# M-7002
# User Manual

ICP DAS CO., LTD.
Revision: 1.5
2024/02/17

# Table of Contents

# 1. Introduction

## Features

- Voltage or Current Input
- +/-240 Vrms Overvoltage Protection
- High Resolution: 16-bit
- 2500 $V_{DC}$ Intra-module Isolation
- Sink and Source Type Digital Inputs
- Photocoupler Isolation
- Supports Relay Outputs
- DIN-Rail Mountable
- Dual Watchdog
- Wide Operating Temperature Range: -25 ~ +75℃

## Applications

- Building Automation
- Factory Automation
- Machine Automation
- Remote Maintenance
- Remote Diagnosis
- Testing Equipment

## More Information

Refer to Chapter 1 of the “I-7000 Bus Converter User Manual” for more information regarding the following:

1.1. I-7000 Overview

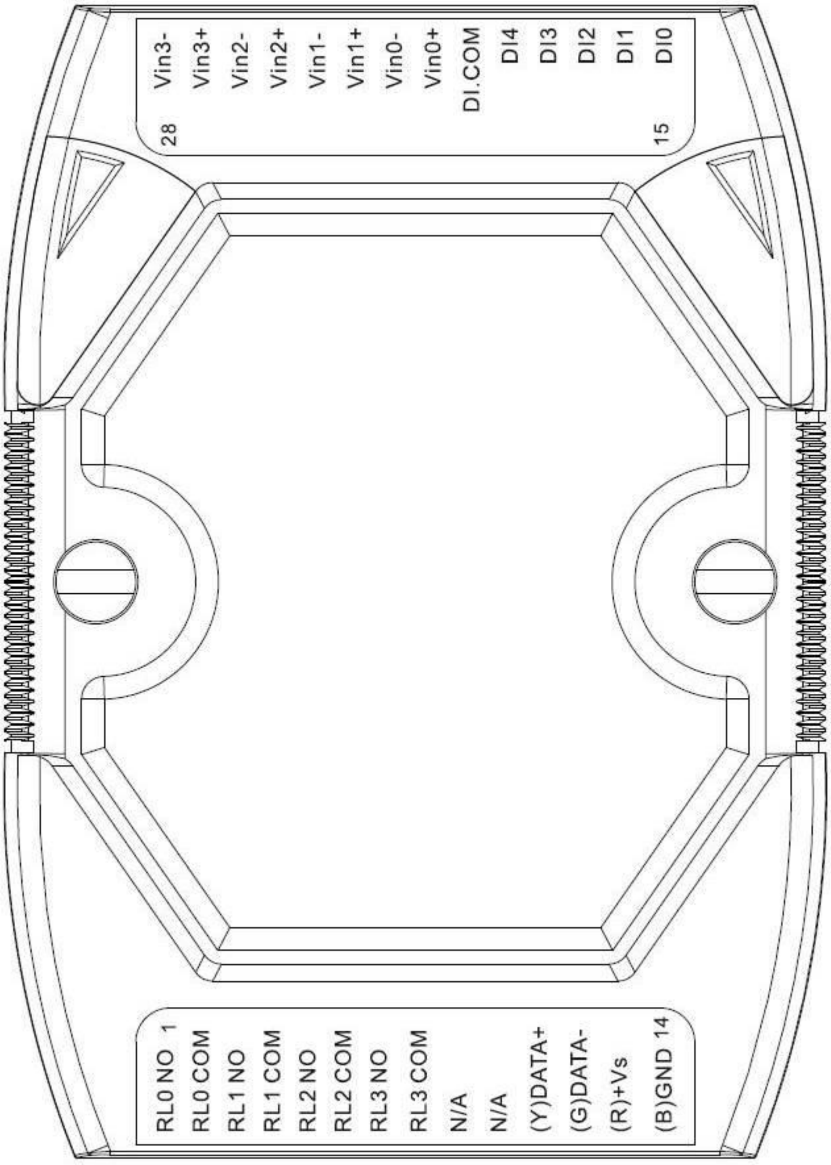
1.2. I-7000 Related Documentation

1.3. I-7000 Common Features

1.4. I-7000 System Network Configuration

1.5. I-7000 Dimensions

## 1.1. Pin Assignments



## 1.2. Specifications

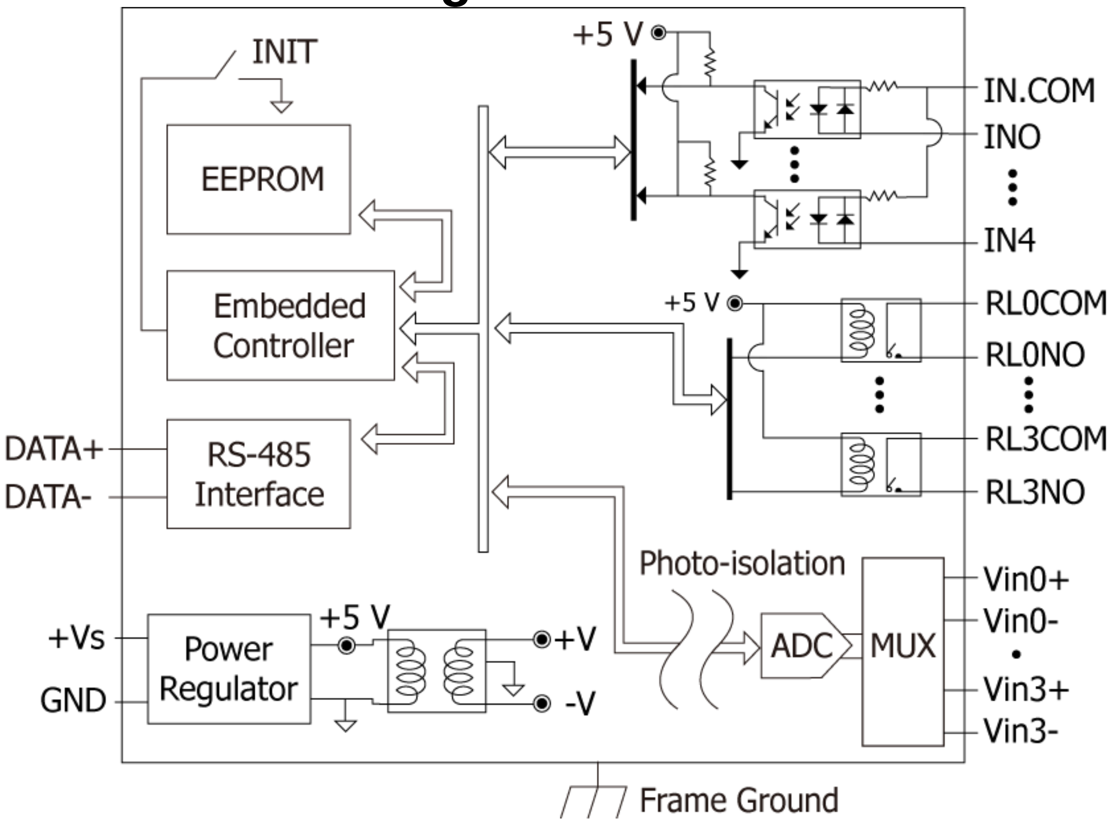
### Syste

| Commur | |
|---|---|
| Interface | |
| Format | |
| Baud Rat | |
| Protocol | |
| Dual Wat | |
| **LED Ind** | |
| System L | ator |
| I/O LED | |
| 7-Segmer | |
| **Isolation** | |
| Intra-Mo<br>to-Logic | |
| **EMS Pr** | |
| ESD (IEC | |
| EFT (IEC | |
| Surge (IE | |
| **Power** | |
| Reverse F | |
| Input Vol | |
| Consump | |
| **Mechani** | |
| Dimensic | |
| Installatic | |
| **Environ** | |
| Operating | |
| Storage T | |
| Humidity | |

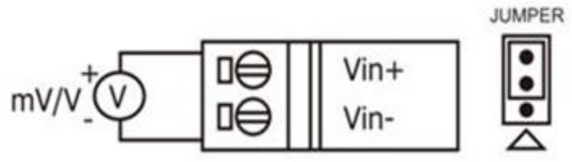
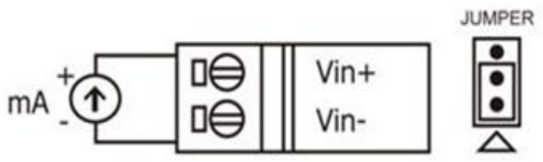
## I/O Specifications

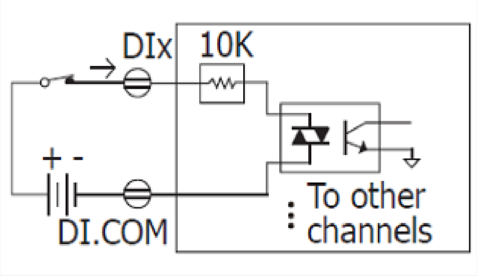
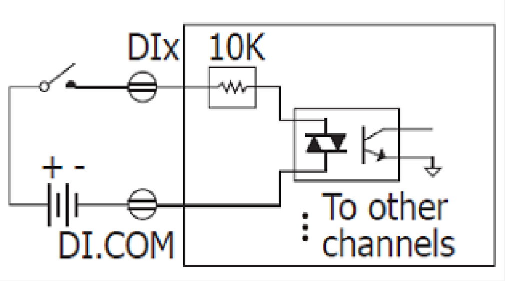
| **Analog Input** | | |
|---|---|---|
| Channels | | 4 |
| Wiring | | Differential |
| Input Range | | +/-150 mV, +/-500 mV, +/-1 V , +/-5 V, +/-10 V<br>+/-20 mA , 0~20 mA, 4~20 mA (jumper selectable) |
| Resolution | | 12/16-bit |
| Accuracy | Normal Mode | 0.1% |
| | Fast Mode | 0.5% |
| Sampling Rate | Normal Mode | 10 Hz |
| | Fast Mode | 60 Hz |
| Input Impedance | Voltage | 2 MΩ |
| | Current | 139 Ω |
| Common Voltage Protection | | +/-200 $V_{DC}$ |
| Individual Channel Configuration | | Yes |
| Overcurrent Protection | | 50 mA max. at 110 $V_{DC}$/$V_{AC}$ max. |
| Overvoltage Protection | | 240 Vrms |
| **Digital Input/Counter** | | |
| Channels | | 5 |
| Contact | | Wet |
| Sink/Source (NPN/PNP) | | Sink/Source |
| On Voltage Level | | 10 ~ 50 $V_{DC}$ |
| Off Voltage Level | | +4 $V_{DC}$ Max. |
| Counter (50 Hz, 16-bit) | | Yes |
| Input Impedance | | 10 kΩ |
| Overvoltage Protection | | +/-70 $V_{DC}$ |
| Isolation Voltage | | 3750 $V_{DC}$ |
| **Relay Output** | | |
| Channels | | 4 |
| Type | | Power Relay (Form A) |
| Contact Rating | | 5 A @ 250 $V_{AC}$<br>5 A @ 30 $V_{DC}$ |
| Surge Strength | | 3000 $V_{DC}$ |
| Operation Time | | 6 ms |
| Release Time | | 3 ms |
| Mechanical Endurance | | 2 × $10^7$ ops. |
| Electrical Endurance | | $10^5$ ops. |
| Power-on Values | | Yes |
| Safe Values | | Yes |

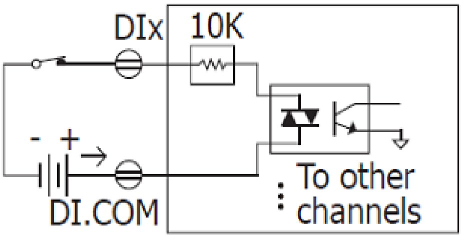
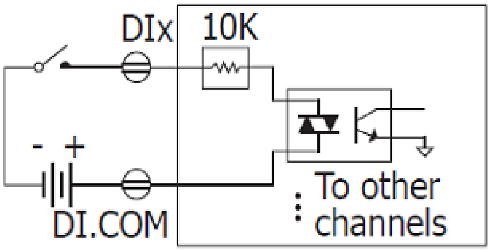
## 1.3. Block Diagram

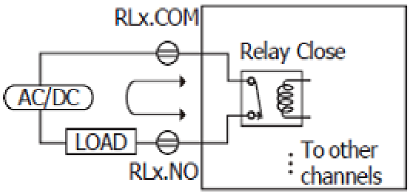
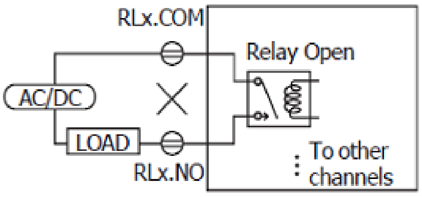


## 1.4. Application Wiring

| Voltage Input Wire Connection | Current Input Wire Connection |
|---|---|
|  |  |

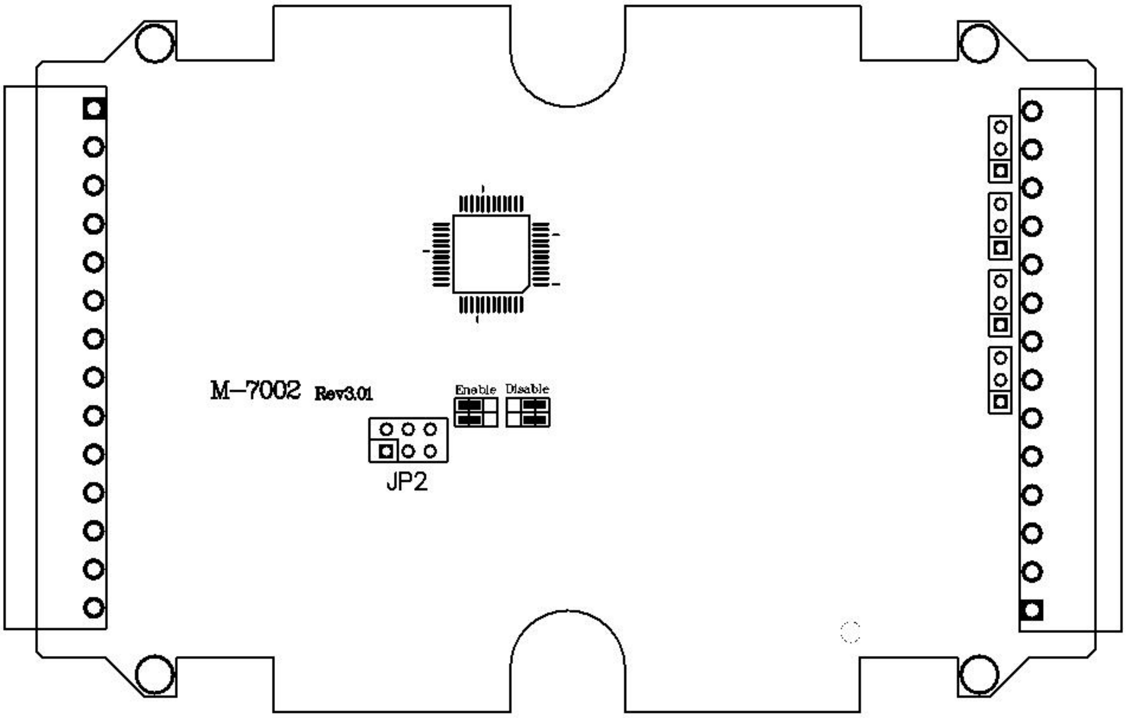
| Digital Input/Counter | Read back as 1 | Read back as 0 |
|---|---|---|
| Sink | +10 ~ +50 $V_{DC}$ | OPEN or < 4 V |
| |  |  |
| Source | +10 ~ +50 $V_{DC}$ | OPEN or < 4 V |

