DIN-100 SERIES USERS MANUAL

REVISED: 06/2000

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

TELEPHONE: 603-622-0452

FAX: 603-622-0487

URL: http://www.dghcorp.com

DIN-100 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

DGH warrants each DIN-100 series module to be free from defects in materials and workmanship under normal conditions of use and service and will replace any component found to be defective, on its return to DGH, transportation charges prepaid within one year of its original purchase. DGH assumes no liability, expressed or implied, beyond its obligation to replace any component involved. Such warranty is in lieu of all other warranties expressed or implied.

WARNING

The circuits and software contained in DIN-100 series modules are proprietary. Purchase of these products does not transfer any rights or grant any license to the circuits or software used in these products. Disassembling or decompiling of the software program is explicitly prohibited. Reproduction of the software program by any means is illegal.

As explained in the setup section, all setups are performed entirely from the outside of the DIN-100 module. There is no need to open the module because there are no user-serviceable parts inside. Removing the cover or tampering with, modifying, or repairing by unauthorized personnel will automatically void the warranty. DGH is not responsible for any consequential damages.

RETURNS

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

Shipping Address:

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

Default Mode

All DIN-100 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 is difficult to establish communications with a module whose address and baud rate are unknown. To overcome this, each module has an input pin labeled DEFAULT*. By connecting this pin to Ground, the module is put in a known communications setup called Default Mode.

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

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

A module in Default Mode will respond to any address except the six 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-485 Quick Hook-Up

Software is not required to begin using your DIN-100 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 is scaled in °C which has been preset at the factory. Once you have a response from the module you can turn to the Chapter 4 and get familiar with the command set.

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

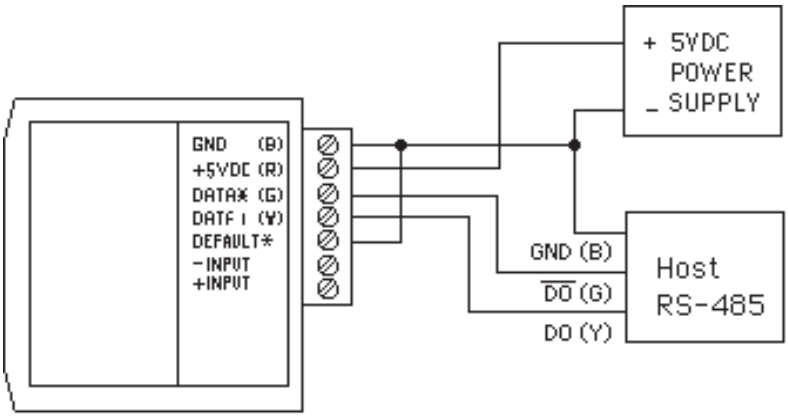


Figure 1.1 RS-485 Quick Hook-Up.

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

An RS-485 module may be easily interfaced to an RS-232C terminal for evaluation purposes. This connection is only suitable for benchtop operation and should never be used for a permanent installation. Figure 1.2 shows the hook-up. This connection will work provided the RS-232C transmit output is current limited to less than 50mA and the RS-232C receive threshold is greater than 0V. All terminals that use 1488 and 1489 style interface IC's will satisfy this requirement. With this connection, characters generated by the terminal will be echoed back. To avoid double characters, the local echo on the terminal should be turned off.

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

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

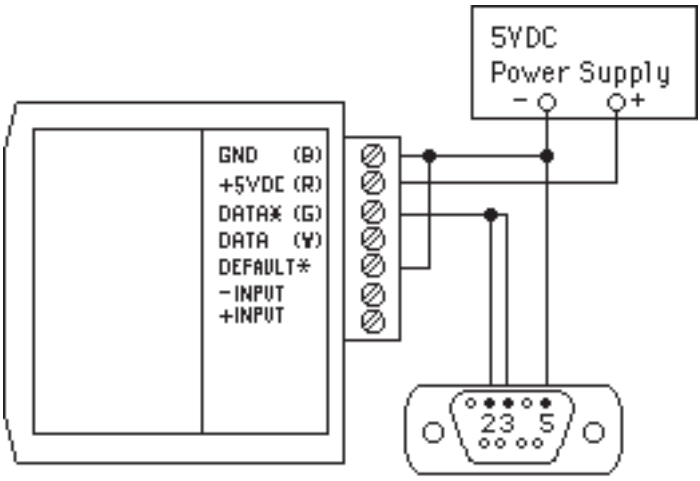


Figure 1.2 RS-485 Quick Hook-Up with RS-232C Port.

Chapter 2
Functional Description

A functional diagram of a typical module is shown in Figure 2.1. It is a useful reference that shows 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/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 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 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 has many 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 value in the offset register. The offset register data is nonvolatile.

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, the same data value may be read back several times before it is updated with a new A/D conversion.

The DIN-170 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.

The DIN-170 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 available varies with the module type.

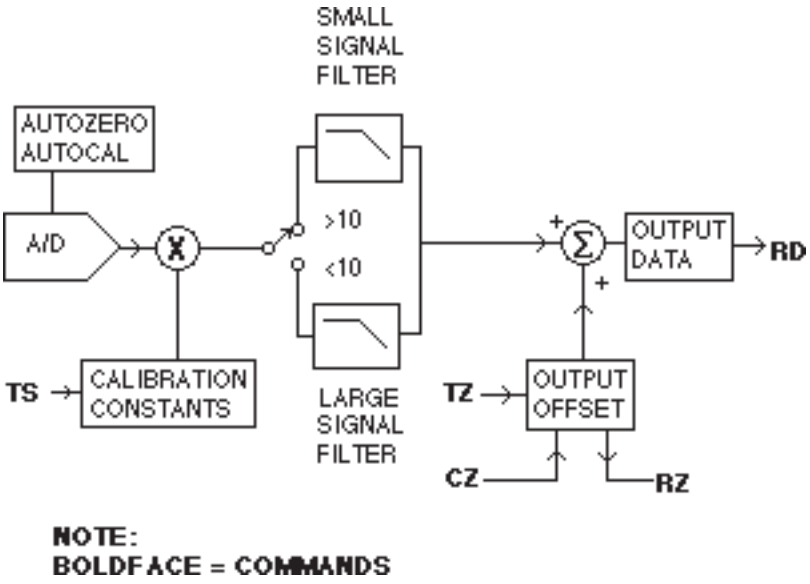


Figure 2.1 Analog Input Block Diagram.

Chapter 3
Communications

Introduction

The DIN-100 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. The ASCII format makes system debugging easy with a dumb terminal.

This system allows multiple modules to be connected to a communications port with a single 4-wire cable. Up to 32 RS-485 modules may be strung together on one cable; 122 with repeaters. The modules communicate with the host on a polling system; that is, each module responds to its own unique address and must be interrogated by the host. A module can never initiate a communications sequence. A simple command/response protocol must be strictly observed to avoid communications collisions and data errors.

Communications to the DIN-100 modules is performed with two-character or three-character ASCII command codes such as RD to Read Data from the analog input. A complete description of all commands is given in the Chapter 4. A typical command/response sequence would look like this:

Command: **$1RD**
Response: ***+00123.00**

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

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

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

Mnemonic	Timeout
DI,DO,RD	10 mS
All other commands	100 mS

Table 3.1 Response Timeout Specifications.

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

Data Format

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

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).

RS-485

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

1) balanced line gives excellent noise immunity
2) can communicate with D1000 modules at 115200 baud
3) communications distances up to 4,000 feet.
4) true multidrop; modules are connected in parallel
5) can disconnect modules without losing communications
6) up to 32 modules on one line; 122 with repeaters
7) no communications delay due to multiple modules
8) simplified wiring using standard telephone cable

RS-485 does have disadvantages. Very few computers or terminals have built-in support for this new standard. Interface boards are available for the IBM PC and compatibles. As RS-485 system usually requires an interface.

The DIN-190 will convert RS-232 signals to RS-485 or repeat RS-485 signals. The DIN-190 connected as an RS-485 repeater can be used to

extend an existing RS-485 network or connect up to 122 modules on one serial communications port.

RS-485 Multidrop System

Figure 3.1 illustrates the wiring required for multiple-module RS-485 system. Notice that every module has a direct connection to the host system. Any number of modules may be unplugged without affecting the remaining modules. Each module must be setup with a unique address and the addresses can be in any order. All RS-485 modules must be setup for no echo to avoid bus conflicts (see Setup). Also note that the connector pins on each module are labelled with notations (B), (R), (G), and (Y). This designates the colors used on standard 4-wire telephone cable:

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

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

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

To minimize unwanted reflections on the transmission line, the bus should be arranged as a line going from one module to the next. ‘Tree’ or random structures of the transmission line should be avoided. When using long transmission lines and/or high baud rates, the data lines should be terminated at each end with 200 ohm resistors. Standard values of 180 ohms or 220 ohms are acceptable.

