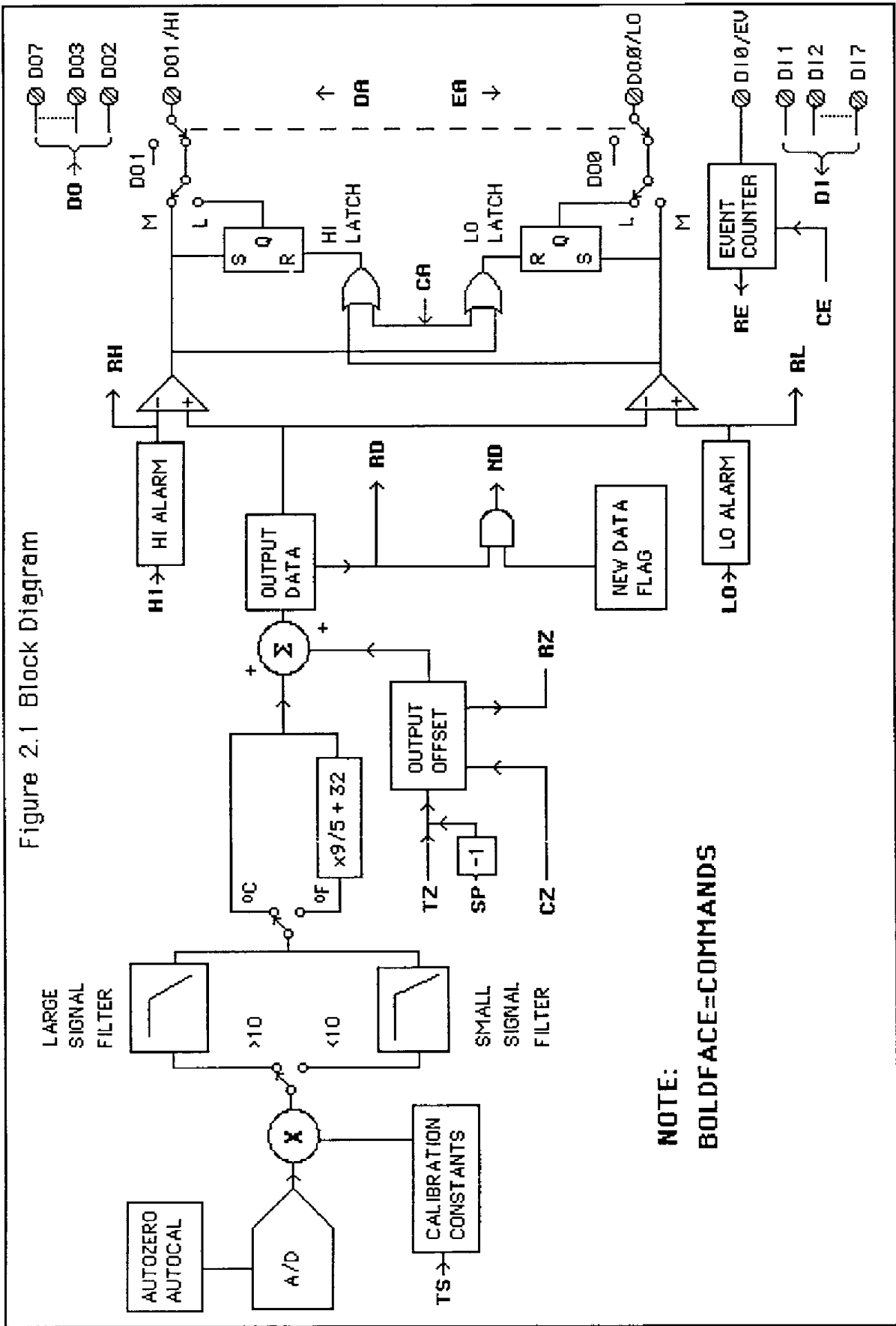
The state of the alarms may be read with the Digital Input (DI) command. Also, the alarm outputs may be used to activate digital outputs on the module connector to turn on alarms or to perform simple control functions. To help limit the number of terminals required on the module connector, the alarm outputs are shared with the general purpose digital output bits DO0 and DO1. To connect the alarm outputs to the connector, the Enable Alarm (EA) command is used. The connector pins may be switched back to the general-purpose digital outputs using the Disable Alarms (DA) command. The EA/DA selection is nonvolatile.

The general-purpose digital outputs are open-collector transistor switches that may be controlled by the host with the Digital Output (DO) command. They are designed to activate external solid-state relays to control AC or DC power circuits. The output may also be used to interface to other logic-level devices. The number of digital outputs available depends on the module type, with eight being the maximum.

The Digital Input (DI) command is used to sense the logic levels on the digital input pins DI0-DI7. The digital inputs are used to read logic levels generated by other devices. They are also useful to sense the state of electro-mechanical limit switches. The number of digital inputs varies with the module type.

The DI0 input is shared with the input to the Event Counter. The Event Counter is used to accumulate the number of positive transitions that have occurred on the DI0/EV connector pin. The counter can accumulate up to 9999999 (decimal) events and may be read with the Read Events (RE) command. The counter input is filtered and uses a Schmitt-trigger input to provide a bounce-free input for mechanical switches. The counter value may be zeroed with the Clear Events (CE) command.

Figure 2.1 Block Diagram



NOTE:
BOLDFACE=COMMANDS

CHAPTER 3

COMMUNICATIONS

Introduction

The M1000 series of interface modules has 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 standard interfaces such as RS-232C, no special machine language software drivers are necessary for operation. The modules can also be connected to auto-answer modems for long-distance operation without the need for a supervisory computer. The ASCII format also makes system debugging easy with a dumb terminal.

The MetraByte system is designed to allow 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 M1000 modules is performed with two-character ASCII command codes such as RD to Read Data from the analog input. A complete description of all commands is given in the Command Set section. 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 following table lists the timeout specification for each command:

Mnemonic	Timeout
DI,DO,RD,WE	10 ms.
ND	See text
All other commands	100 ms.

Table 3.1 Response Timeout Specifications.

This timeout specification indicates the turn-around time from the receipt of a command to when the module will start to transmit a response.

RS-232C

RS-232C is the most widely used communications standard for information transfer between computing equipment. RS-232C versions of the M1000 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 MetraByte 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 delays in multiple-module systems
- 5) less reliable—loss of one module results in no communications
- 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 M1000 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.

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

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

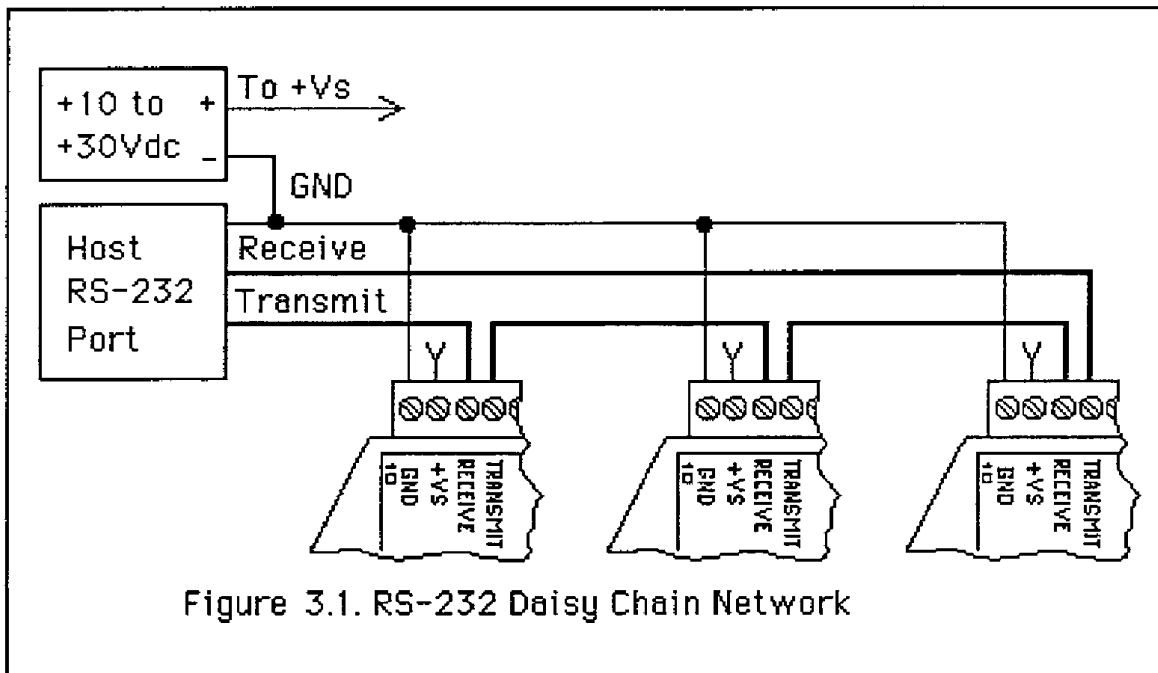


Figure 3.1. RS-232 Daisy Chain Network

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 usually 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 M1000 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	33.30 ms.
600	16.70 ms.
1200	8.33 ms.
2400	4.17 ms.
4800	2.08 ms.
9600	1.04 ms.

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 $4 \times 8.33 \text{ ms.} = 33.3 \text{ ms.}$ 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 Setup section) 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, a line break, 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 when using a computer 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

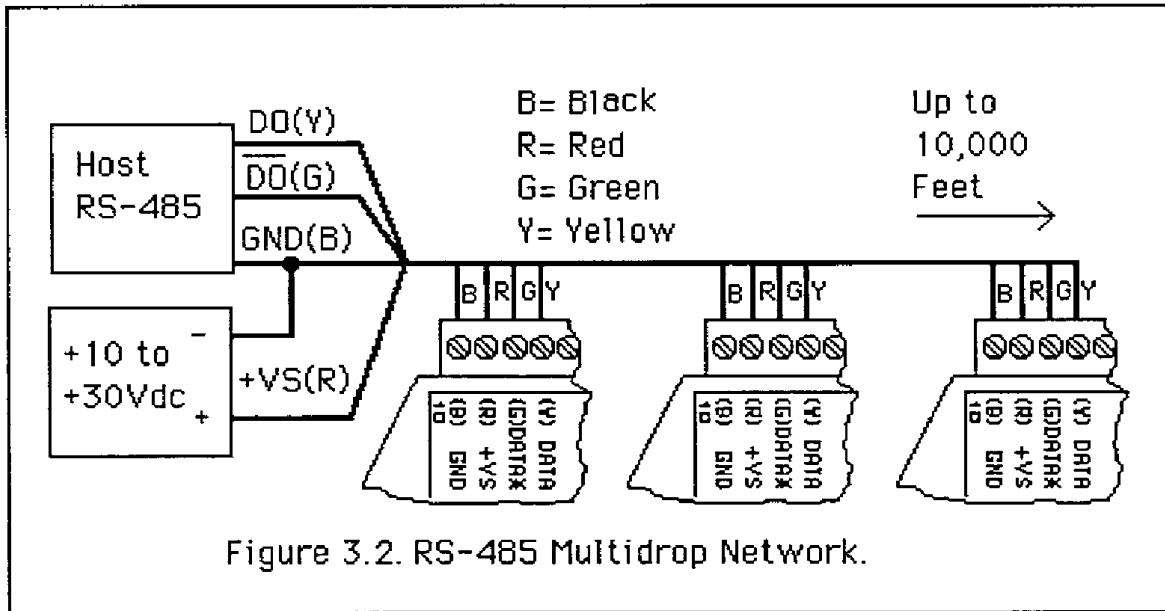
RS-485 is a recently developed communications standard to satisfy the need for multidropped systems that can communicate at high data rates over long distances. RS-485 is similar to RS-422 in that it uses a balanced differential pair of wires switching from 0 to 5 volts to communicate data. RS-485 receivers 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 be used to communicate with M1000 modules at 38400 baud
- 3) communications distances up to 10,000 feet.
- 4) true multidrop; modules are connected in parallel
- 5) modules can be disconnected without breaking 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

Of course, RS-485 does have its 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. This means that an RS-485 system usually requires the extra expense of an interface.

MetraByte Corp. offers interface converters to convert RS-232C and RS-422 to RS-485. For systems that require more than a few modules, long wiring distances, or high speed, RS-485 is recommended



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 operation of 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). These notations designate 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 on all MetraByte RS-485 equipment 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.

The notation '*' on the label DATA* is simply used to indicate the complement of DATA (negative true).

To minimize unwanted reflections on the transmission line, the bus should be arranged as a line going from one module to the next. 'Tree' or random structures of the transmission line should be avoided. For wire runs greater than 500 feet total, the end of the line should be terminated with a 100 ohm resistor connected between DATA and DATA*.

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. The resistance of 20 gauge wire commonly used in telephone cable is about 10 ohms per 1000 feet. The maximum current draw from a single module is 75 ma. Using Ohm's Law, the maximum allowable resistance in the GND lead to be within the -5V. common-mode condition is 66.7 ohms. For 20 gauge wire this results in a maximum bus length of 6670 feet for a single module. These calculations can be reduced to a general rule-of-thumb by taking the number of modules on the bus and multiplying it by the bus length in feet. The resultant number must be less than the number given in the following Table :

Wire Gauge	Maximum modules X feet
22	4000
20	6000
18	10000

Communications Delay

All M1000 modules with RS-485 outputs are setup at the factory to provide two units of communications delay after a command has been received (see Setup section). 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 M1000 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 (the long response format will be covered a little later).

The prompt character must be followed by a single address character identifying the module to which the command is directed. Each module attached to a common communications port must be setup with its own unique address so that commands may be directed to the proper unit. Module addresses are assigned by the user with the SetUp (SU) command. For best results, use 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-character command which identifies the function to be performed by the module. All of the available commands are listed in Table 4.1 along with a short function definition. All commands are described in full later in this section. Commands must be transmitted as upper-case characters.

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

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

Data Structure

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

The 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 M1000 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 specified, even though some digits may not be significant. Failure to do this will result 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, the M1300 thermocouple input modules feature 1 degree output resolution. A typical analog data value from this type of module could be +00123.00 . The two digits to the right of the decimal point have no significance in this particular model. 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, an alarm value may be stored in a M1000 module using the 'H!' command:

Command: \$1HI+12345.67M
Response: *

The alarm value may then be read back with the Read High (RH) command:

Command: \$1RH
Response: *+12345.60M

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 nonexistent.

Overload values of analog data are represented by +99999.99 and -99999.99 .

Data read back from the Event Counter with the Read Events (RE) command is in the form of a seven-digit decimal number with no sign or decimal point. Round-off errors do not occur on the event counter. For example:

Command: \$1RE
Response: *0000123

The Digital Input, Digital 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. These commands are write protected to guard against accidental loss of setup data. All write-protected commands must be preceded by a Write Enable (WE) command before the protected command may be executed.

Miscellaneous Protocol Notes

The address character must 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 M1000 module begin with either an asterisk '*' (ASCII \$2A) or a question mark '?' (ASCII \$3F) prompt. The '*' prompt indicates acknowledgment of a valid command. The '?' prompt precedes an error message. All response messages are terminated with a CR. Many commands simply return a single '*' character to acknowledge that the command has been executed by the module. Other commands send data information following the '*' prompt. The response format of all commands may be found in the detailed command description.

The maximum response message length is 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 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 4.1, a communications time-out error has occurred. After the communications time-out it is assumed that no response data is forthcoming. This error usually results when an improper command prompt or address is transmitted.

Long Form Responses

When the pound sign '#' command prompt is used, the module will respond with a 'long form' response. This type of response will echo the command message, supply the necessary response data, and will add a two-character checksum to the end of the message. Long form responses are used in cases where the host wishes to verify the command received by the module. The checksum is included to verify the integrity of the response data. The '#' command prompt may be used with any command. For example:

Command: \$1RD (short form)
Response: *+00072.10

Command: #1RD (long form)
Response: *1RD+00072.10A4 (A4=checksum)

Checksum

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

Command Checksum

A two-character checksum may be appended to any command to the M1000 module as a user option. When a module interprets a command, it looks for the two extra characters and assumes that it is a checksum. If the checksum is not present, the module will perform the command normally. If the two extra characters are present, the module will calculate the checksum for the

message. If the calculated checksum does not agree with the transmitted checksum, the module will respond with a 'BAD CHECKSUM' error message and the command will be aborted. If the checksums agree, the command will be executed. If the module receives a single extra character, it will respond with a 'SYNTAX ERROR' and the command will be aborted. For example:

Command: \$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 #1DOFF

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

Sum (hex addition) 23 + 31 + 44 + 4F + 46 + 46 = 173

The checksum is 73 (hex). Append the characters 7 and 3 to the end of the message: #1DOFF73

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:

$$\begin{array}{cccccccccccc} * & 1 & R & D & + & 0 & 0 & 0 & 7 & 2 & . & 1 & 0 \\ 2A+ & 31+ & 52+ & 44+ & 2B+ & 30+ & 30+ & 30+ & 37+ & 32+ & 2E+ & 31+ & 30 = 2A4 \end{array}$$

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

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

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

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

Parity bits are never included in the checksum calculation.

Table 4.1 M1000 Command Set

Command and Definition		Typical Command Message (\$ prompt)	Typical Response Message
DI	Read Alarms/Digital Inputs	\$1DI	*0003
DO	Set Digital Outputs	\$1DOFF	*
ND	New Data	\$1ND	*+00072.00
RD	Read Data	\$1RD	*+00072.00
RE	Read Event Counter	\$1RE	*0000107
RL	Read Low Alarm Value	\$1RL	*+00000.00 L
RH	Read High Alarm Value	\$1RH	*+00510.00 L
RS	Read Setup	\$1RS	*31070142
RZ	Read Zero	\$1RZ	*+00000.00
WE	Write Enable	\$1WE	*

Write Protected Commands.

CA	Clear Alarms	\$1CA	*
CE	Clear Events	\$1CE	*
CZ	Clear Zero	\$1CZ	*
DA	Disable Alarms	\$1DA	*
EA	Enable Alarms	\$1EA	*
HI	Set High Alarm Limit	\$1HI+12345.67L	*
LO	Set Low Alarm Limit	\$1LO+12345.67L	*
RR	Remote Reset	\$1RR	*
SU	Setup Module	\$1SU31070142	*
SP	Set Setpoint	\$1SP+00600.00	*
TS	Trim Span	\$1TS+00600.00	*
TZ	Trim Zero	\$1TZ+00000.00	*

M1000 User Commands

Note that in all command and response examples given below, a carriage return is implied after every character string.

Clear Alarms (CA)

The clear alarms command turns both the HI and LO alarms OFF. This command does not affect the enable/disable or momentary/latching alarm conditions. The alarms will continue to be compared to the input data after the CA command is given. In cases where the alarm condition persists, the alarms will be set at the end of the next input data conversion. The primary purpose of the CA command is to clear latching alarms. See the Alarms section for more information.

Command: \$1CA
Response: *

Command: #1CA
Response: *1CADF

Clear Events (CE)

The Clear Events command clears the events counter to 0000000.

Command: \$1CE
Response: *

Command: #1
Response: *1CEE3

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

Clear Zero (CZ)

The Clear Zero command clears the output offset register value to +00000.00. This command clears any data resulting from a Trim Zero (TZ) or SetPoint (SP) command.

Command: \$1CZ
Response: *

Command: #1CZ
Response: *1CZF8

Disable Alarms (DA)

Most models in the M1000 series feature LO/DO0 and HI/DO1 pins on the module connector. These pins serve a dual function and can be used to output either the alarm outputs or digital outputs 0 and 1. The Disable Alarms command is used to connect the digital outputs 0 and 1 to the connector pins. The alarm settings are not affected in any way except that the alarm outputs are disconnected from the module connector. The alarm status can still be read with the Digital Input (DI) command. The complement to the DA command is the Enable Alarms (EA) command.

Command: \$1DA

Response: *

Command: #1DA

Response: *1DAE0

Digital Input (DI)

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

Command: \$1DI

Response: *0003

Command: #1DI

Response: *1DI0003AB

Listed below are the four possible alarm states in the first digital input byte and their hex values.

- 00 Both HI and LO alarms off.
- 01 HI alarm off. LO alarm on.
- 02 HI alarm on. LO alarm off.
- 03 Both HI and LO alarms on.

The second byte displays the hex value of the digital input status. The number of digital inputs varies depending on module type.

<u>Digital Inputs</u>	<u>DI7</u>	<u>DI6</u>	<u>DI5</u>	<u>DI4</u>	<u>DI3</u>	<u>DI2</u>	<u>DI1</u>	<u>DI0</u>
Data Bits	7	6	5	4	3	2	1	0

For example: A typical response from a \$1DI command could be: *01FE. This response indicates that the HI alarm is off, the LO alarm is on, DI0 = 0 and all other digital inputs are = 1.

All digital inputs that are not implemented or left unconnected are read as '1'.

Digital input 0 serves a dual function. It is both a digital input and the Event Counter input.

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

Digital Output (DO)

The DO command controls eight bits of digital outputs on the module connector. The number of digital outputs implemented depends on the model used. The digital outputs allow the module to control external circuits under host command. The DO command requires an argument of two hex characters specifying the eight bits of output data.

Digital Outputs	DO7	DO6	DO5	DO4	DO3	DO2	DO1	DO0
Data Bits	7	6	5	4	3	2	1	0

The electrical implementation of the digital output consists of open-collector transistors wired to the module connector. If a digital output is set to '1' the corresponding transistor is turned on and sinks current. Note that when a digital output bit is set to '1' the electrical output is near 0 volts. If a digital output is set to '0' the corresponding transistor is turned off and sinks no current.

Assume a module has two digital outputs, and you wish to turn both outputs on (sinking current). Set data bit 0 and data bit 1 to '1'. Since the module has only two digital outputs, all the other bits are 'don't cares'. For example, this command will turn both outputs 'on':

Command: \$1DOFF
Response: •

To turn both outputs off you could use the command:

Command: \$1DO00
Response: •

Digital outputs 0 and 1 share connector pins with the HI and LO alarms. The Disable Alarms (DA) command is used to configure these pins as digital outputs.

Digital output settings are not stored in nonvolatile memory. If a power failure occurs, all digital outputs will be set to 0 upon power up.

The DO command is the only means of changing digital outputs. There is no software provision to read the state of the digital outputs.

Enable Alarms (EA)

Digital outputs DO0/LO and DO1/HI serve a dual purpose as digital outputs and alarms. Digital output 0 is shared with the LO alarm and digital output 1 is shared with the HI alarm. The Enable Alarms (EA) command configures the shared outputs to indicate alarm conditions and disconnects digital outputs 0 and 1. The EA command only affects the electrical output of the alarms to the pins. The alarm status can be read at any time with the Digital Input (DI) command. The complement to the EA command is the Disable Alarms (DA) command.