| | | |
|---|---|---|
| |  |  |

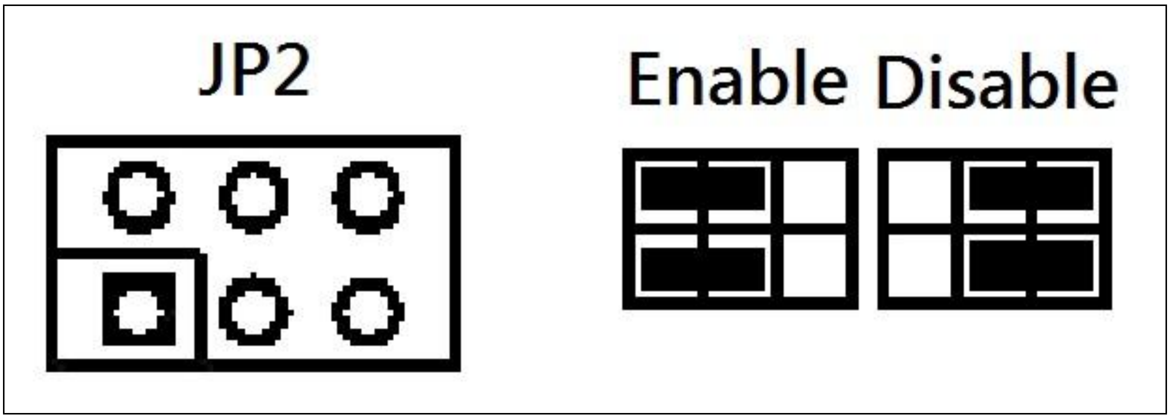
| Power Relay | ON State<br>Read back as 1 | OFF State<br>Read back as 0 |
|---|---|---|
| Relay Output | Relay On | Relay Off |
| |  RLx.COM<br>Relay Close<br>AC/DC<br>LOAD<br>RLx.NO<br>To other<br>channels |  RLx.COM<br>Relay Open<br>AC/DC<br>LOAD<br>RLx.NO<br>To other<br>channels |

## 1.5. Jumper Settings

For the M-7002 PCB version 3.01 and later, the JP2 jumper can be used to enable providing the RS-485 bias. The position of the JP2 jumper is shown in the figure below.



The settings for the JP2 jumper is as follows.



## 1.6. Default Settings

The default settings for the M-7002 are:

- Module address: 01
- Analog input type: Type 08, -10V to 10V
- Protocol: Modbus protocol
- Baud Rate: 9600 bps
- Checksum disabled
- Engineering units format
- Filter set at 60Hz rejection

## 1.7. Calibration

**Warning:** *It is not recommended that calibration be performed until the process is fully understood.*

### 1.7.1 Analog Input

The calibration procedure is as follows:

1. Warm up the module for 30 minutes.
2. Set the type code to the type you want to calibrate. Refer to Section 2.12 for details.
3. Enable calibration. Refer to Section 2.31 for details.
4. Apply the zero calibration voltage/current.
5. Send the “zero calibration” command. Refer to Section 2.6 for details.
6. Apply the span calibration voltage/current.
7. Send the “span calibration” command. Refer to Section 2.5 for details.
8. Repeat steps 3 to 7 three times.

**Notes:**

1. Connect the calibration voltage/current to channel 0.
2. When calibrating type 0D, the jumper for channel 0 should be set to the “current input” position.
3. Calibration voltages and currents are shown below.

Calibration voltage/current:

| Type Code | 08 | 09 | 0A | 0B | 0C | 0D |
|---|---|---|---|---|---|---|
| Zero Input | 0 V | 0 V | 0 V | 0 mV | 0 mV | 0 mA |
| Span Input | +10 V | +5 V | +1 V | +500 mV | +150 mV | +20 mA |

# 1.8. Configuration Tables

## Baud Rate Settings (CC)

Bits 5:0

| Code | 03 | 04 | 05 | 06 | 07 | 08 | 09 | 0A |
|---|---|---|---|---|---|---|---|---|
| Baud Rate | 1200 | 2400 | 4800 | 9600 | 19200 | 38400 | 57600 | 115200 |

Bits 7:6

00: no parity, 1 stop bit  
01: no parity, 2 stop bits  
10: even parity, 1 stop bit  
11: odd parity, 1 stop bit

## Analog Input Type Settings (TT)

| Type Code | Analog Input Type | Range |
|---|---|---|
| 07 | +4 ~ +20 mA | +4 mA ~ +20 mA |
| 08 | +/-10 V | -10 V ~ +10 V |
| 09 | +/-5 V | -5 V ~ +5 V |
| 0A | +/-1 V | -1 V ~ +1 V |
| 0B | +/-500 mV | -500mV ~ +500 mV |
| 0C | +/-150 mV | -150 mV ~ +150 mV |
| 0D | +/-20 mA | -20 mA ~ +20 mA |
| 1A | 0 ~ +20 mA | 0 ~ +20 mA |

**Note:**

When types 07, 0D or 1A are selected, the jumper for the corresponding channel should be set to the “current input” position.

## Data Format Settings (FF)

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| FS | CS | MS | Reserved | | | DF | |

| Key | Description |
|---|---|
| DF | Data format<br>00: Engineering units<br>01: % of FSR (full scale range)<br>10: 2’s complement hexadecimal |
| MS | Mode settings<br>0: Normal mode (16 bits)<br>1: Fast mode (12 bits) |
| CS | Checksum settings<br>0: Disabled<br>1: Enabled |
| FS | Filter settings<br>0: 60Hz rejection<br>1: 50Hz rejection |

**Note**: Reserved bits should be zero.

# Analog Input Type and Data Format Table

| Type code | Input Type | Data Format | +F.S | -F.S. |
|---|---|---|---|---|
| 07 | +4 to +20 mA | Engineering units | +20.000 | +04.000 |
| | | % of FSR | +100.00 | +000.00 |
| | | 2’s comp HEX | FFFF | 0000 |
| 08 | -10 to +10 V | Engineering units | +10.000 | -10.000 |
| | | % of FSR | +100.00 | -100.00 |
| | | 2’s comp HEX | 7FFF | 8000 |
| 09 | -5 to +5 V | Engineering units | +5.0000 | -5.0000 |
| | | % of FSR | +100.00 | -100.00 |
| | | 2’s comp HEX | 7FFF | 8000 |
| 0A | -1 to +1 V | Engineering units | +1.0000 | -1.0000 |
| | | % of FSR | +100.00 | -100.00 |
| | | 2’s comp HEX | 7FFF | 8000 |
| 0B | -500 to +500 mV | Engineering units | +500.00 | -500.00 |
| | | % of FSR | +100.00 | -100.00 |
| | | 2’s comp HEX | 7FFF | 8000 |
| 0C | -150 to +150 mV | Engineering units | +150.00 | -150.00 |
| | | % of FSR | +100.00 | -100.00 |
| | | 2’s comp HEX | 7FFF | 8000 |
| 0D | -20 to +20 mA | Engineering units | +20.000 | -20.000 |
| | | % of FSR | +100.00 | -100.00 |
| | | 2’s comp HEX | 7FFF | 8000 |
| 1A | 0 to +20 mA | Engineering units | +20.000 | +00.000 |
| | | % of FSR | +100.00 | +000.00 |
| | | 2’s comp HEX | FFFF | 0000 |

## 1.9. M-7000 Notes

The main difference between the I-7000 and M-7000 series is that the M-7000 series has additional support for the Modbus RTU communication protocol, which is the default protocol of the M-7000 series. The communication Baud Rates for the Modbus RTU protocol can be in the range of 1200 bps to 115200 bps, and the parity, data and stop bits are fixed as no parity, 8 data bits and 1 stop bit.

Modbus functions supported by the module are described in Chapter 3.

### 1.9.1 Protocol Switching

To switch to the DCON protocol:

1. Uses sub-function 06h of the function 46h and set byte 8 to a value of 1. See Section 3.4.4 for details.
2. After a power-on reset, the communication protocol will be changed to DCON.

To switch to the Modbus RTU protocol:

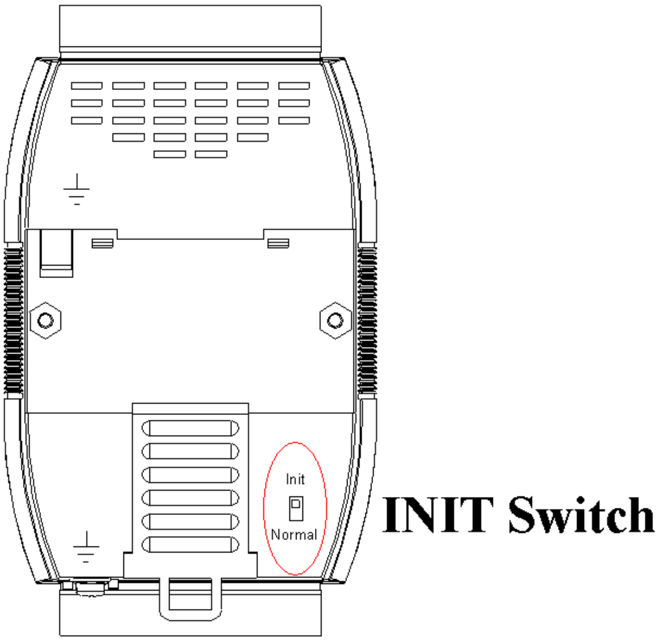
1. Sends the $AAPN command and set N to a value of 1. Note that the slide switch on the rear side of the module should be set to INIT position, see the figure on the next page. See Section 2.19 for details.
2. After a power-on reset, the communication protocol will be changed to Modbus RTU protocol.

### 1.9.2 INIT Mode

When the module is powered on, with the rear slide switch set to INIT position as shown in the figure below, the module is in INIT mode (Section 5.1), and the communication settings are as follows:

1. Address: 00
2. Baud Rate: 9600 bps
3. No checksum
4. Protocol: DCON

If communication with the module is not possible, set the module to INIT mode and use the above settings to communicate with the module. To read the current settings, send the commands $AA2 (Section 2.7), and $AAPN (Section 2.19). The new communication settings will be effective after the next power-on reset.



# 2. DCON Protocol

All communication with the module consists of commands generated by the host and responses transmitted by the module. Each module has a unique ID number that is used for addressing purposes and is stored in non-volatile memory. The ID is 01 by default and can be changed by transmitting the appropriate using a user command. All commands to the modules contain the ID address, meaning that only the addressed module will respond. The only exception to this is command #** (Section 2.2) and command ~** (Section 2.22), which is sent to all modules, but, in these cases, the modules do not reply to the command.

### Command Format:

| Leading Character | Module Address | Command | [CHKSUM] | CR |
|---|---|---|---|---|

### Response Format:

| Leading Character | Module Address | Data | [CHKSUM] | CR |
|---|---|---|---|---|

**CHKSUM** A 2-character checksum that is present when the checksum setting is enabled. See Sections 2.1 and 5.1 for details.

**CR** End of command character, carriage return (0x0D)

## Checksum Calculation:

1. Calculate the ASCII code sum of all the characters in the command/response string, except for the carriage return character (CR).
2. The checksum is equal to the sum masked by 0FFh.

## Example:

Command string: $012(CR)

1. The sum of the string = “$”+”0”+”1”+”2” = 24h+30h+31h+32h = B7h
2. Therefore the checksum is B7h, and so CHKSUM = “B7”
3. The command string with the checksum = $012B7(CR)

Response string: !01200600(CR)

1. The sum of the string = “!”+”0”+”1”+”2”+”0”+”0”+”6”+”0”+”0” = 21h+30h+31h+32h+30h+30h+36h+30h+30h = 1AAh
2. Therefore the checksum is AAh, and so CHKSUM = “AA”
3. The response string with the checksum = !01200600AA(CR)

## Note:

All characters should be in upper case.

| General Command Sets | | | |
|---|---|---|---|
| Command | Response | Description | Section |
| %AANNTTCCFF | !AA | Sets the configuration of the module | 2.1 |
| $AA2 | !AANNTTCCFF | Reads the configuration of the module | 2.7 |
| $AA5 | !AAS | Reads the reset status of the module | 2.9 |
| $AAC | !AA | Clears the DI/DO latches of the module | 2.14 |
| $AAF | !AA(Data) | Reads the firmware version information | 2.15 |
| $AAI | !AAS | Reads the status of the INIT switch | 2.16 |
| $AALS | !(Data) | Reads the status of the DI and DO latches | 2.17 |
| $AAM | !AA(Data) | Reads the name of the module | 2.18 |
| $AAP | !AASC | Reads the communication protocol | 2.19 |
| $AAPN | !AA | Sets the communication protocol | 2.20 |
| ~AAD | !AAVV | Reads the miscellaneous settings | 2.29 |
| ~AADVV | !AA | Sets the miscellaneous settings | 2.30 |
| ~AADT | !AAE | Reads the counting on overflow setting | 2.58 |
| ~AADTE | !AA | Sets the counting on overflow setting | 2.59 |
| ~AAI | !AA | Sets the software INIT modification to enabled | 2.32 |
| ~AAO(Data) | !AA | Sets the name of the module | 2.33 |
| ~AARD | !AATT | Reads the response delay time | 2.34 |
| ~AARDTT | !AA | Sets the response delay time | 2.35 |
| ~AATnn | !AA | Sets the Software INIT timeout value | 2.36 |
| @AACECi | !AA | Clears the DI counter | 2.37 |
| @AADI | !AAOOII | Reads the status of the DO and DI channels | 2.45 |
| @AADODD | !AA | Sets the status of the DO channels | 2.46 |
| @AARECi | !AA | Reads the counter for a specific DI channel | 2.51 |

| Analog Input Command Sets | | | |
|---|---|---|---|
| Command | Response | Description | Section |
| #** | No Response | Synchronized sampling | 2.2 |
| #AA | >(Data) | Reads the analog inputs of all channels | 2.3 |
| #AAN | >(Data) | Reads the analog input of a specific channel | 2.4 |
| $AA0 | !AA | Performs an analog input span calibration | 2.5 |
| $AA1 | !AA | Performs an analog input zero calibration | 2.6 |
| $AA4 | >AAS(Data) | Reads the synchronized data | 2.8 |
| $AA5VV | !AA | Enables/Disables the analog input channels | 2.10 |
| $AA6 | !AAVV | Reads the enabled/disabled status of all analog input channels | 2.11 |

| | | | |
|---|---|---|---|
| $AA7CiRrr | !AA | Sets the type code for a specific channel | 2.12 |
| $AA8Ci | !AACiRrr | Reads the type code for a specific channel | 2.13 |
| $AAS1 | !AA | Reloads the default calibration parameters | 2.21 |
| ~AAEV | !AA | Enables/Disables the analog input calibration | 2.31 |
| @AACH | !AA | Clears the high latch value for all channels | 2.38 |
| @AACHi | !AA | Clears the high latch value for a specific channel | 2.39 |
| @AACHCi | !AA | Clears the status of the high alarm | 2.40 |
| @AACL | !AA | Clears the low latch value for all channels | 2.41 |
| @AACLi | !AA | Clears the low latch value for a specific channel | 2.42 |
| @AACLCi | !AA | Clears the status of the low alarm | 2.43 |
| @AADA | !AA | Disables the analog input alarm | 2.44 |
| @AAEAt | !AA | Enables the momentary/latch function | 2.47 |
| @AAHI(Data)Ci | !AA | Sets the analog input high alarm | 2.48 |
| @AALO(Data)Ci | !AA | Sets the analog input low alarm | 2.49 |
| @AARAO | !AAHHLL | Reads the activated alarms associated with the DO channels of a module | 2.50 |
| @AARH | !AA(Data) | Reads the high latch value for all channels | 2.52 |
| @AARHi | !AA(Data) | Reads the high latch value for a specific channel | 2.53 |
| @AARHCi | !AA(Data) | Reads the status of the analog input high alarm | 2.54 |
| @AARL | !AA(Data) | Reads the low latch value for all channels | 2.55 |
| @AARLi | !AA(Data) | Reads the low latch value for a specific channel | 2.56 |
| @AARLCi | !AA(Data) | Reads the status of the analog input low alarm | 2.57 |

| Host Watchdog Command Sets | | | |
|---|---|---|---|
| **Command** | **Response** | **Description** | **Section** |
| ~** | No Response | Informs all modules that host is OK | 2.22 |
| ~AA0 | !AASS | Reads the status of the Host Watchdog | 2.23 |
| ~AA1 | !AA | Resets the status of the Host Watchdog | 2.24 |
| ~AA2 | !AAEVV | Reads the Host Watchdog timeout settings | 2.25 |
| ~AA3EVV | !AA | Sets the Host Watchdog timeout settings | 2.26 |
| ~AA4 | !AAPPSS | Reads the DO power-on value and the safe value | 2.27 |
| ~AA5PPSS | !AA | Sets the DO power-on value and the safe value | 2.28 |

## 2.1. %AANNTTCCFF

### Description:

This command is used to set the configuration for a specific module.