During normal operation, there are periods of time where all RS-485 drivers are off and the communications lines are in an 'idle' high impedance condition. During this condition, the lines are susceptible to noise pickup which may be interpreted as random characters on the communications line. To prevent noise pickup, all RS-485 systems should incorporate 1K ohm bias resistors as shown in Figure 3.1. The resistors will maintain the data lines in a 'mark' condition when all drivers are off.

DIN-191 and DIN-192 modules have the 1KΩ resistors built-in.

Special care must be taken with very long busses (greater than 1000 feet) to ensure error-free operation. Long busses must be terminated as described above. The use of twisted cable for the DATA and DATA* lines will greatly enhance signal fidelity. Use parity and checksums along with the ‘#’

form of all commands to detect transmission errors. In situations where many modules are used on a long line, voltage drops in the power leads becomes an important consideration. The GND wire is used both as a power connection and the common reference for the transmission line receivers in the modules. Voltage drops in the GND leads appear as a common-mode voltage to the receivers. The receivers are rated for a maximum of -7V. of common-mode voltage. For reliable operation, the common mode voltage should be kept below -5V.

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

Communications Delay

All DIN-100 modules are setup at the factory to provide two units of communications delay after a command has been received (see Chapter 5). This delay is necessary when using host computers that transmit a carriage return as a carriage return-linefeed string. Without the delay, the linefeed character may collide with the first transmitted character from the module, resulting in garbled data. If the host computer transmits a carriage return as a single character, the delay may be set to zero to improve communications response time.

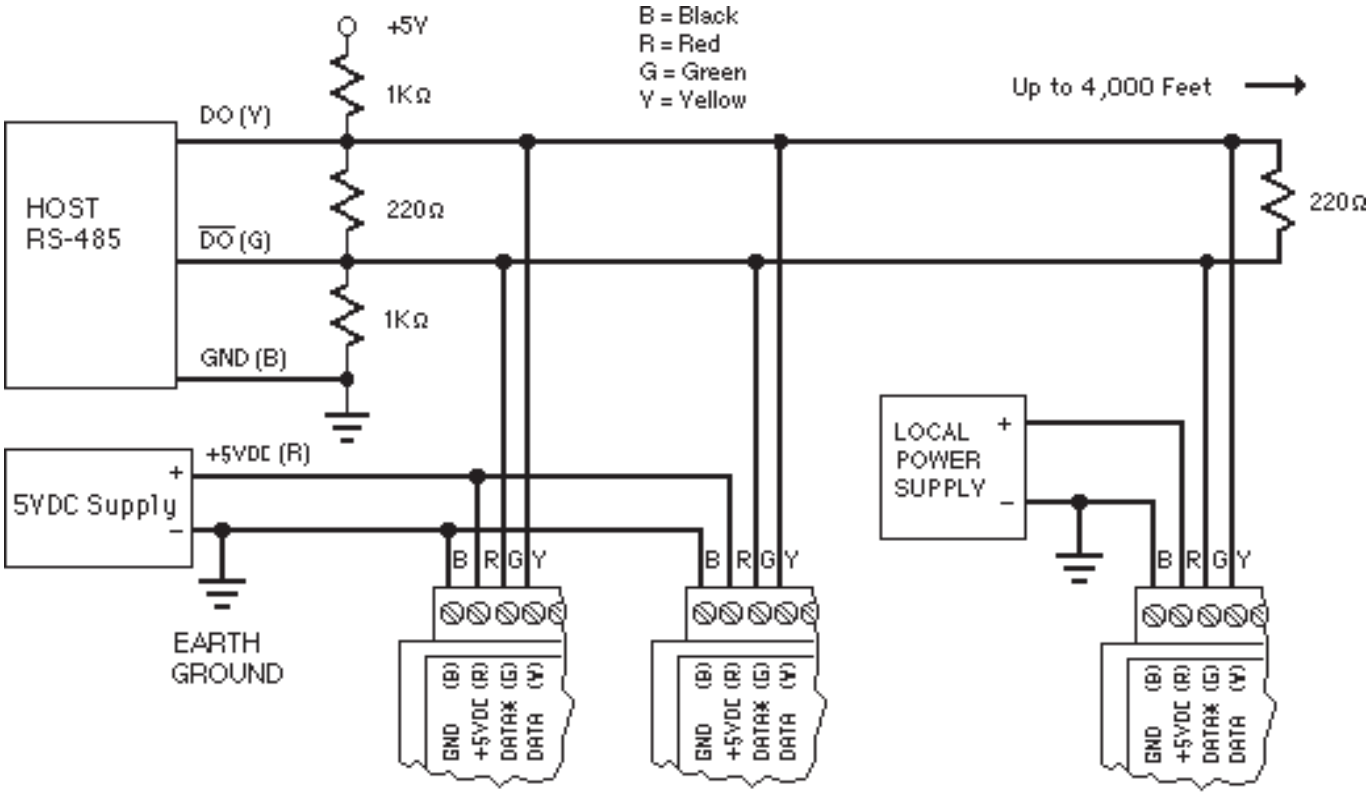


Figure 3.1 RS-485 Network.

Chapter 4
ASCII Command Set

The DIN-100 modules operate with a simple command/response protocol to control all module functions. A command must be transmitted to the module by the host computer or terminal before the module will respond with useful data. A module can never initiate a communications sequence. A variety of commands exists to exploit the full functionality of the modules. A list of available commands and a sample format for each command is listed in Table 4.1.

Command Structure

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

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

The address character is followed by a two-character command that identifies the function to be performed by the module. All of the available commands are listed in Table 4.1 along with a short function definition. All commands are described in Chapter 4. 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' in Chapter 4 .

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 DIN-100 series. Analog data is represented as a nine-character string consisting of a sign, five digits, decimal point, and two additional digits. The string represents a decimal value in engineering units. Examples:

+12345.68
+00100.00
-00072.10
-00000.00

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

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

In many cases, some of the digits in the analog data may not be significant. For instance, the DIN-130 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.

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

Miscellaneous Protocol Notes

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

The length of a command message is limited to 20 printable characters. If a properly addressed module receives a command message of more than 20 characters the module will abort the whole command sequence and no response will result.

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

Response Structure

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

The maximum response message length is 20 characters.

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

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

When a module receives a valid command, it must interpret the command, perform the desired function, and the communicate the response back to the host. Each command has an associated delay time in which the module is busy calculating the response. If the host does not receive a response in an appropriate amount of time specified in Table 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 responds with a 'long form' response. This type of response will echo the command message, supply the necessary response data and will add a two-character checksum to the end of the message. Long form responses are used when the host wishes to verify the command received by the module. The checksum is included to verify the integrity of the response data. The ' # ' command prompt may be used with any command. For example:

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

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

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

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

Command: **$1RD** **(no checksum)**
Response: ***+00072.10**

Command: **$1RDEB** **(with checksum)**
Response: ***+00072.10**

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

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

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

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

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

Checksum Calculation

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

Example: Append a checksum to the command #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:

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

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

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

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

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

Parity bits are never included in the checksum calculation.

Table 4.1 DIN-100 Command Set

Command and Definition		Typical Command Message	Typical Response Message ($ prompt)
DI	Read Alarms/Digital Inputs	$1DI	*0003
DO	Set Digital Outputs	$1DOFF	*
RD	Read Data	$1RD	*+00072.00
RS	Read Setup	$1RS	*31070142
RZ	Read Zero	$1RZ	*+00000.00
WE	Write Enable	$1WE	*
Write Protected Commands			
MBR	Set Modbus Address	$1MBR01	*
MBD	Modbus Disable	$1MBD	*
RR	Remote Reset	$1RR	*
SU	Setup Module	$1SU31070142	*
TS	Trim Span	$1TS+00600.00	*
TZ	Trim Zero	$1TZ+00000.00	*

DIN-100 User Commands

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

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).

Command: $1CZ
Response: *

Command: #1CZ
Response: *1CZF8

Digital Input (DI)

The DI command reads the status of the digital inputs on the DIN-171. The response to the DI command is four hex characters representing two bytes of data. The second byte contains the digital input data.

Command: $1DI
Response: *0003

Command: #1DI
Response: *1DI0003AB

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

Digital Inputs	DI5	DI4	DI3	DI2	DI1	DI0
Data Bits	5	4	3	2	1	0

For example: A typical response from a $1DI command could be: *00FE. This response indicates that 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 DIN-172 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 output settings are not stored in nonvolatile memory. If a power failure occurs, all digital outputs will be 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 digital outputs.