Command: \$1EA

Response: *

Command: #1EA

Response: *1EAE1

High Alarm Limit (HI)

The high alarm command sets the value and type of the high alarm. The data specified by the HI command is stored in nonvolatile memory and compared with the sensor data after every A/D conversion. The high alarm is activated if the input data is greater than the value stored by the HI command. The high alarm status may be read using the Digital Input (DI) command. The alarm may be used to activate a digital output by using the Enable Alarms (EA) command. The HI command also specifies whether the high alarm is momentary or latching. A letter indicating the alarm type, "L" for latching or "M" for momentary, must follow the alarm value. For example:

Command: \$1HI+00100.00M

Response: *

Command: #1HI+00100.00M

Response: *1HI+00100.00ME3

The alarm limit should be set within the output range of the module. If the alarm limit is set beyond the output range, the alarm will be activated only on an overload condition.

The high alarm value may be read back with the Read High Alarm (RH) command.

A latched alarm may be cleared with the Clear Alarms (CA) command. More information on alarms may be found in the Digital I/O section.

Low Alarm Limit (LO)

The low alarm command sets the value and type of the low alarm. The data specified with the LO command is stored in nonvolatile memory and compared with the sensor data after every A/D conversion. If the input data is less than the low limit, the low alarm is activated. The low alarm status may be read using the Digital Input (DI) command. The alarm may be used to activate a digital output by using the Enable Alarms (EA) command. A letter indicating the alarm type, "L" for latching or "M" for momentary, must follow the alarm value. For example:

Command: \$1LO+00000.00M

Response: *

Command: #1LO+00000.00M

Response: *1LO+00000.00MEC

The alarm limit should be set within the output range of the module. If the alarm limit is set beyond the output range, the alarm will be activated only on an overload condition.

The low limit value may be read back with the Read Low Limit (RL) command.

More information on alarms may be found in the Digital I/O section.

New Data Command (ND)

The New Data (ND) command is a variation of the Read Data (RD) command used to read sensor data from the module. The ND command guarantees that the output data has not been previously read.

The M1000 module acquires analog input data eight times a second and stores the result in the output buffer (see Figure 2.1). The Read Data (RD) command simply reads the results stored in the output buffer. A fast host communicating at a high baud rate could possibly read the output buffer several times before the information is updated with a new A/D conversion. This results in redundant information which may be confusing or may be a waste of host processor time.

Associated with the output buffer is the New Data Flag (see Figure 2.1). This flag is cleared each time an RD or ND command is performed. The flag is set when the module's microprocessor loads the output buffer with the result of the most recent A/D conversion. The ND command will output data only when the New Data Flag is set. If the flag is cleared when an ND command is received, the module will wait until new data is present in the output buffer before responding to the command. Thus, the output data obtained with an ND command is always the result of a new A/D conversion.

The ND command is especially useful with computers that handle communications on an interrupt basis. The ND command may be used to get the maximum possible throughput without producing redundant data.

Command: \$1ND
Response: *+00072.00

Command: #1ND
Response: *1ND+00072.009F

A special condition exists when using the ND command with the M1600 frequency/pulse modules. These modules differ from the other sensor input modules in that they require an input trigger signal to obtain new data. If no signal exists on the input of the M1600, an ND command will wait indefinitely for new data and the module will not respond.

In order to escape this condition, a single control-C (\$03) may be issued by the host to abort the ND command. The aborted ND command will respond with the data value currently stored in the output buffer. Be aware that on an RS-485 system, the control-C character may interfere with the ND output data, causing a communications collision.

Read Data (RD)

The read data command is the basic command used to read the buffered sensor data. The output buffer (Figure 2.1) allows the data to be read immediately without waiting for an input A/D conversion. For example:

Command: \$1RD
Response: *+00072.00

Command: #1RD
Response: *1RD+00072.10A4

Since the RD command is the most frequently used command in normal operation, a special shortened version of the command is available. If a module is addressed without a two-letter command, the module interprets the string as an RD command.

Command: \$1
Response: *+00072.10

Command: #1
Response: *1RD+00072.10A4

READ EVENTS (RE)

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

Command: \$1RE
Response: *0000107

Command: #1RE
Response: *1RE00001074A

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

The Event Counter count is not stored in nonvolatile memory. If power is removed, the Event Counter will reset to all 0's upon power up.

The Remote Reset (RR) command or a line break does not effect the value of the Event Counter.

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

Read High Alarm (RH)

The Read High alarm command reads the value and type of the high alarm previously loaded by the HI command. The alarm type can be either latching or momentary. A letter indicating the alarm type, "L" for latching or "M" for momentary, will follow the alarm value. For example:

Command: \$1RH
Response: *+00510.00L

Command: #1RH
Response: *1RH+00510.00LF0

The RH command may be used to verify the data loaded into nonvolatile memory by the HI command.

Read Low Alarm (RL)

The Read Low alarm command reads the value and type of the low alarm. The alarm type can be either latching or momentary. A letter indicating the alarm type, "L" for latching or "M" for momentary, will follow the alarm value. For example:

Command: \$1RL
Response: *+00000.00L

Command: #1RL
Response: *1RL+00000.00LEE

The RL command may be used to verify data loaded into the nonvolatile memory with the LO command.

Remote Reset (RR)

The reset command allows the host to perform a program reset on the module's microprocessor. This may be necessary if the module's internal program is disrupted by static or other electrical disturbances. Once a reset command is received, the module will recalibrate itself. The calibration process takes approximately 2 seconds. For example:

Command: \$1RR
Response: *

Command: #1RR
Response: *1RRFF

In general, the state of the digital outputs and the event counter will not be affected by the RR command. However, if data in the microprocessor's RAM (Random Access Memory) has been lost, the RR command will result in a full power-up reset.

Any commands sent to the module during the self-calibration sequence will result in a NOT READY error.

Read Setup (RS)

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

Command: \$1RS
Response: *31070142

Command: #1RS
Response: *1RS3107014292

The response contains the module's channel address, baud rate, averaging constants, °C/°F, and other parameters. Refer to the setup command (SU), and the Setup section of this manual for a full list of parameters contained within the setup information.

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

Read Zero (RZ)

The Read Zero command reads back the value stored in the Output Offset Register (Figure 2.1).

Command: \$1RZ
Response: *+00000.00

Command: #1RZ
Response: *1RZ+00000.00B0

The data read back from the Output Offset Register may be interpreted in several ways. The commands that affect this value are the Trim Zero (TZ), SetPoint (SP), and Clear Zero (CZ).

Setpoint (SP)

The data specified by the setpoint command is multiplied by -1 and loaded into the Output Offset Register (Figure 2.1). The SP command is useful in on-off controller applications and is described in detail in the Digital I/O section of this manual. The SP command may be used to null out sensor data to obtain a deviation output when the RD or ND commands are used.

Command: \$1SP+00450.00

Response: *

Command: #1SP+00450.00

Response: *1SP+00450.00B0

It is possible to load setpoint data that is beyond the output range of the sensor. In this case, the setpoint can never be reached by the sensor data unless an overload is present.

To clear a setpoint, use the Clear Zero (CZ) command.

The SP command will write over any data written into the Output Offset Register by the Trim Zero (TZ) command. If the Output Offset Register is used as a trim value, this must be accounted for by the host before using the SP command. The value stored in the offset register may be read back using the Read Zero (RZ) command.

The setpoint data or trim data in the Output Offset Register is saved in nonvolatile memory.

Setup Command (SU)

Each M1000 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 Section 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

Trim Span (TS)

The trim span command is the basic means of trimming the accuracy of a M1000 sensor module. The TS command loads a calibration factor into nonvolatile memory to trim the full-scale output of the signal conditioning circuitry. It is intended only to compensate for long-term drifts due to aging of the analog circuits, and has a useful trim value of $\pm 10\%$ of the nominal calibration set at the factory. It is not intended to be used to change the basic transfer function of the module. Full information on the use of the TS command may be found in the Calibration section.

Command: \$1TS+00500.00

Response: *

Command: #1TS+00500.00

Response: *1TS+00500.00B0

Caution! TS is the only command associated with the span trim. There is no provision to read back or clear erroneous information loaded by the TS command. Unwarranted use of the TS command may destroy the calibration of the unit which can only be restored by using laboratory calibration instruments in a controlled environment.

Trim Zero (TZ)

The Trim Zero command is used to load a value into the Output Offset Register (Figure 2.1) to null out an undesirable offset in the output data. It may be used to trim offsets created by sensors such as strain gages. It may also be used to null out data to create a deviation output.

Example: Assume a M1511 bridge input module is being used with a load cell for weight measurement. An initial reading of the load cell with no weight applied may reveal an initial offset error:

Command: \$1RD

Response: *+00005.00

With no weight applied, we would like to trim the output to read zero. To trim the system, use the TZ command and specify the desired output reading:

Command: \$1TZ+00000.00 (zero output)
Response: *

The TZ command will load a data value into the Output Offset Register to force the output to read zero. The module will compensate for any previous value loaded into the Output Offset Register. If another output reading is taken, it will show that the offset has been eliminated:

Command: \$1RD
Response: *+00000.00

Although the TZ command is most commonly used to null an output to zero, it may be used to offset the output to any specified value. Assume that with the previously nulled load cell system we performed this command:

Command: \$1TZ-000100.00
Response: *

The new data output with no load applied would be:

Command: \$1RD
Response: *-000100.00

The load cell output is now offset by -100.

The offset value stored by the TZ command is stored in nonvolatile memory and may be read back with the Read Zero (RZ) command and cleared with the Clear Zero (CZ) command.

The SetPoint (SP) command will write over any value loaded by the TZ command.

Write Enable (WE)

Each MetraByte module is write protected against accidental changing of alarms, limits, setup, or span and zero trims. To change any of these write protected parameters, 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 remains write enabled until a command is successfully completed resulting in an '*' prompt. This lets the user correct the command error without having to execute another WE command.

ERROR MESSAGES

The M1000 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 the error description. The error messages have the same format for either the '\$' or '#' command prompts. For example:

?1 SYNTAX ERROR

There are eight possible error messages, and each error message description begins with a different character. This makes it easy for a computer program to identify the error without having to read the entire 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 occurs due to noise or interference on the communications line. In many cases, repeating the command will solve the problem. If the error persists, either the checksum is being calculated incorrectly or

there is a problem with the communications channel. In some cases, more reliable transmissions may be obtained by using a lower baud rate.

COMMAND ERROR

This error occurs when the two-character 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.

NOT READY

If a module is reset, it performs a self-calibration routine which takes 2-3 seconds to complete. Any commands sent to the module during the self-calibration period will result in a NOT READY error. When this occurs, simply wait a couple seconds and repeat the command.

The module may be reset in four ways: a power-up reset, a Remote Reset (RR) command, a line break, or an internal reset. All modules contain a 'watchdog' timer to ensure proper operation of the microprocessor. The timer may be tripped if the microprocessor is executing its program improperly due to power transients or static discharge.

If the NOT READY error persists for more than 30 seconds, check the power supply to be sure it is within specifications.

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 also be changed or turned off by using the Default Mode.

SYNTAX ERROR

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

VALUE ERROR

This error results when an incorrect character is used as a numerical value. Data values can only contain decimal digits 0-9. Hex values used in the SetUp (SU) and Digital Output (DO) commands can range from 0-F.

WRITE PROTECTED

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

CHAPTER 5

SETUP INFORMATION & COMMAND

The MetraByte M1000 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 MetraByte 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)**
- Alarm enable/disable**
- Alarm momentary/latching**
- CJC disable (M1300 series)**
- RTD 3/4 wire (M1400 series)**
- Positive/negative edge trigger (M1600 series)**
- Fahrenheit/Celsius**
- Echo**
- Communication delay (0-6 characters)**
- Number of displayed digits**
- Large-signal filter constant**
- Small-signal filter constant**

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. Once you are completely familiar with the setups, you can refer to Table 5.6 for a summary of all of the setup information.

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: \$1SU31070182.

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. The Appendix 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 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 \$1SU31070080, 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: \$1SU32070080. 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 M1000 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 MetraByte module 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, performing a line break (see Communications), or powering down. This extra level of write protection is necessary to ensure that communications to the module is not accidentally lost. This is very important when changing the baud rate of an RS-232C string. For more information on changing baud rate, refer to the Communications section.

Let's run through an example of changing the baud rate. Assume our sample module contains the setup data value of '31070080'. 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: *31070080

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: \$1SU31020080
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: *31020080

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 the Communications section for instructions.

Bits 3 and 4

These two bits of byte 2 are not used by the M1000 series and should be set to '0'.

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

BYTE2

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

Byte 3

This byte contains the setup information for several seldom-used options. The default value for this byte is '01'.

Alarm Enable

Bit 7 determines if the outputs from the LO and HI alarms are connected to module terminal block. If the value is '0' the alarms are not connected to the terminal block. In this condition the outputs are controlled by the Digital Output (DO) command. If bit 7 is '1' the alarms are connected to the terminal block. This bit is also controlled by the Enable Alarms (EA) command which sets the bit to '1'. The Disable Alarms (DA) command clears the bit to '0'.

Low Alarm Latch

Bit 6 determines whether the LO Alarm is latching or momentary. A '1' indicates that the alarm is latching; '0' indicates a momentary alarm. Bit 6 is also controlled by the LO Alarm (LO) command.

High Alarm Latch

Bit 5 determines whether the HI Alarm is latching or momentary. A '1' indicates latching. Bit 5 is also controlled individually by the HI Alarm (HI) command.

Disable CJC

RTD 3/4 Wire

Trigger Edge Select

The setup information stored in bit 4 has different meanings depending on the M1000 model number.

Disable CJC; this functions pertains only to the M1300 series of thermocouple input modules. If the bit is set to '1' the Cold Junction Compensation is disabled. The module calculates the temperature output with a fixed cold junction temperature of 0°C. This setup is useful for calibrating the module or in cases where remote CJC is used. Normally this bit is cleared to '0'.

RTD 3/4 Wire; this functions pertains only to the M1400 series of RTD input modules. If the bit is set to '1', the module provides the correct lead-compensation calculation for 4-wire RTD's. If the bit is cleared to '0', the module calculates the correct lead compensation for 3-wire RTD's. Measurement errors may result if the module is not set to the correct sensor type.

Trigger Edge Select; this function pertains only to the M1600 series frequency and pulse modules. Bit 4 determines the polarity of the edge used to trigger the measurement cycle. If bit 4 is '1' then the measurement cycle is

started on a positive-going edge. The measurement cycle is started on the negative-going edge if bit 4 is '0'. In general, this setup has very little effect on frequency inputs. It is primarily used with pulse inputs where the pulse train deviates from 50% duty cycle.

Celsius/Fahrenheit

The default scaling for temperature output modules is Celsius which is selected by making bit 3 = 0. To change the scaling to Fahrenheit, set bit 3 to '1'. All modules that do not have temperature output must have bit 3 cleared to zero. The scaling factors are operative only on the sensor data; HI and LO limits and setpoints must be modified by appropriate commands to reflect a scaling change (see Figure 2.1).

Echo

When bit 2 is set to '1', the M1000 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 the Communications section for a more complete description.

Delay

Bits 0 and 1 specify a minimum turn-around delay between a command and the module response. This delay time is useful on host systems that are not fast enough to capture data from quick-responding commands such as RD. This is particularly true for systems that use software UART's. The specified delay is added to the typical command delays listed in the Communications section. Each unit of delay specified by bits 0 and 1 is equal to 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 M1000 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
ALARMS OFF	0
ALARMS ON	1
HIGH ALARM MOMENTARY	0
HIGH ALARM LATCHING	1
LOW ALARM MOMENTARY	0
LOW ALARM LATCHING	1
CJC (D1300'S)	0
NO CJC (D1300'S)	1
3 WIRE (D1400'S)	0
4 WIRE (D1400'S)	1
- STARTING EDGE (D1600'S)	0
+ STARTING EDGE (D1600'S)	1
CELSIUS	0
FAHRENHEIT	1
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 digital filter time constants.

Number of displayed digits

For ease of use, the data outputs of all M1000 modules are standardized to a common 7-digit output consisting of sign, 5 digits, decimal point, and two more digits. Typical output data looks like: +00100.00. However, best-case resolution of the A/D converter is 1 part in 50,000 or about 4 1/2 digits. In some cases, the resolution of the output format is much greater than the resolution of the measurement system. In such cases, the trailing digits of the response would display meaningless information. Bits 6 and 7 are used to insert trailing zeros into the output data 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)

For example, the M1411 model for RTD's has .1 degree output resolution. The appropriate number of digits for this module is 6, to mask off the .01 digit which has no meaningful data. In some cases, the user may want to limit the output resolution to 1 degree. To do this, select bits 6 and 7 to display 5 digits. With this selection, the rightmost-two digits will always be set to '0'.

The number of displayed digits affects only data received from an RD or ND command.

Large Signal Filter, Bits 3,4,5

Small Signal Filter, Bits 0,1,2

The M1000 series modules contain a versatile single-pole, low-pass digital filter to smooth out unwanted noise caused by interference or small signal variations. The digital filter offers many advantages over traditional analog filters. The filtering action is done completely in firmware and is not affected by component drifts, offsets, and circuit noise typically found in analog filters. The filter time constant is programmable through the SetUp (SU) command and can be changed at any time, even if the module is remote from the host.

The digital filter features separate time constants for large and small signal variations. The Large Signal Filter time constant is controlled by bits 3,4,5. This time constant is used when large signal variations are present on the input. The

Small Signal Filter time constant is controlled by bits 0,1,2. This filter time constant is automatically selected when input signal variations are small. The microprocessor in the MetraByte module automatically selects the correct filter constant after every A/D conversion. The constant selected depends on the magnitude of the change of the input signal and the setup for the number of digits displayed. The microprocessor always keeps the value of the last calculated output to compare to a new data conversion. If the new data differs from the last output by more than ten counts of the last displayed digit, the large signal time constant is used in the digital filter. If the result of the most recent A/D conversion differs from the last output value by less than ten counts of the last displayed digit, the small signal time constant is used. Let's look at an example:

The M1411 module for RTD's has a standard output resolution of .1 degrees. The standard number-of -displayed-digits setup for this module is 6 digits, specified by byte 4 of the setup data. Therefore, the large signal filter will be selected if a new input conversion differs from the previous value by more than 1.0 degree:

Previous data	New data	Filter selected (large/small)
+00100.00	+00100.50	small
+00100.00	+00101.50	large
+00100.00	+00099.90	small
+00100.00	+00098.90	large
-00050.50	-00050.00	small
-00050.50	-00060.00	small

If the number of displayed digits is changed to reduce output resolution, the filter selection is also affected. If the number of displayed digits in the previous example is changed to 5, the output resolution becomes 1.0 degree.

In this case the large signal time constant is used if the new reading differs from the old by more than 10.0 degrees:

Previous data	New data	Filter selected (large/small)
+00100.00	+00105.00	small
+00100.00	+00111.00	large
+00100.00	+00091.00	small
+00100.00	+00085.00	large
-00050.00	-00045.00	small
-00050.00	-00039.00	large

Large Signal Time Constant

The large signal filter time constant is specified by bits 3,4,5 of byte 4. It may be specified from 0 (no filter) to 16 seconds. The time constant for a first-order filter is the time required for the output to reach 63% of its final value for a step input.

Small Signal Time Constant

Bits 0,1,2 specify the filter time constant for small signals. Its values are similar to the ones for the large signal filter. Most sensors can benefit from a small amount of small signal filtering such as $T = 0.5$ sec. In most applications, the small signal time constant should be larger than the large signal time constant. This gives stable readings for steady-state inputs while providing fast response to large signal changes.

Table 5.4 Byte 4 Displayed Digits and Filter Time Constants.

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						
NO LARGE SIGNAL FILTERING			0	0	0			
0.25 SECOND TIME CONSTANT			0	0	1			
0.5 SECOND TIME CONSTANT			0	1	0			
1.0 SECOND TIME CONSTANT			0	1	1			
2.0 SECOND TIME CONSTANT			1	0	0			
4.0 SECOND TIME CONSTANT			1	0	1			
8.0 SECOND TIME CONSTANT			1	1	0			
16.0 SECOND TIME CONSTANT			1	1	1			
NO SMALL SIGNAL FILTERING						0	0	0
0.25 SECOND TIME CONSTANT						0	0	1
0.5 SECOND TIME CONSTANT						0	1	0
1.0 SECOND TIME CONSTANT						0	1	1
2.0 SECOND TIME CONSTANT						1	0	0
4.0 SECOND TIME CONSTANT						1	0	1
8.0 SECOND TIME CONSTANT						1	1	0
16.0 SECOND TIME CONSTANT						1	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 M1111 unit and you wish to set the unit to echo so that it may be used in a daisy-chain (See Communications). Read out the current setup with the Read Setup command:

Command: \$1RS
Response: *310701C2

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: \$1SU310705C2
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.

The DA, EA, HI, and LO commands affect some of the bits of the setup data that are associated with alarms. If these commands are performed, the setup data read back with the Read Setup (RS) command may not correspond exactly with the data previously written with the SetUp (SU) command.

Table 5.5 Factory Setups by Model.

(All modules from the factory are set for address '1', 300 baud, no parity)

<u>Model</u>	<u>Setup Message</u>
M111X, M121X, M123X, M125X	310701C2
M112X, M124X	31070182
M113X, M114X	31070142
M13XX	31070142
M14XX	31070182
M15XX	310701C2
M16XX	310701C0
M170X	31070100

CHAPTER 6

DIGITAL I / O FUNCTION

The M1000 series features versatile digital I/O capability to interface to auxiliary equipment. The functions available are:

- 1) Digital Outputs
- 2) Digital Inputs
- 3) Alarm Outputs
- 4) Events Counter

Digital Outputs

A digital output consists of an open-collector transistor controlled by the host, using the Digital Output (DO) command (See Figure 6.1). The number of digital outputs implemented depends on the specific M1000 model number. Most sensor modules contain two digital outputs and the M1701/2 has eight digital outputs. The open-collector configuration is used to provide maximum versatility in interfacing to solid state relays (SSR's) or to standard logic levels such as TTL or CMOS. Each digital output can sink up to 30mA and can withstand up to 30V. Power in the transistor must be limited to 300 mW. The emitter of each transistor is tied to the GND terminal on the input connector.