### Syntax:

**%AANNTTCCFF[CHKSUM](CR)**

**%** Delimiter character

**AA** The address of the module to be configured in hexadecimal format (00 to FF)

**NN** The new address of the module in hexadecimal format (00 to FF)

**TT** Not used by the M-7002 and should be set to 00.

**CC** The new Baud Rate code, see Section 1.7 for details. To change the Baud Rate, the module should first be switched to INIT* mode.

**FF** The command used to set the data format, checksum, and filter settings. See Section 1.7 for details of the data format. To change the checksum settings, the module should first be switched to INIT* mode.

### Response:

Valid Response: **!AA[CHKSUM](CR)**

Invalid Response: **?AA[CHKSUM](CR)**

**!** Delimiter for a valid response

**?** Delimiter for an invalid response
(If the Baud Rate or checksum settings are changed without first switching to INIT* mode, the module will return an invalid response.)

**AA** The address of the responding module in hexadecimal format (00 to FF)

There will be no response if the command syntax is incorrect, there is a communication error, or there is no module with the specified address.

## Note:

Changes to the address, Type Code, Data Format and Filter settings take effect immediately after a valid command is received. Changes to the Baud Rate and checksum settings take effect at the next power-on reset.

## Examples:

Command: %0102000600 Response: !02

Changes the address of module 01 to 02 and returns a valid response.

Command: %0202000602 Response: !02

Sets the data format of module 02 to type 2 (2’s complement hexadecimal). The module returns a valid response.

Command: %0101000A00 Response: ?01

Attempts to change the Baud Rate of module 01 to 115200 bps, but returns an invalid response because the module was not switched to INIT* mode before sending the command.

Command: %0101000A00 Response: !01

Changes the Baud Rate of module 01 to 115200 bps and the module is in INIT* mode. The module returns a valid response.

## Related Commands:

Section 2.7 $AA2

## Related Topics:

Section 1.7 Configuration Tables

## 2.2. #**

### Description:

When this command is received, it allows every analog input module to read data from every input channel and the data will be stored in the buffer for later retrieval.

### Syntax:

**#**[CHKSUM](CR)**

| | |
|---|---|
| # | Delimiter character |
| ** | The synchronized sampling command |

### Response:

There is no response to this command. To access the data, another command, $AA4, must be sent, see Section 2.8 for details.

### Examples:

Command: #**                              No response

Sends the synchronized sampling command.

Command: $014                              Response:

```
>011+025.12+020.45+012.78+018.97
```

Sends the command to read the synchronized data. The status byte of the response is 1, which means that it is the first time the synchronized data has been read since the previous #** command was sent.

Command: $014                              Response:

```
>010+025.12+020.45+012.78+018.97
```

Sends the command to read the synchronized data. The status byte of the response is 0, which means that it is not the first time the synchronized data has been read since the previous #** command was sent.

### Related Commands:

Section 2.8 $AA4

## 2.3. #AA

### Description:

This command is used to read the data from all analog input channels of a specified module.

### Syntax:

**#AA[CHKSUM](CR)**

| | |
|---|---|
| # | Delimiter character |
| **AA** | The address of the module to be read in hexadecimal format (00 to FF) |

### Response:

| | |
|---|---|
| Valid Response: | **>(Data)[CHKSUM](CR)** |
| Invalid Response: | **?AA[CHKSUM](CR)** |

| | |
|---|---|
| **>** | Delimiter character for a valid response |
| **?** | Delimiter character for an invalid response |
| **(Data)** | The data from all analog input channels. See Section 1.7 for details of the data format. |
| **AA** | The address of the responding module in hexadecimal format (00 to FF) |

There will be no response if the command syntax is incorrect, there is a communication error, or there is no module with the specified address.

## Examples:

Command: #01 Response:
>+025.12+020.45+012.78+018.97

Reads module 01 and receives a valid response with the data in engineering units format.

Command: #02 Response:
>4C532628E2D683A2

Reads module 02 and receives a valid response with the data in hexadecimal format.

Command: #03 Response:
>-9999.9-9999.9-9999.9-9999.9

Attempts to read module 03, but returns an invalid response indicating that the data is out of range.

## Related Commands:

Section 2.1 %AANNTTCCFF, Section 2.7 $AA2, Section 2.4 #AAN

## Related Topics:

Section 1.7 Configuration Tables

## 2.4. #AAN

### Description:

This command is used to read the analog input data from channel N of a specified module.

### Syntax:

**#AAN[CHKSUM](CR)**

# Delimiter character

**AA** The address of the module to be read in hexadecimal format (00 to FF)

**N** The channel to be read, zero based

### Response:

Valid Response: **>(Data)[CHKSUM](CR)**

Invalid Response: **?AA[CHKSUM](CR)**

> Delimiter character for a valid response

**?** Delimiter character for an invalid response

**AA** The address of the responding module in hexadecimal format (00 to FF)

**(Data)** The analog input data from the specified channel. See Section 1.7 for details of the data format.

There will be no response if the command syntax is incorrect, there is a communication error, or there is no module with the specified address.

### Examples:

Command: #032 Response: >+025.13

Reads data from channel 2 of module 03 and returns a valid response indicating a value of +025.13.

Command: #02F Response: ?02

Attempts to read data from channel 15 of module 02, but returns an invalid response because channel 15 does not exist.

### Related Commands:

Section 2.1 %AANNTTCCFF, Section 2.3 #AA, Section 2.7 $AA2

### Related Topics:

Section 1.7 Configuration Tables

## 2.5. $AA0

### Description:

This command is used to perform an analog input span calibration on a specified module.

### Syntax:

**$AA0[CHKSUM](CR)**

| | |
|---|---|
| **$** | Delimiter character |
| **AA** | The address of the module to be calibrated in hexadecimal format (00 to FF) |
| **0** | The command to perform the analog input span calibration |

### Response:

| | |
|---|---|
| Valid Response: | **!AA[CHKSUM](CR)** |
| Invalid Response: | **?AA[CHKSUM](CR)** |

| | |
|---|---|
| **!** | Delimiter character for a valid response |
| **?** | Delimiter character for an invalid response |
| **AA** | The address of the responding module in hexadecimal format (00 to FF) |

There will be no response if the command syntax is incorrect, there is a communication error, or there is no module with the specified address.

### Note:

The “enable calibration” command, ~AAEV Section 2.31, must be sent before this command is used. See Section 1.6 for details.

### Examples:

| | |
|---|---|
| Command: $010 | Response: ?01 |

Attempts to perform an analog input span calibration on module 01, but returns an invalid response because the "enable calibration" command, "~AAEV", was not sent in advance.

| | |
|---|---|
| Command: ~01E1 | Response: !01 |

Enables calibration on module 01 and returns a valid response.

| | |
|---|---|
| Command: $010 | Response: !01 |

Performs an analog input span calibration on module 01 and returns a valid response.

### Related Commands:

Section 2.6 $AA1, Section 2.31 ~AAEV

### Related Topics:

Section 1.6 Calibration

## 2.6. $AA1

### Description:

This command is used to perform an analog input zero calibration on a specified module.

### Syntax:

**$AA1[CHKSUM](CR)**

**$** Delimiter character

**AA** The address of the module to be calibrated in hexadecimal format (00 to FF)

**1** The command to perform the analog input zero calibration

### Response:

Valid Response: **!AA[CHKSUM](CR)**

Invalid Response: **?AA[CHKSUM](CR)**

**!** Delimiter character for a valid response

**?** Delimiter character for an invalid response

**AA** The address of the responding module in hexadecimal format (00 to FF)

There will be no response if the command syntax is incorrect, there is a communication error, or there is no module with the specified address.

### Notes:

The “enable calibration” command, ~AAEV Section 2.31, must be sent before this command is used, see Section 1.6 for details.

## Examples:

Command: $011 Response: ?01

Attempts to perform an analog input zero calibration on module 01, but returns an invalid response because the “enable calibration” command, “~AAEV”, was not sent in advance.

Command: ~01E1 Response: !01

Enables calibration on module 01 and returns a valid response.

Command: $011 Response: !01

Performs an analog input zero calibration on module 01 and returns a valid response.

## Related Commands:

Section 2.5 $AA0, Section 2.31 ~AAEV

## Related Topics:

Section 1.6 Calibration

## 2.7. $AA2

### Description:

This command is used to read the configuration of a specified module.

### Syntax:

**$AA2[CHKSUM](CR)**

**$** Delimiter character

**AA** The address of the module to be read in hexadecimal format (00 to FF)

**2** The command to read the configuration of the module

### Response:

Valid Response: **!AATTCCFF[CHKSUM](CR)**

Invalid Response: **?AA[CHKSUM](CR)**

**!** Delimiter character for a valid response

**?** Delimiter character for an invalid response

**AA** The address of the responding module in hexadecimal format (00 to FF)

**TT** Not used by the M-7002 and should be 00

**CC** The Baud Rate code for the module. See Section 1.7 for details of the data format..

**FF** The data format, checksum and filter settings for the module. See Section 1.7 for details of the data format.

There will be no response if the command syntax is incorrect, there is a communication error, or there is no module with the specified address.

## Examples:

Command: $012 Response: !01000A00

Reads the configuration of module 01 and returns a valid response indicating that the Baud Rate is 115200 bps, the data format is Engineering units and the checksum is disabled.

Command: $022 Response: !02000602

Reads the configuration of module 02 and returns a valid response indicating that the Baud Rate is 9600 bps, data format is 2’s compliment hexadecimal and checksum is disabled.

## Related Commands:

Section 2.1 %AANNTTCCFF

## Related Topics:

Section 1.7 Configuration Tables

## 2.8. $AA4

### Description:

This command is used to read the synchronization data from a specified module that was stored when the last #** command (Section 2.2) was sent.

### Syntax:

**$AA4[CHKSUM](CR)**

**$** Delimiter character

**AA** The address of the module to be read in hexadecimal format (00 to FF)

**4** The command to read the synchronization data

### Response:

Valid Response: **!AAS(Data)[CHKSUM](CR)**

Invalid Response: **?AA[CHKSUM](CR)**

**!** Delimiter character for a valid response

**?** Delimiter character for an invalid response

**AA** The address of the responding module in hexadecimal format (00 to FF)

**S** The status of the synchronization data
0: The data has been read before
1: The data is being read for the first time

**(Data)** The synchronization data. See Section 1.7 for details of the data format.

There will be no response if the command syntax is incorrect, there is a communication error, or there is no module with the specified address.

## Examples:

Command: #** No response

Sends the synchronized sampling command. There will be no response to this command.

Command: $014 Response:
>011+00.000+00.100+01.000+10.000

Reads the synchronization data from module 01 and returns a valid response containing the data, and sets the status byte to 1 to indicate that the synchronized data is being read for the first time.

Command: $014 Response:
>010+00.000+00.100+01.000+10.000

Reads the synchronized data from module 01 and returns a valid response containing the data, and sets the status byte to 0 to indicate that the synchronized data has already been read before.

## Related Commands:

Section 2.2 #**

## Related Topics:

Section 1.7 Configuration Tables

## 2.9. $AA5

### Description:

This command is used to read the reset status of a specified module.

### Syntax:

**$AA5[CHKSUM](CR)**

**$** Delimiter character

**AA** The address of the module to be read in hexadecimal format (00 to FF)

**5** The command to read the reset status

### Response:

Valid Response: **!AAS[CHKSUM](CR)**

Invalid Response: **?AA[CHKSUM](CR)**

**!** Delimiter character for a valid response

**?** Delimiter character for an invalid response

**AA** The address of the responding module in hexadecimal format (00 to FF)

**S** The reset status of the module:

0: This is not the first time the command has been sent since the module was powered on, which denotes that there has been no module reset since the last $AA5 command was sent.

1: This is the first time the command has been sent since the module was powered on.

There will be no response if the command syntax is incorrect, there is a communication error, or there is no module with the specified address.

**Examples:**

Command: $015 Response: !011

Reads the reset status of module 01. The module returns a valid response indicating that it is the first time the $AA5 command has been sent since the module was powered on.

Command: $015 Response: !010

Reads the reset status of module 01. The module returns a valid response indicating that there has been no module reset since the last $AA5 command was sent.

## 2.10. $AA5VV

### Description:

This command is used to specify which channel(s) of a specified module are to be enabled.

### Syntax:

**$AA5VVVV[CHKSUM](CR)**

**$** Delimiter character

**AA** The address of the module to be set in hexadecimal format (00 to FF)

**5** The command to set the channel(s) to enabled

**VV** A two-digit hexadecimal value, where bit 0 corresponds to channel 0, bit 1 corresponds to channel 1, etc. When the bit is 0, it means that the channel is disabled and 1 means that the channel is enabled.

### Response:

Valid Response: **!AA[CHKSUM](CR)**

Invalid Response: **?AA[CHKSUM](CR)**

**!** Delimiter character for a valid response

**?** Delimiter character for an invalid response. An invalid response is returned if an attempt is made to enable a channel that is not present or does not exist.

**AA** The address of the responding module in hexadecimal format (00 to FF)

There will be no response if the command syntax is incorrect, there is a communication error, or there is no module with the specified address.

### Examples:

Command: $0150A Response: !01

Enables channels 1 and 3 of module 01 and disables all other channels. The module returns a valid response.

Command: $016 Response: !010A

Reads the channel status of module 01 and returns a valid response of “0A”, meaning that channels 1 and 3 are enabled and all other channels are disabled.

### Related Commands:

Section 2.11 $AA6

## 2.11. $AA6

### Description:

This command is used to read whether each channel of a specified module is enabled or disabled.