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**

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 3 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 and other parameters. Refer to the setup command (SU), and Chapter 5 for a list of parameters in the setup information.

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

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: Trim Zero (TZ) and Clear Zero (CZ).

Setup Command (SU)
Each DIN-100 module contains an EEPROM (Electrically Erasable Programmable Read Only Memory) which is used to store module setup information such as address, baud rate, parity, etc. The EEPROM is a special type of memory that will retain information even if power is removed from the module. The EEPROM is used to replace the usual array of DIP switches normally used to configure electronic equipment.

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

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

Command: **$1SU31070182**
Response: *****

Command: **#1SU31070182**
Response: ***1SU3107018299**

Trim Span (TS)
The trim span command is the basic means of trimming the accuracy of a DIN-100 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 ±10% of the nominal calibration set at the factory. It is not 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 Chapter 9.

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 errors loaded by the TS command. Misuse 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. An input signal must be applied when using this command.

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 offset in the output data. It may be used to trim offsets created by sensors. It may also be used to null out data to create a deviation output.

Example: Assume a DIN-151 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, trim the output to read zero. To trim, use the TZ command and specify the desired output reading:

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

With no weight applied, trim the output to read zero. To trim, 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-00100.00
Response: *

The new data output with no load applied would be:

Command: $1RD
Response: *-00100.00

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 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 will remain write enabled until a command is successfully completed resulting in an ' * ' prompt. This allows the user to correct the command error without having to execute another WE command.

Write Extended Address (WEA)

ERROR MESSAGES

The DIN-100 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.

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

?1 SYNTAX ERROR

There are eight error messages, and each error message begins with a different character. It is easy for a computer program to identify the error without havi**ng to read the** entire string.

ADDRESS ERROR

There are six ASCII values that are illegal for use as a module address: NULL ($00), CR ($0D), $ ($24), # ($23), { ($7B) and } ($7D). 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 th**an $7F will p**roduce an error.

BAD CHECKSUM

This error is caused by an incorrect checksum included in the command string. The module recognizes any two hex characters appended to a command string as a checksum. Usually a BAD CHECKSUM error is due to noise or interference on the communications line. Often, repeating the command solves the problem. If the error persists, either the checksum is calculated incorrectly or there is a problem with the communications channel. More reliable transmissions might be obtaine**d by using a l**ower 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 v**alid comma**nds 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 three ways: a power-up reset, a Remote Reset (RR) command, 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 be changed or turn**ed off by usi**ng 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 ra**nge from 0-F.

WRITE PROTECTED

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

Chapter 5
Setup Information/SetUp Command

The DIN-100 modules feature a wide choice of user configurable options which gives them the flexibility to operate on virtually any computer or terminal based system. The user options include a choice of baud rate, parity, address, and many other parameters. The particular choice of options for a module is referred to as the setup information.

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

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

The following options can be specified by the SetUp command:

Channel address (122 values)
Linefeeds
Parity (odd, even, none)
Baud rate (300 to 38,400)
Addressing Mode: Extended/Normal
CJC disable (DIN-130 series)
RTD 3/4 wire (DIN-140 series)
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.

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 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' and 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.

are six ASCII codes that are illegal for use as an address. These codes are $00, $0D, $24, $23, $7B, $7D which are ASCII codes for the characters NUL, CR, $, #, { and }. Using these codes for an address will cause an ADDRESS ERROR and the setup data will remain unchanged. This leaves a total of 122 possible addresses that can be loaded with the SU command. It is highly recommended that only ASCII codes for printable characters be used ($21 to $7E) which greatly simplifies system debugging with a dumb terminal. Refer to Appendix A for a list of ASCII codes. Table 5.1 lists the printable ASCII codes that may be used as addresses.

Table 5.1 Byte 1 ASCII Printable Characters.

HEX	ASCII	HEX	ASCII	HEX	ASCII	HEX	ASCII
21	!	3A	:	51	Q	68	h
22	“	3B	;	52	R	69	i
25	%	3C	<	53	S	6A	j
26	&	3D	=	54	T	6B	k
27	‘	3E	>	55	U	6C	l
28	(	3F	?	56	V	6D	m
29	)	40	@	57	W	6E	n
2A	*	41	A	58	X	6F	o
2B	+	42	B	59	Y	70	p
2C	,	43	C	5A	Z	71	q
2D	-	44	D	5B	[	72	r
2E	.	45	E	5C	\	73	s
2F	/	46	F	5D	]	74	t
30	0	47	G	5E	^	75	u
31	1	48	H	5F	_	76	v
32	2	49	I	60	‘	77	w
33	3	4A	J	61	a	78	x
34	4	4B	K	62	b	79	y
35	5	4C	L	63	c	7A	z
36	6	4D	M	64	d	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 DIN-100 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 D1000 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 0-3 specify the communications baud rate. The baud rate can be selected from ten 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 (see Communications) or powering down. This extra level of write protection is necessary to ensure that communications to the module is not accidently lost. This is very important when

changing the baud rate of an RS-485 string. For more information on changing baud rate, refer to Chapter 3.

Let's run through an example of changing the baud rate. Assume our sample module contains the setup data value of '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 '0010' (from Table 5.2). This would change byte 2 to '02'. To perform the SU command we must first send a Write Enable command because SU is write protected:

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

Command: **$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.

Bit 4
Bit 4 is used to enable or disable extended addressing mode.

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

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					
NORMAL ADDRESSING				0				
EXTENDED ADDRESSING				1				
38400 BAUD					0	0	0	0
19200 BAUD					0	0	0	1
9600 BAUD					0	0	1	0
4800 BAUD					0	0	1	1
2400 BAUD					0	1	0	0
1200 BAUD					0	1	0	1
600 BAUD					0	1	1	0
300 BAUD					0	1	1	1

Byte 3
This byte contains the setup information for several seldom-used options. The default value for this byte is ‘01’.

Disable CJC
RTD 3/4 Wire
Trigger Edge Select
The setup information stored in bit 4 has different meanings depending on the DIN-100 model number.

Disable CJC; this function pertains only to the DIN-130 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 degrees Celsius. 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 function pertains only to the DIN-140 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. This function has no affect on DIN-145 or DIN-146 Thermistor inputs.

Delay

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

In some systems, such as IBM BASIC, a carriage return (CR) is always followed by a linefeed (LF). The DIN-100 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.

FUNCTION	DATA BIT							
	7	6	5	4	3	2	1	0
CJC (DIN-130'S)				0				
NO CJC (DIN-130'S)				1				
3 WIRE (DIN-140'S)				0				
4 WIRE (DIN-140'S)				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 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 32,768. 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 DIN-141 model for RTD's has 0.1 degree output resolution. The appropriate number of digits for this module is 6, to mask off the 0.01 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 right-most 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 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 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 DIN-141 RTD module has a standard output resolution of 0.1 degrees. The standard number-of-displayed-digits setup for this module is 6 digits, from 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 > 1.0 degree:

Previous data	New data	Filter selected
+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, 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
+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 seconds 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 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.

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

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

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

DIN-100 Series Model Number	Setup Message
111, 115, 125	310701C2
110, 113, 114	31070142
13X	31070142
141, 142, 143 , 112	31070182
145, 15X	310701C2
16X	310701C0
17X	31070100

Chapter 6
Digital I/O Functions

The DIN-100 series features the DIN-171 module with six digital inputs and the DIN-172 module with six digital outputs .

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 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 100mA and can withstand up to 30V. Power in the transistor must be limited to 300mW. The emitter of each transistor is tied to the GND terminal on the input connector.

A typical connection of a digital output is shown in Figure 6.1. In this case, a solid state relay is controlled by the DIN-172 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

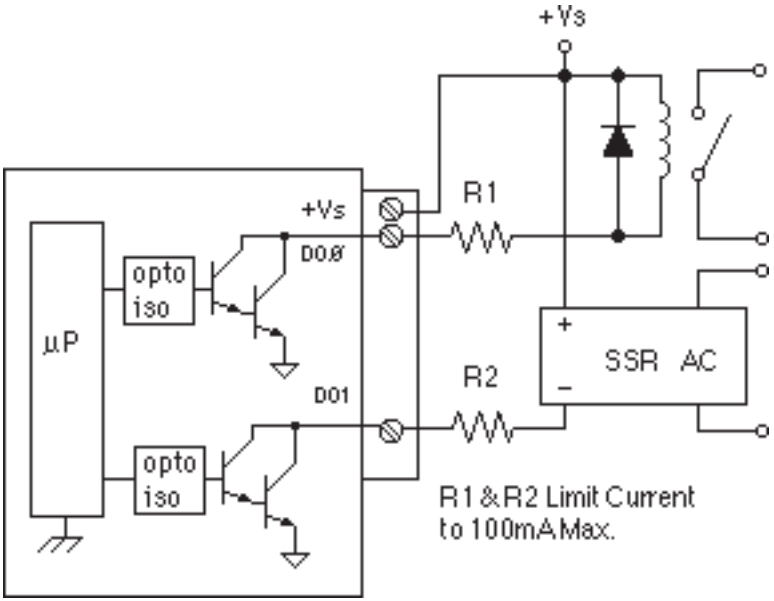


Figure 6.1 Digital Outputs Used With Relays

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 100mA.