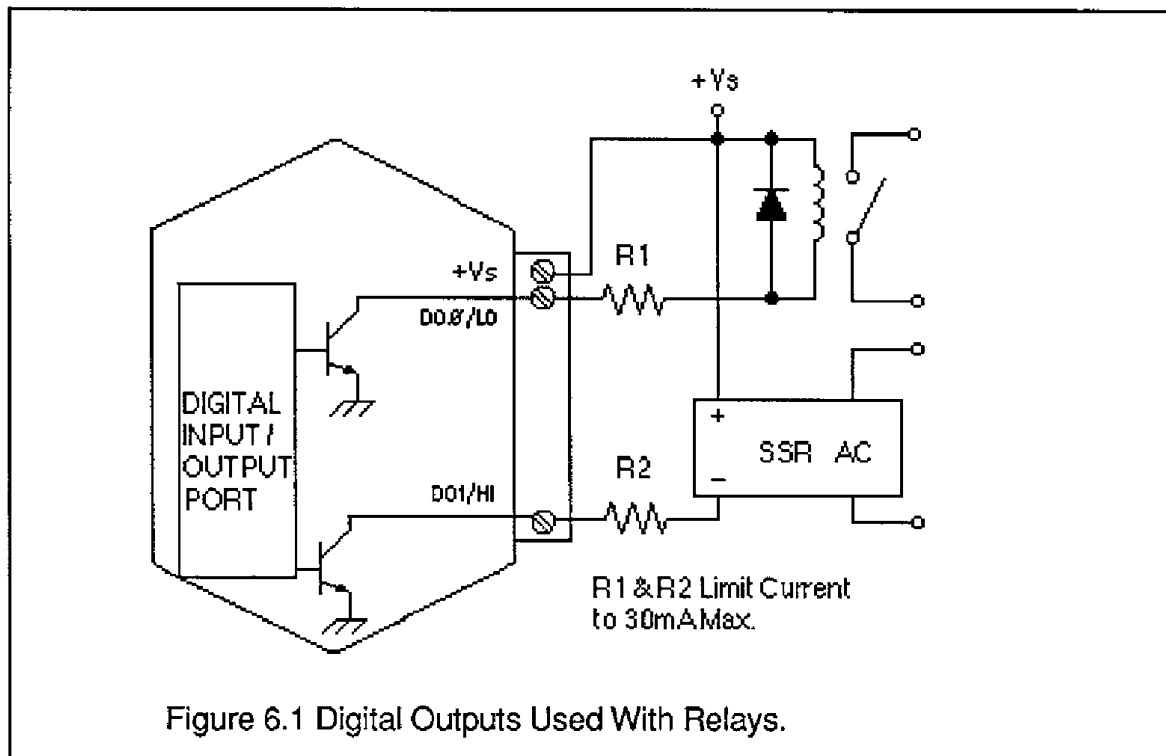


Figure 6.1 Digital Outputs Used With Relays.

A typical connection of a digital I/O output is shown in Figure 6.1. In this case, a solid state relay is controlled by the M1000 module. The SSR can then be used to control AC power to alarms, heaters, pumps, etc. A typical connection to a logic input is shown in Figure 6.2. In some cases, the common-mode voltage of the GND terminal may be significantly different from the ground potential of the logic input to be interfaced. This may occur when the module is powered remotely. In this case, an opto-isolator may be used to eliminate the common-mode voltage. See Figure 6.2. In all cases, the current switched by the transistor may not be more than 30 mA.

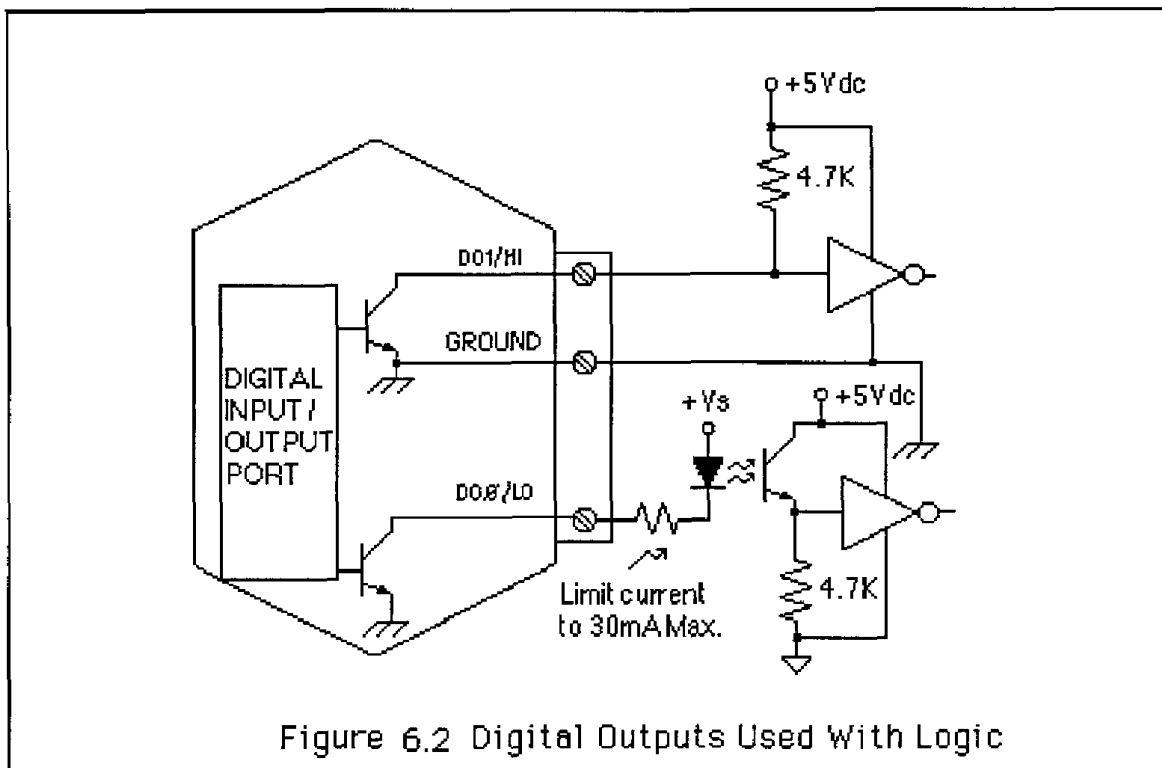


Figure 6.2 Digital Outputs Used With Logic

Only three commands can effect the Digital Output. The Enable Alarms (EA) and Disable Alarms (DA) commands select the function of the DO0/LO and DO1/Hi pin outputs. To route digital outputs to these pins, use the Disable Alarms (DA) command. The Enable Alarms (EA) command configures these two pins as alarm outputs. The Enable Alarms and Disable Alarms commands do not affect the other digital outputs, DO2-DO7. The digital outputs are controlled by the host with the Digital Output (DO) command.

If the module loses power, the digital outputs are turned off. The outputs will remain off until switched by a Digital Output (DO) command. The function of the shared pins, DO0/LO and DO1/Hi is not affected if power is lost, since this information is stored in nonvolatile memory.

The digital outputs are not affected by the Remote Reset (RR) command or a reset caused by a line break.

DIGITAL INPUTS

Digital inputs are used to sense switch closures and the state of digital signals. The inputs are protected to voltages up to $\pm 30V$ and are normally pulled up to the logic "1" condition (see Figure 6.3). Digital inputs can be read by the Digital Input (DI) command. Voltage inputs less than 1 V are read back as '0'. Signals greater than 3.5 V are read as '1'. No other commands have any affect on the inputs.

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 0.5 mA; 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.

Connection to logic outputs is shown in Figure 6.2. Opto-isolation is used for isolation and where common-mode exists between the M1000 module and the signal being sensed.

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

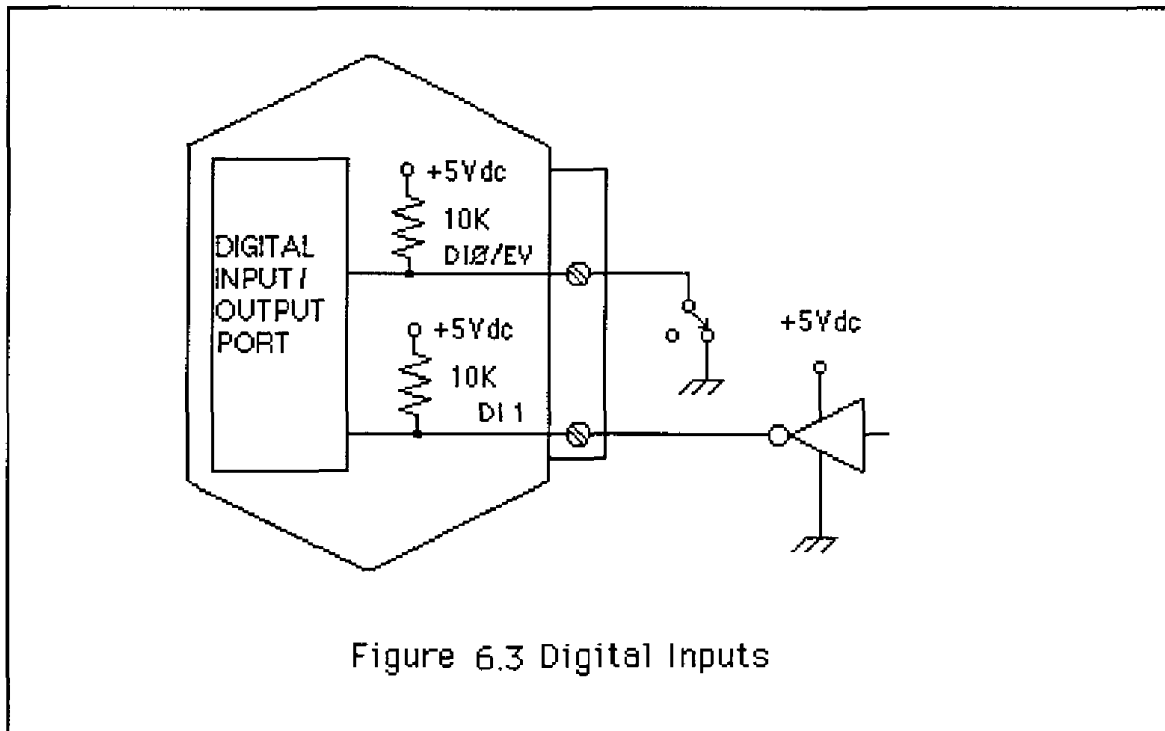


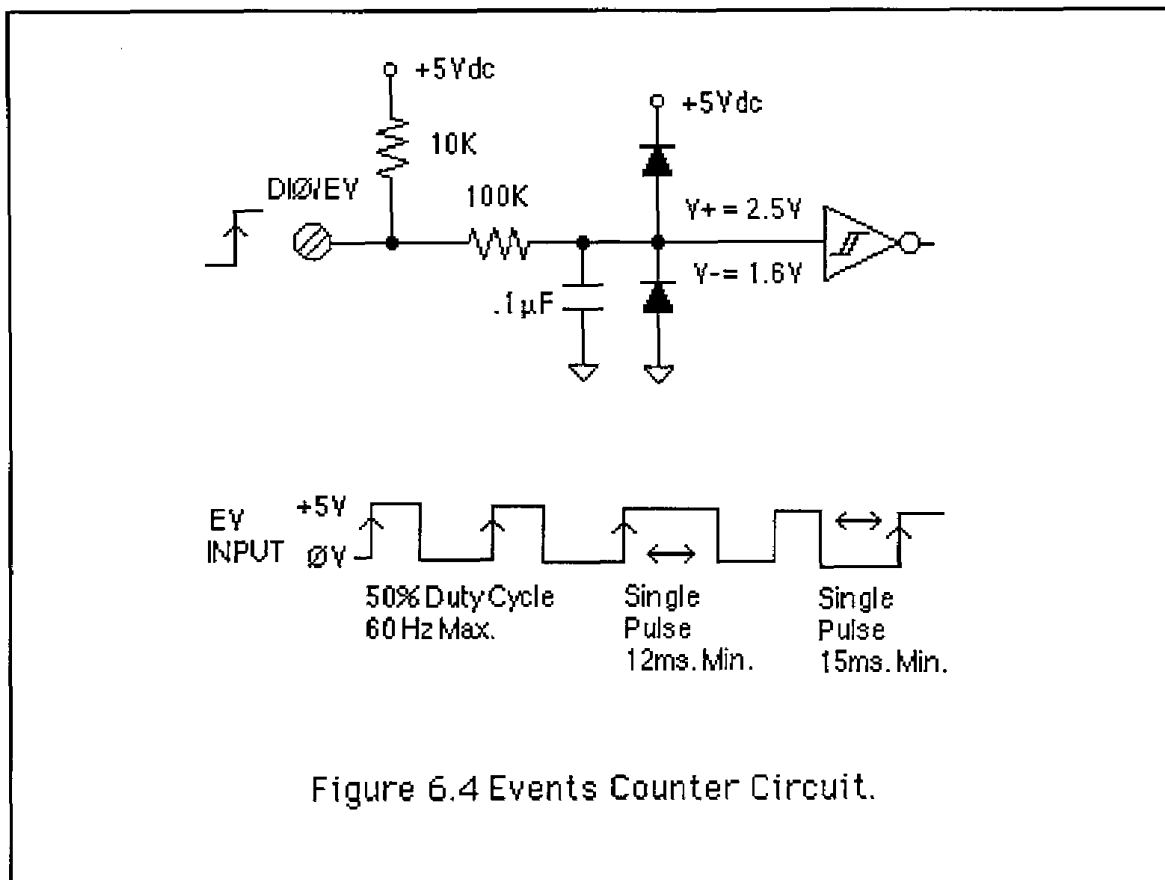
Figure 6.3 Digital Inputs

EVENT COUNTER

The Event Counter input is connected to the Digital Input 0 terminal. It can be used to count any low speed event that occurs on the DI0/EV input. Any of the interfacing techniques described for Digital Inputs may be used. The input pulses must meet the specifications in Figure 6.4 to avoid missing counts. Switch inputs are filtered to eliminate contact bounce.

The Event Counter is read by using the Read Events (RE) command. The maximum accumulated count is 9,999,999. If the maximum count is reached, counting stops. The Event Counter may be cleared to zero with the Clear Events (CE) command.

The Event Counter is not nonvolatile and the count will be lost if power to the module goes down. Upon power up, the counter is cleared to zero. The Remote Reset (RR) command or a line break will not affect the counter.



ALARM OUTPUTS

The M1000 sensor input modules perform HI/LO limit checking by comparing the sensor input value to downloaded HI/LO limit values stored in memory (see HI and LO commands). The result of the limit check can be used to control special HI and LO digital outputs.

The DO0/LO and DO1/HI output pins can be configured to be alarm outputs by using the Enable Alarms (EA) command. After performing an EA command, the state of the DO0/LO and DO1/HI pins will be controlled by the alarm settings. The EA command does not affect the other digital outputs, DO2-DO8. The Disable Alarms (DA) command is used to disconnect the alarms from the output pins whereupon they are controlled by the Digital Output (DO) command.

Since the Alarm Outputs share the same circuits with the Digital Outputs, all electrical interfacing considerations are the same.

Alarm limit values are loaded into the module with the Low limit (LO) and Hi limit (HI) commands. The limit values are stored in nonvolatile memory so they will not be lost when power is removed. The HI and LO commands are also used to specify whether the alarms are momentary or latching. If an alarm is specified as momentary, the alarm is activated as long as the alarm condition exists. The alarm output will turn off when the input is within limits. A Latching alarm is activated when the specified limit is exceeded and will remain on even if the input value returns within limits. A Latching alarm can be turned off with the Clear Alarms (CA) command. The HI alarm output is turned on (sinking current) when the measured sensor input is greater than the high limit loaded in with the HI command. The LO alarm output is turned on (sinking current) when the input value is less than the stored low limit.

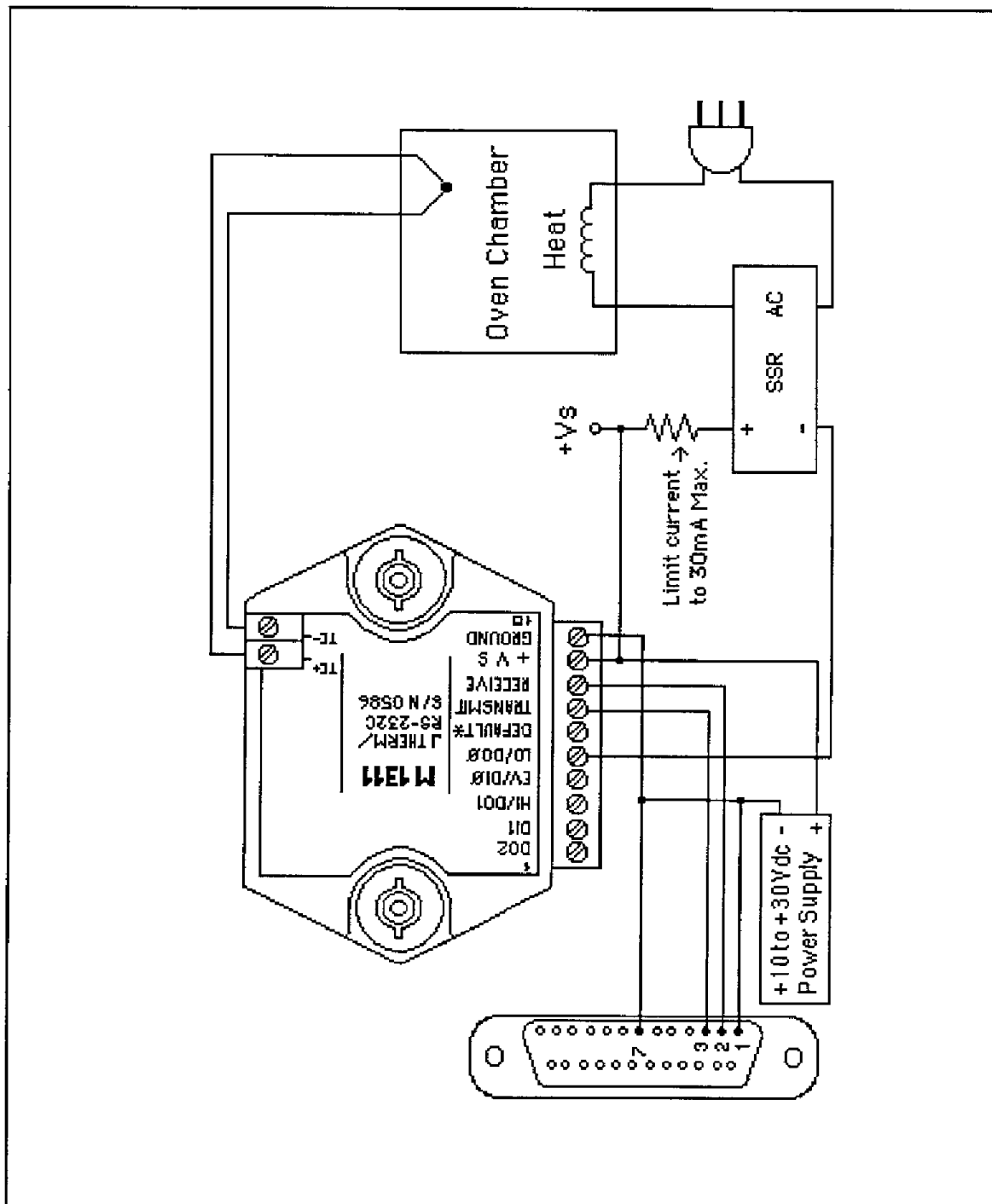
The alarm limit values may be read back at any time using the Read Low (RL) or Read High (RH) commands.

ON-OFF CONTROLLERS

The alarm capabilities of the M1000 sensor-input modules may be utilized to construct simple ON-OFF controllers that operate without host intervention. In fact, since all the alarm information is stored in nonvolatile memory, the module can act as a stand-alone controller with the communications lines disconnected.

The simplest controller connection is to use a momentary alarm output to control the process. A typical application would have a temperature input module controlling a heater, as shown in Figure 6.5. To maintain a constant temperature, set the low limit to the setpoint desired and specify the alarm output to be momentary. Use the LO alarm output to control the heater. If the temperature measurement exceeds the low limit, the heater will be turned off.

When the temperature goes below the limit, the LO alarm output goes on, turning on the heater. The negative feedback action of the control output will keep the temperature at the desired value. The high limit is still available to activate an alarm or shut down the system if the temperature goes out of limit.



ON-OFF CONTROLLER WITH HYSTERESIS

The simple single-value controller, by its very nature, suffers from erratic output that may not be acceptable, particularly when high-power equipment is being controlled. To lengthen the control cycle and to make the control action smoother, hysteresis (also known as dead band) is often used in on-off controllers. With hysteresis, the process variable is controlled between the two setpoints in order to lengthen the duty cycle of the control output. To increase the control duty cycle, the hysteresis, or difference between the setpoints, must be increased. Figure 6.6 shows the effect of hysteresis on the control output.

The high and low alarm limits on the M1000 sensor modules may be set to provide on-off control with hysteresis. The two limits specify the two control setpoints. The difference between the high limit and the low limit is the hysteresis value. The high limit must be greater than the low limit for proper operation. The alarm output used to control the process must be set to the Latching mode. If the control output is turned on, it will remain on until the input data exceeds the second alarm value. At this point the control output is turned off.

A typical example of a controller with hysteresis is illustrated in Figure 6.5. A temperature sensor module such as a M1311 J-Thermocouple model maybe used to regulate the temperature of an oven. The thermocouple is used to sense the oven temperature. The LO alarm output controls a solid state relay (SSR) which in turn controls the oven heater. The Enable Alarms (EA) command must be used to activate the alarm outputs. In this case the desired regulated temperature is 100°C. The Lo alarm is set to 95°C in the latching mode with the LO command. The HI alarm command is used to set the upper limit to 105°C in the momentary mode. The total hysteresis is the difference between the two alarm values, or 10°C. In the steady state condition, the oven temperature will oscillate between 95°C and 100°C (ideally).

Assume the oven temperature is below 95°C. This value is less than the value loaded into the low limit, therefore the LO alarm output is turned on. Since the low alarm is set for latching mode, the control output stays on even as the oven temperature goes above the 95°C low limit. The control output will stay on until the temperature reaches the value loaded into the high limit, in this case 105°C. At this point the latched LO alarm is turned off, turning off the heater. The control output will remain off as the oven cools down through heat losses. When the oven cools to 95°C, the LO alarm is again turned on, and the control process repeats indefinitely. The control signals are shown in Figure 6.6.