### Syntax:

**$AA6[CHKSUM](CR)**

| | |
|---|---|
| **$** | Delimiter character |
| **AA** | The address of the module to be read in hexadecimal format (00 to FF) |
| **6** | The command to read the channel status |

### Response:

| | |
|---|---|
| Valid Response: | **!AAVV[CHKSUM](CR)** |
| Invalid Response: | **?AA[CHKSUM](CR)** |

| | |
|---|---|
| **!** | Delimiter character for a valid response |
| **?** | Delimiter character for an invalid response |
| **AA** | The address of the responding module in hexadecimal format (00 to FF) |
| **VV** | A two-digit hexadecimal value, where bit 0 corresponds to channel 0, bit 1 corresponds to channel 1, etc.  When the bit is 0 it means that the channel is disabled, and 1 means that the channel is enabled. |

There will be no response if the command syntax is incorrect, there is a communication error, or there is no module with the specified address.

### Examples:

Command: $0150A Response: !01

Enables channels 1 and 3 of module 01 and disables all other channels. The module returns a valid response.

Command: $016 Response: !010A

Reads the channel status of module 01 and returns a valid response of “0A”, meaning that channels 1 and 3 are enabled and all other channels are disabled.

### Related Commands:

Section 2.10 $AA5VV

## 2.12. $AA7CiRrr

### Description:

This command is used to set the type code for a specific channel of a specified module.

### Syntax:

**$AA7CiRrr[CHKSUM](CR)**

| | |
|---|---|
| **$** | Delimiter character |
| **AA** | The address of the module to be set in hexadecimal format (00 to FF) |
| **7** | The command to set the channel range code |
| **Ci** | i specifies the channel to be set, zero based |
| **Rrr** | rr represents the type code for the channel to be set. Refer to the Analog Input Type Settings table in Section 1.7 for details. |

### Response:

| | |
|---|---|
| Valid Response: | **!AA[CHKSUM](CR)** |
| Invalid Response: | **?AA[CHKSUM](CR)** |

| | |
|---|---|
| **!** | Delimiter character for a valid response |
| **?** | Delimiter character for an invalid response or an invalid type code |
| **AA** | The address of the responding module in hexadecimal format (00 to FF) |

There will be no response if the command syntax is incorrect, there is a communication error, or there is no module with the specified address.

## Examples:

Command: $017C0R08 Response: !01

Sets the type code for channel 0 of module 01 to 08 (-10 ~ +10 V) and the module returns a valid response.

Command: $018C0 Response: !01C0R08

Reads the analog input type code information for channel 0 of module 01 and returns a valid response of 08, which means that the input type is -10~+10 V.

Command: $037C1RFF Response: ?03

Attempts to set the type code for channel 1 of module 03 to FF, but returns an invalid response because the type code is invalid.

## Related Commands:

Section 2.13 $AA8Ci

## Related Topics:

Section 1.7 Configuration Tables

## 2.13. $AA8Ci

### Description:

This command is used to read the analog input type code information for a specific channel of a specified module.

### Syntax:

**$AA8Ci[CHKSUM](CR)**

| | |
|---|---|
| **$** | Delimiter character |
| **AA** | The address of the module to be read in hexadecimal format (00 to FF) |
| **8** | The command to read the type code of the channel |
| **Ci** | i specifies which analog input channel to access for the type code information |

### Response:

| | |
|---|---|
| Valid Response: | **!AACiRrr[CHKSUM](CR)** |
| Invalid Response: | **?AA[CHKSUM](CR)** |

| | |
|---|---|
| **!** | Delimiter character for a valid response |
| **?** | Delimiter character for an invalid response or an invalid channel |
| **AA** | The address of the responding module in hexadecimal format (00 to FF) |
| **Ci** | i specifies the analog input channel that was accessed to retrieve the type code information. |
| **Rrr** | rr represents the type code for the specified analog input channel. Refer to the Analog Input Type Settings table in Section 1.7 for details. |

There will be no response if the command syntax is incorrect, there is a communication error, or there is no module with the specified address.

### Examples:

Command: $017C0R08 Response: !01

Sets the type code for channel 0 of module 01 to 08 (-10 ~ +10 V) and the module returns a valid response.

Command: $018C0 Response: !01C0R08

Reads the analog input type code information for channel 0 of module 01 and returns a valid response of 08 which means that the input type is -0~+10 V.

Command: $018CF Response: ?01

Attempts to read the analog input type code information for channel 15 of module 01, but returns an invalid response because analog input channel 15 does not exist.

### Related Commands:

Section 2.12 $AA7CiRrr

### Related Topics:

Section 1.7 Configuration Tables

## 2.14. $AAC

### Description:

This command is used to clear the digital input/output latch data for a specified module.

### Syntax:

**$AAC[CHKSUM](CR)**

| | |
|---|---|
| **$** | Delimiter character |
| **AA** | The address of the module to be cleared in hexadecimal format (00 to FF) |
| **C** | The command to clear the digital input/output latch data |

### Response:

| | |
|---|---|
| Valid Response: | **!AA[CHKSUM](CR)** |
| Invalid Response: | **?AA[CHKSUM](CR)** |

| | |
|---|---|
| **!** | Delimiter character for a valid response |
| **?** | Delimiter character for an invalid response |
| **AA** | The address of the responding module in hexadecimal format (00 to FF) |

There will be no response if the command syntax is incorrect, there is a communication error, or there is no module with the specified address.

### Examples:

Command: $01C Response: !01

Clears the latched data for module 01 and returns a valid response.

### Related Command:

Section 2.17 $AALS

## 2.15. $AAF

### Description:

This command is used to read the firmware version information for a specified module.

### Syntax:

**$AAF[CHKSUM](CR)**

| | |
|---|---|
| **$** | Delimiter character |
| **AA** | The address of the module to be read in hexadecimal format (00 to FF) |
| **F** | The command to read the firmware version information |

### Response:

| | |
|---|---|
| Valid Response: | **!AA(Data)[CHKSUM](CR)** |
| Invalid Response: | **?AA[CHKSUM](CR)** |

| | |
|---|---|
| **!** | Delimiter character for a valid response |
| **?** | Delimiter character for an invalid response |
| **AA** | The address of the responding module in hexadecimal format (00 to FF) |
| **(Data)** | A string indicating the firmware version information for the module |

There will be no response if the command syntax is incorrect, there is a communication error, or there is no module with the specified address.

### Examples:

Command: $01F Response: !01A2.0

Reads the firmware version information for module 01 and returns a valid response showing that it is version A2.0.

## 2.16. $AAI

### Description:

This command is used to read the status of the INIT switch on a specified module.

### Syntax:

**$AAI[CHKSUM](CR)**

| | |
|---|---|
| **$** | Delimiter character |
| **AA** | The address of the module to be read in hexadecimal format (00 to FF) |
| **I** | The command to read the status of the INIT switch on the module |

### Response:

Valid Response: **!AAS[CHKSUM](CR)**
Invalid Response: **?AA[CHKSUM](CR)**

| | |
|---|---|
| **!** | Delimiter character for a valid response |
| **?** | Delimiter character for an invalid response |
| **AA** | The address of the responding module in hexadecimal format (00 to FF) |
| **S** | The status of the INIT switch on the module<br>0: The INIT switch is currently in the INIT position<br>1: The INIT switch is currently in the Normal position |

There will be no response if the command syntax is incorrect, there is a communication error, or there is no module with the specified address.

**Examples:**

Command: $01I Response: !010

Reads the status of the INIT switch on module 01 and shows that it is currently in the INIT position.

## 2.17. $AALS

**Description:**

This command is used to read the status of the digital input/output latch for each channel of a specified module.

**Syntax:**

**$AALS[CHKSUM](CR)**

**$** Delimiter character

**AA** The address of the module to be read in hexadecimal format (00 to FF)

**L** The command to read the status of the latches for each channel

**S** The status of the latch
0 = Latch low
1 = Latch high

**Response:**

Valid Response: **!(Data)[CHKSUM](CR)**

Invalid Response: **?AA[CHKSUM](CR)**

**!** Delimiter character for a valid response

**?** Delimiter character for an invalid response

**AA** The address of the responding module in hexadecimal format (00 to FF)

**(Data)** A four-digit hexadecimal value followed by 00 representing the status of the latched digital output/input channels

There will be no response if the command syntax is incorrect, there is a communication error, or there is no module with the specified address.

**Examples:**

Command: $01L1 Response: !030100

Reads module 01 and returns a valid response showing that the latches are high on digital output channels 0 and 1 and digital input channel 0.

Command: $01C Response: !01

Clears the digital input and output latch data for module 01 and returns a valid response.

Command: $01L1 Response: !000000

Reads module 01 and returns a valid response showing that high latches have not occurred on any digital input or output channels.

Command: $01L2 Response: ?01

Attempts to read the module 01, but returns an invalid response because the parameter “2” is outside the range of valid value.

**Related Commands:**

Section 2.14 $AAC, Section 2.46 @AADODD

## 2.18. $AAM

### Description:

This command is used to read the name of a specified module.

### Syntax:

**$AAM[CHKSUM](CR)**

| | |
|---|---|
| **$** | Delimiter character |
| **AA** | The address of the module to be read in hexadecimal format (00 to FF) |
| **M** | The command to read the name of the module |

### Response:

| | |
|---|---|
| Valid Response: | **!AA(Data)[CHKSUM](CR)** |
| Invalid Response: | **?AA[CHKSUM](CR)** |

| | |
|---|---|
| **!** | Delimiter character for a valid response |
| **?** | Delimiter character for an invalid response |
| **AA** | The address of the responding module in hexadecimal format (00 to FF) |
| **(Data)** | A string showing the name of the module |

There will be no response if the command syntax is incorrect, there is a communication error, or there is no module with the specified address.

### Examples:

Command: $01M                          Response: !017002

Reads module 01 and returns a valid response with the name “7002”.

### Related Commands:

Section 2.33 ~AAO(Data)

## 2.19. $AAP

### Description:

This command is used to read which communication protocol is supported and being used by a specified module.

### Syntax:

**$AAP[CHKSUM](CR)**

**$** Delimiter character

**AA** The address of the module to be read in hexadecimal format (00 to FF)

**P** The command to read the communication protocol

### Response:

Valid Response: **!AASC[CHKSUM](CR)**

Invalid Response: **?AA[CHKSUM](CR)**

**!** Delimiter character for a valid response

**?** Delimiter character for an invalid response

**AA** The address of the responding module in hexadecimal format (00 to FF)

**S** Indicates which protocol is supported
0: Only the DCON protocol is supported
1: Both the DCON and Modbus RTU protocols are supported

C Indicates which protocol is currently being used
0: The protocol set in the EEPROM is DCON
1: The protocol set in the EEPROM is Modbus RTU

There will be no response if the command syntax is incorrect, there is a communication error, or there is no module with the specified address.

### Examples:

Command: $01P1 Response: !01

Sets the communication protocol for module 01 to Modbus RTU and returns a valid response.

Command: $01P Response: !0111

Reads which communication protocol is being used by module 01 and returns a valid response of 10, meaning that it supports both the DCON and Modbus RTU protocols, and the protocol that will be used at the next power-on reset is Modbus RTU.

### Related Commands:

Section 2.20 $AAPN

## 2.20. $AAPN

### Description:

This command is used to set the communication protocol to be used by a specified module.

### Syntax:

**$AAPN[CHKSUM](CR)**

**$** Delimiter character

**AA** The address of the module to be set in hexadecimal format (00 to FF)

**P** The command to set the communication protocol

**N** The protocol to be used
0: DCON
1: Modbus RTU

### Note:

Before using this command, the INIT switch must be in the INIT position, see Section 5.1 for details. The settings for the new protocol are saved in the EEPROM and will become effective after the next power-on reset.

### Response:

Valid Response: **!AA[CHKSUM](CR)**
Invalid Response: **?AA[CHKSUM](CR)**

**!** Delimiter character for a valid response

**?** Delimiter character for an invalid response

**AA** The address of the responding module in hexadecimal format (00 to FF)

There will be no response if the command syntax is incorrect, there is a communication error, or there is no module with the specified address.

### Examples:

Command: $01P1 Response: ?01

Attempts to set the communication protocol for module 01 to Modbus RTU, but returns an invalid response because the INIT switch is not in INIT position.

Command: $01P1 Response: !01

Sets the communication protocol for module 01 to Modbus RTU and returns a valid response. The new protocol will become effective after the next power-on reset.

Command: $01P Response: !0111

Reads which communication protocol is being used by module 01 and returns a valid response of 10 meaning that it supports both the DCON and Modbus RTU protocols, and the protocol that will be used at the next power-on reset is Modbus RTU.

### Related Commands:

Section 2.19 $AAP

### Related Topics:

Section 5.1 INIT Mode

## 2.21. $AAS1

**Description:**
This command is used to reload the factory default calibration parameters for a specified module, including the internal calibration parameters.

**Syntax:**
**$AAS1[CHKSUM](CR)**

| | |
|---|---|
| **$** | Delimiter character |
| **AA** | The address of the module where the default calibration parameters are to be reloaded in hexadecimal format (00 to FF) |
| **S1** | The command to reload the factory default calibration parameters |

**Response**:

| | |
|---|---|
| Valid Response: | **!AA[CHKSUM](CR)** |
| Invalid Response: | **?AA[CHKSUM](CR)** |

| | |
|---|---|
| **!** | Delimiter character for a valid response |
| **?** | Delimiter character for an invalid response |
| **AA** | The address of the responding module in hexadecimal format (00 to FF) |

There will be no response if the command syntax is incorrect, there is a communication error, or there is no module with the specified address.

**Examples:**

Command: $01S1 Response: !01

Sends a command to module 01 to reload the factory default calibration parameters and returns a valid response.

**Related Commands:**

Section 2.5 $AA0, Section 2.6 $AA1, Section 2.31 ~AAEV

**Related Topics:**

Section 1.6 Calibration

## 2.22. ~**

### Description:

This command is used to inform all modules on the network that the host is OK.

### Syntax:

**~**[CHKSUM](CR)**

| | |
|---|---|
| ~ | Delimiter character |
| ** | The “Host OK” command |

### Response:

There is no response to this command.

### Examples:

Command: ~** No response

Sends a “Host OK” command to all modules on the network.

### Related Commands:

Section 2.23 ~AA0, Section 2.24 ~AA1, Section 2.25 ~AA2, Section 2.26 ~AA3EVV, Section 2.27 ~AA4, Section 2.28 ~AA5PPSS

## 2.23. ~AA0

### Description:

This command is used to read the status of the Host Watchdog for a specified module.

### Syntax:

**~AA0[CHKSUM](CR)**

| | |
|---|---|
| ~ | Delimiter character |
| **AA** | The address of the module to be read in hexadecimal format (00 to FF) |
| **0** | The command to read the status of the module’s Host Watchdog |

### Response:

| | |
|---|---|
| Valid Response: | **!AASS[CHKSUM](CR)** |
| Invalid Response: | **?AA[CHKSUM](CR)** |

**!** Delimiter character for a valid response

**?** Delimiter character for an invalid response

**AA** The address of the responding module in hexadecimal format (00 to FF)

**SS** A two-digit hexadecimal value that represents the status of the Host Watchdog, where:

Bit 2: 0 indicates that no Host Watchdog timeout has occurred, and 1 indicates that a Host Watchdog timeout has occurred.