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 digital outputs are not affected by the Remote Reset (RR) command.

Digital Inputs

Digital inputs are used to sense switch closures and the state of digital signals. The inputs are protected to voltages up to ±30V and are normally pulled up to the logic “1” condition (see Figure 6.2). Digital inputs can be read by the Digital Input (DI) command. Voltage inputs less than 1V are read back as ‘0’. Signals greater than 3.5V 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 with the addition of an external 10K pull-up resistor to +5Vdc. The pull-up supplies only 0.5ma; therefore, self-wiping switches designed for low current operation should be used.

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

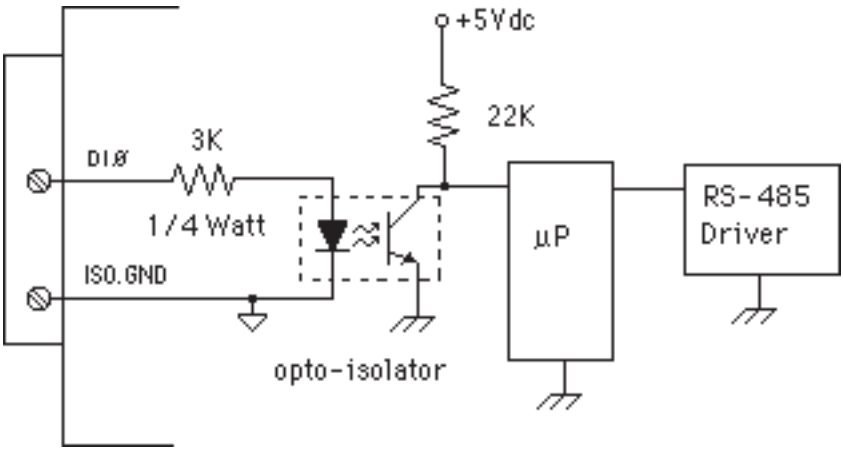


Figure 6.2 Typical Digital Input Circuit.

Chapter 7
Power Supply

DIN-100 modules require a regulated +5Vdc power-supply. The modules contain a low voltage detection circuit that shuts down all circuits in the module at approximately 4.5 Vdc. 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.

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

The low voltage detection circuit shuts down the module at approximately 4.5Vdc. If the module is interrogated while in a low power supply condition, the module will not respond. Random NOT READY error messages could indicate that the power supply voltage is periodically drooping below the 4.7V minimum.

In some cases, a small number of modules may be operated by “stealing” power from a host computer or terminal.

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 DIN-100 modules are protected against power supply reversals.

Chapter 8
Troubleshooting

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

• RS-485 Module is not responding to commands

1. Using a voltmeter, measure the power supply voltage at the +Vs and GND terminals to verify the power supply voltage is constantly 5Vdc ± 5%.

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

3. When using a serial communications converter (DIN-191) ensure that the communications Baud Rate switch is set to the proper Baud Rate value.

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

5. If the Baud Rate value being used in the application is greater than 300 Baud and the module will only communicate 300 Baud then make sure that the DEFAULT* terminal is not connected to Ground (GND).

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

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

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

• Module responds with ?1 COMMAND ERROR to every command

Ensure that characters in the command message are uppercase characters. All commands consist of uppercase characters only.

• Characters in each response message appear as graphics characters

1. Set the communications software parity setting to "M" for 'MARK' parity type and 7 data bits. Or, utilize any parity type in both the module and software other than "NO" parity.

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

Chapter 9
Calibration

The DIN-100 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. Calibration procedure is as follows.

Voltage and current inputs: clear the output offset register using the Clear Zero (CZ) command. Zero trims are not neccessary 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. 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 with the RD command. For example to trim a DIN-112 module:

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 use a +900mV input voltage 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 300µV. To trim:

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

Command: **$1TS+00900.00**
Response: *****

This sequence will trim the output to +00900.00. Verify:

Command: **$1RD**
Response: ***+00900.00 (The module is calibrated.)**

Thermocouples: 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, 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
DIN-110	+9000μV	+09000.00
DIN-111	+90mV	+00090.00
DIN-112	+900mV	+00900.00
DIN-113	+4.5V	+04500.00
DIN-114	+9V	+09000.00
DIN-115	+90V	+00090.00
DIN-125	+20mA	+00020.00
DIN-131	+39.13mV	+00700.00
DIN-132	+41.269mV	+01000.00
DIN-133	+17.816mV	+00350.00
DIN-134	+68.783mV	+01000.00
DIN-135	+17.445mV	+01500.00
DIN-136	+15.576mV	+01500.00
DIN-137	+10.094mV	+01500.00
DIN-138	+33.442mV	+01982.00
DIN-141	300.00Ω	+00558.00
DIN-142	300.00Ω	+00547.60
DIN-143	134.91Ω	+00115.00
DIN-145	206.1Ω	+00090.00
DIN-146	3018Ω	+00140.00
DIN-151	25mV	+00025.00
DIN-152	25mV	+00025.00
DIN-153	90mV	+00090.00
DIN-154	90mV	+00090.00
DIN-161	18Khz	+18000.00

Chapter 10
Extended Addressing

The DIN-100 may be configured to a special command format called Extended Addressing. This mode uses a different prompt, either '{' or '}' to distinguish it from the regular command syntax. The major difference in syntax for the Extended Addressing mode is that it uses a two-character address. A typical command in Extended Address mode would look like this:

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

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

There are two benefits to using Extended Addressing with the DIN-100:

1) Greatly expanded addressing capability.

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

With single-byte addressing of the normal command structure, address space is limited to 122 points. Extended addressing allows an addressing range of 249 points.

Structured Addressing
Even for a relatively small system, it can be advantageous to employ a hierarchical addressing system as used in Fig. 7.1. This is particularly true in systems that consist of many sites that are identical. From a host software standpoint, each site can be treated identically with the same module addresses, with each site having a different DIN-100 address.

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

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

Command examples with Extended Address '01':

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

Command: **}01WE**
Response: ***01WE27**

Command: **{01RS**
Response: ***31070000** **(typical)**

Command: **}01RS**
Response: ***01RS31070000BB** **(typical)**

Checksums may be appended to commands:

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

All commands that are available with single-byte addressing may be accessed with Extended Addressing, and vice-versa.

Appendix A
ASCII Table

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

A	D	Hex	Binary	D	Hex	Binary
^@	0	00	00000000	128	80	10000000
^A	1	01	00000001	129	81	10000001
^B	2	02	00000010	130	82	10000010
^C	3	03	00000011	131	83	10000011
^D	4	04	00000100	132	84	10000100
^E	5	05	00000101	133	85	10000101
^F	6	06	00000110	134	86	10000110
^G	7	07	00000111	135	87	10000111
^H	8	08	00001000	136	88	10001000
^I	9	09	00001001	137	89	10001001
^J	10	0A	00001010	138	8A	10001010
^K	11	0B	00001011	139	8B	10001011
^L	12	0C	00001100	140	8C	10001100
^M	13	0D	00001101	141	8D	10001101
^N	14	0E	00001110	142	8E	10001110
^O	15	0F	00001111	143	8F	10001111
^P	16	10	00010000	144	90	10010000
^Q	17	11	00010001	145	91	10010001
^R	18	12	00010010	146	92	10010010
^S	19	13	00010011	147	93	10010011
^T	20	14	00010100	148	94	10010100
^U	21	15	00010101	149	95	10010101
^V	22	16	00010110	150	96	10010110
^W	23	17	00010111	151	97	10010111
^X	24	18	00011000	152	98	10011000
^Y	25	19	00011001	153	99	10011001
^Z	26	1A	00011010	154	9A	10011010
^[	27	1B	00011011	155	9B	10011011
^\	28	1C	00011100	156	9C	10011100
^]	29	1D	00011101	157	9D	10011101
^^	30	1E	00011110	158	9E	10011110
^_	31	1F	00011111	159	9F	10011111
	32	20	00100000	160	A0	10100000
!	33	21	00100001	161	A1	10100001
“	34	22	00100010	162	A2	10100010

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

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

A	D	Hex	Binary	D	Hex	Binary
u	117	75	01110101	245	F5	11110101
v	118	76	01110110	246	F6	11110110
w	119	77	01110111	247	F7	11110111
x	120	78	01111000	248	F8	11111000
y	121	79	01111001	249	F9	11111001
z	122	7A	01111010	250	FA	11111010
{	123	7B	01111011	251	FB	11111011
|	124	7C	01111100	252	FC	11111100
}	125	7D	01111101	253	FD	11111101
~	126	7E	01111110	254	FE	11111110
	127	7F	01111111	255	FF	11111111

Appendix B
DIN-160 Data Sheet

The Frequency Input modules feature a versatile input stage that can be used in a variety of applications. Figure B-1 is a block diagram of the input signal conditioning.