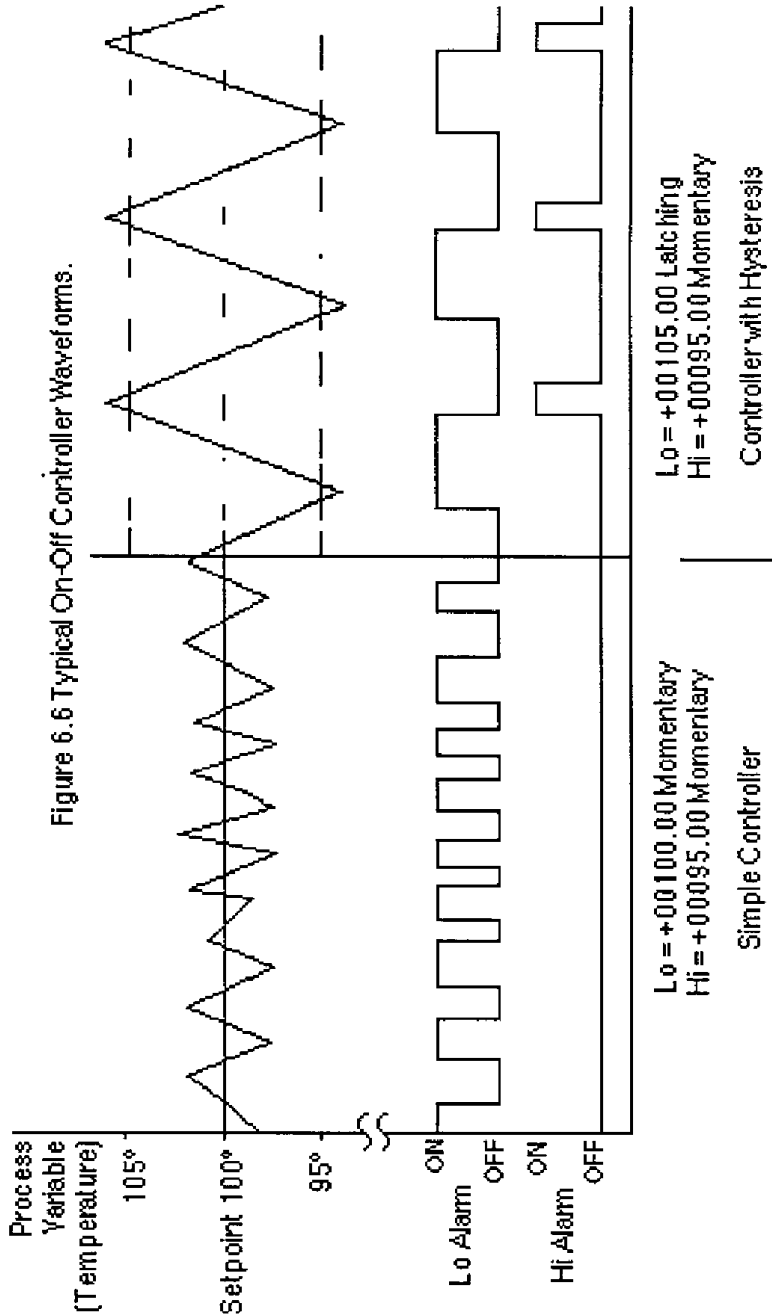


Figure 6.6 Typical On-Off Controller Waveforms.

In this case the high alarm was set to momentary mode. The high alarm could have been set to the latching mode without affecting the LO alarm output. However, the output at the HI alarm terminal would change. If the high alarm is set to Latching, the alarm output is simply the complement of the LO alarm. Either alarm output may be used for control depending on which one will result in negative feedback. For example, in a refrigeration system, the HI output may be used to control the refrigeration compressor and the low alarm value is used only to set the desired hysteresis value.

SETPOINT

In the preceding example, the low and high alarm limits are used to specify a hysteresis value around a desired setpoint. To change the desired setpoint, both the low and high alarm values must be changed. In this type of controller operation, the Read Data (RD) or New Data (ND) commands will read out the actual value of the process variable.

The M1000 modules provide a means of downloading a setpoint value without affecting the desired hysteresis by using the Setpoint (SP) command. The Setpoint command is used to load the desired control value into the output offset register (see Figure 2.1). The value in the output offset register is always added to the data derived from the sensor input. For instance, if the sensor data is +00100.00 and the output offset register contains +00050.00, a Read Data command will yield an output of +00150.00. The Setpoint command loads a value into the offset register to null out the sensor data. If the command \$1SP+00100.00 is given to a module with address 1, the effect of the command is to load the output offset register with -00100.00. An RD command will now result in the deviation of the input data from the downloaded setpoint value.

A careful look at Figure 2.1 will reveal that the alarm limits are checked after the output offset is added the input data. To construct a controller using the SP command, the high and low alarms must be loaded with the hysteresis values referred to the deviation from the setpoint value. In the oven controller example, the hysteresis was set to $\pm 5^{\circ}\text{C}$ from the desired control temperature of 100°C . When using the SP command, the high limit would be set to +00005.00 and the low limit would be set to -00005.00 to get the same hysteresis affect. The Latching modes of the alarm limits are used in the same manner as previously described.

Let's look at the oven controller again using the Setpoint command. The desired oven temperature is 100°C . This time we'll use the SP command to load the 100°C value into the temperature module. As before, we would like a hysteresis band of ± 5 from the nominal temperature of 100°C . In this case, set the low limit to -00005.00 latching and the high limit to +00005.00. The high and low limits are now used solely to define the hysteresis band. If the oven temperature is low, say 90°C , the resulting deviation from the setpoint of 100°C is -10°C . This value exceeds the low limit and the LO alarm control output is

turned on to activate the heater. The latched LO alarm will stay on until the measured temperature exceeds 105°C. At this point the deviation from the setpoint is greater than $\pm 5^\circ\text{C}$, the value loaded into the high limit. When the high limit is exceeded, the latched LO alarm output is turned off, turning off the heater. The control action is identical to the controller described in Figure 6.6.

The benefit of using SP command is that only one command is necessary to change the setpoint value. The hysteresis is stored in the HI and LO alarm registers and does not have to be changed when a new setpoint is used.

The SP command makes it particularly easy to construct a controller whose setpoint is a time varying function downloaded from a host computer. The SP command can also be used without control functions whenever a deviation output is desired.

The setpoint value may be read back by using the Read Zero (RZ) command. The RZ simply reads back the contents of the output offset register. The RZ command will always read back the setpoint value with the sign changed.

The setpoint value is stored in the same register as the output offset trim (see TZ command). In cases where the output offset register is used to hold a calibration trim value, the SP command will erase the trim. In most cases, an offset calibration trim is not necessary and the trim value would read back as +00000.00 using the Read Zero (RZ) command. If the trim is non-zero, it must be read and stored by the host before the SP command is executed. To download setpoint values the host must then subtract the trim value from the desired setpoint to derive the proper data for the SP command. To restore the trim, use the SP command to download the negative of the trim that was previously read back with the RZ command.

CHAPTER 7 POWER SUPPLY

MetraByte modules may be powered with an unregulated +10 to +30V dc. 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 M1000 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. M1000 modules without sensor excitation consume a maximum of .75 watts and this figure should be used in determining the power supply current requirement. For example, assume a 24 volt power supply will be used to power four modules. The total power requirement is $4 \times .75 = 3$ watts. The power supply must be able to provide $3 / 24 = .125$ amps.

For modules with sensor excitation, consult individual data sheets for power requirements.

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 D25 connector.

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

For best reliability, modules operated on long 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 MetraByte modules are protected against power supply reversals.

CHAPTER 8

TROUBLESHOOTING

Symptom: No module response.
Events counter not counting properly.
Error in displayed value.

• No module response

Assuming that the module is in a system, complete the following steps before removing it from the system. These steps will check the obvious things before removing.

1. Using a voltmeter, measure the power supply voltage at the module +Vs and Ground terminals to ensure that the power supplied is between +10 and +30 Vdc.
2. Check to see that the communications lines are connected properly and that there are no breaks in the lines.
3. If you are using the RS-232C to RS-485 converter, make sure the baud rate switch is set to the correct position.

If the above steps do not correct the problem, remove the module from the system and return to the bench. Complete the following steps.

1. Connect a working power supply (between +10 and +30 Vdc) to the Vs and Ground terminals on the module.
2. Connect the module to a dumb terminal as in the Quick Hook Up procedure. Your terminal should be set to 300 baud rate and no parity. Also if you are using the RS-485 converter, make sure its baud rate is set to 300 baud.
3. Connect a jumper wire from the Ground terminal to the Default* terminal.
4. Turn the power supply on, type \$1RD on the terminal and press the Return key.

If the module still does not respond, call the factory for assistance. If the module responds, send a \$1RS command to see if the information in the module's setup message is correct.

1. Check that the baud rate is correct.
2. If daisy-chaining RS-232C modules, be sure that the echo bit is set to 1.
3. If using byte time delay make sure that the proper delay is set. The above procedure basically makes sure that the module and the system are speaking the same language. Reinstall module in the system and try again.

- **Events counter not counting properly**

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

- **Error in displayed value**

1. Make sure that the °C/°F bit is set to a 0. Otherwise the values will be scaled by the °F equation.

CHAPTER 9 CALIBRATION

The M1000 module is initially calibrated at the factory and has a recommended calibration interval of one year.

Calibration constants are stored in the EEPROM and may be trimmed using the Trim Span (TS) and Trim Zero (TZ) commands. There are no pots to adjust. Calibration procedure is as follows.

Voltage and current inputs: clear the output offset register using the Clear Zero (CZ) command. Zero trims are not necessary due to the built-in auto-zero function. Apply a known calibrated voltage or current to the input of the module. The calibrated stimulus should be adjusted to be near 90% of the full scale output of the modules for best results. Obviously, the accuracy of the calibrated voltage or current must be better than the rated accuracy of the module, which in most cases is 0.02% of full scale. Use the Read Data (RD) command to obtain an output reading. If the output corresponds to the applied input, no calibration is necessary. If the output is in overload, check the circuit connections or use a different input value to obtain an output within the operating range of the module.

To trim the output, use the Trim Span (TS) command. The argument of the TS command should correspond to the desired module output. After performing the TS command, verify the trim using the RD command. For example: To trim a M1121 module, the following steps should be performed.

1. Clear the output offset register.

Command: \$1WE
Response: • (CZ is write protected)

Command: \$1CZ
Response: *

2. Apply an input voltage near 90% of rated full scale. In this case we will use a +900 mV input voltage that is accurate to at least 0.02%. Obtain an output reading.

Command: \$1RD
Response: *+00900.30

In this case, the output of the module is off by 300 μ V.

To trim:

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

Command: \$1TS+00900.00

Response: *

This sequence will trim the output to the desired calibrated value of +00900.00.

Verify:

Command: \$1RD

Response: *+00900.00

The module is calibrated.

Thermocouples: To start the calibration, disable the cold junction compensation by setting bit 4 in byte 3 of the setup data with the SetUp (SU) command. The module may now be calibrated using a known input voltage. Perform the calibration as described for a voltage input module. Table 9.1 gives recommended calibration points. Due to the nonlinear nature of thermocouples, it may be necessary to repeat the TS command to obtain the desired output. After calibration is complete, be sure to enable the cold junction compensation by clearing bit 4 in byte 3 of the setup data.

RTD: Use a calibrated resistor mounted directly on the module connector to avoid lead resistance errors. The resistor must be accurate to 0.01% for proper calibration. Recommended calibration points are listed in Table 9.1. Follow the command sequence described for voltage inputs to calibrate the module. Due to the nonlinear nature of RTD's it may be necessary to repeat the TS command to obtain the desired output.

Table 9.1 Calibration Values

Model	Input Stimulus	Output Data	0F
M111X	+90mV	+00090.00	
M112X	+900mV	+00900.00	
M113X	+4.5V	+04500.00	
M114X	+9V	+09000.00	
M121X	+9000 μ A	+09000.00	
M123X	+90mA	+00090.00	
M124X	+900mA	+00900.00	
M125X	+20mA	+00020.00	
M131X	+39.13	+00700.00	+01292.00
M132X	+41.269mV	+01000.00	+01832.00
M133X	+17.816mV	+00350.00	+00662.00
M134X	+68.783mV	+01000.00	+01832.00
M135X	+17.445mV	+01500.00	+02732.00
M136X	+15.576mV	+01500.00	+02732.00
M137X	+10.094mV	+01500.00	+02732.00
M138X	+33.442mv	+01982.00	+03600.00
M141X	300.00 Ω	+00558.00	+01036.40
M142X	300.00 Ω	+00547.60	+01017.70
M151X	25mV	+00025.00	
M152X	90mV	+00090.00	
M160X	18KHZ	+18000.00	
M161X	25 Sec	+25000.00	

APPENDIX A ASCII TABLES

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
#	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
L	76	4C	01001100	204	CC	11001100
M	77	4D	01001101	205	CD	11001101
N	78	4E	01001110	206	CE	11001110
O	79	4F	01001111	207	CF	11001111
P	80	50	01010000	208	D0	11010000
Q	81	51	01010001	209	D1	11010001
R	82	52	01010010	210	D2	11010010
S	83	53	01010011	211	D3	11010011
T	84	54	01010100	212	D4	11010100
U	85	55	01010101	213	D5	11010101
V	86	56	01010110	214	D6	11010110

W	87	57	01010111	215	D7	11010111
X	88	58	01011000	216	D8	11011000
Y	89	59	01011001	217	D9	11011001
Z	90	5A	01011010	218	DA	11011010
[91	5B	01011011	219	DB	11011011
\	92	5C	01011100	220	DC	11011100
]	93	5D	01011101	221	DD	11011101
^	94	5E	01011110	222	DE	11011110
~	95	5F	01011111	223	DF	11011111
	96	60	01100000	224	E0	11100000
a	97	61	01100001	225	E1	11100001
b	98	62	01100010	226	E2	11100010
c	99	63	01100011	227	E3	11100011
d	100	64	01100100	228	E4	11100100
e	101	65	01100101	229	E5	11100101
f	102	66	01100110	230	E6	11100110
g	103	67	01100111	231	E7	11100111
h	104	68	01101000	232	E8	11101000
i	105	69	01101001	233	E9	11101001
j	106	6A	01101010	234	EA	11101010
k	107	6B	01101011	235	EB	11101011
l	108	6C	01101100	236	EC	11101100
m	109	6D	01101101	237	ED	11101101
n	110	6E	01101110	238	EE	11101110
o	111	6F	01101111	239	EF	11101111
p	112	70	01110000	240	F0	11110000
q	113	71	01110001	241	F1	11110001
r	114	72	01110010	242	F2	11110010
s	115	73	01110011	243	F3	11110011
t	116	74	01110100	244	F4	11110100
u	117	75	01110101	245	F5	11110101
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

M1400 DATA SHEET

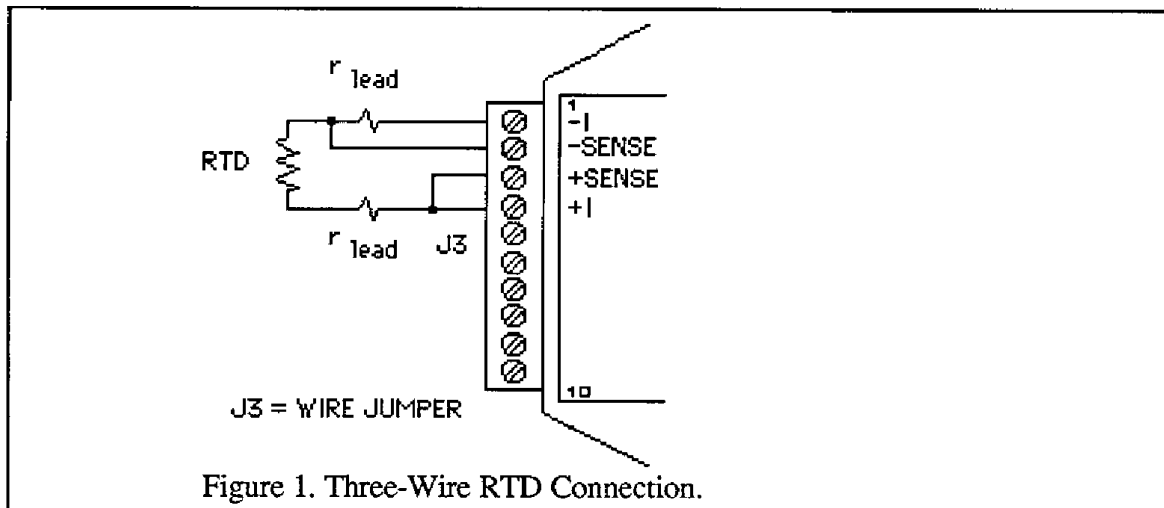
SPECIFICATIONS: (Typical @ 25°C, V+ = +15V)

RTD Types: $\approx .00385, .00392$ 100 @ 0°C
Resolution: 0.1°
Accuracy: $\pm 0.3^\circ$
Input connections: 2, 3, or 4 wire
Excitation current: 0.25 mA
Max. Lead resistance: 65Ω
Input protection to 120 Vac
Automatic linearization and lead compensation
User selectable °C or °F
Lead resistance effect: 3 wire 2.5°C per Ω of imbalance
4 wire Negligible

SENSOR HOOKUPS

The RTD sensor must be connected as shown in the accompanying diagrams to insure proper operation.

3-WIRE - The M1400 modules are shipped from the factory configured for 3-wire operation. Connect the RTD sensor as shown in the diagram. The wires connected to the +I and -I terminals should be matched in length and gauged for proper lead compensation. The +I and +SENSE terminals must be tied together at the connector with a short wire jumper. For proper 3-wire lead compensation, the RTD 3/4 wire set-up bit must be 0 (see Set-Up (SU) command). A typical set-up for 3-wire operation would be 31070182.



4-WIRE - For 4-wire operation, connect the sensor as shown in the diagram. If the RTD is equipped with heavy excitation wires, they should be connected to the +I and -I terminals. For proper 4-wire operation, the RTD set-up bit must be set to 1 (see Set-Up (SU) command). A typical set-up for 4-wire operation would be 31071182.

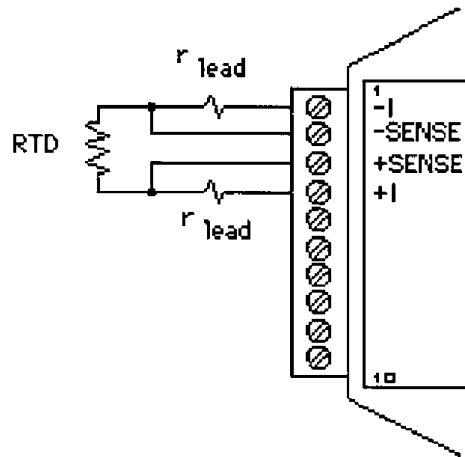
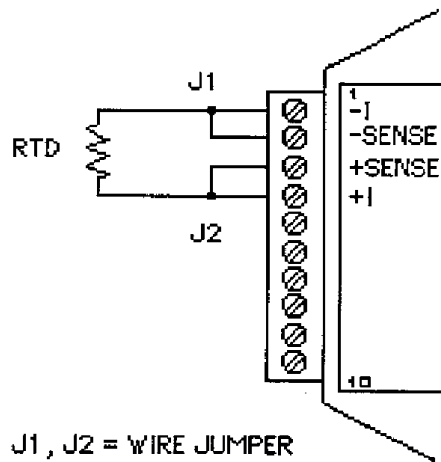


Figure 2. Four-Wire RTD Connection.

2-WIRE - The 2-wire connection requires two jumpers on the connector as shown in the diagram. This connection provides no lead compensation. The RTD set-up bit can be set to either 0 or 1 for this connection.



J1, J2 = WIRE JUMPER

Figure 3. Two-Wire RTD Connection.

START-UP: During normal operation, the RTD lead resistance is periodically scanned and filtered by the M1400 module. This may result in large initial errors if the RTD sensor is connected while the M1400 is powered up. To avoid this error, wire the sensor should to the connector before power is applied. The error may also be eliminated by performing a Remote Reset (RR) command.

LEAD RESISTANCE OVERLOAD: If the lead resistance exceeds 65Ω, the output data is set to +99999.99.

SENSOR GROUNDING: The sensor input is electrically isolated from the power and communications inputs for common-mode voltages up to 500V. If the sensor is to be grounded or shielded, the ground connection should be made to the -I terminal for best performance.

APPENDIX C

M1500 DATA SHEET

The M1500 Bridge Sensor Interface Modules contain all of the signal conditioning functions necessary to interface Strain Gage and other resistive bridge devices to an RS-232C or RS-485 computer port. Each module contains excitation, an instrumentation amplifier, and a smart analog to digital converter to convert resistive bridge sensor signals to ASCII data.

The user should become familiar with the generic D1000 information described in the D1000 User's Manual before attempting any of the procedures outlined below.

DATA FORMAT

The ASCII output data is expressed in millivolts with 10 microvolt resolution.

For Example:

Command: \$1RD (Read Data)
Response: *+00012.34

In this case, the output data is 12.34 millivolts.

Modules that are configured for ± 30 mV and have a usable span of ± 60 mV. Modules configured for ± 100 mV have a usable span of ± 120 mV. The extra overhead is used to trim any bridge offsets.

SETUP DATA

The factory setup for all versions of M1500 modules is 310701C2

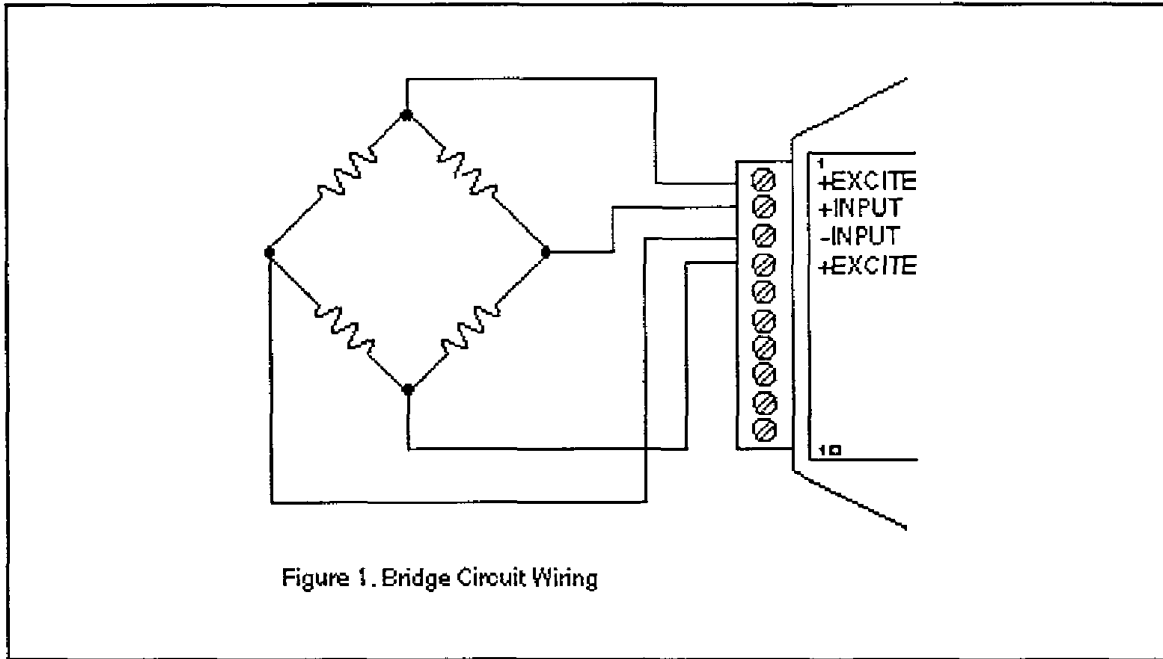
SENSOR CONNECTIONS

See Figure 1 for the proper bridge sensor connections. Shields or grounds should be connected to the -Excitation terminal.

OFFSET TRIM

The M1500 modules do not provide any means of trimming the analog offset of the sensor bridge. However, sensor offsets may be nulled from the output data with the Trim Zero (TZ) command. This method of trimming is convenient because the offset may be trimmed through the communications port at any time. There is no need to have access to the module since the trimming is performed remotely.

The input signal conditioning circuitry of the M1500 modules have a wide input range to accommodate large sensor offsets without the need for external trims. Modules rated for ± 30 mV. have an input range capability of ± 60 mV. Modules specified for ± 100 mV have an input range of 120 mV.