Note: The status information for the Host Watchdog is stored in EEPROM and can only be reset using the ~AA1 command.

Bit 7: 0 indicates that the Host Watchdog is disabled, and 1 indicates that the Host Watchdog is enabled.

There will be no response if the command syntax is incorrect, there is a communication error, or there is no module with the specified address.

## Examples:

Command: ~010 Response: !0100

Reads the status of the Host Watchdog for module 01 and returns a valid response with a value of 00, meaning that the Host Watchdog is disabled and no Host Watchdog timeout has occurred.

Command: ~020 Response: !0204

Reads the status of the Host Watchdog for module 02 and returns a valid response with a value of 04, meaning that a Host Watchdog timeout has occurred.

## Related Commands:

Section 2.22 ~**, Section 2.24 ~AA1, Section 2.25 ~AA2, Section 2.26 ~AA3EVV

## Related Topics:

Section 5.2 Dual Watchdog Operation

## 2.24. ~AA1

### Description:

This command is used to reset the timeout status of the Host Watchdog for a specified module.

### Syntax:

**~AA1[CHKSUM](CR)**

| | |
|---|---|
| ~ | Delimiter character |
| **AA** | The address of the module to be reset in hexadecimal format (00 to FF) |
| **1** | The command to reset the timeout status of the Host Watchdog |

### Response:

| | |
|---|---|
| Valid Response: | **!AA[CHKSUM](CR)** |
| Invalid Response: | **?AA[CHKSUM](CR)** |

| | |
|---|---|
| **!** | Delimiter character for a valid response |
| **?** | Delimiter character for an invalid response |
| **AA** | The address of the responding module in hexadecimal format (00 to FF) |

There will be no response if the command syntax is incorrect, there is a communication error, or there is no module with the specified address.

### Examples:

Command: ~010 Response: !0104

Reads the status of the Host Watchdog for module 01 and returns a valid response showing that a Host Watchdog timeout has occurred.

Command: ~011 Response: !01

Resets the Host Watchdog timeout for module 01 and returns a valid response.

Command: ~010 Response: !0100

Reads the status of the Host Watchdog for module 01 and shows that no Host Watchdog timeout has occurred.

### Related Commands:

Section 2.22 ~**, Section 2.23 ~AA0, Section 2.25 ~AA2, Section 2.26 ~AA3EVV

## 2.25. ~AA2

### Description:

This command is used to read the Host Watchdog timeout value for a specified module.

### Syntax:

**~AA2[CHKSUM](CR)**

| | |
|---|---|
| ~ | Delimiter character |
| **AA** | The address of the module to be read in hexadecimal format (00 to FF) |
| **2** | The command to read the Host Watchdog timeout value |

### Response:

Valid Response: **!AAEVV[CHKSUM](CR)**
Invalid Response: **?AA[CHKSUM](CR)**

| | |
|---|---|
| **!** | Delimiter character for a valid response |
| **?** | Delimiter character for an invalid response |
| **AA** | The address of the responding module in hexadecimal format (00 to FF) |
| **E** | The status of the Host Watchdog<br>0: The Host Watchdog is disabled<br>1: The Host Watchdog is enabled |
| **VV** | A two-digit hexadecimal value that represents the Host Watchdog timeout value in tenths of a second, for example, 01 means 0.1 seconds and FF means 25.5 seconds. |

There will be no response if the command syntax is incorrect, there is a communication error, or there is no module with the specified address.

## Examples:

Command: ~013164 Response: !01

Enables the Host Watchdog for module 01 and sets the Host Watchdog timeout value to 10.0 seconds. The module returns a valid response.

Command: ~012 Response: !011FF

Reads the Host Watchdog timeout value for module 01 and returns a valid response with a value of 1FF, meaning that the Host Watchdog is enabled and the Host Watchdog timeout value is 25.5 seconds.

## Related Commands:

Section 2.22 ~**, Section 2.23 ~AA0, Section 2.24 ~AA1, Section 2.26 ~AA3EVV

## Related Topics:

Section 5.2 Dual Watchdog Operation

## 2.26. ~AA3EVV

### Description:

This command is used to enable or disable the Host Watchdog for a specified module and to set the Host Watchdog timeout value.

### Syntax:

**~AA3EVV[CHKSUM](CR)**

| | |
|---|---|
| ~ | Delimiter character |
| **AA** | The address of the module to be set in hexadecimal format (00 to FF) |
| **3** | The command to set the Host Watchdog |
| **E** | The command to enable or disable the Host Watchdog<br>0: Disables the Host Watchdog<br>1: Enables the Host Watchdog |
| **VV** | A two-digit hexadecimal value to represent the Host Watchdog timeout value in tenths of a second, for example, 01 means 0.1 seconds and FF means 25.5 seconds. |

### Response:

| | |
|---|---|
| Valid Response: | **!AA[CHKSUM](CR)** |
| Invalid Response: | **?AA[CHKSUM](CR)** |

| | |
|---|---|
| **!** | Delimiter character for a valid response |
| **?** | Delimiter character for an invalid response |
| **AA** | The address of the responding module in hexadecimal format (00 to FF) |

There will be no response if the command syntax is incorrect, there is a communication error, or there is no module with the specified address.

### Examples:

Command: ~013164 Response: !01

Enables the Host Watchdog for module 01 and sets the Host Watchdog timeout value to 10.0 seconds. The module returns a valid response.

Command: ~012 Response: !01164

Reads the Host Watchdog timeout value for module 01 and returns a valid response with a value of 164, meaning that the Host Watchdog is enabled and the Host Watchdog timeout value is 10.0 seconds.

### Related Commands:

Section 2.22 ~**, Section 2.23 ~AA0, Section 2.24 ~AA1, Section 2.25 ~AA2

### Related Topics:

Section 5.2 Dual Watchdog Operation

## 2.27. ~AA4

**Description:**
This command is used to read the digital output power-on value and the safe value for a specified module.

**Syntax:**
**~AA4[CHKSUM](CR)**

~ Delimiter character

**AA** The address of the module to be read in hexadecimal format (00 to FF)

**4** The command to read the digital output power-on value and the safe value

**Response:**
Valid Rommand: **!AAPPSS[CHKSUM](CR)**
Invalid Rommand: **?AA[CHKSUM](CR)**

**!** Delimiter character for a valid response

**?** Delimiter character for an invalid response

**AA** The address of the responding module in hexadecimal format (00 to FF)

**PP** A two-digit hexadecimal value that represents the digital output power-on value

**SS** A two-digit hexadecimal value that represents the digital output safe value

There will be no response if the command syntax is incorrect, there is a communication error, or there is no module with the specified address

**Note:**

Neither the power-on value nor the safe value have any effect on digital outputs that are associated with alarm outputs.

**Examples:**

Command: ~0150300 Response: !01

Sets the digital output power-on value to 03 and sets the digital output safe value to 00, and returns a valid response.

Command: ~014 Response: !010300

Reads the digital output power-on value and the digital output safe value for module 01 and returns a valid response indicating a power-on value of 03 and safe value of 00.

**Related Commands:**

Section 2.22 ~**, Section 2.23 ~AA0, Section 2.24 ~AA1, Section 2.25 ~AA2, Section 2.26 ~AA3EVV, Section 2.28 ~AA5PPSS

## 2.28. ~AA5PPSS

**Description:**
This command is used to set the digital output power-on value and the safe value for a specified module.

**Syntax:**
**~AA5PPSS[CHKSUM](CR)**

|  |  |
|---|---|
| ~ | Delimiter character |
| **AA** | The address of the module to be set in hexadecimal format (00 to FF) |
| **5** | The command to set the digital output power-on value and the safe value |
| **PP** | A two-digit hexadecimal value to represent the digital output power-on value |
| **SS** | A two-digit hexadecimal value to represent the digital output safe value |

**Response:**

|  |  |
|---|---|
| Valid Response: | **!AA[CHKSUM](CR)** |
| Invalid Response: | **?AA[CHKSUM](CR)** |

|  |  |
|---|---|
| **!** | Delimiter character for a valid response |
| **?** | Delimiter character for an invalid response |
| **AA** | The address of the responding module in hexadecimal format (00 to FF) |

There will be no response if the command syntax is incorrect, there is a communication error, or there is no module with the specified address.

**Note:**

Neither the power-on value nor the safe value have any effect on digital outputs that are associated with alarm outputs.

**Examples:**

Command: ~0150102 Response: !01

Sets the digital output power-on value to 01 and sets the digital output safe value to 02, and returns a valid response.

Command: ~014 Response: !010102

Reads the digital output power-on value and the digital output safe value for module 01 and returns a valid response with a value of 0102, which denotes that the digital output power-on value is 01 and the digital output safe value is 02.

**Related Commands:**

Section 2.22 ~**, Section 2.23 ~AA0, Section 2.24 ~AA1, Section 2.25 ~AA2, Section 2.26 ~AA3EVV, Section 2.27 ~AA4

## 2.29. ~AAD

### Description:

This command is used to read the miscellaneous settings for a specified module.

### Syntax:

**~AAD[CHKSUM](CR)**

~ Delimiter character

**AA** The address of the module to be read in hexadecimal format (00 to FF)

**D** The command to read the miscellaneous settings

### Response:

Valid Response: **!AAVV[CHKSUM](CR)**

Invalid Response: **?AA[CHKSUM](CR)**

**!** Delimiter character for a valid response

**?** Delimiter character for an invalid response

**AA** The address of the responding module in hexadecimal format (00 to FF)

**VV** A two-digit hexadecimal value that represents the miscellaneous settings, as indicated in the following tables:

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| Reserved | | | | | | OA | IA |

| Key | Description |
|---|---|
| OA | DO active state<br>0: Output value 0 indicates the relay is inactive<br>Output value 1 indicates the relay is active<br>1: Output value 0 indicates the relay is active<br>Output value 1 indicates the relay is inactive |
| IA | DI active state<br>0: Input value 0 indicates high voltage<br>Input value 1 indicates that there is no signal or the voltage is low<br>1: Input value 0 indicates that there is no signal or the voltage is low<br>Input value 1 indicates high voltage |

There will be no response if the command syntax is incorrect, there is a communication error, or there is no module with the specified address.

## Examples:

Command: ~$01D01 Response: !01

Sets the miscellaneous settings for module 01 to 01, meaning that the digital input will be inversed, and returns a valid response.

Command: ~$01D Response: !0101

Reads the miscellaneous settings of module 01 and returns a valid response with a value of 01 indicating that the digital input will be inversed.

## Related Commands:

Section 2.30 ~AADVV

## 2.30. ~AADVV

### Description:

This command is used to set the miscellaneous settings for a specified module.

### Syntax:

**~AADVV[CHKSUM](CR)**

~ Delimiter character

**AA** The address of the module to be set in hexadecimal format (00 to FF)

**D** The command to set the miscellaneous settings

**VV** A two-digit hexadecimal value that represents the miscellaneous settings, as indicated in the following tables:

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| Reserved | | | | | | OA | IA |

| Key | Description |
|---|---|
| OA | DO active state<br>0: Output value 0 indicates the relay is inactive<br>Output value 1 indicates the relay is active<br>1: Output value 0 indicates the relay is active<br>Output value 1 indicates the relay is inactive |
| IA | DI active state<br>0: Input value 0 indicates high voltage<br>Input value 1 indicates that there is no signal or the voltage is low<br>1: Input value 0 indicates that there is no signal or the voltage is low<br>Input value 1 indicates high voltage |

**Response**:

| | |
|---|---|
| Valid Response: | **!AA [CHKSUM](CR)** |
| Invalid Response: | **?AA[CHKSUM](CR)** |

| | |
|---|---|
| **!** | Delimiter character for a valid response |
| **?** | Delimiter character for an invalid response |
| **AA** | The address of the responding module in hexadecimal format (00 to FF) |

There will be no response if the command syntax is incorrect, there is a communication error, or there is no module with the specified address.

**Examples:**

Command: ~$01D01　　Response: !01

Sets the miscellaneous settings for module 01 to 01, meaning that the digital input will be inversed, and returns a valid response.

Command: ~$01D　　Response: !0101

Reads the miscellaneous settings of module 01 and returns a valid response with a value of 01 indicating that the digital input will be inversed.

**Related Commands:**

Section 2.29 ~AAD

## 2.31. ~AAEV

### Description:

This command is used to enable or disable calibration on a specified module.

### Syntax:

**~AAEV[CHKSUM](CR)**

| | |
|---|---|
| ~ | Delimiter character |
| **AA** | The address of the module to be set in hexadecimal format (00 to FF) |
| **E** | The command to enable/disable calibration |
| **V** | 0: Disables calibration<br>1: Enables calibration |

### Response:

Valid Response: **!AA[CHKSUM](CR)**

Invalid Response: **?AA[CHKSUM](CR)**

| | |
|---|---|
| **!** | Delimiter character for a valid response |
| **?** | Delimiter character for an invalid response |
| **AA** | The address of the responding module in hexadecimal format (00 to FF) |

There will be no response if the command syntax is incorrect, there is a communication error, or there is no module with the specified address.

### Examples:

| | |
|---|---|
| Command: $010 | Response: ?01 |

Attempts to send the command to perform a span calibration on module 01, but returns an invalid response because the “enable calibration” command, ~AAEV, was not sent in advance.

| | |
|---|---|
| Command: ~01E1 | Response: !01 |

Enables calibration on module 01 and returns a valid response.

| | |
|---|---|
| Command: $010 | Response: !01 |

Sends the command to perform a span calibration on module 01 and returns a valid response.

### Related Commands:

Section 2.5 $AA0, Section 2.6 $AA1

### Related Topics:

Section 1.6 Calibration

## 2.32. ~AAI

**Description:**

This command is used to enable modification of the Baud Rate and checksum settings for a specified module using the software INIT function only.

**Syntax:**

**~AAI[CHKSUM](CR)**

| | |
|---|---|
| **~** | Delimiter character |
| **AA** | The address of the module to be set in hexadecimal format (00 to FF) |
| **I** | The command to set the software INIT function |

**Response:**

| | |
|---|---|
| Valid Response: | **!AA[CHKSUM](CR)** |
| Invalid Response: | **?AA[CHKSUM](CR)** |

| | |
|---|---|
| **!** | Delimiter character for a valid response |
| **?** | Delimiter character for an invalid response |
| **AA** | The address of the responding module in hexadecimal format (00 to FF) |

There will be no response if the command syntax is incorrect, there is a communication error, or there is no module with the specified address.

**Examples:**

Command: ~01T10 Response: !01

Sets the timeout value for the software INIT function on module 01 to 16 seconds and returns a valid response.

Command: $01I Response: !01

Sets the software INIT function on module 01 to enabled and returns a valid response.

Command: %0101000600 Response: !01

Sets the Baud Rate for module 01 to 9600 bps and returns a valid response.

**Related Commands:**

Section 2.1 %AANNTTCCFF, Section 2.36 ~AATnn

## 2.33. ~AAO(Data)

### Description:

This command is used to set the name of a specified module

### Syntax:

**~AAO(Data)[CHKSUM](CR)**

| | |
|---|---|
| ~ | Delimiter character |
| **AA** | The address of the module to be set in hexadecimal format (00 to FF) |
| **O** | The command to set the name of the module |
| **(Data)** | The new name of the module (max. 6 characters) |

### Response:

| | |
|---|---|
| Valid Response: | **!AA[CHKSUM](CR)** |
| Invalid Response: | **?AA[CHKSUM](CR)** |

| | |
|---|---|
| **!** | Delimiter character for a valid response |
| **?** | Delimiter character for an invalid response |
| **AA** | The address of the responding module in hexadecimal format (00 to FF) |

There will be no response if the command syntax is incorrect, there is a communication error, or there is no module with the specified address.