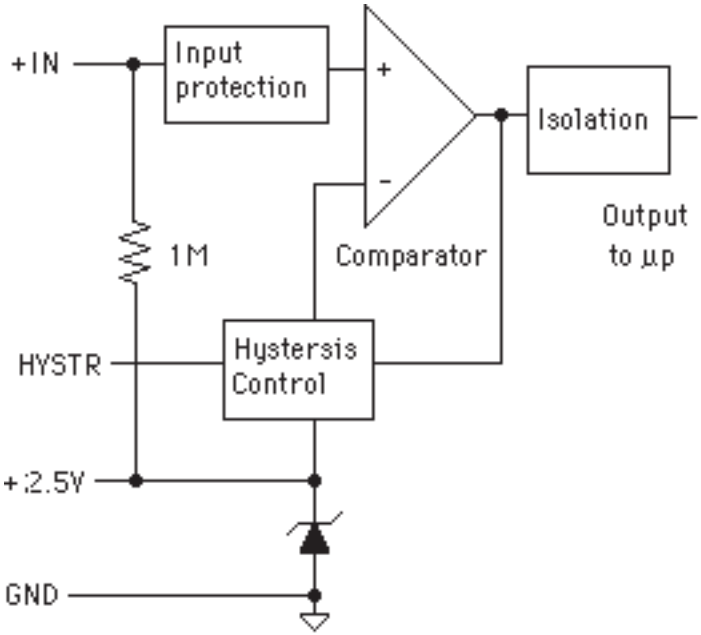


Figure B-1. DIN-161 Input Signal Conditioning Block Diagram.

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. Figure 2 shows the most frequently used connection.

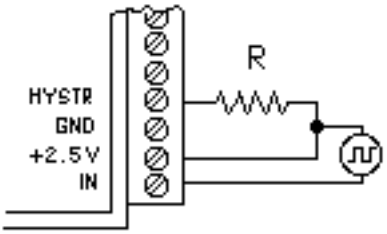
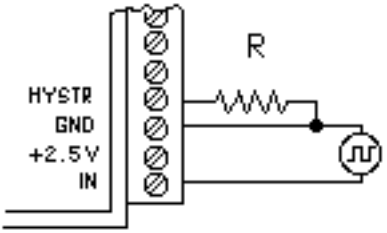


Figure B-3. Controlling Hysteresis For Bipolar Signals.

Figure B-4. Controlling Switching Level and Hysteresis.



R	$V_{switching} \pm V_{hysteresis}$
Open	2.5V ± 0.5V
ØΩ	1.7V ± 5mV

For $V_{hysteresis}$ >5mV and <0.5V:

$$R\ (\text{in K}\Omega) = \frac{34\ V_{hysteresis}}{0.5 - V_{hysteresis}}$$

$$V_{switching} = 2.5 - \frac{14}{17 + R}$$

The hysteresis control may also be connected to ground (GND), which produces another set of switching levels. This connection is shown in figure B-4. If the HYSTR terminal is shorted to GND the nominal switching point is 1.6V with ±5mV 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 1MΩ resistor connected from the +IN to +2.5V for biasing. A .01 uf cap may be used for frequencies down to 10HZ.

Appendix C
DIN-140 Data Sheet

SPECIFICATIONS: (Typical @ 25°C, V+ = +5V)

RTD Types: =.00385, .00388, .00392 100Ω @ 0°C
Resolution: 0.1°
Accuracy: ±0.3°C
Input connections: 2, 3, or 4 wire
Excitation current: .25 mA
Max. Lead resistance: 50Ω
Input protection to 120Vac
Automatic linearization and lead compensation
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 DIN-140 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.

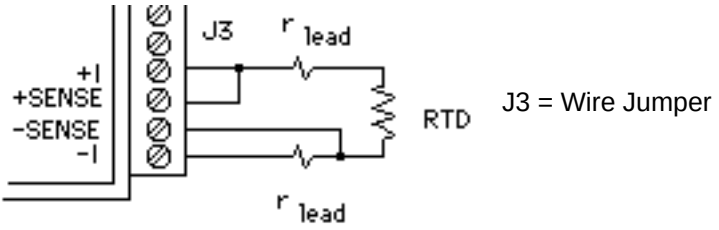


Figure C-1. 3-wire RTD Configuration

4-Wire: For 4-wire operation, connect the RTD as shown in the diagram. If the RTD has 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.

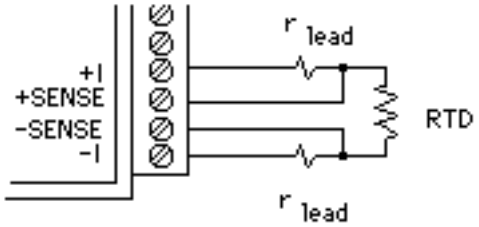


Figure C-2. 4-Wire RTD Configuration.

2-Wire: The 2-wire connection requires two jumpers on the connector (J1 & J2) as shown in the diagram. This connection provides no lead compensation. The RTD set-up bit can be either 0 or 1 for this connection.

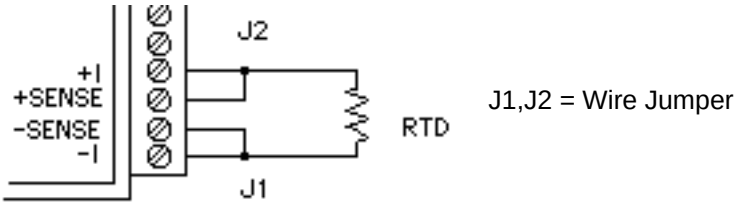


Figure C-3. 2-Wire RTD Configuration.

Start-Up: During normal operation, the RTD lead resistance is periodically scanned and filtered by the DIN-140 module. This may result in large initial errors if the RTD sensor is connected while the DIN-140 is powered up. To avoid this error, the sensor should be wired 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 50Ω, 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.

Appendix D
DIN-150 Data Sheet

The DIN-150 Bridge Sensor Interface Modules contain all of the signal conditioning functions necessary to interface Strain Gage and other resistive bridge devices to an 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 DIN-100 information described in the DIN-100 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 ±30mV and have a usable span of ±60mV. Modules configured for ±100mV have a usable span of ±120mV. The extra overhead is used to trim any bridge offsets.

Setup Data

The factory setup for all versions of DIN-150 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 DIN-150 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 DIN-150 modules have a wide input range to accommodate large sensor offsets without the need for external trims. Modules rated for ±30mV have an input range capability of ±60mV. Modules rated for ±100mV have an input range of ±120mV.

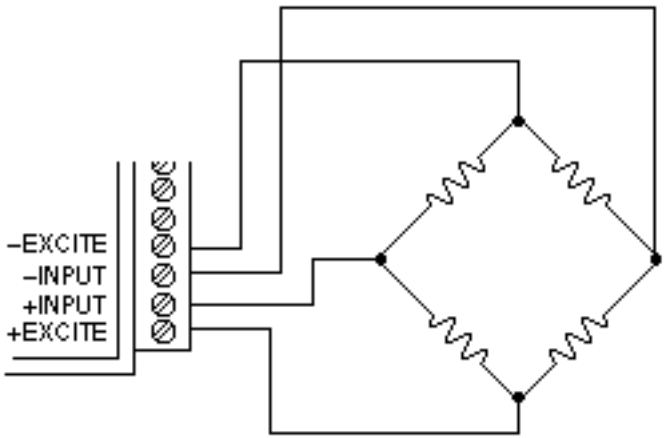


Figure D-1 Bridge Circuit Wiring

To perform an initial offset trim, attach the bridge unit to the module (as shown in Figure D-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 ±60mV or ±120mV. 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 Figure D-2). If the initial offset is acceptable, the offset may be trimmed with the Trim Zero (TZ) command.