To perform an initial offset trim, attach the bridge unit to the module (as shown in Fig. 1). Clear out any previous offset trims with the Clear Zero (CZ) command. Apply the desired zero condition to the bridge sensor. For a Strain Gage Bridge this would be the relaxed or unstrained condition. For load cells, the zero condition could include any tare weight due to a weighing platform or other attachments that would affect the zero balance. Obtain an initial reading using the Read Data (RD) command. The output data will indicate the total offset of the system. Subtract the offset value from the usable input range of your module, either ± 60 mV or ± 120 mV. The result is the maximum usable "input overhead". If the overhead is not sufficient for your application, the bridge must be trimmed externally to lower the offset to an acceptable value. The bridge may be trimmed with a small series resistance or a large shunt resistance to the appropriate leg of the bridge (as shown is Fig. 2). If the initial offset is acceptable, the offset may be trimmed with the Trim Zero (TZ) command.

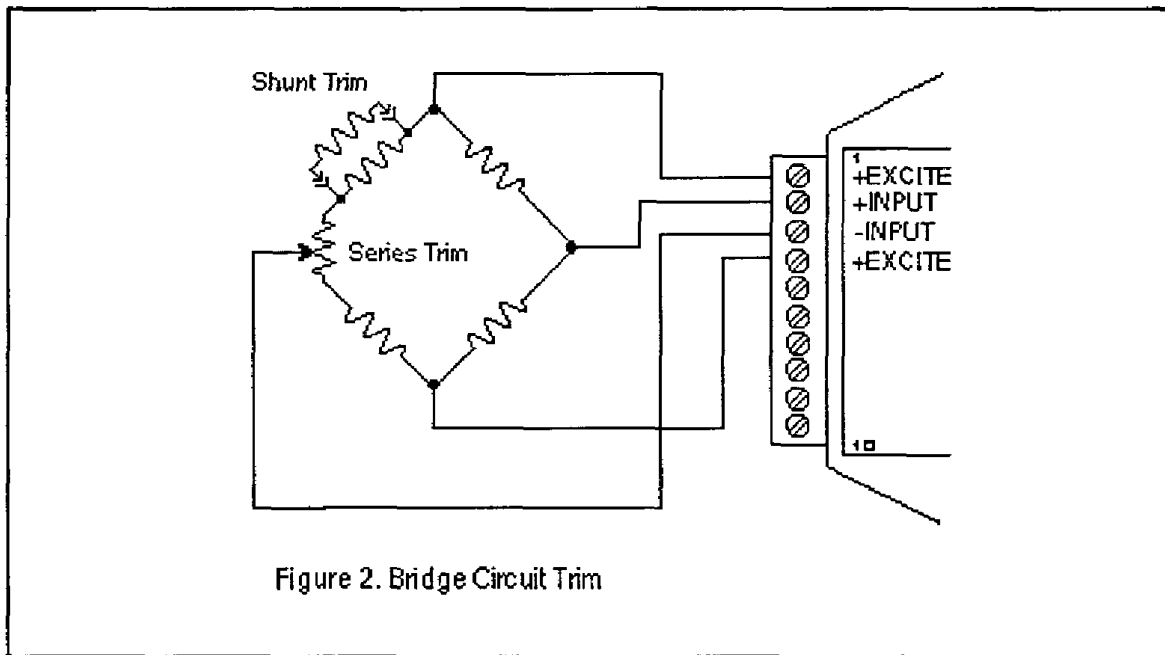


Figure 2. Bridge Circuit Trim

Example 1:

A load cell to be used in a weighing application is mated to a M1521 module. The load cell is rated for 3 mV/V, which results in a maximum ± 30 mV with 10 V excitation. However, in this particular application, the load cell is used only in tension so its ideal output will be from 0 to +30 mV.

The load cell is mounted in its final position with the weighing attachments. Clear any offset data that may be stored in the M1521 module:

Command: \$1WE (CZ is write-protected)
Response: •

Command: \$1CZ (Clear Zero)
Response: •

Verify that the Zero Trim is cleared:

Command: \$1RZ (Read Zero)
Response: *+00000.00

Obtain an initial offset reading from the load cell with no weight attached:

Command: \$1RD (Read Data)
Response: *+00002.34

The initial offset is +2.34 mV. The M1521 has a useful input range of ± 60 mV. After subtracting the offset the "input overhead" is -62.34 mV and +57.66 mV.

The expected 0 to +30 mV output of the load cell easily falls within the overhead range and no external trimming is necessary.

To Trim Zero:

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

Command: \$1TZ +00000.00 (zero is the desired output)
Response: *

Now read the data output to verify the trim:

Command: \$1RD (Read Data)
Response: *+00000.00

The load cell system has been trimmed to zero.

Example 2:

A strain gage bridge will be used to measure both compression and tensile strains on a structural member. The bridge is attached to a M1521 module and the ideal output from the bridge is ± 30 mV full scale.

Clear the Zero Trim:

Command: \$1WE
Response: *

Command: \$1CZ (Clear Zero)
Response: *

Measure the initial offset from the bridge:

Command: \$1RD
Response: *-00043.21

In this case, the bridge exhibits a large initial offset of -43.21mV. Subtract this value from the ± 60 mV useful range of the M1521 to obtain an "input overhead" value of -16.79 mV to 103.21 mV. In this case the -16.79 mV overhead is not large enough to cover the -30 mV that may be obtained from the bridge. The bridge must be trimmed externally to bring the offset to within ± 30 mV. It is not necessary to obtain an exact zero with the external trim.

After the external trim has been performed, check the offset:

Command: \$1RD
Response: *-00022.22

This value is within the ± 30 mV offset necessary to provide enough headroom for the strain gage bridge.

The remaining offset may be trimmed out with the Trim Zero (TZ) command:

Command: \$1WE
Response: *

Command: \$1TZ +00000.00
Response: *

The bridge is now trimmed to zero.

Verify:

Command: \$1RD
Response: * +00000.00

The Trim Zero (TZ) command may be used at any time to balance out offsets due to temperature, residual stress, tare, etc.

Excitation

M1500 modules may be ordered with either 5 V or 10 V excitation. Maximum excitation current available is 60 mA. Modules with 10 V excitation may be used with bridges that have input impedances of 166 ohms or greater. Half-bridges of 120 Ω strain gages may be used with 10 V excitation if the bridge is completed with 350 Ω resistors. Modules with 5 V excitation will source bridges of 85 Ω and up.

The actual excitation voltage may vary ± 0.5 V from the nominal values of +10 V and +5 V. However, the module's internal microprocessor constantly monitors the actual excitation voltage and provides compensation for any deviation from the nominal value. This results in a constant data output for a constant bridge load even if the excitation changes. From a user's point of view, the excitation voltage will appear to be exactly +10 V or +5 V.

CALIBRATION

Since the M1500 modules use a ratiometric technique to compensate for variances in the excitation voltage, special consideration is required to properly calibrate the unit. Figure 3 shows the calibration setup. The Digital Voltmeter (DVM) must be capable of measuring the excitation voltage to 4 digit accuracy. The voltage source must be able to provide millivolt signals accurate to ± 5 microvolts. The resistive divider may be constructed from 1% resistors of equal value from 100 to 1000 Ω . The resistor divider places the voltage source in the center of the common-mode range of the input amplifier for best accuracy.

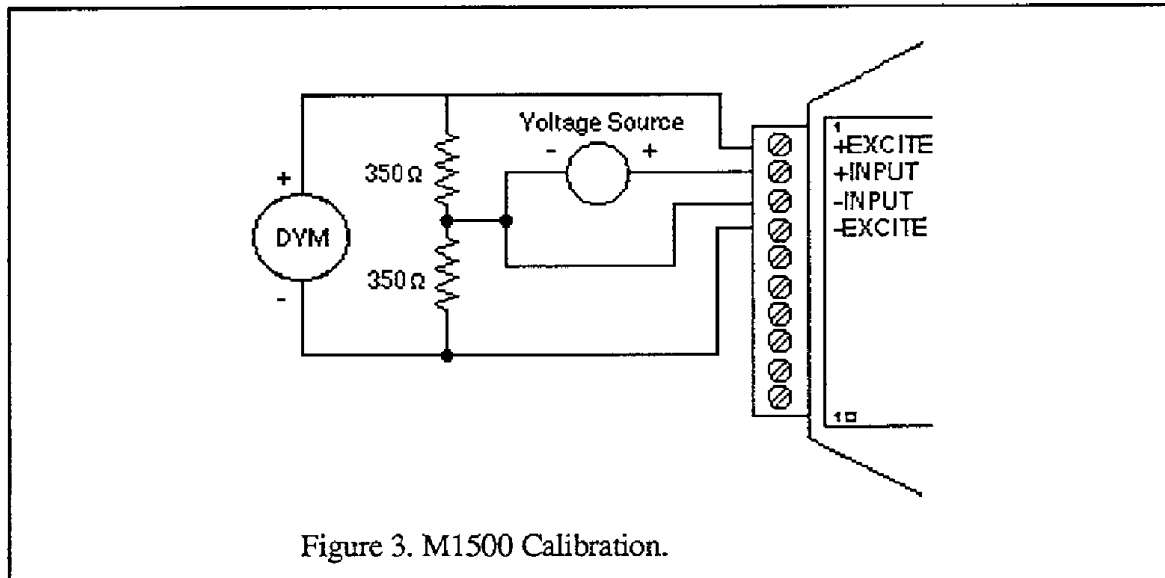


Figure 3. M1500 Calibration.

Step 1: power up the unit under test and let it warm up for at least two minutes.

Step 2: set the voltage source to 0 volts (short). Perform a TZ+00000.00 (Trim Zero) command to eliminate any common-mode offset errors.

Step 3: measure the excitation voltage with the DVM. Divide the result by the nominal excitation voltage, either 10 V or 5 V, to obtain a "compensation factor" = CF

Step 4: calculate the correct calibration voltage to apply to the unit.

For ± 30 mV units the voltage is $V = +50 \text{ mV} \times \text{CF}$

For ± 100 mV units the voltage is $V = +100 \text{ mV} \times \text{CF}$

Set the voltage source to the calculated voltage V.

Step 5: trim the unit with the Trim Span (TS) command.

For ± 30 mV modules the command is \$1TS+00050.00

For ± 100 mV modules the command is \$1TS+00100.00

Step 6: verify the trim using the \$1RD command. The result should be either *+00050.00 or *+00100.00

Calibration Example:

We wish to calibrate a M1511 module. This unit contains 5 V excitation and a ± 30 mV input.

Step 1 is straightforward and needs no further explanation.

Step 2: set the voltage source to 0 volts. Trim zero:

Command: \$1WE

Response: *

Command: \$1TZ+00000.00

Response: *

Step 3: measure the excitation voltage with the DVM. In this example the measured voltage is 4.954 V Calculate the "compensation factor":

$$CF = 4.954 / 5 = .9908$$

Step 4: calculate the calibration voltage:

$$V = + 50 \text{ mV} \times .9908 = + 49.54 \text{ mV.}$$

Set the voltage standard to + 49.54 mV.

Step 5: perform the Trim Span command:

Command: \$1WE

Response: *

Command: \$1TS+00050.00

Response: *

Step 6: verify the calibration, continuing to apply + 49.54 mV to the input:

Command: \$1RD

Response: *+00050.00

The span trim is now complete. The Trim Zero (TZ) command may be used to trim sensor offsets without affecting the span trim.

OPTIONS:

Digital Output:

All M1500 units come standard with a Digital Input / Event Counter input. This connector pin may be factory configured for a Digital Output / Low Alarm output. Consult factory.

Continuous Output:

Any of the D1000 sensor input modules may be factory configured to provide continuous output data without interrogation from the host. This option is ideal for use with LED display panels to provide a continuous visual output. To specify continuous output, add a "C" suffix to the model number; M1511C for example.

Programmable Scaling:

The Metrabyte D2500 series of interface modules are bridge units similar to the M1500 series except that the input/output transfer function may be programmed by the user. Output data may be scaled to any desired engineering units such as pounds, psi, Newtons, etc. Non-linear functions may also be programmed into the module. All scaling data is stored in non-volatile memory and may be re-programmed any number of times. Call factory for details.

Bridge Completion Resistors:

For convenience, standard bridge completion resistors may be obtained from Metrabyte. Standard values available are 120 Ω and 350 Ω .

APPENDIX D M1600 DATA SHEET

The M1601/2 Frequency Input modules feature a versatile input stage that can be used in a variety of applications. Fig. 1 is a block diagram of the input signal conditioning.

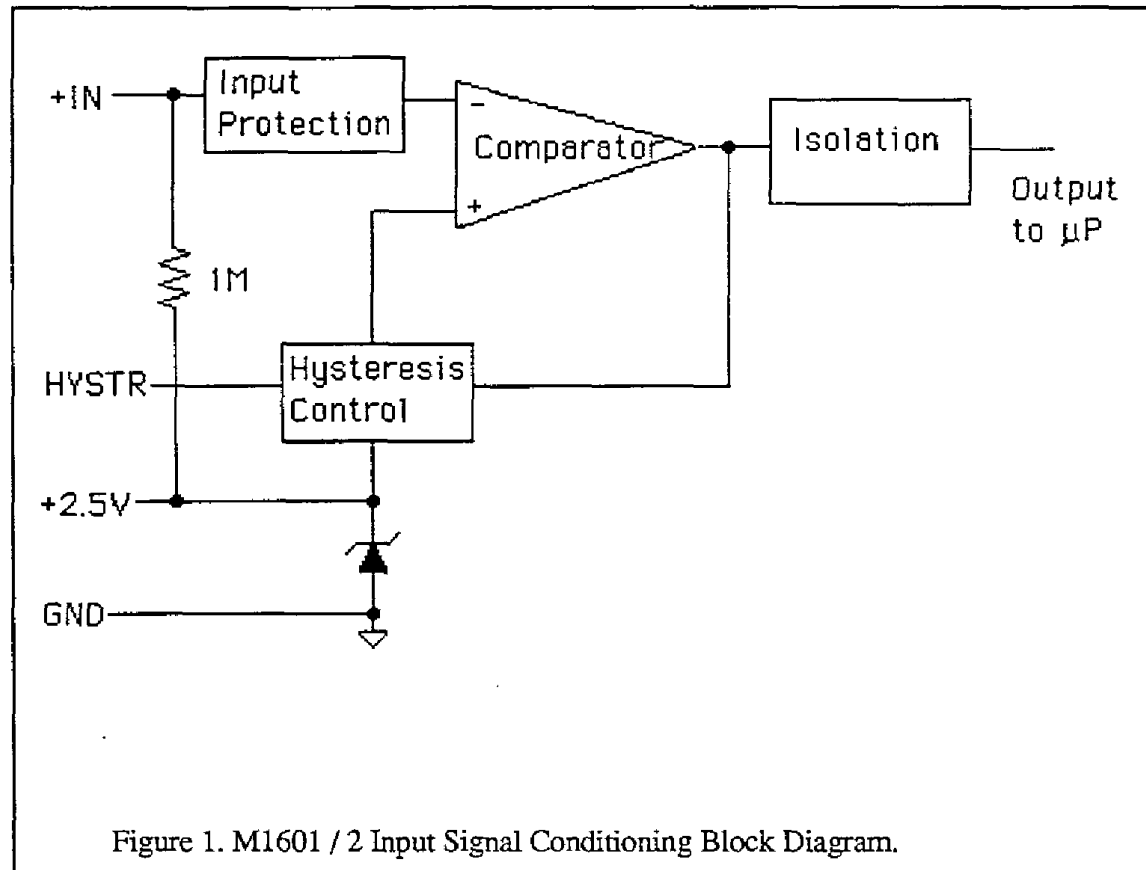
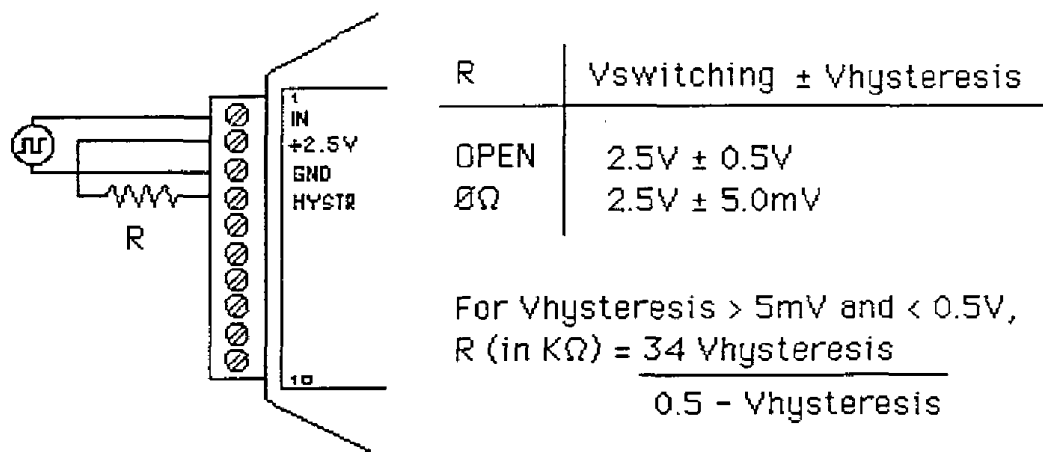


Figure 1. M1601 / 2 Input Signal Conditioning Block Diagram.

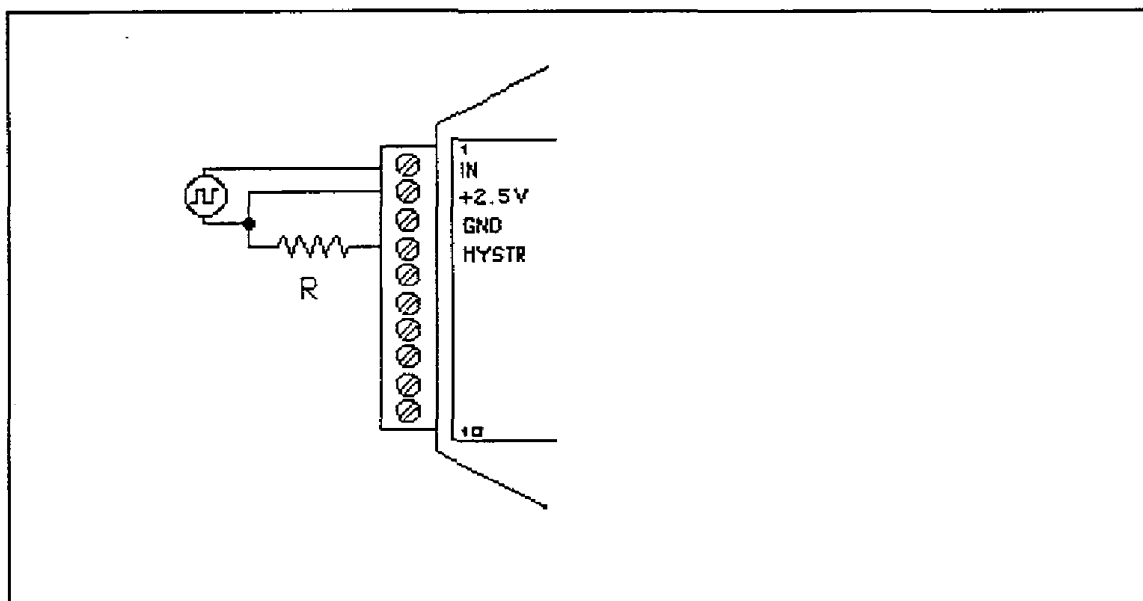
The input signal is applied to a precision comparator through the + input. Input protection is provided to withstand inputs up to 230Vac. The comparator output is then fed through an opto-isolator to the module's microprocessor for scaling and formatting. The input section is completely isolated from the power and communications lines. The isolation allows up to 500V of common-mode voltage between the input ground and the power connections.

The input comparator employs hysteresis to provide reliable readings with noisy or slow input signals. The amount of hysteresis may be controlled by connecting the hysteresis control line (HYSTR) to ground or the 2.5V terminal through an external resistor. Fig. 2 shows the most frequently used connection.

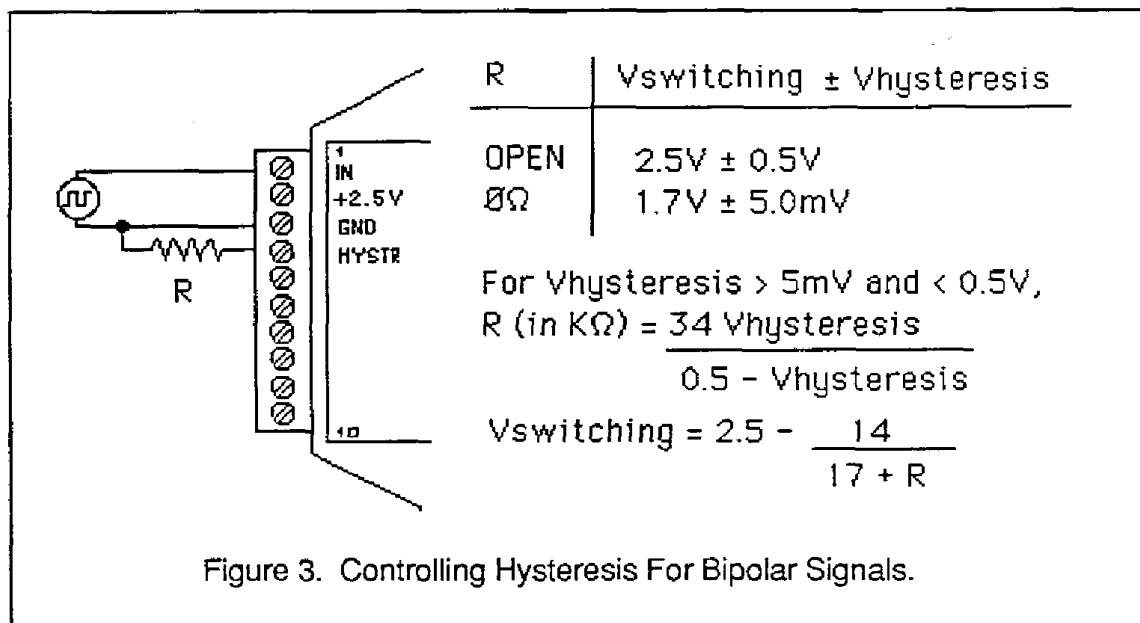
Figure 2. Controlling Hysteresis For Positive-Going Signals



This connection is used for unipolar positive-going frequency signals. The hysteresis is centered around a +2.5V switching level. If R is left open, the switching levels are +3V and +2V, or $2.5V \pm 0.5V$. If R is shorted, the hysteresis decreases with resulting switching levels of $2.5V \pm 5mV$. Any hysteresis value from $\pm 5mV$ to $\pm 0.5V$ may be obtained by selecting an appropriate value for R. Fig. 2 shows the relationship between the hysteresis and R.