## Examples:

Command: ~01O7002 Response: !01

Sets the name of module 01 to “7002” and returns a valid response.

Command: $01M Response: !017002

Reads module 01 and returns a valid response with the name “7002”.

## Related Commands:

Section 2.18 $AAM

## 2.34. ~AARD

**Description:**

This command is used to read the response delay time for a specified module.

**Syntax:**

**~AARD[CHKSUM](CR)**

~ Delimiter character

**AA** The address of the module to be read in hexadecimal format (00 to FF)

**RD** The command to read the response delay time

**Response:**

Valid Response: **!AATT[CHKSUM](CR)**

Invalid Response: **?AA[CHKSUM](CR)**

**!** Delimiter character for a valid response

**?** Delimiter character for an invalid response

**AA** The address of the responding module in hexadecimal format (00 to FF)

**TT** A two-digit hexadecimal value that represents the response delay time value in milliseconds. For example, 01 denotes 1 millisecond and 1A denotes 26 milliseconds. The value must be less than or equal to 1E.

There will be no response if the command syntax is incorrect, there is a communication error, or there is no module with the specified address.

**Examples:**

Command: ~01RD10 Response: !01

Sets the response delay time to 16 milliseconds and returns a valid response..

Command: ~01RD Response: !0110

Reads the response delay time and returns a valid response with a value of 10 indicating 16 milliseconds. The response will be sent after 16 milliseconds have elapsed.

**Related Commands:**

Section 2.35 ~AARDTT

## 2.35. ~AARDTT

**Description:**

This command is used to set the response delay time for a specified module.

**Syntax:**

**~AARDTT[CHKSUM](CR)**

~ Delimiter character

**AA** The address of the module to be set in hexadecimal format (00 to FF)

**RD** The command to set the response delay time

**TT** A two-digit hexadecimal value that represents the response time value in milliseconds. For example, 01 denotes 1 millisecond and 1A denotes 26 milliseconds. The value must be less than or equal to 1E.

**Response:**

Valid Response: **!AA [CHKSUM](CR)**

Invalid Response: **?AA[CHKSUM](CR)**

**!** Delimiter character for a valid response

**?** Delimiter character for an invalid response

**AA** The address of the responding module in hexadecimal format (00 to FF)

There will be no response if the command syntax is incorrect, there is a communication error, or there is no module with the specified address.

**Examples:**

Command: ~01RD10 Response: !01

Sets the response delay time to 16 milliseconds and returns a valid response..

Command: ~01RD Response: !0110

Reads the response delay time and returns a valid response with a value of 10 indicating 16 milliseconds. The response will be sent after 16 milliseconds have elapsed.

**Related Commands:**

Section 2.34 ~AARD

## 2.36. ~AATnn

**Description:**

This command is used to set the timeout value for the software INIT function on a specified module.

**Syntax:**

**~AARDTT[CHKSUM](CR)**

**~** Delimiter character

**AA** The address of the module to be set in hexadecimal format (00 to FF)

**T** The command to set the timeout value for the software INIT function

**nn** A two-digit hexadecimal value that represents the timeout value for the software INIT function in seconds. For example, 01 denotes 1 second and 1A denotes 26 seconds. The value must be less than or equal to 3C.

**Response:**

Valid Response: **!AA [CHKSUM](CR)**

Invalid Response: **?AA[CHKSUM](CR)**

**!** Delimiter character for a valid response

**?** Delimiter character for an invalid response

**AA** The address of the responding module in hexadecimal format (00 to FF)

There will be no response if the command syntax is incorrect, there is a communication error, or there is no module with the specified address.

**Examples:**

Command: ~01T10 Response: !01

Sets the timeout value for the software INIT function on module 01 to 16 seconds and returns a valid response.

Command: $01I Response: !01

Sets the software INIT function on module 01 to enabled and returns a valid response.

Command: %0101000600 Response: !01

Sets the Baud Rate for module 01 to 9600 bps and returns a valid response.

Command: ~01TFF Response: ?01

Attempts to set the timeout value for the software INIT function on module 01 to 255 seconds, but returns an invalid response because the duration is greater than the permitted value (3C).

**Related Commands:**

Section 2.1 %AANNTTCCFF, Section 2.32 ~AAI

## 2.37. @AACECi

**Description:**

This command is used to reset the counter for a specific channel of a specified module.

**Syntax:**

**@AACECi[CHKSUM](CR)**

| | |
|---|---|
| **@** | Delimiter character |
| **AA** | The address of the module to be reset in hexadecimal format (00 to FF) |
| **CE** | The command to reset the counter |
| **Ci** | i specifies the channel to be reset, zero based |

**Response:**

| | |
|---|---|
| Valid Response: | **!AA [CHKSUM](CR)** |
| Invalid Response: | **?AA[CHKSUM](CR)** |

| | |
|---|---|
| **!** | Delimiter character for a valid response |
| **?** | Delimiter character for an invalid response |
| **AA** | The address of the responding module in hexadecimal format (00 to FF) |

There will be no response if the command syntax is incorrect, there is a communication error, or there is no module with the specified address.

**Example:**

Command: @01REC1 Response: !0100000008

Reads data from channel 1 of module 01 and returns a valid response indicating a counter value of 00000008.

Command: @01CEC1 Response: !01

Resets the counter for channel 1 of module 01 to the preset value and returns a valid response.

Command: @01REC1 Response: !0100000000

Reads data from channel 1 of module 01 and returns a valid response indicating a counter value of 00000000.

Command: @01CECF Response: !01

Attempts to reset the counter for channel 15 of module 01 to the preset value, but returns an invalid response because channel 15 does not exist.

**Related Commands:**

Section 2.51 @AARECi

## 2.38. @AACH

**Description:**

This command is used to clear the high latch values for all channels of a specified module.

**Syntax:**

**@AACH [CHKSUM](CR)**

| | |
|---|---|
| **@** | Delimiter character |
| **AA** | The address of the module to be cleared in hexadecimal format (00 to FF) |
| **CH** | The command to clear the high latch values |

**Response:**

| | |
|---|---|
| Valid Response: | **!AA[CHKSUM](CR)** |
| Invalid Response: | **?AA[CHKSUM](CR)** |

| | |
|---|---|
| **!** | Delimiter character for a valid response |
| **?** | Delimiter character for an invalid response |
| **AA** | The address of the responding module in hexadecimal format (00 to FF) |

There will be no response if the command syntax is incorrect, there is a communication error, or there is no module with the specified address.

**Examples:**

Command: @01RH0 Response: !01+05.000

Reads the high latch value for channel 0 of module 01 and returns a valid response of +05.000 (5 V) indicating a value of 5 V.

Command: @01CH Response: !01

Clears the high latch value for all channels of module 01 and returns a valid response.

Command: @01RH0 Response: !01+00.000

Reads the high latch value for channel 0 of module 01 and returns a valid response of +00.000 (0 V) indicating a value of 0 V.

**Related Commands:**

Section 2.39 @AACHi, Section 2.52 @AARH, Section 2.53 @AARHi

## 2.39. @AACHi

**Description:**
This command is used to clear the high latch value for a specific channel of a specified module.

**Syntax:**
**@AACHi [CHKSUM](CR)**
**@** Delimiter character
**AA** The address of the module to be cleared in hexadecimal format (00 to FF)
**CH** The command to clear the high latch value
**i** The channel to be cleared, zero based

**Response:**
Valid Response: **!AA[CHKSUM](CR)**
Invalid Response: **?AA[CHKSUM](CR)**
**!** Delimiter character for a valid response
**?** Delimiter character for an invalid response
**AA** The address of the responding module in hexadecimal format (00 to FF)

There will be no response if the command syntax is incorrect, there is a communication error, or there is no module with the specified address.

**Examples:**

Command: @01RH1 Response: !01+06.000

Reads the high latch value for channel 1 of module 01 and returns a valid response with a value of +06.000 (6 V).

Command: @01CH1 Response: !01

Clears the high latch value for channel 1 of module 01 and returns a valid response.

Command: @01RH1 Response: !01+00.000

Reads the high latch value for channel 1 of module 01 and returns a valid response with a value of +00.000 (0 V).

Command: @01CHF Response: !01

Attempts to clear the high latch value for channel 15 of module 01 and returns an invalid response because channel 15 does not exist.

**Related Commands:**

Section 2.38 @AACH, Section 2.52 @AARH, Section 2.53 @AARHi

## 2.40. @AACHCi

**Description:**

This command is used to clear the status of the high alarm for a specific channel of a specified module.

**Syntax:**

**@AACHCi [CHKSUM](CR)**

**@** Delimiter character

**AA** The address of the module to be cleared in hexadecimal format (00 to FF)

**CH** The command to clear the status of the high alarm

**Ci** i specifies the channel to be cleared, zero based

**Response:**

Valid Response: **!AA[CHKSUM](CR)**

Invalid Response: **?AA[CHKSUM](CR)**

**!** Delimiter character for a valid response

**?** Delimiter character for an invalid response

**AA** The address of the responding module in hexadecimal format (00 to FF)

There will be no response if the command syntax is incorrect, there is a communication error, or there is no module with the specified address.

**Examples:**

Command: @01CHC0 Response: !01

Clears the status of the high alarm for channel 0 of module 01 and returns a valid response.

Command: @01CHCF Response: !01

Attempts to clear the status of the high alarm for channel 15 of module 01, but returns an invalid response because channel 15 does not exist.

**Related Commands:**

Section 2.48 @AAHI(Data)Ci, Section 2.50 @AARAOj

## 2.41. @AACL

**Description:**

This command is used to clear the low latch value for all channels of a specified module.

**Syntax:**

**@AACL [CHKSUM](CR)**

| | |
|---|---|
| **@** | Delimiter character |
| **AA** | The address of the module to be cleared in hexadecimal format (00 to FF) |
| **CL** | The command to clear the low latch values |

**Response:**

| | |
|---|---|
| Valid Response: | **!AA[CHKSUM](CR)** |
| Invalid Response: | **?AA[CHKSUM](CR)** |

| | |
|---|---|
| **!** | Delimiter character for a valid response |
| **?** | Delimiter character for an invalid response |
| **AA** | The address of the responding module in hexadecimal format (00 to FF) |

There will be no response if the command syntax is incorrect, there is a communication error, or there is no module with the specified address.

**Examples:**

Command: @01RL0 Response: !01-05.000

Reads the low latch value for channel 0 of module 01 and returns a valid response with a value of -05.000 (-5 V).

Command: @01CL Response: !01

Clears the low latch value for all channels on module 01 and returns a valid response.

Command: @01RL0 Response: !01+00.000

Reads the low latch value for channel 0 of module 01 and returns a valid response with a value of +00.000 (0 V).

**Related Commands:**

Section 2.42 @AACLi, Section 2.55 @AARL, Section 2.56 @AARLi

## 2.42. @AACLi

**Description:**
This command is used to clear the low latch value for a specific channel of specified module.

**Syntax:**
**@AACLi [CHKSUM](CR)**
**@** Delimiter character
**AA** The address of the module to be cleared in hexadecimal format (00 to FF)
**CL** The command to clear the low latch value
**i** The channel to be cleared, zero based

**Response:**
Valid Response: **!AA[CHKSUM](CR)**
Invalid Response: **?AA[CHKSUM](CR)**
**!** Delimiter character for a valid response
**?** Delimiter character for an invalid response
**AA** The address of the responding module in hexadecimal format (00 to FF)

There will be no response if the command syntax is incorrect, there is a communication error, or there is no module with the specified address.

**Examples:**

Command: @01RL1 Response: !01-06.000

Reads the low latch value for channel 1 of module 01 and returns a valid response with a value of -06.000 (-6 V).

Command: @01CL1 Response: !01

Clears the low latch value for channel 1 of module 01 and returns a valid response.

Command: @01RL1 Response: !01+00.000

Reads the low latch value for channel 1 of module 01 and returns a valid response with a value of +00.000 (0 V).

**Related Commands:**

Section 2.41 @AACL, Section 2.55 @AARL, Section 2.56 @AARLi

## 2.43. @AACLCi

**Description:**

This command is used to clear the status of the low alarm for a specific channel of a specified module.

**Syntax:**

**@AACLCi [CHKSUM](CR)**

**@** Delimiter character

**AA** The address of the module to be cleared in hexadecimal format (00 to FF)

**CL** The command to clear the status of the low alarm

**Ci** i specifies the channel to be cleared, zero based

**Response:**

Valid Response: **!AA[CHKSUM](CR)**

Invalid Response: **?AA[CHKSUM](CR)**

**!** Delimiter character for a valid response

**?** Delimiter character for an invalid response

**AA** The address of the responding module in hexadecimal format (00 to FF)

There will be no response if the command syntax is incorrect, there is a communication error, or there is no module with the specified address.

**Examples:**

Command: @01CHC7 Response: !01

Clears the low alarm status for channel 7 of module 01 and returns a valid response.

Command: @01CHCF Response: !01

Attempts to clears the status of the low alarm for channel 15 of module 01 and returns an invalid response because channel 15 does not exist.

**Related Commands:**

Section 2.49 @AALO(Data)Ci, Section 2.50 @AARAOj

## 2.44. @AADA

**Description:**

This command is used to disable the alarm function of a specified module.

**Syntax:**

**@AADA [CHKSUM](CR)**

| | |
|---|---|
| **@** | Delimiter character |
| **AA** | The address of the module to be set in hexadecimal format (00 to FF) |
| **DA** | The command to disable the alarm function |

**Response:**

| | |
|---|---|
| Valid Response: | **!AA[CHKSUM](CR)** |
| Invalid Response: | **?AA[CHKSUM](CR)** |

| | |
|---|---|
| **!** | Delimiter character for a valid response |
| **?** | Delimiter character for an invalid response |
| **AA** | The address of the responding module in hexadecimal format (00 to FF) |

There will be no response if the command syntax is incorrect, there is a communication error, or there is no module with the specified address.

**Examples:**

Command: @01EAM Response: !01

Enables the momentary alarm function of module 01 and returns a valid response.

Command: @01DA Response: !01

Disables the alarm function of module 01 and returns a valid response.

**Related Commands:**

Section 2.47 @AAEAt, Section 2.48 @AAHI(Data)Ci, Section 2.49 @AALO(Data)Ci, Section 2.50 @AARAO, Section 2.54 @AARHCi, Section 2.57 @AARLCi

## 2.45. @AADI

**Description:**

This command is used to read the status of the digital input and digital output channels of a specified module.

**Syntax:**

**@AADI [CHKSUM](CR)**

| | |
|---|---|
| **@** | Delimiter character |
| **AA** | The address of the module to be read in hexadecimal format (00 to FF) |
| **DI** | The command to read the status of the DI/DO channels |

**Response:**

| | |
|---|---|
| Valid Response: | **!AASHHLL[CHKSUM](CR)** |
| Invalid Response: | **?AA[CHKSUM](CR)** |

| | |
|---|---|
| **!** | Delimiter character for a valid response |
| **?** | Delimiter character for an invalid response |
| **AA** | The address of the responding module in hexadecimal format (00 to FF) |
| **S** | The type of alarm<br>0: Alarm function is disabled<br>1: Momentary alarm<br>2: Latched alarm |
| **HH** | A two-digit hexadecimal value to denote the DO status, where bit 0 corresponds to channel 0, bit 1 corresponds to channel 1, etc. When the bit is 0, it denotes that the DO channel is off, and 1 denotes that the DO channel is on. |

**LL** A two-digit hexadecimal value to denote the DI status, where bit 0 corresponds to channel 0, bit 1 corresponds to channel 1, etc. When the bit is 0, it denotes that the DI channel is off, and 1 denotes that the DI channel is on.