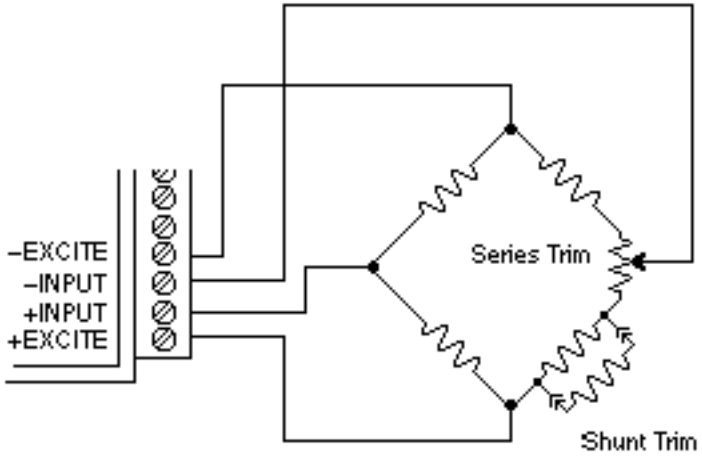


Figure D-2 Bridge Circuit Trim

Example 1:
A load cell to be used in a weighing application is mated to a DIN-152 module. The load cell is rated for 3mV/V, which results in a maximum ±30mV with 10V excitation. However, in this application, the load cell is used only in tension so its ideal output will be from 0 to +30mV.

The load cell is mounted in position with the weighing attachments. Clear any offset data that may be stored in the DIN-152 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 +2.34mV. The DIN-152 has a useful input range of ±60mV. After subtracting the offset the "input overhead" is -62.34mV and +57.66mV. The expected 0 to +30mV 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 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 DIN-152 module and the ideal output from the bridge is ±30mV 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 ±60mV useful range of the DIN-152 to obtain and "input overhead" value of -16.79mV to 103.21mV. In this case the -16.79mV overhead is not large enough to cover the -30mV that may be obtained from the bridge. The bridge must be trimmed externally to bring the offset to within ±30mV. 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 ±30mV offset necessary to provide enough head-room for the strain gage bridge.

Trim out the remaining offset with the Trim Zero (TZ) command:

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

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

The bridge is now trimmed to zero. Verify:

C**ommand:** **$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

DIN-150 modules may be ordered with either 5V or 10V excitation. Maximum excitation current available is 40mA. Modules with 10V 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 10V excitation if the bridge is completed with 350 Ω resistors. Modules with 5V excitation will source bridges of 85 Ω and up.

The actual excitation voltage may vary ±0.5V from the nominal values of +10V and +5V. 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 +10V or +5V.

Calibration

Since the DIN-150 modules use a ratiometric technique to compensate for variances in the excitation voltage, special consideration is required to properly calibrate the unit. Figure D-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.

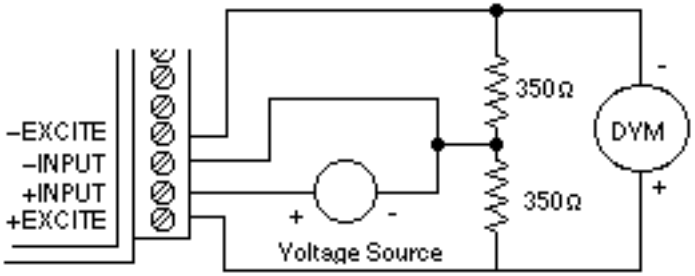


Figure D-3 DIN-150 Calibration

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 10V or 5V, to obtain a "compensation factor" = CF.

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

For ±30mV units the voltage is V = +50mV X CF

For ±100mV units the voltage is V = +100mV X CF

Set the voltage source to the calculated voltage V.

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

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

For ±100mV 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 DIN-151 module. This unit contains 5V excitation and a ±30mV 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 4.954V Calculate the "compensation factor":

CF=4.954/5=0.9908

Step 4: calculate the calibration voltage:

V=+50mV X 0.9908=+49.54mV.

Set the voltage standard to +49.54mV.

Step 5: perform the Trim Span command:

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

Command: **$1TS+00050.00**
Response: ***

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

Appendix E
DIN-100 Specifications

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

Analog
- Single channel analog input.
- Maximum CMV, input to output at 60Hz: 500V rms.
- Leakage current, input to output at 115Vrms, 60Hz: <2µA rms.
- 15 bit measurement resolution.
- 8 conversions per second.
- Autozero & autocalibration—no adjustment pots.

Digital
- 8-bit CMOS microcomputer.
- Digital scaling, linearization and calibration.
- Nonvolatile memory eliminates pots and switches.

Digital filtering
- Small and large signal with user selectable time constants from 0 to 16 seconds.

Communications
- Communications in ASCII via RS-232C, RS-485 ports.
- Selectable baud rates: 300, 600, 1200, 2400, 4800, 9600, 19200, 38400.
- NRZ asynchronous data format; 1 start bit, 7 data bits, 1 parity bit and 1 stop bit.
- Parity: odd, even, none.
- User selectable channel address.
- ASCII format command/response protocol.
- Up to 122 multidrop modules per host serial port.
- Communications distance up to 4,000 feet (RS-485)**.**
- Transient suppression on RS-485 communications lines.
- Communications error checking via checksum.
- Can be used with "dumb terminal".
- Scan up to 250 channels per second.
- All communications setups stored in EEPROM.

Power
Requirements: 5V±0.5Vdc, 0.75W max (DIN-150, 2.0W max.).
Internal switching regulator.
Protected against power supply reversals.

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

Warranty
12 months on workmanship and material.

DIN-110 Voltage Inputs
• Voltage ranges: ±10mV,±100mV, ±1V, ±5V, ±10V, ±100Vdc.
• Resolution: 0.01% of FS (4 digits).
• Accuracy: ±0.02% of FS max.
• Common mode rejection: 100dB at 50/60Hz.
• Zero drift: ±1 count max (autozero).
• Span tempco: ±50ppm/°C max.
• Input burnout protection to 250Vac .
• Input impedance: ≤ ±1V input = 100MΩ min.
≥ ±5V input = 1MΩ min.

DIN-120 Current Inputs
• Current ranges: 4-20mAdc.
• Resolution: ±0.04% of FS.
• Accuracy: ±0.04% of FS (4-20mA).
• Common mode rejection: 100dB at 50/60Hz.
• Zero drift: ±1 count max (autozero).
• Span tempco: ±50ppm/°C max.
• Voltage drop: ±0.1V max.

DIN-130 Thermocouple Inputs
• Thermocouple types: J, K, T, E, R, S, B, C (factory set).
• Ranges: J = -200°C to +760°C B = 0°C to +1820°C
K = -150°C to +1250°C S = 0°C to +1750°C
T = -200°C to +400°C R = 0°C to +1750°C
E = -100°C to +1000°C C = 0°C to +2315°C
• Resolution: ±1°.
• Overall Accuracy (error from all sources) from 0 to +40°C ambient:
±1.0 °C max (J, K, T, E).
±2.5 °C max (R, S, B, C)(300°C TO FS).
• Common mode rejection: 100dB at 50/60Hz.
• Input impedance: 100MΩ min.
• Lead resistance effect: <20μV per 350Ω.
• Open thermocouple indication.
• Input burnout protection to 250Vac.
• Overrange indication.
• Automatic cold junction compensation and linearization.

DIN-140 RTD Inputs

- RTD types: α = .00385, .00392, 100Ω at 0°C,
  .00388, 100Ω at 25°C.
- Ranges: .00385 = -200°C to +850°C.
  .00392 = -200°C to +600°C.
  .00388 = -100°C to +125°C.
- Resolution: 0.1°.
- Accuracy: ±0.3°C.
- Span tempco: ±50 ppm/°C max.
- Common mode rejection: 100dB at 50/60Hz.
- Input connections: 2, 3, or 4 wire.
- Excitation current: 0.25mA.
- Lead resistance effect: 3 wire - 2.5°C per Ω of imbalance.
  4 wire - negligible.
- Max lead resistance: 50Ω.
- Input burnout protection to 120Vac .
- Automatic linearization and lead compensation.

DIN-145 Thermistor Inputs

- Thermistor types: 2252Ω at 25°C, TD Series
- Ranges: 2252Ω = -0°C to +100°C.
  TD = -40°C to +150°C.
- Resolution: 2252Ω = 0.01°C or F.
  TD = 0.1°C or F
- Accuracy: 2252Ω = ±0.1°C.
  TD = ±0.2°C
- Common mode rejection: 100dB at 50/60Hz.
- Input protection to 30Vdc .