The input comparator may be setup for comparisons around zero volts by using the connections in Fig. 3. This connection is useful for AC or bipolar signals. Since the input section is isolated, the +2.5V pin may be connected to any signal with a common-mode voltage up to 500V. With the hysteresis control connected as in Fig. 3, the switching points occur symetrically on either side of the +2.5V level. Since the low side of the input signal is connected to the +2.5V pin, the switching points appear to be symmetrical to zero, as referenced to the input signal. The hysteresis may be varied from $\pm 5\text{mV}$ to $\pm 0.5\text{V}$ as shown in Fig. 2.



The hysteresis control may also be connected to ground (GND), which produces another set of switching levels. This connection is shown in fig. 4. If the HYSTR terminal is shorted to GND the nominal switching point is 1.6V with $\pm 5\text{mV}$ of hysteresis.

To measure AC signals super-imposed on a DC value, the input may be AC coupled by simply placing a capacitor in series with the +IN terminal. The module contains an internal $1\text{M}\Omega$ resistor connected from the +IN to +2.5V for biasing. A .01 uf cap may be used for frequencies down to 10 HZ.

M2000 SERIES

PROGRAMMING MANUAL

TABLE OF CONTENTS

CHAPTER 1	Linear Scaling	1-1
	Nonlinear Functions	1-2
CHAPTER 2	Block Diagram	2-1
	Programming Table	2-2
	Breakpoints	2-5
CHAPTER 3	BreakPoint Command	3-1
	MiNimum Command	3-2
	MaXimum Command	3-3
CHAPTER 4	Programming Software	4-1
	General Guidelines	4-1
	Function Programming	4-3
	Linear Scaling	4-4
CHAPTER 5	Programming Steps	5-2
	Examples	5-4

Chapter 1 Introduction

The MetraByte M2000 series of intelligent analog-to-computer interfaces are designed to solve many difficult interfacing problems that cannot be performed with existing standard interfaces. The M2000 series may be programmed to create custom transfer functions to interface to non-standard sensors or to scale the outputs to any engineering units desired.

The M2000 series is an enhancement of the MetraByte M1000 series of standard interfaces. The M2000 series is similar to the M1000 series in every respect except that the M2000 interfaces allow custom input-to-output transfer functions. As shipped from the factory, the M2000 modules operate in the same manner as their M1000 counterparts. For example, a M2111 shipped from the factory contains the same transfer function as a M1111 module; in this case they are both ± 100 mV inputs and communicate with RS-232C. Before any attempt is made to program a M2000, you must first be familiar with the operation of a M1000 module as described in the M1000 manual.

The M2000 contains built-in commands to create custom functions. All programming is performed through the communications port of the M2000 module. There is never any need to open the module case. Modules may be re-ranged remotely as many times as desired. Function data is stored in nonvolatile memory to retain the scaling even if power is removed.

Linear Scaling

The basic concept of the M2000 series is to create interfaces which output data in engineering units that may be instantly read and interpreted without any data conversion necessary by a host computer. In fact, the M2000 interfaces may be used with a dumb terminal to provide data readings in easy-to-understand engineering units. For example, a typical pressure sensor might provide a 1 to 5V. linear output for pressures of 0 to 1000 psi. Using a M1131 module or an unprogrammed M2131 unit the output data would look like this:

<u>Pressure (psi)</u>	<u>Sensor Output</u>	<u>M2131 Output (mV)</u>
0	1V	+01000.00
500	3V	+03000.00
1000	5V	+05000.00

The standard output of the M2131 reads out in units of millivolts. Even though the M2131 will faithfully output the sensor voltage, the real parameter of interest is pressure, not voltage, and

(1-2) MetraByte M2000 Programming Manual

the voltage readings may be difficult to interpret. To make the output data more readable, the M2131 may be programmed to output the data in units of pressure:

<u>Pressure (psi)</u>	<u>Sensor Output</u>	<u>M2131 Output (psi)</u>
0	1V	+00000.00
500	3 V	+00500.00
1000	5 V	+01000.00

In some cases, the desired output may be more specific to a particular application. Assume that the same pressure sensor is used to measure the "fullness" of a pressure vessel, such as a cylinder of compressed air. The M2131 could be scaled to output in units of "percent" and in this case we will assume that if the cylinder reads 750 psi it is 100% full:

<u>Pressure</u>	<u>Volts</u>	<u>Output (%)</u>
0	1	+00000.00
375	2.5	+00050.00
750	4	+00100.00

Nonlinear Functions

As we have shown with the linear pressure sensor example, the output may be scaled to any units we desire. However, the real power of the M2000 series is that they may be programmed to provide a nonlinear transfer function. This capability may be used to provide outputs in engineering units for nonlinear sensors. The M2000 uses a linear piece-wise approximation technique to describe nonlinear functions. Up to 24 linear segments may be used to approximate a function, as shown in Figure 1. Figure 2 shows some of the variety of curves that may be programmed into the M2000.

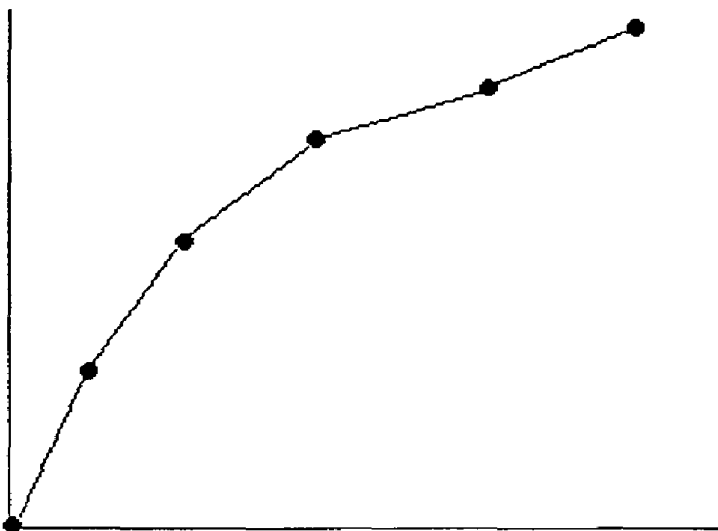


Figure 1. Linear piece-wise approximation

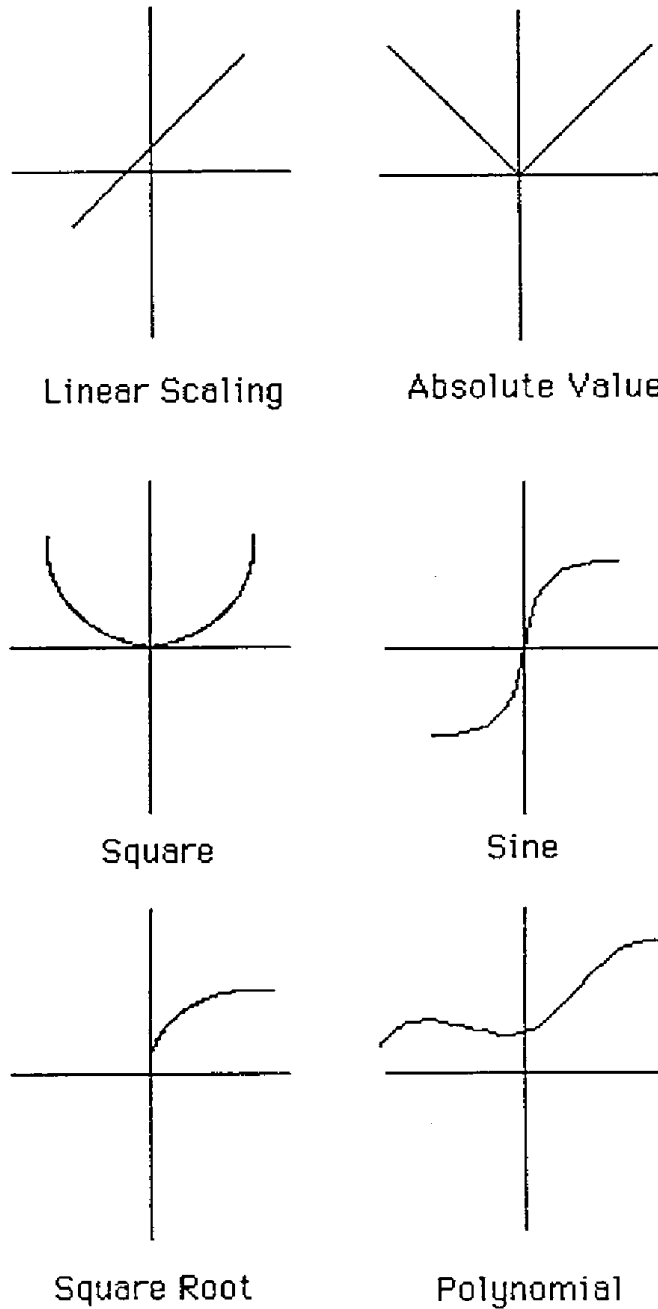


Figure 2. Typical curves that can be programmed by the M2000

The M2000 modules may also be programmed in the field to specific test inputs where the actual nonlinearity is not known.

Chapter 2 Theory of Operation

The M2000 performs all scaling functions in firmware using the module's internal microprocessor. All scaling and nonlinear function data is stored in a table contained in EEPROM nonvolatile memory. Scaling data stored in the memory will remain intact indefinitely even if power is removed. M2000 modules may be re-scaled up to 10,000 times.

All re-scaling operations are performed with simple commands given to the module through its communications port. The M2000 series command set encompasses all the the M1000 commands plus additional commands to perform function programming. There is no need to open or have access to the module to perform re-scaling. In many cases the modules may be re-scaled remotely after they have been installed. Detailed descriptions of the M2000 programming commands are given in Chapter 5.

Figure 3 is a simplified block diagram of the M2000, showing only the portions related to re-scaling. The microprocessor reads the raw Analog-to-Digital Converter (ADC) data after every conversion. The μP takes the raw ADC data and looks it up in a table held in EEPROM. The table contains entries which map the raw ADC data to output data in engineering units. If an exact match is not found, the data is interpolated between the two closest table entries. The resulting data in engineering units is stored in a memory buffer where it may be read by the Read Data (RD) or New Data (ND) Commands.

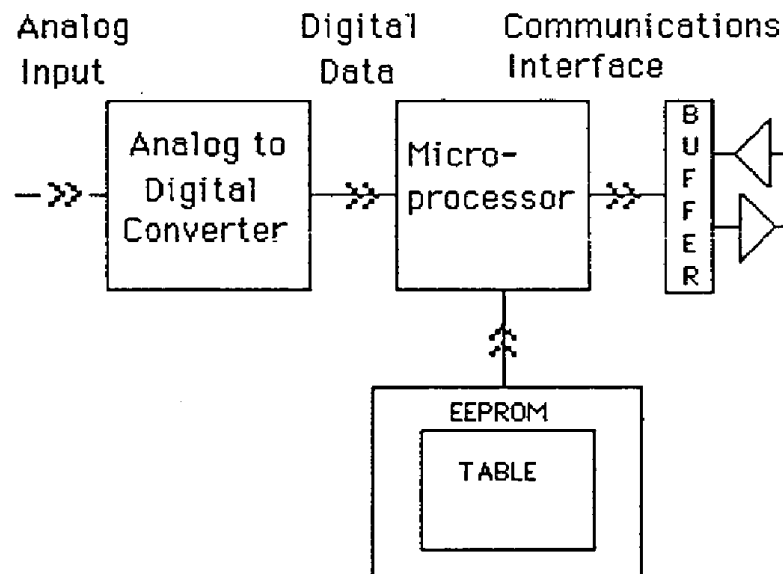


Figure 3. Block Diagram

Note that the re-scaling operation acts on the output of the analog-to-digital converter. The basic input-to-output transfer function of the ADC is fixed and cannot be changed. For example, a M2131 module with a ± 5 V. input range cannot be re-scaled to ± 10 V or any other range. Analog input scaling is performed by selecting the M2000 model that best matches the sensor signal. The ADC data is then manipulated with the function table to provide output data in engineering units.

(2-2) MetraByte M2000 Programming Manual

Programming Table

Figure 4 shows a programmer's model of the table used to program the input-output transfer function of the M2000. The table values are intentionally left blank so that it may be copied and used as a worksheet to help program the modules.

	ANALOG INPUT	DATA OUTPUT
MINIMUM	X_{MIN}	Y_{MIN}
MAXIMUM	X_{MAX}	Y_{MAX}
BREAKPOINT 00	X_{00}	Y_{00}
BREAKPOINT 01	X_{01}	Y_{01}
BREAKPOINT 02	X_{02}	Y_{02}
BREAKPOINT 03	X_{03}	Y_{03}
BREAKPOINT 04	X_{04}	Y_{04}
BREAKPOINT 05	X_{05}	Y_{05}
BREAKPOINT 06	X_{06}	Y_{06}
BREAKPOINT 07	X_{07}	Y_{07}
BREAKPOINT 08	X_{08}	Y_{08}
BREAKPOINT 09	X_{09}	Y_{09}
BREAKPOINT 0A	X_{0A}	Y_{0A}
BREAKPOINT 0B	X_{0B}	Y_{0B}
BREAKPOINT 0C	X_{0C}	Y_{0C}
BREAKPOINT 0D	X_{0D}	Y_{0D}
BREAKPOINT 0E	X_{0E}	Y_{0E}
BREAKPOINT 0F	X_{0F}	Y_{0F}
BREAKPOINT 10	X_{10}	Y_{10}
BREAKPOINT 11	X_{11}	Y_{11}
BREAKPOINT 12	X_{12}	Y_{12}
BREAKPOINT 13	X_{13}	Y_{13}
BREAKPOINT 14	X_{14}	Y_{14}
BREAKPOINT 15	X_{15}	Y_{15}
BREAKPOINT 16	X_{16}	Y_{16}

Figure 4. Breakpoint Table.

MetraByte M2000 Programming Manual (2-3)

The two most important points in the table are the Minimum and Maximum points. These two table entries specify the minimum and maximum endpoints of the transfer function curve. For instance, a M2121 has a range of $\pm 1V$, and the standard table values are:

	<u>Analog Input</u>	<u>Data Output</u>
Minimum	-1 V	-01000.00
Maximum	+1 V	+01000.00

Plotted on a graph (Figure 5), these two points specify the endpoints of the transfer curve. In this case, the analog input variable X is in terms of voltage. The X values in the table specify the minimum and maximum voltages that may be applied to the analog input that will result in a linearized output. (The X voltage values are actually stored in memory in terms of ADC binary data). Voltage values applied to the analog input that are more negative than Xmin will result in an overload output of -99999.99. Similarly, voltage values greater than Xmax will result in +99999.99.

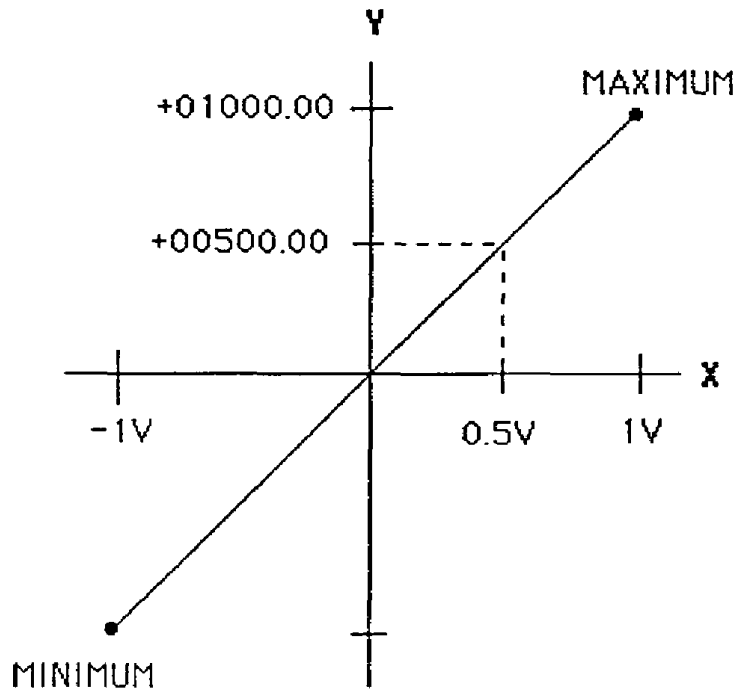


Figure 5 Function Endpoints

The corresponding Y values in the table specify the output data of the minimum and maximum points. In this case, a -1V input corresponds to an output of -01000.00 millivolts. The Y values are always stored in the standard data format of sign, 5 digits, decimal point and two additional digits.

(2-4) MetraByte M2000 Programming Manual

The minimum and maximum points are the only table values necessary to specify a linear transfer function. For analog input values between Xmin and Xmax, the output values are determined by linearly interpolating between the minimum and maximum points. For instance, in the case of the M2121, an analog input value of +.5V is linearly interpolated to an output value of +00500.00 (Figure 5).

It should be apparent at this point that a M2000 module may be re-scaled by modifying the minimum and maximum values in the table. This may be accomplished by using the Minimum (MN) command and the Maximum (MX) command. Using the M2121 ± 1 volt module as an example, we may use the MN and MX commands to alter the table to look like this:

	<u>Analog Input</u>	<u>Data Output</u>
Minimum	0 V	+00100.00
Maximum	+1 V	+00800.00

In this case the minimum point is 0 V, corresponding to the output data +00100.00. The maximum point is +1 V input and +00800.00 output. The graph of this equation is shown in Figure 6.

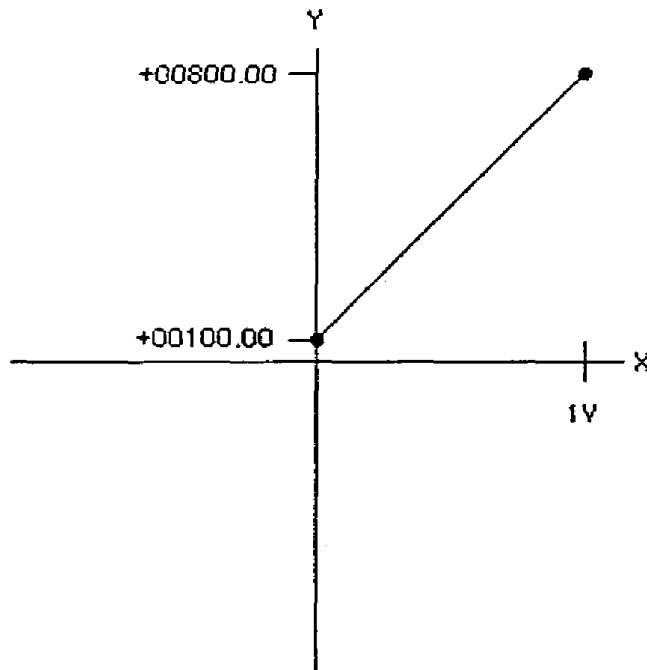


Fig. 6

By changing the minimum and maximum values in the table, an infinite number of linear functions may be specified, bounded by X values of ± 1 V and Y values of ± 99999.99 . Figure 7 shows a few possibilities.

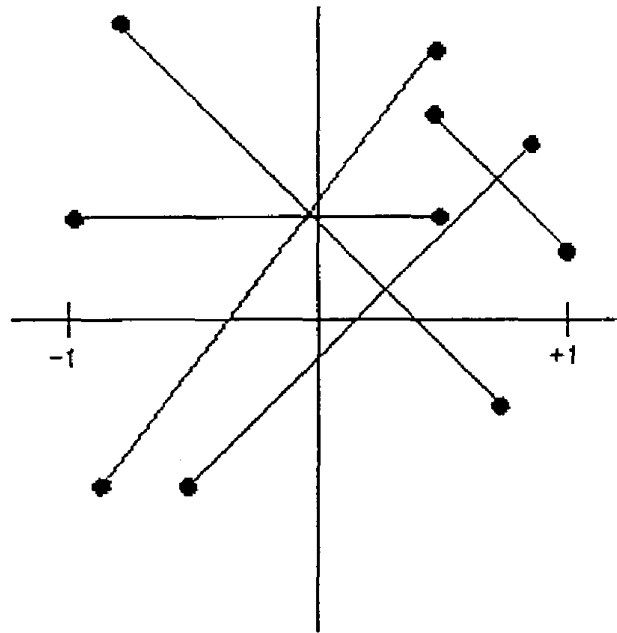


Figure 7.

The exact procedure necessary to program the maximum and minimum points is described in Chapter 5.

Breakpoints

Looking back at Figure 4, we can see that most of the transfer function table is reserved for "Breakpoints". Breakpoints are used to modify the basic linear curve defined by the Minimum and Maximum points to create nonlinear functions.

Nonlinear functions in the M2000 are approximated by using linear segments which are specified by the data values held in the Breakpoint Table. Up to 23 breakpoints may be programmed to specify up to 24 linear segments. Figure 8 illustrates the action of the breakpoints. Figure 8a shows a basic linear transfer function described by the Minimum and Maximum points. Figure 8b shows the effect of one breakpoint used to modify the linear function. Notice that the breakpoint has created a nonlinear function described by two linear segments joined at the breakpoint. Figure 8c shows that two breakpoints may be used to specify a nonlinear curve described by three linear segments. Up to 23 breakpoints may be used to create complex nonlinear curves.

Breakpoints are stored in the EEPROM table in the same fashion as the minimum and maximum points. Each breakpoint is described by an X-Y pair specifying the analog input value at which the breakpoint occurs and the corresponding output data value. When the microprocessor reads the analog (X) data from the ADC, it searches the breakpoint table to find the X value closest to the input data. The micro then linearly interpolates between two breakpoints to calculate the resulting output data.

(2-6) MetraByte M2000 Programming Manual

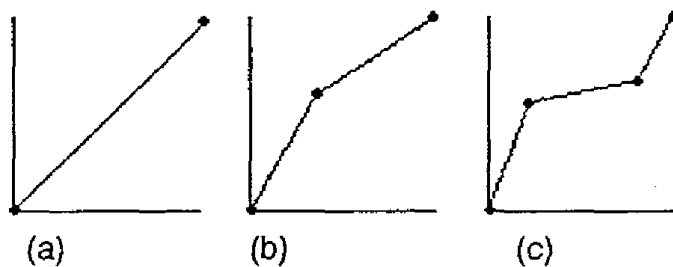


Figure 8. Breakpoint Examples.

Any number of breakpoints up to 23 values may be specified. The breakpoint table must be filled progressively starting with Breakpoint 00 to Breakpoint 16 (hex). Unused or "erased" breakpoints are not used in the function calculation.

Let's use the M2121 $\pm 1V$ module again as an illustrative example to show the effect of a breakpoint. Figure 9 shows the M2121 function table with 1 breakpoint programmed:

	<u>Analog Input</u>	<u>Data Output</u>
Minimum	-1 V	-01000.00
Maximum	+1 V	+01000.00
Breakpoint 00	+2 V	+00800.00
Breakpoint 01	_____	_____
....		
....		