There will be no response if the command syntax is incorrect, there is a communication error, or there is no module with the specified address.

**Examples:**

Command: @01DO01 Response: !01

Sets digital output channel 0 of module 01 to on, and sets digital output channel 1, 2 and 3 of module 01 to off and returns a valid response.

Command: @01DI Response: !0100102

Reads the status of the DI/DO channels of module 01 and returns a response of 00102 indicating that the alarm function is disabled, DO0 is on, DO1, DO2 and DO3 are off, DI1 is on and DI0, DI2, DI3 and DI4 are off.

**Related Commands:**

Section 2.48 @AAHI(Data)Ci, Section 2.49 @AALO(Data)Ci, Section 2.46 @AADODD

## 2.46. @AADODD

**Description:**

This command is used to set the status of the digital output for a specified module.

**Syntax:**

**@AADODD[CHKSUM](CR)**

**@** Delimiter character

**AA** The address of the module to be set in hexadecimal format (00 to FF)

**DO** The command to set the digital output ports

**DD** A two-digit hexadecimal value that represent the DO status, where bit 0 corresponds to DO channel 0, bit 1 corresponds to DO channel 1, etc. When the bit is 0, it denotes that the digital output port is off, and 1 denotes that the digital output port is on.

**Response:**

Valid Response: **!AA[CHKSUM](CR)**

Invalid Response: **?AA[CHKSUM](CR)**

**!** Delimiter character for a valid response

**?** Delimiter character for an invalid response

**AA** The address of the responding module in hexadecimal format (00 to FF)

There will be no response if the command syntax is incorrect, there is a communication error, or there is no module with the specified address.

**Notes:**

**1.** If the digital output port is already set as an alarm output port, the value written to the port will be ignored.

**2.** If a Host Watchdog timeout occurs, the module will return an invalid response for this command and the DO value that was sent will be ignored.

**Examples:**

Command: @01DO09 Response: !01

Sets digital output channel 0 and 3 of module 01 to on, and sets digital output channel 1 and 2 to off and returns a valid response.

Command: @01DI Response: !0100902

Reads the status of the DI/DO channels of module 01 and returns a response of 00902 indicating that the alarm function is disabled, DO0 and DO3 are on, DO1 and DO2 are off, DI1 is on and DI0, DI2, DI3 and DI4 are off.

**Related Commands:**

Section 2.45 @AADI

## 2.47. @AAEAt

**Description:**

This command is used to set the momentary or latch alarm function on a specified module.

**Syntax:**

**@AADA [CHKSUM](CR)**

**@** Delimiter character

**AA** The address of the module to be set in hexadecimal format (00 to FF)

**EA** The command to enable the alarm function

**t** The type of alarm
M: Momentary alarm
L: Latched alarm

**Response:**

Valid Response: **!AA[CHKSUM](CR)**

Invalid Response: **?AA[CHKSUM](CR)**

**!** Delimiter character for a valid response

**?** Delimiter character for an invalid response

**AA** The address of the responding module in hexadecimal format (00 to FF)

There will be no response if the command syntax is incorrect, there is a communication error, or there is no module with the specified address.

**Examples:**

Command: @01EAM Response: !01

Enables the momentary alarm function for module 01 and returns a valid response.

Command: @01DA Response: !01

Disables the alarm function for module 01 and returns a valid response.

**Related Commands:**

Section 2.44 @AADA, Section 2.48 @AAHI(Data)Ci, Section 2.49 @AALO(Data)Ci, Section 2.50 @AARAO, Section 2.54 @AARHCi, Section 2.57 @AARLCi

## 2.48. @AAHI(Data)Ci

**Description:**

This command is used to set the high alarm limit for a specific channel of a specified module.

**Syntax:**

**@AAHI(Data)Ci [CHKSUM](CR)**

**@** Delimiter character

**AA** The address of the module to be set in hexadecimal format (00 to FF)

**HI** The command to set the high alarm

**(Data)** The high alarm limit. This should be consistent with the data format. Refer to Section 1.7 for details.

**Ci** i specifies the channel to be set, zero based

**Response:**

Valid Response: **!AA[CHKSUM](CR)**

Invalid Response: **?AA[CHKSUM](CR)**

**!** Delimiter character for a valid response

**?** Delimiter character for an invalid response

**AA** The address of the responding module in hexadecimal format (00 to FF)

There will be no response if the command syntax is incorrect, there is a communication error, or there is no module with the specified address.

**Examples:**

Command: @01HI+09.000C0        Response: !01

Sets the high alarm limit for channel 0 of module 01 to +09.000 (9 V) and returns a valid respons.

Command: @01RHC0        Response: !01+09.000

Reads the high alarm limit for channel 0 of module 01 and returns a valid response with a value of +09.000 (9V).

Command: @01HI+09.000CF        Response: ?01

Attempts to set the high alarm limit for channel 15 of module 01 to +09.000 (9 V), but returns an invalid response because channel 15 does not exist.

**Related Commands:**

Section 2.40 @AACHCi, Section 2.50 @AARAOj, Section 2.54 @AARHCi

**Related Topics:**

Section 1.7 Configuration Tables

## 2.49. @AALO(Data)Ci

**Description:**
This command is used to set the low alarm limit for a specific channel of a specified module.

**Syntax:**
**@AALO(Data)Ci [CHKSUM](CR)**

| | |
|---|---|
| **@** | Delimiter character |
| **AA** | The address of the module to be set in hexadecimal format (00 to FF) |
| **LO** | The command to set the low alarm |
| **(Data)** | The low alarm limit. This should be consistent with the data format. Refer to Section 1.7 for details. |
| **Ci** | i specifies the channel to be set, zero based |

**Response:**

| | |
|---|---|
| Valid Response: | **!AA[CHKSUM](CR)** |
| Invalid Response: | **?AA[CHKSUM](CR)** |

| | |
|---|---|
| **!** | Delimiter character for a valid response |
| **?** | Delimiter character for an invalid response |
| **AA** | The address of the responding module in hexadecimal format (00 to FF) |

There will be no response if the command syntax is incorrect, there is a communication error, or there is no module with the specified address.

**Examples:**

Command: @01LO-03.000C1 Response: !01

Sets the low alarm limit for channel 1 of module 01 to -03.000 (-3 V) and returns a valid response.

Command: @01RLC1 Response: !010-03.000

Reads the low alarm limit for channel 1 of module 01 and returns a valid response with a value -03.000 (-3 V).

Command: @01LO-03.000CF Response: ?01

Attempts to set the low alarm limit for channel 15 of module 01 to -03.000 (-3 V), but returns an invalid response because channel 15 does not exist.

**Related Commands:**

Section 2.43 @AACLCi, Section 2.50 @AARAOj, Section 2.57 @AARLCi

**Related Topics:**

Section 1.7 Configuration Tables

## 2.50. @AARAO

**Description:**

This command is used to read which currently activated alarms are associated with a specified module.

**Syntax:**

**@AARAO[CHKSUM](CR)**

| | |
|---|---|
| **@** | Delimiter character |
| **AA** | The address of the module to be read in hexadecimal format (00 to FF) |
| **RAO** | The command to read the currently activated alarms associated with the module channel |

**Response:**

| | |
|---|---|
| Valid Response: | **!AAHHLL[CHKSUM](CR)** |
| Invalid Response: | **?AA[CHKSUM](CR)** |

| | |
|---|---|
| **!** | Delimiter character for a valid response |
| **?** | Delimiter character for an invalid response |
| **AA** | The address of the responding module in hexadecimal format (00 to FF) |
| **HH** | A two-digit hexadecimal value that represents the currently activated high alarms associated with the digital output channel, where bit 0 corresponds to channel 0, bit 1 corresponds to channel 1, etc. When the bit is 0, it denotes that there are no activated high alarms associated with the channel. When the bit is 1, it denotes that there is an activated high alarm associated with the channel. |

**LL** A two-digital hexadecimal value that represents the currently activated low alarms associated with the digital output channel, where bit 0 corresponds to channel 0, bit 1 corresponds to channel 1, etc. When the bit is 0, it denotes that there are no activated low alarms associated with the channel. When the bit is 1, it denotes that there is an activated low alarm associated with the channel.

There will be no response if the command syntax is incorrect, there is a communication error, or there is no module with the specified address.

**Examples:**

Command: @01RAO Response: !010102

Reads the currently activated alarms associated with module 01. The module returns a valid response with a value of 0102, which denotes that there is currently an activated high alarm associated with channel 0, and an activated low alarm associated with channel 1.

**Related Commands:**

Section 2.40 @AACHCi, Section 2.43 @AACLCi, Section 2.48 @AAHI(Data)Ci, Section 2.49 @AALO(Data)Ci

## 2.51. @AARECi

**Description:**

This command is used to read the value of the digital input counter for a specific channel of a specified module.

**Syntax:**

**@AARECi[CHKSUM](CR)**

| | |
|---|---|
| **@** | Delimiter character |
| **AA** | The address of the module to be read in hexadecimal format (00 to FF) |
| **RE** | The command to read the value of the digital input counter |
| **Ci** | i specifies the channel to be read, zero based |

**Response:**

| | |
|---|---|
| Valid Response: | **!AA(Data)[CHKSUM](CR)** |
| Invalid Response: | **?AA[CHKSUM](CR)** |

| | |
|---|---|
| **!** | Delimiter character for a valid response |
| **?** | Delimiter character for an invalid response |
| **AA** | The address of the responding module in hexadecimal format (00 to FF) |
| **(Data)** | The value of the digital input counter for the specified channel |

There will be no response if the command syntax is incorrect, there is a communication error, or there is no module with the specified address.

**Examples:**

Command: @01REC1 Response: !0100000008

Reads the counter data from channel 1 of module 01 and returns a valid response with a value of 00000008.

Command: @01CEC1 Response: !01

Resets the counter for channel 1 of module 01 to the preset value and returns a valid response.

Command: @01REC1 Response: !0100000000

Reads the counter data from channel 1 of module 01 and returns a valid response with a value of 00000000.

**Related Commands:**

Section 2.37 @AACECi

## 2.52. @AARH

**Description:**

This command is used to read the high latch values for all channels of a specified module.

**Syntax:**

**@AARH [CHKSUM](CR)**

**@** Delimiter character

**AA** The address of the module to be read in hexadecimal format (00 to FF)

**RH** The command to read the high latch values for all channels

**Response:**

Valid Response: **!AA(Data)[CHKSUM](CR)**

Invalid Response: **?AA[CHKSUM](CR)**

**!** Delimiter character for a valid response

**?** Delimiter character for an invalid response

**AA** The address of the responding module in hexadecimal format (00 to FF)

**(Data)** The high latch values for all channels. See Section 1.7 for details of the data format.

There will be no response if the command syntax is incorrect, there is a communication error, or there is no module with the specified address.

**Examples:**

Command: @01RH                    Response: !01+08.000+00.000+00.000+00.000

Reads the high latch values from all channels of module 01 and returns a valid response with the data in engineering format.

Command: @01CH                    Response: !01

Clears the high latch values for all channels of module 01 and returns a valid response.

Command: @01RH                    Response: !01+00.000+00.000+00.000+00.000

Reads the high latch values from all channels of module 01 and returns a valid response with the data in engineering format.

**Related Commands:**

Section 2.38 @AACH, Section 2.39 @AACHi, Section 2.53 @AARHi

**Related Topics:**

Section 1.7 Configuration Tables

## 2.53. @AARHi

**Description:**
This command is used to read the high latch value for a specific channel of a specified module.

**Syntax:**
**@AARHi [CHKSUM](CR)**

| | |
|---|---|
| **@** | Delimiter character |
| **AA** | The address of the module to be read in hexadecimal format (00 to FF) |
| **RH** | The command to read the high latch value |
| **i** | The channel to be read, zero based |

**Response:**

| | |
|---|---|
| Valid Response: | **!AA(Data)[CHKSUM](CR)** |
| Invalid Response: | **?AA[CHKSUM](CR)** |

| | |
|---|---|
| **!** | Delimiter character for a valid response |
| **?** | Delimiter character for an invalid response |
| **AA** | The address of the responding module in hexadecimal format (00 to FF) |
| **(Data)** | The high latch value for the specified channel. See Section 1.7 for details of the data format. |

There will be no response if the command syntax is incorrect, there is a communication error, or there is no module with the specified address.

**Examples:**

Command: @01RH0 Response: !01+08.000

Reads the high latch value for channel 0 of module 01 and returns a valid response with a value of +08.000 (8 V) in engineering format.

Command: @01CH Response: !01

Clears the high latch value for all channels of module 01 and returns a valid response.

Command: @01RH0 Response: !01+00.000

Reads the high latch value for channel 0 of module 01 and returns a valid response with a value of +00.000 (0 V) in engineering format.

Command: @01RHF Response: ?01

Attempts to read the high latch value for channel 15 of module 01, but returns an invalid response because channel 15 does not exist.

**Related Commands:**

Section 2.38 @AACH, Section 2.39 @AACHi, Section 2.52 @AARH

**Related Topics:**

Section 1.7 Configuration Tables

## 2.54. @AARHCi

**Description:**

This command is used to read the status of the high alarm for a specific channel of a specified module.

**Syntax:**

**@AARHCi [CHKSUM](CR)**

**@** Delimiter character

**AA** The address of the module to be read in hexadecimal format (00 to FF)

**RH** The command to read the status of the high alarm

**Ci** i specifies the channel to be read, zero based

**Response:**

Valid Response: **!AA(Data) [CHKSUM](CR)**

Invalid Response: **?AA[CHKSUM](CR)**

**!** Delimiter character for a valid response

**?** Delimiter character for an invalid response

**AA** The address of the responding module in hexadecimal format (00 to FF)

**(Data)** The status of the high alarm for the specified channel. See Section 1.7 for details of the data format.

There will be no response if the command syntax is incorrect, there is a communication error, or there is no module with the specified address.

**Examples:**

Command: @01HI+08.000C0 Response: !01

Sets the high alarm limit for channel 0 of module 01 to +08.000 (8 V) and returns a valid response.

Command: @01RHC0 Response: !01+08.000

Reads the high alarm limit for channel 0 of module 01 and returns a valid response indicating that the high alarm limit is +08.000 (8 V).

Command: @01RHCF Response: ?01

Attempts to read the high alarm limit for channel 15 of module 01, but returns an invalid response because channel 15 does not exist.

**Related Commands:**

Section 2.40 @AACHCi. Section 2.48 @AAHI(Data)Ci

**Related Topics:**

Section 1.7 Configuration Tables

## 2.55. @AARL

**Description:**

This command is used to read the low latch values for all channels of a specified module.