DIN-150 Bridge Inputs

- Voltage Ranges: ±30mV, ±100mV.
- Resolution: 10µV (mV spans).
  0.02% of FS (V span).
- Accuracy: ±0.05% of FS max.
- Common mode rejection: 100dB at 50/60Hz.
- Input burnout protection to 30Vdc .
- Offset Control: Full input range.
- Excitation Voltage: 5V, 10Vdc, 60mA max.
- Zero drift: ±1µV/°C max.
- Span tempco: ±50ppm/°C max.

DIN-160 Timer and Frequency Inputs

- Input impedance: 1M Ω.
- Switching level: selectable 0V, +2.5V.
- Hysteresis: Adjustable 10mV-1.0V.
- Input burnout protection: 250Vac.

Frequency Input
•Range: 1Hz to 20KHz.
•Resolution: 0.005% of reading + 0.01Hz.
•Accuracy: ±0.01% of reading ±0.01Hz.
•Tempco: ±20ppm/ °C.

DIN-170 Digital Inputs/Outputs
• 6 digital inputs or 6 output bits.
• Input voltage levels: 0-30V without damage.
• Input switching levels: High, 3.5V min., Low, 1.0V max.
• Outputs: Open collector to 30V, 100mA max. load.
• Vsat: 1.0V max @ 100mA.
• Inputs/Outputs are read/set in parallel.
• Isolated from power supply ground.

DIN-190 RS-232/485 Converter/Repeater
• Baud Rates: 300-115200 (Dip-switch selectable).
• Termination and biasing resistors included (selectable via internal jumpers).

Appendix F
Modbus Protocol

MODBUS PROTOCOL OVERVIEW

This document describes the Modbus RTU protocol option included in the DIN-100 series of data acquisition modules. This implementation of the Modbus protocol is a subset of the protocol as described in the Modicon Modbus Protocol Reference Guide PI-MBUS-300 Rev F. Only the RTU version of the protocol has been implemented.

Modbus RTU mode communicates in standard NRZ asynchronous format with one start bit, eight data bits, one parity bit, and one stop bit. Even and odd parity is supported. If no parity is specified, the number of stop bits can be user configured for either one or two stop bits.

Baud rates supported at this time are: 300, 600, 1200, 2400, 4800, 9600, 19,200 and 38,400.

Modbus uses the RS-485 electrical specification for multidrop communications. The RS-232 electrical specification is not supported.

Modbus is a registered trademark of AEG Modicon Inc.

The Modbus RTU protocol transmits data in 8-bit binary bytes (not ASCII). To illustrate the data in this document, the 8-bit byte is described as two hexadecimal nibbles. For example, the binary byte value "0101 1101" will be written as 5D.

A typical Modbus RTU command may look like this:

01 04 00 00 00 01 31 CA

Remember, this command string and others throughout this document are actually transmitted to a module as eight 8-bit binary characters.

The actual format of the data is dependent on the type of command desired. The example above is the Modbus 'Read Input Registers' function.

The '01' is the address of the slave device (DIN-100 module) being commanded. Each slave device must have its own unique address.

The '04' specifies the Modbus 'Read Input Registers' function. This is equivalent to the 'Read Data' command to obtain analog input data.

The next two characters ‘00 00’ specify the starting address of the registers to be read. The first Modicon input register 30001 is addressed as ‘00 00’. Register 30005 is addressed as ‘00 04’, etc.

The next two characters of this command specify the number of registers to be read, including the starting register. In this case the two binary characters ‘00 01’ indicates only one register is to be read.

The final two characters of the command string make up the Cyclical Redundancy Check (CRC), used to check for errors in the message.

There are no prompt or terminating characters in the messages. All messages must be transmitted as continuous strings. Messages are terminated by a ‘silent’ interval of at least 3.5 character times. A ‘silent’ interval of more than 1.5 character times marks the beginning of the next message. Therefore it is mandatory that the RS-485 bus must be biased in the MARK condition during the ‘silent’ interval. This is usually accomplished by pull-up and pull-down resistors on the communications line.

A typical response to this example command could be:

01 04 02 80 00 D8 F0

The ‘01’ and ‘04’ characters echo the slave address and the command function.

For this particular command function, the ‘02’ character indicates the number of data characters to follow, in this case, 2 characters.

The two character string ‘80 00’ is the value read from Modicon input register 30001. Register data is read back as 16 bits.

The remaining two characters, ‘D8 F0’ is the CRC for the response.

The A1000 and DIN-190 series of RS-232 to RS-485 protocol converters and repeaters will not operate with the 9-bit data characters used by the Modbus protocol.

Getting Started

The DIN-100 series modules are initialized at the factory to communicate using the DIN-100 ASCII protocol. This allows for all setup and configurations to be easily performed using the DIN-100 setup software or a dumb terminal. After the setup process has been completed the DIN-100 can be placed in Modbus RTU protocol mode using the “MBR” command. Disable the Modbus RTU mode using the Modbus Disable (“MBD”) command.

Quick start steps:

1 Connect a power supply to the DIN-100 between +Vs terminal and GND terminal. The regulated supply voltage must be +5.0Vdc.

2. Properly connect the DIN-100 series to a computer using the “quick hook-up” diagrams in chapter #1 of this manual using either an RS-232 or RS-485 serial port.

3. Locate the Windows Utility software CD-ROM and run the setup.exe file to install the software. A DGH Data Acquisition menu selection will be added to the Windows “Start, Programs” menu. The Utility Software will be listed under that selection.

4. Configure the host computer serial ports by selecting main menu selection “Edit” and “Serial Ports”. Select the correct COMx: port, baud rate and Parity type. Note: If the “Default*” pin on DIN-100 is connected to GND then select 300 baud as host computer baud rate and select no parity.

5. Select main menu “Setup” and enter the DIN-100 device address and model number. For example, select “111 DIN +/-100mV/RS-485 Out”.

6. At the next configuration screen make the necessary alterations to Baud Rate, Parity type and other required parameters. Drop down the “Communications Protocol” list box and select “Modbus RTU Protocol”. Specifiy the correct Modbus slave address and press the <APPLY> to transmit the new setup values. Once the values have been transmitted press the <ESC> key back to the program main menu.

7. Remove the connection between “Default*” and GND, which performs internal reset, to enable Modbus RTU mode. If there was no connection between “Default*” and GND then cycle the power on device to force a reset and enable Modbus Mode.

The device is now configured for Modbus RTU mode and can be connected

to a RS-485 based Modbus master system.

MODBUS Function Codes

Modbus protocol compatible devices communicate using a master-slave technique similar to that used in ASCII protocol. In a master-slave communications system only one device (the master) can initiate a communications sequence. All others devices (the slaves) respond when requested by the master. Typical master devices can be personal computers or PLCs. Typical slave devices are DIN-100 modules.

The master can address any slave device. Slave devices return a message to any command that was addressed specifically to them. The returned messages are considered response messages.

The Modbus protocol format used by a master consists of a device address, a command function code which defines the operation to be performed, data required with the command, and an error checking value. The slave response message contains any required data and an error checking value. If an error occurs, an exception code will be generated.

The supported master function codes are discussed below.

01 - Read Coil Status (Digital Inputs)
04 - Read Input Register (Analog Inputs)
05 - Force Single Coil (Digital Output)
06 - Preset Single Register (Return to ASCII protocol)
15 - Force Multiple Coils (Digital Outputs)

Function (01) Read Coil Status (Digital Inputs)

Modbus function (01) Read Coil Status will read the status of both the digital inputs and digital outputs. Digital outputs are read as the state of the data on the microprocessor output port before being buffered by the open-collector transistor. If the coil status of a digital output returns as '1', this means that this particular bit (coil) is turned "on" or sinking current on the corresponding module digital output pin. Depending on the module type, some of the digital outputs may not be implemented.

Modbus relay input coils are considered digital inputs on the DIN-100 series modules. Modbus relay output coils are considered digital outputs on the DIN-100 series modules. This function can be used to read status of the digital inputs or the combined status of both the digital inputs and digital outputs.

DIN-100 digital output bits B00 to B05 correspond to Modbus coils 00 00 to 00 05.

DIN-100 digital input bits B00 to B05 correspond to Modbus coils 00 08 to 00 0D.

The DIN-100 series digital inputs and outputs are register mapped as two 8-bit bytes (16-bits), one byte for inputs and one byte for outputs. The least-significant byte represent the status of up to 8 digital outputs. The most-significant byte represents the status of up to 8 digital input bits. The register contents can be interrogated as 8-bits of digital input data or together as 16-bits of digital inputs and outputs data.

Exception errors will be generated by the module if attempting to read or write to more than 16 bits.