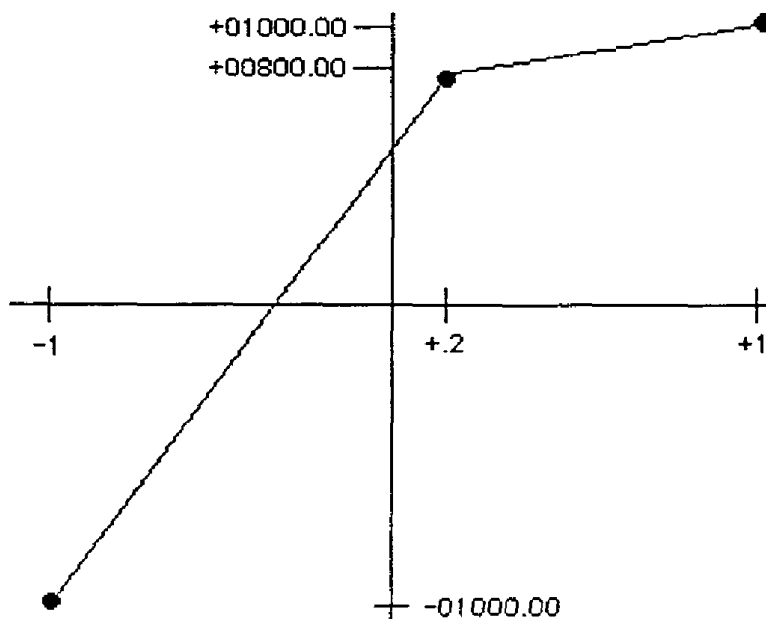


Fig. 9

Breakpoints 01 through 16 (hex) are erased and do not enter the function calculation. The Minimum and Maximum table entries contain the standard data values of ± 01000.00 mV. The new curve is shown in Figure 9.

Notice how the breakpoint has affected the whole curve, creating a nonlinear function. Here are a few samples of the input-output values that may be obtained from this curve:

<u>Analog Input</u>	<u>Data Output</u>
-.8 V	-00700.00
-.6 V	-00400.00
-.4 V	-00100.00
-.2 V	+00200.00
0 V	+00500.00
+.2 V	+00800.00
+.4 V	+00850.00
+.6 V	+00900.00
+.8 V	+00950.00

The procedure to create a breakpoint table is detailed in Section 4.

Chapter 3 Command Set

The M2000 module series incorporates the same command set as the M1000 series, with new commands added to facilitate custom range programming. The added M2000 commands are used only for programming. For normal operational commands, refer to the M1000 manual.

CAUTION: THE M2000 PROGRAMMING COMMANDS MUST BE USED WITH CARE. EACH OF THE COMMANDS IS CAPABLE OF DESTROYING FACTORY CALIBRATION.

All of the commands added to the M2000 series are write-protected to guard against accidentally altering data values stored in the module's EEPROM. Therefore, all programming commands must be preceded with a Write Enable (WE) command.

All of the M1000 command-response protocol rules apply to the M2000.

This section is intended only to describe the new commands. For programming information refer to Chapter 5.

BREAK POINT (BP)

Nonlinear functions may be approximated in the M2000 by describing the function curve with a series of line segments (see Figure 1). The line segments are programmed into the M2000 using the BreakPoint (BP) command. A breakpoint specifies the intersection between two linear segments used to approximate the nonlinear transfer function. Up to 23 breakpoints may be used to specify 24 linear segments in a curve.

To program a breakpoint, a known analog stimulus must be applied to the sensor input of the M2000 module. This specifies the input variable (X-axis) location of the breakpoint. The corresponding output data (Y-axis) of the breakpoint is specified as an argument to the BreakPoint (BP) command.

Example: (Spaces have been added to the command for clarity)

Command: \$1 BP 03 +00100.00

Response: *

Command: #1 BP 03 +00100.00

Response: *1BP 03 +00100.00FA (FA is the checksum)

The first two characters following the "BP" command specify the breakpoint number. Up to 23 breakpoints may be programmed into the M2000. In the sample command above, breakpoint

(3-2) MetraByte M2000 Programming Manual

number "03" is being specified. Breakpoint numbers are expressed in two-digit hexadecimal notation, ranging from "00" to "16" for a total of 23 (decimal) points. During a normal programming operation, breakpoints are entered in sequence in progressively-increasing X values starting from the minimum value (see Minimum (MN) command). Breakpoint programming must start with Breakpoint "00". It is not necessary to specify all the breakpoints; any number up to 23 may be used. However, a breakpoint sequence must start at "00" and be entered sequentially. Any remaining breakpoints may be left unspecified.

Following the breakpoint number, the output (Y-axis) data must be specified. The data must be in standard M1000 format: sign, five digits, decimal point, 2 digits. The output data specifies the module's output response for the test stimulus applied to the module input.

Before setting the breakpoints with the BreakPoint (BP) Command, the overall function span must be specified by the MiNimum (MN) and MaXimum (MX) commands. (See Chapter 5 for programming instructions.)

Erase Breakpoints (EB)

The EB command erases all previously entered breakpoints from the module's EEPROM. Erased data cannot be recovered. Therefore, before using the EB command, be prepared to re-program all of the breakpoints in the unit. The EB command is used to provide a clean slate before entering a breakpoint sequence. Previous end-point data entered by the MiNimum (MN) and MaXimum (MX) commands are not affected.

Command: \$1EB

Response: *

Command: #1EB

Response: *1EBE2 (E2 is the checksum)

MiNimum (MN)

The MiNimum (MN) command is used to define an endpoint of a transfer function programmed into a M2000 module. The minimum endpoint defines the most negative value allowed on the analog input before an overload will occur.

In effect, the minimum value is the starting breakpoint in a programmed transfer function. To use the MiNimum (MN) command, a known analog test stimulus must be applied to the analog input of the module. The test stimulus must correspond to the most negative value of the desired analog input range. The analog input stimulus specifies the starting input value (X-axis) of the transfer function. The test input must lie within the specified full scale input range of the module.

The argument of the MN command specifies the starting output value (Y-axis) of the transfer function.

Command: \$1MN -00100.00

Response: *

Command: #1MN -00100.00

Response: *1MN-00100.00A2 (A2 is the checksum)

The argument of the MN function must be in standard analog data format.

MaXimum (MX)

The MaXimum (MX) command specifies the most positive analog input allowed before an overload indication will occur. The MaXimum command also defines the positive end point of a transfer function programmed into the M2000. To perform a MaXimum command, a known analog stimulus must be applied to the sensor input of the M2000 unit. This test input must correspond to most positive value of the programmed transfer function. The analog test signal must remain within the specified input range of the M2000 module. The analog input establishes the maximum input value (X-axis) for the transfer function. The maximum output value (Y-axis) is specified as the argument of the MaXimum command.

Command: \$1MX +00500.00

Response: •

Command: #1MX +0500.00

Response: *1MX+00500.00AE (AE is the checksum)

Chapter 4 Programming

This section will cover the mechanics of programming a custom transfer function into the M2000. All programming is performed through the communications port of the M2000 using a dumb terminal or a computer operating as a dumb terminal. In field installations where AC power is not readily available, programming may be accomplished with standard battery-operated ASCII terminals. Since all programming is accomplished through the communications port, access to the module is not necessary and ranging may be accomplished remotely.

Programming Software

Although all programming functions may be accomplished with a dumb terminal, the task may be greatly simplified with the use of utility software running on a computer. MetraByte offers programming software which will run on many of the popular personal computers. The software provides many enhancements that are not available through manual programming. In many applications the M2000 modules may be programmed strictly through software methods without the need for external excitation sources. Contact MetraByte for availability.

GENERAL GUIDELINES

Input Scaling

The analog input characteristics of a M2000 module may not be altered by the user. Input scaling is accomplished by selecting the correct M2000 model for the application. Programming a M2000 involves altering the scaling of the unit's A/D converter output. There is no provision for changing the gain or offset of the analog circuitry.

Excitation

When the M2000 modules are programmed manually with a terminal, external excitation sources are necessary to establish calibration points within the module. Excitation may be provided by standard voltage, current and frequency calibration sources. The final absolute accuracy of the module is directly dependent on the accuracy of the excitation sources. In some cases, the excitation may be generated directly by the system being monitored. In situations when excitation sources are not available or impractical, modules may be programmed with MetraByte programming software without excitation.

Output Data Format

One of the preliminary decisions to be made before programming is how the output data will be structured. All MetraByte sensor modules communicate data in a fixed format of sign, five digits, decimal point, and two additional digits; +00100.00 is an example. The fixed format is

(4-2) MetraByte M2000 Programming Manual

used to simplify software in host computers. Despite the fixed format, the programmer has a certain amount of flexibility to structure the output data for the best compromise between resolution and readability. For example, an output indication of +.05 volts could be structured in three different output formats:

+00000.05	(+.05 volts)
+00050.00	(+50 millivolts)
+50000.00	(50,000 microvolts)

The first consideration must be the resolution or the number of output counts available in the output structure. If the overall function span is 0 to +50 millivolts, the first example would only yield 5 counts from +00000.00 to +00000.05. In most applications this resolution would not be acceptable. The next obvious output structure is to output the data in units of millivolts, as shown in the second example. This format would give us 5,000 counts of resolution. Finally, the third example expresses the output data in units of microvolts to give a possible resolution of five million counts.

The second factor that must be considered is the performance limitations of the A/D converter. The best resolution of the ADC is 50,000 counts. Resolution is degraded by round-off errors, noise, etc., so that a practical expectation for usable resolution would be in the range of 5,000 to 20,000 counts. Output resolution may be limited by picking a suitable output format or by using the appropriate 'displayed digits' setup as described in the M1000 Setup section.

In the present example of the 0 to 50 mV. output, probably the best compromise is to use the millivolt form to represent the data. This gives 5,000 count resolution in easy to interpret units of millivolts. In this case the 'displayed digits' setup should be programmed to display all digits.

It may be tempting to use the microvolt output format in an effort to extract 50,000 counts of resolution, but the units digit will tend to be noisy. The uncertainty of the units digit may be counteracted somewhat by using large amounts of digital filtering in the module setup. In this case the setup data should specify a 'displayed digits' setting of the first five digits only, since the digits to the right of the decimal point have no meaning. Also, the microvolt format is a bit more awkward to interpret than the millivolt format.

In some cases it may require a bit of creative thinking to develop a suitable output format. For example, a M2000 module may be required to output data in units of specific gravity. In a typical application, the specific gravity output may range between .5 and 2. The most obvious output format would have the output data ranging from +00000.50 to +00002.00. This format gives only 150 counts of resolution between the minimum and maximum outputs. However, since the specific gravity of water is defined to be 1, the output may be scaled in units of "percent of water". The specific gravity of water would then be 100 percent. The output data in 'percent' units would range from +00050.00 to +00200.00. This format allows up to 15,000 counts of resolution and reads out in units that may be easily interpreted.

Linearity

The analog-to-digital converter used in the M2000 has a typical integral nonlinearity of .1% of full scale. At the factory the ADC linearity is corrected by using breakpoints to reduce the nonlinearity to .01%. If the breakpoint table is erased with the Erase Breakpoints (EB) command, the linearity correction is lost. In some cases when linear re-scaling is performed, the programmer may take advantage of the factory linearity correction (Example L-4). If less than half of the full analog input scale is used, the linearity correction should be erased with the EB command. Linearity may be improved with the use of breakpoints (Example N-5).

M2000 FUNCTION PROGRAMMING

The M2000 transfer function may be programmed by modifying the function table with the MiNimum (MN), MaXimum (MX) and BreakPoint (BP) commands. All three commands operate on the same basic principle. Each command is used to specify an input-output (X,Y) data pair in the function table. To perform a programming command, a known analog excitation must be applied to the analog input of the M2000 module. The excitation may be a voltage, current, frequency, or the output of a resistive bridge, depending on the specific M2000 module type. The known excitation value is used to create the "X" values in the function table. The "Y" table values are loaded with data specified in the command argument.

For example, suppose we have a M2121 ± 1 V module and we'd like to program the minimum table value to $X_{min} = -.5V$, $Y_{min} = -.00100.00$. Apply $-.5$ volts to the module input with a calibrated voltage source. Perform the MiNimum (MN) command with the Y_{min} value as the data argument in the MN command:

Command: \$1WE (MN is write-protected)

Response: *

Command: \$1MN-00100.00

Response: •

When the module executes the MN command, the microprocessor performs two functions. First, it reads the data produced by the A/D converter with the $-.5V$ input. The A/D converter data is stored as X_{min} in the function table. The micro then reads the argument of the MN instruction, which in this case is $-.00100.00$, and stores this value in the table as Y_{min} . This completes the definition of the new minimum point. The module will immediately use this new minimum point data in calculating output data.

Note that the MN command will write over any previous data in the table. The old data is permanently lost. This is also true with the MaXimum (MX) and BreakPoint (BP) commands.

(4-4) MetraByte M2000 Programming Manual

Since the MN, MX, and BP commands affect the calibration of the module, they must not be used indiscriminately unless you are prepared to re-calibrate the unit.

LINEAR SCALING

Rescaling the M2000 to a linear transfer function is the easiest and most common way to reprogram the module. The linear scale function is defined by specifying the two endpoints of the linear function (as shown in Figure 7). Any linear function within the analog input range of the module may be defined.

Custom scaling requires a calibrated analog input signal to define the end points of the linear transfer function. The signal could be a voltage, current, or frequency depending on the specific model type. The MiNimum and MaXimum commands are used to program the end point data into the modules's memory.

Programming procedures:

1. Make sure the module has not been previously programmed with Break Point (BP) Commands. If it has, clear the breakpoints with the Erase Breakpoints (EB) command.
2. Clear any data in the output offset register with the Clear Zero (CZ) command.
3. Determine the endpoints which will be used to define the linear function. The analog input values must lie within the operating range of the module. The analog inputs used to determine the endpoints will also define the overload outputs of the module. Construct an output data format that is best suited for your application.
4. Apply a calibrated analog signal to the module input corresponding to the most negative input of the desired linear scale. Perform a Minimum (MN) command to store the function endpoint into the modules's memory.
5. Apply a calibrated analog signal to the module input corresponding to the most positive analog input value. Perform a Maximum (MX) command to load the endpoint data into the module memory.
6. Verify that the transfer function has been correctly loaded into the module by applying test inputs to the module and reading out the data with the Read Data (RD) command.

Example L-1

Reprogram a M2151, 4-20 mA module to output data in terms of percent; that is, 4 mA will read out to be 0% and 20 mA will read out as 100%.

1. If the module had been previously programmed with breakpoints, erase the function table with the Erase Breakpoints (EB) command:

Command: \$1WE
Response: *

Command: \$1EB
Response: *

2. Clear any offset data with the Clear Zero command:

Command: \$1WE
Response: *

Command: \$1CZ
Response: *

3. The minimum analog input in this case is 4 mA. Any current less than 4 mA will result in a negative over-range (-99999.99). The maximum positive input is 20 mA. Since the minimum value of 4 mA corresponds to 0%, the appropriate output data would be +00000.00. The output data corresponding to 20 mA is +00100.00. This data format gives us whole units of "percent" to the left of the decimal point. To get the maximum resolution from the module, set up the number of displayed digits with the SetUp (SU) so that all digits are displayed.

Command: \$1WE
Response: *

Command: \$1SU310701C2 (typical)
Response: *

4. Apply exactly 4 mA to the current input of the module. Program the endpoint with the MiNimum command:

Command: \$1 WE
Response: *

Command: \$1 MN+00000.00
Response: *

5. Apply exactly 20 mA to the module input and store the maximum endpoint with the MaXimum (MX) command:

Command: \$1 WE
Response: *

Command: \$1 MX+00100.00
Response: *

(4-6) MetraByte M2000 Programming Manual

6. Verify the module response by testing it with various inputs within its range:

Input Current Output Data

8 mA	+00025.00
12 mA	+00050.00
16 mA	+00075.00

Rescaling is now complete.

Example L-2

A paddle-wheel flow sensor will be used to monitor the flow of water in a pipe. The characteristics of the sensor and the size of the pipe results in an output frequency of 10 Hz. per gallon per minute. The operating range is from 1 to 20 gallons per minute.

We would like to scale a M2000 module to output data in units of .1 gallons.

The logical module choice in this application is the M2601 frequency input module. The frequency output of the flow sensor will range from 10 Hz. to 200 Hz., easily within the 5 Hz. to 20 kHz. range of the M2601.

1. Erase Breakpoints:

Command: \$1WE
Response: *

Command: \$1EB
Response: •

2. Clear Zero:

Command: \$1WE
Response: •

Command: \$1CZ
Response: •

3. The minimum endpoint in this case is 10 Hz corresponding to an output of +00001.00 gpm. The maximum frequency at 20 gpm is 200 Hz. The maximum output data is +00020.00. To get .1 gpm resolution, set up the module to display six digits:

Command: \$1WE
Response: *

Command: \$1SU31070182 (typical)
Response: *

4. Using a calibrated frequency generator, apply 10 Hz. to the module input. Set the minimum point:

Command: \$1WE
Response: *

Command: \$1MN+00001.00
Response: *

5. Set the frequency generator to 200 Hz. to program the maximum point:

Command: \$1WE
Response: *

Command: \$1MX+00020.00
Response:

6. Use the frequency generator to verify a few points in the scale:

<u>Analog Input</u>	<u>Data Output</u>
30 Hz	+00003.00
100 Hz	+00010.00
155 Hz	+00015.50

Programming is now complete and the M2601 can be attached to the flow sensor.

Example L-3

In many cases the analog calibration values may be produced directly by the sensors to be used in a system. The module may be re-ranged in the field to encompass any errors due to sensor inaccuracies.

In this example, we wish to use a pressure sensor to measure the volume of water in a cylindrical tank that is 10 feet tall with the capacity of 1500 gallons. The pressure sensor is mounted at the bottom of the tank so that it will produce an output corresponding to the height of water in the tank. The pressure sensor chosen produces 1V @ 0 psi and 5V @ 10 psi. A full tank with 10 feet of water will produce 4.335 psi (1 ft = .4335 psi), well within the range of the pressure sensor. A M2131 ± 5 V input module will be used as the interface.

(4-8) MetraByte M2000 Programming Manual

1. Erase Breakpoints:

Command: \$1WE

Response: *

Command: \$1EB

Response: *

2. Clear Zero:

Command: \$1 WE

Response: •

Command: \$1 CZ

Response: •

3. To produce the analog Xmin and Xmax endpoint values, we will use the actual water levels in the tank to produce a calibration pressure. The accuracy of the pressure transducer is not important, as long as it is stable and linear. To set the minimum value, we will empty the tank and set the minimum value to +00000.00. The maximum value will be programmed with the tank full and the maximum output data will be set to +01500.00 gallons. In this case an output resolution in units of gallons is acceptable and we can set up the module so that 5 digits are displayed. The digits to the right of the decimal point will always read out "00".

Command: \$1WE

Response: •

Command: \$1SU31070142 (typical)

Response: •

4. With the tank empty, program the minimum point:

Command: \$1 WE

Response: •

Command: \$1 MN +00000.00

Response: •

5. Fill the tank with water and program the maximum point:

Command: \$1 WE

Response: •

Command: \$1 MX +01500.00

Response: *

6. Verify the scaling. In this case, it is difficult to verify the scaling quickly and accurately. A "ballpark" check can be made by letting water out of the full tank and checking to see if the module output readings are "reasonable". A more accurate check can be made by filling the tank with known amounts of water and verifying the output readings.

Example L- 4

A M2251 4-20 mA module will be used to provide a computer interface to an existing process 4-20 mA signal. The loop transmitter produces a linear 4-20 mA signal corresponding to a sensor temperature of 0-200 degrees C. In this case we'd like to take advantage of the factory linearity correction in the M2251 for greater accuracy. To do this, we must use the same analog input minimum and maximum points as programmed at the factory. The M2251 minimum and maximum points are 0 mA and 25 mA. The 4-20 mA span of the process transmitter must be extrapolated to 0-25 mA to provide the correct data when using the MN and MX commands. The transfer relationship of the 4-20 mA transmitter can be described by the equation:

$$T = 12.5 \times \text{mA} - 50$$

Plug values of 0 mA and 25 mA into the equation to derive extrapolated values of T:

$$T = 12.5 \times (0) - 50 = -50$$

$$T = 12.5 \times (25) - 50 = +262.5$$

These values will be used in the MN and MX instructions.

Program the module:

1. In this case it is assumed that the M2251 is fresh from the factory and it still contains linearity correction in the breakpoint table. In order to take advantage of the linearity correction, the breakpoints will not be erased.

2. Clear zero:

Command: \$1WE

Response: *

Command: \$1CZ

Response: *

(4-10) MetraByte M2000 Programming Manual

3. The minimum endpoint has been extrapolated to be -00050.00 @ 0 mA. The maximum point is +00262.50 @ 25 mA. We'll setup the module to display temperature with .1 degree resolution:

Command: \$1WE

Response: *

Command: \$1SU31070142 (typical)

Response: *

4. Apply 0 mA (open circuit) to the current input and program the minimum point:

Command: \$1WE

Response: *

Command: \$1MN-00050.00

Response: *

5. Apply exactly 25 mA to the current input to program the maximum point:

Command: \$1WE

Response: *

Command: \$1MX+00262.50

Response: *

6. Apply test currents to the module to verify the scaling:

Apply 4 mA to the input:

Command: \$1

Response: +00000.00

Apply 20 mA to the input:

Command: \$1

Response: +00200.00

Chapter 5 Nonlinear Programming

Nonlinear functions may be created by first specifying a linear function with the MiNimum (MN) and MaXimum (MX) commands. The linear function is then modified by using the BreakPoint (BP) command. Almost any practical nonlinear function may be approximated provided it satisfies two rules:

1) The nonlinear function must be totally enclosed by the rectangular area defined by the minimum and maximum points. Figure 10 gives examples of the "rectangular area".

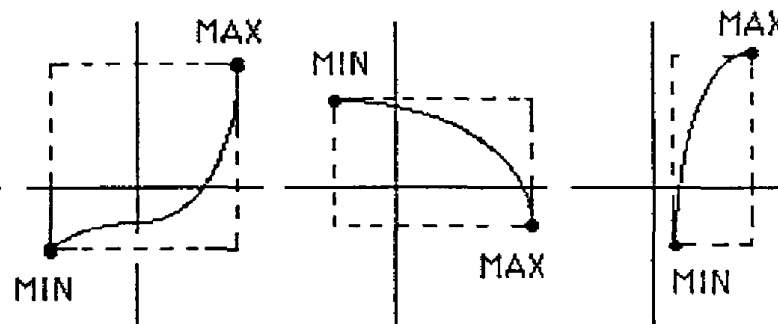


Figure 10

Figure 11 illustrates a function that is not possible since a portion of the curve lies outside of the rectangle. In most cases this limitation may be overcome by simply re-arranging the curve so that the rectangular area is larger. Figure 12 shows the same curve as Figure 11, but slightly modified to allow it to be programmed into the M2000.