The following example can be used to read only the digital input status:

Command: 01 01 00 08 00 08 BC 0E
Response: 01 01 01 FF 11 C8

In the command string:
01 is the slave address
01 is the Read Coil Status command
00 08 is the starting coil number
00 08 is the number of bits to read
BC 0E is the CRC to this message

In the response string:
01 is the slave address
01 is the Read Coil Status command
01 is the number of data bytes returned
FF is the Digital Inputs status data
11 C8 is the CRC for this message

The following example can be used to read the status of both the digital inputs and outputs:

Command: 01 01 00 00 00 10 3D C6
Response: 01 01 02 09 FF FF EC

In the command string:

01 is the slave address
01 is the Read Coil Status command
00 00 is the starting coil address
00 10 is the number digital bits to read
3D C6 is the CRC to this message

In the response string:
01 is the slave address
01 is the Read Coil Status command
02 is the number of data bytes returned
09 is the Digital Output status
FF is the Digital Inputs status
FF EC is the CRC for this message

Function (04) - Read Input Register (Analog Inputs)
Read Input Register function (04) is the primary command to acquire analog input data. This command function supports reading of up to 16 input registers starting from Modbus slave register 30001. The registers are addressed starting from zero meaning registers 1-16 are addressed as 0-15.

The response data for each channel is returned as two bytes that represent a 16-bit binary value. The 16-bit value is scaled as a percentage of the full scale input range. The first byte contains the high order bits and the second contains the low order bits. The binary analog values for each channel can range from 0000-FFFF (hexadecimal).

Only the register values for channel one are valid as each module contains a single analog input. The remaining data values for channels 2-16 will always return as 0000 (hexadecimal).

A typical command and response to read the analog input value from Modbus device address 01 is:

Command: 01 04 00 00 00 01 31 CA
Response: 01 04 02 14 57 F7 CE

In the command string:
01 is the slave address
04 is the Read Input Registers command
00 00 is the starting register to be read (Modbus address 30001)
00 01 specifies the number of registers to be read, in this case, one register.

31 CA is the CRC for this message

In the response string:
01 is the slave address
04 is the command
02 indicated the number of data bytes in the message, in this case, two bytes
14 57 is the analog data
F7 CE is the CRC for this message

This sample command reads two registers:

Command: 01 04 00 00 00 02 71 CB
Response: 01 04 04 14 58 00 00 7F A7

The analog data from Modbus register 30001 is 14 58.
The data from Modbus register 30002 is set to 00 00.

The analog data is scaled so that 00 01 represents the Negative Full Scale value programmed into the module. FF FE represents the Positive Full Scale value programmed into the module.

For example, for a ±10 volt input module:
00 01 corresponds to -10 volts
80 00 corresponds to 0 volts
FF FE corresponds to +10 volts

A negative overload where the analog input exceeds minus full scale value is represented by 00 00 (hexadecimal).

A positive overload where the analog input exceeds the positive full scale value is represented by FF FF (hexadecimal).

Function (05) - Force Single Coil (Digital Output)
The Force Single Coil function (05) is used to set or clear a single Modbus output relay coil. Each output relay coil is considered a digital output on the DIN-100 series modules. Modbus Coil #1 equals DIN-100 series digital output bit B00 and Coil #6 equals digital output bit B05.

The following example can be used to turn on digital output bit B00:

Command: 01 05 00 00 FF 00 8C 3A
Response: 01 05 00 00 FF 00 8C 3A

In the command string:
01 is the slave address
05 is the Force Single Coil command
00 00 is the address of the digital output bit, 00 07 would equal B07
FF 00 indicates that the desired bit will be set or turned on
8C 3A is the CRC for this message

The valid address range of digital output bits is 00 00 to 00 07. Any other address will produce an exception (error) response.

To clear or turn off digital output bit B03, replace the FF 00 string with 00 00. For example:

Command: 01 05 00 03 00 00 3D CA
Response: 01 05 00 03 00 00 3D CA

Command values other than FF 00 or 00 00 will result in an exception (error) response.

Function (06) - Preset Single Register (Return to DIN-100 ASCII Protocol)
The Preset Single Register function (06) can be used to temporarily suspend the Modbus RTU protocol and force the module into DIN-100 ASCII protocol. Write a value of 0000 to Modbus register 40001 to temporarily suspend Modbus RTU mode. The module will then communicate using the DIN-100 ASCII protocol only.

The DIN-100 ASCII protocol can be used to alter or check setup information and/or for troubleshooting purposes. The module will continue to communicate using the ASCII protocol until either a Remote Reset (RR) command (RR) is received or the power is cycled. At which time, the module will return to the Modbus RTU protocol mode.

Refer to the DIN-100 ASCII Modbus Disable command (MBD) for more information on disabling the Modbus protocol.

Command: 01 06 00 00 00 00 89 CA
Response: 01 06 00 00 00 00 89 CA
Function (15) - Force Multiple Coils (Digital Outputs)

The Force Multiple Coils function (15) is used to force multiple Modbus output relay coils to a desired ON or OFF state. This function is similar in operation to the DIN-100 ASCII digital output command (DO) in that it updates the status of all available output coils at once. The state of each output coil is set ON or OFF according to the digital data value received with the function.

The digital output bits are referred to as “output relay coils” in the Modbus protocol. The DIN-100 series digital output bit B00 equals Modbus output relay Coil#1 and digital output bit B05 equals output relay Coil#6. DIN-100 series modules with less than eight digital outputs also equate output bit B00 with output relay Coil#1 and count up by one for each additional output bit.

The following example can be used to turn on two digital output bits on a DIN-113 module.

Command: 01 0F 00 00 00 02 00 03 9F 06
Response: 01 0F 00 00 00 02 D4 0A

In the command string:
01 is the slave address
05 is the Force Single Coil command
00 00 is the starting address of the digital output bits to be changed, 00 07 would equal B07
00 02 specifies the number of output relay coils to be changed
00 03 specifies the digital output data value in HI byte LO byte format. Note: The HI byte will always be zero as DIN-100 series modules contain up to eight digital outputs
9F 06 is the CRC for this message

The valid address range of digital output bits is 00 00 to 00 07. Any other address will produce an exception (error) response.

To clear or turn off the digital output bits, replace the 00 03 string with 00 00. For example:

Command: 01 0F 00 00 00 02 00 00 DF 07
Response: 01 0F 00 00 00 02 D4 0A

Modbus RTU Enable (MBR)

To place any DIN-100 module in Modbus protocol mode use the Modbus RTU (MBR) command. The MBR command must be used to specify the Modbus device address and enable the Modbus protocol mode. The device address consists of a two character hexadecimal value and is stored in EEPROM. The two byte address specified is translated to a one byte, 8 bit address required by the Modbus protocol. The example below can be used to specify a Modbus device address of “01”.

Command: **$1MBR01**
Response: *****

Command: **#1MBR01**
Response: ***1MBR019D**

After the Modbus address is specified, a reset is necessary to activate the Modbus protocol mode. The reset may be accomplished in one of three ways:

1) Removing power for about 10 seconds to perform a power-up reset.

2) Momentarily grounding the Default* pin.

3) Issue a Write Enable (WE) command followed by a Remote Reset (RR) command.

After a reset is performed, the module is in Modbus protocol mode.

Modbus Disable (MBD)
The Modbus Disable (MBD) command is used to disable the Modbus protocol. Any DIN-100 series module in Modbus mode can be returned to DIN-100 ASCII protocol mode by connecting a jumper wire between module pins GND and Default* pin. This places the module in Default Mode, where the module will only communicate at 300 baud, no parity, DIN-100 ASCII protocol, and answer to any address. While in Default mode, transmit an MBD command to internally disable the Modbus protocol.

Following the MBD command a device reset must occur. The reset is necessary to activate the DIN-100 ASCII protocol. A reset can occur by removing the Default* jumper, performing a power-up reset or by transmitting a Write Enable (WE) and Remote Reset (RR) command sequence.

After a reset is performed, the module is in DIN-100 ASCII protocol mode.

Command: **$1MBD**
Response: *****

Command: **#1MBD**
Response: ***1MBD2E**

Modbus Exception Responses
The following standard Modbus exception codes (error messages) are supported:

01 Illegal Function
This exception code is generated when the function code is not recognized by the module.

02 Illegal Data Address
This code is generated when the specified data address in the command is not supported by the module.

03 Illegal Data Value
This exception code is returned if the command data is out of range for the function.

06 Slave Device Busy
After the module is reset by power-up, a 'RR' command, or return from Default Mode, the module performs an initial self-calibration for a period of about 3 seconds. During this time any command sent to the module will result in a 'busy' exception response.