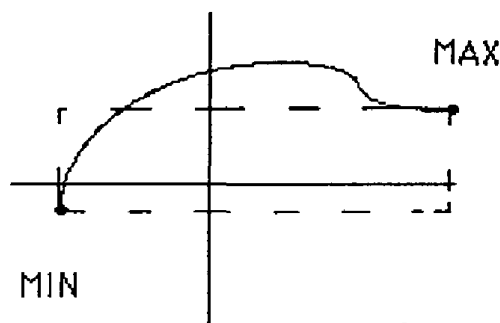


FIGURE 11 ILLEGAL FUNCTION

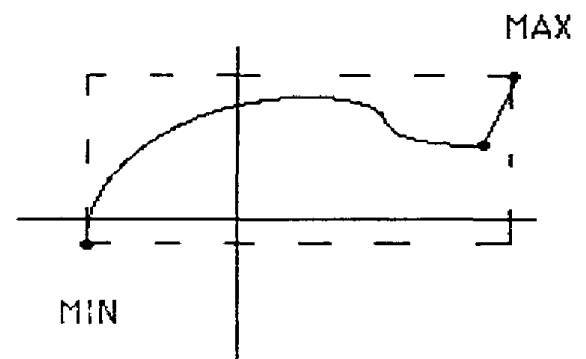


FIGURE 12 Modified Function.

(5-2) MetraByte M2000 Programming Manual

2) The nonlinear function must be a single-valued function of X. That is, for each input value, there can exist only one output value. Figure 13 shows two illegal functions. This limitation is seldom encountered in natural phenomenon.

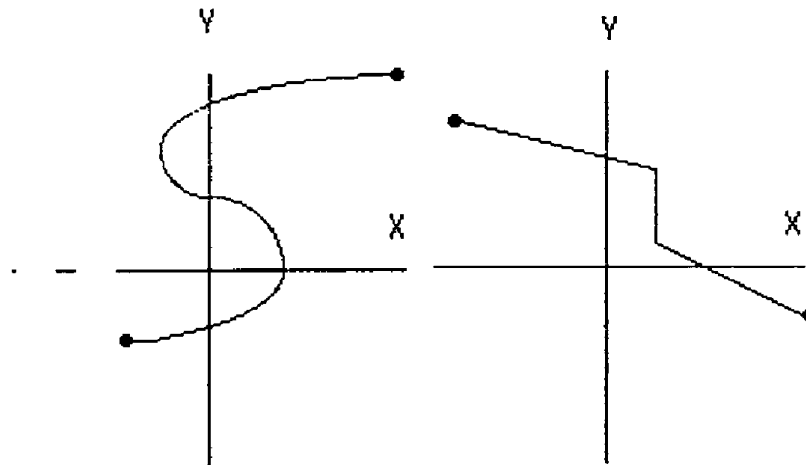


Figure 13. Illegal Functions

Programming Steps

- 1) Define the function data to be programmed
- 2) Erase breakpoints
- 3) Clear zero
- 4) Use SetUp (SU) command to set number of displayed digits
- 5) Program the minimum endpoint
- 6) Program the maximum endpoint
- 7) Program breakpoints
- 8) Verify the function

Step 1. Define the function data to be programmed. The ability of the M2000 to simulate a nonlinear transfer function is highly dependent on the location of the breakpoints selected by the programmer. The ultimate conformity to the desired function is directly dependent on the linear-segment approximation loaded into the module. The M2000 gives the programmer a great deal of flexibility in how the breakpoints are placed. In areas where the function curves sharply, or where greater accuracy is desired, breakpoints may be placed close together for better conformity to the desired function. The chart in Figure 4 is a handy form to help organize the breakpoint data.

Step 2. The breakpoint table should be cleared by using the Erase Breakpoints (EB) command. This command will completely erase the breakpoint table. Any previous breakpoint information will be permanently lost.

Command: \$1WE
Response: *

Command: \$1EB
Response: *

Step 3. Clear any data stored in the output offset register by using the Clear Zero (CZ) command:

Command: \$1WE
Response: *

Command: \$1CZ
Response: *

Step 4. Use the SetUp (SU) command to program the correct number of displayed digits:

Command: \$1WE
Response: *

Command: \$1SU31070182 (typical)
Response: *

Step 5. Start the function programming by setting the minimum point using the MiNimum (MN) command as described in the linear scaling section.

Step 6. Set the maximum function point with the MaXimum command as described in the linear scaling section.

Step 7. Use the BreakPoint (BP) command to program the nonlinear function into memory. Apply the proper excitation to the module for Breakpoint 00. Use the BreakPoint command to enter the data into memory:

Command: \$1WE
Response: *

Command: \$1BP00+00100.00
Response: *

At this point it may be useful to verify that the breakpoint data has indeed been recorded in memory. Without changing the excitation, read the output data:

Command: \$1
Response: *+00100.00

(5-4) MetraByte M2000 Programming Manual

The output data should match the data programmed with the Breakpoint command.

Once Breakpoint 00 has been entered, proceed to Breakpoint 01. Set the analog excitation for the correct value for Breakpoint 01. Load the breakpoint into memory using the BreakPoint command. Be sure to specify '01' in the BreakPoint command:

Command: \$1WE

Response: *

Command: \$1BP01+00200.00

Response: *

Verify that the data has been entered properly:

Command: \$1

Response: +00200.00

Continue this process until all breakpoints have been programmed.

Step 8. Test the input-output transfer function of the module to verify that the breakpoint data has been entered properly. Large errors in the output data are generally caused by improper breakpoint programming. In most cases it is not necessary to repeat the whole breakpoint sequence if the error is confined to one portion of the curve. Breakpoints may be re-programmed individually to correct any errors. However, it is not possible to insert extra breakpoints in the middle of the table to correct for a poor initial function approximation.

Example N-1

A voltage-output pressure sensor produces 0 V @ 100 psi and 5 V @ 600 psi. Its output characteristic is nonlinear and may be described by the equation:

$$P = 100 + 80 V + 4 V^2$$

where

V = sensor output in volts

P = pressure in psi

A simple linear equation may be derived by using the endpoint data:

$$P = 100 + 100 V$$

Unfortunately, describing the sensor output with this equation results in a 25 psi error at $V = 2.5 V$.

To obtain better accuracy, we may approximate the quadratic transfer function using breakpoints. Since the sensor output range is 0 - 5 V., the M2131 with an input range of +/- 5 V is most suitable for this application. For simplicity, we will use only four evenly-spaced breakpoints to plot the function. This will result in a function approximation with a maximum error of 1 psi. For better conformity, more breakpoints may be used.

1. First, construct the function table:

	<u>Analog Input</u>	<u>Output</u>
Minimum	0 V	+00100.00
Maximum	5 V	+00600.00
Breakpoint 00	1 V	+00184.00
Breakpoint 01	2 V	+00276.00
Breakpoint 02	3 V	+00376.00
Breakpoint 03	4 V	+00484.00

Notice that we've broken up the curve into five evenly-spaced voltage segments by using four breakpoints. The breakpoint output values were obtained by plugging the breakpoint voltage values into the quadratic equation that describes the sensor.

2. Prepare the M2131 by erasing any stored breakpoints:

(All programming commands must be preceded by a Write Enable (WE) command. In the interest of simplicity, the Write Enable commands are not shown in this or any of the following examples.)

Command: \$1EB
Response: *

3) Clear any data in the output offset register:

Command: \$1CZ
Response: *

4) We will setup the output data to display psi with .1 resolution:

Command: \$1SU31070182 (typical)
Response: *

(5-6) MetraByte M2000 Programming Manual

(The SU data may vary depending on your particular module setup. See the Setup section in the M1000 manual)

5. Apply 0 volts (short) to the input of the M2131 to enter the minimum point of 100 psi:

Command: \$1MN+00100.00

Response: *

6. To set the maximum point, apply 5 V to the input and program the maximum point to be 600 psi:

Command: \$1MX+00600.00

Response: *

7. Program the first breakpoint:

Apply 1 volt to the input and perform the BreakPoint command:

Command: \$1BP00+00184.00

Response: *

Verify the breakpoint data:

Command: \$1

Response: *+00184.00

Repeat the procedure for the remaining breakpoints:

Apply 2 volts to the input:

Command: \$1BP01+00276.00

Response: *

Command: \$1

Response: *+00276.00

Apply 3 volts to the input:

Command: \$1BP02+00376.00

Response: *

Command: \$1
Response: *+00376.00

Apply 4 volts to the input:

Command: \$1BP03+00484.00
Response: •

Command: \$1
Response: *+00484.00

The function programming is now complete.

8. The transfer function may be verified by applying test inputs to the module and obtaining output data. The data can then be compared to the original quadratic equation to check for conformity error.

Example:

Apply .5 volts to the M2131 input and read data:

Command: \$1
Response: *+00142.00

To check, plug .5 volts into the quadratic equation:

$$P = 100 + 80 (.5) + 4 (.5)^2 = 141$$

The conformity error at this point is +1 psi.

Example N-2

A pressure sensor rated for 0-200 psi has a nonlinear transfer function described by the relationship:

$$V = 4 \times 10^{-3} P + 5 \times 10^{-6} P^2$$

V = 0 to 1 volts

P = 0 to 200 psi

Use a M2121 ± 1 V input module to linearize the sensor output and convert the data to engineering units.

This example differs from Example N-1 because the desired output data in psi is the

(5-8) MetraByte M2000 Programming Manual

independent variable in the equation. One solution to this problem would be to convert the equation to a form of $P = f(V)$ and then proceed as we did in Example N-1. However, this kind of mathematical rigor is not necessary. To program the M2121, we simply need to construct a table of X,Y pairs. In this case, we may choose breakpoints to be in even intervals of psi, and then calculate the matching values of V. Our table with four breakpoints would look like this:

	<u>Input</u>	<u>Output</u>
Minimum	0 V	+00000.00
Maximum	+1 V	+00200.00
Breakpoint 00	.168 V	+00040.00
Breakpoint 01	.352 V	+00080.00
Breakpoint 02	.552 V	+00120.00
Breakpoint 03	.768 V	+00160.00

Notice that in this case, the breakpoints were selected by picking even intervals of pressure. The pressure values are then plugged into the sensor equation to produce the breakpoint voltages. The mechanics of entering the breakpoints is the same as in Example 1.

If better conformity is required, more breakpoints may be used. However, breakpoints cannot be simply added to the table at random. Breakpoints must be entered in sequence starting at the minimum value and progress in ever-increasing values of the X variable. To obtain better conformity, a new function table must be started from the beginning. Therefore, to avoid needless trial and error, it is best to test the breakpoint table on paper to determine if the conformity error is acceptable. Another approach is to simply use all 23 breakpoints available for the best conformity.

For this example, we may improve the conformity error by using nine breakpoints:

	<u>Input</u>	<u>Output</u>
Minimum	0 V	+00000.00
Maximum	1 V	+00200.00
Breakpoint 00	.082 V	+00020.00
Breakpoint 01	.168 V	+00040.00
Breakpoint 02	.258 V	+00060.00
Breakpoint 03	.352 V	+00080.00
Breakpoint 04	.45 V	+00100.00
Breakpoint 05	.552 V	+00120.00
Breakpoint 06	.658 V	+00140.00
Breakpoint 07	.768 V	+00160.00
Breakpoint 08	.882 V	+00180.00

Example N-3

In many cases, the system transfer function may not be known. In these situations, a M2000 may be programmed empirically using test inputs derived by the system itself.

A standpipe in a municipal water system has an irregular shape, as shown in Figure 14. It is desirable to obtain a direct reading of the volume of water contained in the standpipe. Because of the shape, a simple water height measurement would give grossly inaccurate readings of volume. Also, the actual relationship of volume to height is complex and unknown.

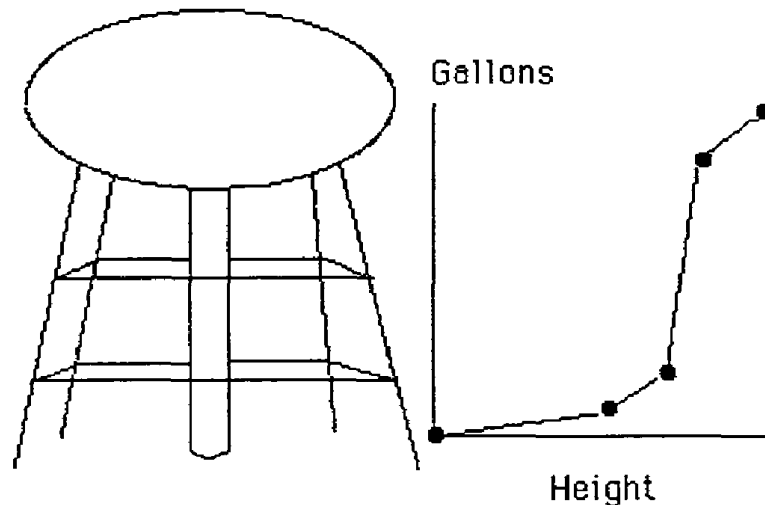


Figure 14. Scaling when function is unknown

The standpipe is 50 feet tall and has a known capacity of 30,000 gallons. A pressure sensor may be used at the base of the standpipe to obtain a reading of the water height. Since 1 foot of water produces a pressure of .4335 psi, the maximum pressure expected is $50 \times .4335 = 21.7$ psi. The pressure sensor we will use produces 0-5 volts for pressures of 0-25 psi. A M2131 ± 5 V input module will be used as the interface.

Install the pressure sensor and the M1231 in place at the standpipe. Prepare the M2131 by erasing breakpoints and clearing zero as detailed in Example N-1. In this case we will setup the M2131 to display four digits which will result in an output resolution of 10 gallons.

Start programming with the standpipe empty. Enter the minimum value:

Command: \$1MN+00000.00

Response: *

In this example, the maximum point may be programmed by filling the standpipe to obtain the maximum pressure output. However, this is awkward and unnecessary. Since the standpipe capacity is known to be 30,000 gallons and the pressure can never reach 25 psi, we can simulate a maximum that we know can never be attained. To do this we may apply 5 V to the module input to simulate 25 psi. The 5 V. source does not have to be accurate. We can set the

(5-10) MetraByte M2000 Programming Manual

maximum value to 35,000 gallons, which is more than the standpipe can hold.

Disconnect the pressure sensor and apply 5 V to the module input:

Command: \$1MX+35000.00

Response: *

Re-connect the pressure sensor to the M2131. Starting with the standpipe empty, we may begin to program the breakpoints. We will set a breakpoint every 1500 gallons for a total of 20 breakpoints.

To set the first breakpoint, fill the standpipe with 1500 gallons of water. Since we will be using actual volumes of water to 'calibrate' the standpipe, the accuracy at which we can measure 1500 gallons will greatly influence the final performance of the system.

With 1500 gallons in the standpipe used as the input excitation, program the first breakpoint:

Command: \$1BP00+01500.00

Response: *

Test:

Command: \$1

Response: *+01500.00

Fill the standpipe with an additional 1500 gallons to program the second breakpoint. The standpipe now holds 3000 gallons:

Command: \$1BP01+03000.00

Response: *

Command: \$1

Response: *+03000.00

Repeat these steps until the standpipe is full. For each step, fill the standpipe with an additional 1500 gallons and program the breakpoint with the accumulated amount of water in the standpipe. When the breakpoint programming is complete, the M2131 will give a very accurate indication of the volume of water in the standpipe directly in units of gallons.

In this example, the actual transfer function of the system is unknown. Instead, the function is plotted in the field by applying known inputs to the system. Note that the voltage produced by the pressure sensor does not have to be known to program the M2131. However it is wise to record the voltages produced by the sensor at each breakpoint. With this information, replacement M2131's may be programmed with a voltage source to avoid repeating the tank filling exercise.

Example N-4

Program a M2141 ± 10 V input module to calculate the square root of the input signal from 0 to 10 V. We'll keep the units in terms of millivolts so that the square root of 10,000 millivolts (10 V) is 100. To keep things simple for this example, we will create a function with nine breakpoints at even 1 V intervals.

1. Construct the function table:

	<u>Analog input</u>	<u>Data Output</u>
Minimum	0 V	+00000.00
Maximum	10 V	+00100.00
Breakpoint 00	1 V	+00031.62
Breakpoint 01	2 V	+00044.72
Breakpoint 02	3 V	+00054.77
Breakpoint 03	4 V	+00063.25
Breakpoint 04	5 V	+00070.71
Breakpoint 05	6 V	+00077.46
Breakpoint 06	7 V	+00083.67
Breakpoint 07	8 V	+00089.44
Breakpoint 08	9 V	+00094.87

2. Erase breakpoints:

Command: \$1EB
Response: *

3. Clear zero:

Command: \$1CZ
Response: *

4. Display all digits:

Command: \$1SU310701C2 (typical)
Response: *

5. Program minimum by applying 0 V (short) to input:

Command: \$1MN+00000.00
Response: *

(5-12) MetraByte M2000 Programming Manual

6. Program maximum by applying exactly 10 volts to the input:

Command: \$1MX+00100.00

Response: *

7. Program breakpoints:

Apply 1 volt to the input:

Command: \$1BP00+00031.62

Response: *

Apply 2 volts to the input:

Command: \$1BP01+00044.72

Response: *

Continue until all breakpoints are programmed.

8. Verify the transfer function.

To get better conformity more breakpoints may be programmed, especially near the minimum end of the scale where the function curvature is greatest. There is no particular requirement to have breakpoints at regular intervals. The breakpoint intervals may be varied to achieve the best overall conformity. Small breakpoint intervals assure better conformity in areas where the function curvature is the greatest.

Example N-5

Breakpoints may be used to improve the linearity of a module in linear output applications.

The M2141 module programmed in Example N-4 is to be programmed back to its original ± 10 V input-output transfer function.

1. Define function data:

	<u>Analog Input</u>	<u>Data Output</u>
Minimum	- 10 V	-10000.00
Maximum	+10 V	+10000.00

(No Breakpoints)

2. Erase breakpoints.
3. Clear zero
4. Setup the displayed output for five digits:

Command: \$1SU31070142 (typical)
Response: *

5. Program minimum by applying exactly -10 V to the input:

Command: \$1MN-10000.00
Response: *

6. Program maximum by applying +10 V to the input:

Command: \$1MX+10000.00
Response: *

7. There are no breakpoints to be programmed.
8. Verify the function scaling.

<u>Analog Input</u>	<u>Data Output</u>
- 5 V	-05010.00
0 V	-00020.00
+ 5 V	+04990.00

During the verification process, we find that the module exhibits some errors. This is due to the .1% typical error inherent in the analog-to-digital converter. The nonlinearity may be corrected by using breakpoints. In this case, instead of using the breakpoints to create a nonlinear function, they will be used to 'straighten' the nonlinearity of the ADC. Only a few breakpoints

(5-14) MetraByte M2000 Programming Manual

are necessary to reduce the linearity error to .02 % or less. In this case we will use three breakpoints to linearize the module:

	<u>Analog Input</u>	<u>Data Output</u>
Minimum	- 10 V	-10000.00
Maximum	+ 10 V	+10000.00
Breakpoint 00	- 5 V	-05000.00
Breakpoint 01	0 V	+00000.00
Breakpoint 02	+5 V	+05000.00

Since the minimum and maximum data have already been programmed, only Step 7 is necessary to program in the breakpoints.

After the breakpoints have been entered, verify the module transfer function:

<u>Analog Input</u>	<u>Data Output</u>
- 7.5 V	-07502.00
0 V	+00000.00
+7.5 V	+07498.00

This time the module output is in error by .02 % or less due to linearizing effect of the breakpoints.

Example N-6

A M2141 module may be programmed to create an absolute-value function as shown in Figure 15. However, this function violates the 'rectangular area' rule. To overcome this limitation, the function may be re-drawn as shown in Figure 16. This curve satisfies the 'rectangular area' rule. The function table for this curve looks like this:

	<u>Analog Input</u>	<u>Data Output</u>
Minimum	- 10 V	-00010.00
Maximum	+10 V	+10000.00
Breakpoint 00	-9.990 V	+09990.00
Breakpoint 01	0 V	+00000.00

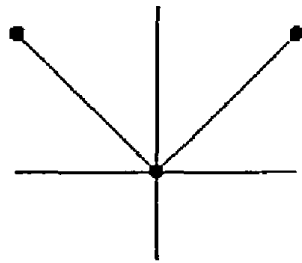


Figure 15.

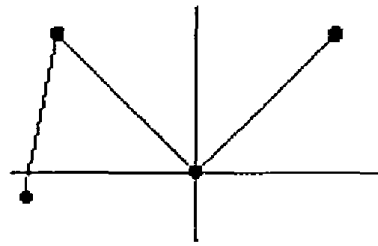


Figure 16.

The absolute-value function will be valid for inputs between - 9.990 V and + 10 V. This technique may be used for other functions that violate the 'rectangular area' rule.

Specifications are subject to change without notice.

All Keithley trademarks and trade names are the property of Keithley Instruments, Inc. All other trademarks and trade names are the property of their respective companies.

KEITHLEY

Keithley Instruments, Inc.

28775 Aurora Road • Cleveland, Ohio 44139 • 440-248-0400 • Fax: 440-248-6168
1-888-KEITHLEY (534-8453) • www.keithley.com

Sales Offices: BELGIUM:

Bergensesteenweg 709 • B-1600 Sint-Pieters-Leeuw • 02-363 00 40 • Fax: 02/363 00 64

CHINA:

Yuan Chen Xin Building, Room 705 • 12 Yumin Road, Dewai, Madian • Beijing 100029 • 8610-6202-2886 • Fax: 8610-6202-2892

FINLAND:

Tietäjantie 2 • 02130 Espoo • Phone: 09-54 75 08 10 • Fax: 09-25 10 51 00

FRANCE:

3, allée des Garays • 91127 Palaiseau Cédex • 01-64 53 20 20 • Fax: 01-60 11 77 26

GERMANY:

Landsberger Strasse 65 • 82110 Germering • 089/84 93 07-40 • Fax: 089/84 93 07-34

GREAT BRITAIN:

Unit 2 Commerce Park, Brunel Road • Theale • Berkshire RG7 4AB • 0118 929 7500 • Fax: 0118 929 7519

INDIA:

Flat 2B, Willocrissa • 14, Rest House Crescent • Bangalore 560 001 • 91-80-509-1320/21 • Fax: 91-80-509-1322

ITALY:

Viale San Gimignano, 38 • 20146 Milano • 02-48 39 16 01 • Fax: 02-48 30 22 74

JAPAN:

New Pier Takeshiba North Tower 13F • 11-1, Kaigan 1-chome • Minato-ku, Tokyo 105-0022 • 81-3-5733-7555 • Fax: 81-3-5733-7556

KOREA:

2FL., URI Building • 2-14 Yangjae-Dong • Seocho-Gu, Seoul 137-888 • 82-2-574-7778 • Fax: 82-2-574-7838

NETHERLANDS:

Postbus 559 • 4200 AN Gorinchem • 0183-635333 • Fax: 0183-630821

SWEDEN:

c/o Regus Business Centre • Frosundaviks Allé 15, 4tr • 169 70 Solna • 08-509 04 679 • Fax: 08-655 26 10

SWITZERLAND:

Kriesbachstrasse 4 • 8600 Dübendorf • 01-821 94 44 • Fax: 01-820 30 81

TAIWAN:

1FL., 85 Po Ai Street • Hsinchu, Taiwan, R.O.C. • 886-3-572-9077 • Fax: 886-3-572-9031