**Syntax:**

**@AARL [CHKSUM](CR)**

**@** Delimiter character

**AA** The address of the module to be read in hexadecimal format (00 to FF)

**RL** The command to read the low latch values for all channels

**Response:**

Valid Response: **!AA(Data)[CHKSUM](CR)**

Invalid Response: **?AA[CHKSUM](CR)**

**!** Delimiter character for a valid response

**?** Delimiter character for an invalid response

**AA** The address of the responding module in hexadecimal format (00 to FF)

**(Data)** The low latch values for all channels. See Section 1.7 for details of the data format.

There will be no response if the command syntax is incorrect, there is a communication error, or there is no module with the specified address.

**Examples:**

Command: @01RL Response:
!01-02.000+00.000+00.000+00.000

Reads the low latch values from all channels of module 01 and returns a valid response with the data in engineering format.

Command: @01CL Response: !01

Clears the low latch value for all channels of module 01 and returns a valid response.

Command: @01RH Response:
!01+00.000+00.000+00.000+00.000

Reads the low latch values from all channels of module 01 and returns a valid response with the data in engineering format.

**Related Commands:**

Section 2.41 @AACL, Section 2.42 @AACLi, Section 2.56 @AARLi

**Related Topics:**

Section 1.7 Configuration Tables

## 2.56. @AARLi

**Description:**
This command is used to read the low latch value for a specific channel of a specified module.

**Syntax:**
**@AARLi [CHKSUM](CR)**

| | |
|---|---|
| **@** | Delimiter character |
| **AA** | The address of the module to be read in hexadecimal format (00 to FF) |
| **RL** | The command to read the low latch value |
| **i** | The channel to be read, zero based |

**Response:**

| | |
|---|---|
| Valid Response: | **!AA(Data)[CHKSUM](CR)** |
| Invalid Response: | **?AA[CHKSUM](CR)** |

| | |
|---|---|
| **!** | Delimiter character for a valid response |
| **?** | Delimiter character for an invalid response |
| **AA** | The address of the responding module in hexadecimal format (00 to FF) |
| **(Data)** | The high latch value for the specified channel. See Section 1.7 for details of the data format. |

There will be no response if the command syntax is incorrect, there is a communication error, or there is no module with the specified address.

**Example:**

Command: @01RL0 Response: !01-02.000

Reads the low latch value for channel 0 of module 01 and returns a valid response with a value of -02.000 (-2 V) in engineering format.

Command: @01RLF Response: ?01

Attempts to read the low latch value for channel 15 of module 01, but returns an invalid response becausechannel 15 does not exist.

**Related Commands:**

Section 2.41 @AACL, Section 2.42 @AACLi, Section 2.55 @AARL

**Related Topics:**

Section 1.7 Configuration Tables

## 2.57. @AARLCi

**Description:**

This command is used to read the status of the low alarm for a specific channel of a specified module.

**Syntax:**

**@AARLCi [CHKSUM](CR)**

| | |
|---|---|
| **@** | Delimiter character |
| **AA** | The address of the module to be read in hexadecimal format (00 to FF) |
| **RL** | The command to read the status of the low alarm |
| **Ci** | i specifies the channel to be read, zero based |

**Response:**

| | |
|---|---|
| Valid Response: | **!AA(Data) [CHKSUM](CR)** |
| Invalid Response: | **?AA[CHKSUM](CR)** |

| | |
|---|---|
| **!** | Delimiter character for a valid response |
| **?** | Delimiter character for an invalid response |
| **AA** | The address of the responding module in hexadecimal format (00 to FF) |
| **(Data)** | The status of the low alarm for the specified channel. See Section 1.7 for details of the data format. |

There will be no response if the command syntax is incorrect, there is a communication error, or there is no module with the specified address.

**Example:**

Command: @01LO-03.000C0 Response: !01

Sets the low alarm limit for channel 0 of module 01 to -03.000 (-3 V) and returns a valid response.

Command: @01RLC0 Response: !01-03.000

Reads the status of the low alarm for channel 0 of module 01 and returns a valid response indicating that the low alarm limit is -03.000 (-3 V), the type is momentary and the low alarm output channel is digital output channel 1.

Command: @01RLCF Response: ?01

Attempts to read the status of the low alarm for channel 15 of module 01, but returns an invalid response because channel 15 does not exist.

**Related Commands:**

Section 2.43 @AACLCi, Section 2.49 @AALO(Data)Ci

**Related Topics:**

Section 1.7 Configuration Tables

## 2.58. ~AADT

**Description:**
This command is used to read the setting of counting on overflow.

**Syntax:**
**~AADT[CHKSUM](CR)**

| | |
|---|---|
| ~ | Delimiter character |
| **AA** | The address of the module to be read in hexadecimal format (00 to FF) |
| **DT** | The command to read the counting on overflow setting |

**Response:**

| | |
|---|---|
| Valid Response: | **!AAE[CHKSUM](CR)** |
| Invalid Response: | **?AA[CHKSUM](CR)** |

| | |
|---|---|
| **!** | Delimiter character for a valid response |
| **?** | Delimiter character for an invalid response |
| **AA** | The address of the responding module in hexadecimal format (00 to FF) |
| **E** | 0 for disable and 1 for enable. |

There will be no response if the command syntax is incorrect, there is a communication error, or there is no module with the specified address.

**Examples:**

| | |
|---|---|
| Command: ~01DT1 | Response: !01 |

Sets the counting on overflow setting to enable..

| | |
|---|---|
| Command: ~01DT | Response: !011 |

Reads the counting on overflow setting and returns a valid response with a value of 1 indicating the setting is enable.

**Related Commands:**

Section 2.59 ~AADTE

## 2.59. ~AADTE

**Description:**

This command is used to set the counting on overflow setting for a specified module.

**Syntax:**

**~AARDTT[CHKSUM](CR)**

| | |
|---|---|
| ~ | Delimiter character |
| **AA** | The address of the module to be set in hexadecimal format (00 to FF) |
| **DT** | The command to set the counting on overflow setting |
| **E** | 0 for disable and 1 for enable. |

**Response:**

| | |
|---|---|
| Valid Response: | **!AA [CHKSUM](CR)** |
| Invalid Response: | **?AA[CHKSUM](CR)** |

| | |
|---|---|
| **!** | Delimiter character for a valid response |
| **?** | Delimiter character for an invalid response |
| **AA** | The address of the responding module in hexadecimal format (00 to FF) |

There will be no response if the command syntax is incorrect, there is a communication error, or there is no module with the specified address.

**Examples:**

Command: ~01DT1 Response: !01

Sets the counting on overflow setting to enable..

Command: ~01DT Response: !011

Reads the counting on overflow setting and returns a valid response with a value of 1 indicating the setting is enable.

**Related Commands:**

Section 2.58 ~AADT

# 3. Modbus RTU Protocol

The Modbus protocol was developed by Modicon Inc., and was originally developed for Modicon controllers. Detailed information can be found at http://www.modicon.com/techpubs/toc7.html. You can also visit http://www.modbus.org to find more valuable information.

| Function code | Description | Section |
|---|---|---|
| 02 (0x02) | Reads the digital input status | 3.1 |
| 04 (0x04) | Reads the analog input channels | 3.2 |
| 05 (0x05) | Writes a single digital output | 3.3 |
| 70 (0x46) | Reads/writes the module settings | 3.4 |

## Error Responses

| | | | |
|---|---|---|---|
| 00 | Address | 1 Byte | 1 to 247 |
| 01 | Function code | 1 Byte | Function code \| 0x80 |
| 02 | Exception code | 1 Byte | 01 |

**Note: If a CRC mismatch occurs, the module will not respond.**

## 3.1. Function 02 (0x02)-Read the Digital Input Status

This function code is used to read the status of the wire connection for a module. (Supports types 0x7 and 0x1A only)

**Request**

| | | | |
|---|---|---|---|
| 00 | Address | 1 Byte | 1 to 247 |
| 01 | Function code | 1 Byte | 0x02 |
| 02 ~ 03 | Starting channel | 2 Bytes | 0x20 to 0x24, where 0x20 corresponds to channel 0, 0x21 corresponds to channel 1, etc. |
| 04 ~ 05 | Number of input channels (N) | 2 Bytes | N, 1 to 5 (Starting channel + N) |

**Response**

| | | | |
|---|---|---|---|
| 00 | Address | 1 Byte | 1 to 247 |
| 01 | Function code | 1 Byte | 0x02 |
| 02 | Byte count | 1 Byte | 1 |
| 03 | Data from input channels | 1 Byte | A bit corresponds to a channel. When the bit is 1, it denotes that the channel is either over-range or under-range. If the bit is 0, it denotes that the channel is normal. |

**Error Response**

| | | | |
|---|---|---|---|
| 00 | Address | 1 Byte | 1 to 247 |
| 01 | Function code | 1 Byte | 0x82 |
| 02 | Exception code | 1 Byte | 03: (the starting channel + the number of input channels) is out of range, or an incorrect number of bytes were received. |

## 3.2. Function 04 (0x04)-Read the Analog Input Channels

This function code is used to read from contiguous analog input channels.

**Request**

| | | | |
|---|---|---|---|
| 00 | Address | 1 Byte | 1 to 247 |
| 01 | Function code | 1 Byte | 0x04 |
| 02 ~ 03 | Starting channel | 2 Bytes | 0 to 3 |
| 04 ~ 05 | Number of input channels (N) | 2 Bytes | N, 1 to 4 (Starting channel + N) |

**Response**

| | | | |
|---|---|---|---|
| 00 | Address | 1 Byte | 1 to 247 |
| 01 | Function code | 1 Byte | 0x04 |
| 02 | Byte count | 1 Byte | 2 x N |
| 03 ~ | Data from input channels | 2 x N Bytes | Data is in either 2's complement hex format or engineering format. |

**Error Response**

| | | | |
|---|---|---|---|
| 00 | Address | 1 Byte | 1 to 247 |
| 01 | Function code | 1 Byte | 0x84 |
| 02 | Exception code | 1 Byte | 03: (The starting channel + the number of input channels) is out of range, or an incorrect number of bytes were received. |

## 3.3. Function 05 (0x05)-Write a Single Digital Output

This function code is used to write from contiguous digital output channels.

**Request**

| | | | |
|---|---|---|---|
| 00 | Address | 1 Byte | 1 to 247 |
| 01 | Function code | 1 Byte | 0x05 |
| 02 ~ 03 | Starting channel | 2 Bytes | 0 to 3 |
| 04 ~ 05 | Data | 2 Bytes | FF00h for ON and 0000h for OFF |

**Response**

| | | | |
|---|---|---|---|
| 00 | Address | 1 Byte | 1 to 247 |
| 01 | Function code | 1 Byte | 0x04 |
| 02 ~ 03 | Byte count | 1 Byte | 0 to 3 |
| 04 ~ 05 | Data | 2 Bytes | FF00h for ON and 0000h for OFF |

**Error Response**

| | | | |
|---|---|---|---|
| 00 | Address | 1 Byte | 1 to 247 |
| 01 | Function code | 1 Byte | 0x85 |
| 02 | Exception code | 1 Byte | 03: (The starting channel + the number of input channels) is out of range, or an incorrect number of bytes were received. |

## 3.4. Function 70 (0x46)-Read/Write Module Settings

This function code is used to either read or change the settings of the module. The following sub-function codes are supported.

| Sub-function Code | Description | Section |
|---|---|---|
| 00 (0x00) | Reads the name of the module | 3.4.1 |
| 04 (0x04) | Sets the address | 3.4.2 |
| 05 (0x05) | Reads the communication settings | 3.4.3 |
| 06 (0x06) | Sets the communication settings | 3.4.4 |
| 07 (0x07) | Reads the type code | 3.4.5 |
| 08 (0x08) | Sets the type code | 3.4.6 |
| 32 (0x20) | Reads the firmware version information | 3.4.7 |
| 37 (0x25) | Reads the enabled/disabled status of all channels | 3.4.8 |
| 38 (0x26) | Sets the channel to either enabled or disabled | 3.4.9 |
| 41 (0x29) | Reads the miscellaneous settings | 3.4.10 |
| 42 (0x2A) | Writes the miscellaneous settings | 3.4.11 |

If the module does not support the sub-function code specified in the message, then it will respond as follows:

**Error Response**

| | | | |
|---|---|---|---|
| 00 | Address | 1 Byte | 1 to 247 |
| 01 | Function code | 1 Byte | 0xC6 |
| 02 | Exception code | 1 Byte | 02: Indicates an invalid sub-function code |

### 3.4.1 Sub-function 00 (0x00)-Read the name of the module

This sub-function code is used to read the name of a module.

**Request**

| | | | |
|---|---|---|---|
| 00 | Address | 1 Byte | 1 to 247 |
| 01 | Function code | 1 Byte | 0x46 |
| 02 | Sub-function code | 1 Byte | 0x00 |

**Response**

| | | | |
|---|---|---|---|
| 00 | Address | 1 Byte | 1 to 247 |
| 01 | Function code | 1 Byte | 0x46 |
| 02 | Sub-function code | 1 Byte | 0x00 |
| 03 ~ 06 | Module name | 4 Bytes | 0x00 0x70 0x02 0x00 |

**Error Response**

| | | | |
|---|---|---|---|
| 00 | Address | 1 Byte | 1 to 247 |
| 01 | Function code | 1 Byte | 0xC6 |
| 02 | Exception code | 1 Byte | 03: An incorrect number of bytes were received |

### 3.4.2 Sub-function 04 (0x04)-Set the module address

This sub-function code is used to set the address of a module.

**Request**

| | | | |
|---|---|---|---|
| 00 | Address | 1 Byte | 1 to 247 |
| 01 | Function code | 1 Byte | 0x46 |
| 02 | Sub-function code | 1 Byte | 0x04 |
| 02 | New address | 1 Byte | 1 to 247 |
| 04 ~ 06 | Reserved | 3 Bytes | 0x00 0x00 0x00 |

**Response**

| | | | |
|---|---|---|---|
| 00 | Address | 1 Byte | 1 to 247 |
| 01 | Function code | 1 Byte | 0x46 |
| 02 | Sub-function code | 1 Byte | 0x04 |
| 03 | Set address result | 1 Byte | 0: OK, others: error |
| 04 ~ 06 | Reserved | 3 Bytes | 0x00 0x00 0x00 |

**Error Response**

| | | | |
|---|---|---|---|
| 00 | Address | 1 Byte | 1 to 247 |
| 01 | Function code | 1 Byte | 0xC6 |
| 02 | Exception code | 1 Byte | 03: The new address is out of range, reserved bits should be filled with zero, or an incorrect number of bytes were received |

### 3.4.3 Sub-function 05 (0x05)-Read the communication settings

This sub-function code is used to read the communication protocol settings for a module.

**Request**

| | | | |
|---|---|---|---|
| 00 | Address | 1 Byte | 1 to 247 |
| 01 | Function code | 1 Byte | 0x46 |
| 02 | Sub-function code | 1 Byte | 0x05 |
| 03 | Reserved | 1 Byte | 0x00 |

**Response**

| | | | |
|---|---|---|---|
| 00 | Address | 1 Byte | 1 to 247 |
| 01 | Function code | 1 Byte | 0x46 |
| 02 | Sub-function code | 1 Byte | 0x05 |
| 03 | Reserved | 1 Byte | 0x00 |
| 04 | Baud Rate | 1 Byte | Baud Rate code, see Section 1.7 for details. |
| 05 ~ 07 | Reserved | 3 Bytes | 0x00 0x00 0x00 |
| 08 | Mode | 1 Byte | 0: DCON protocol<br>1: Modbus RTU protocol |
| 09 ~ 10 | Reserved | 2 Bytes | 0x00 0x00 |

**Error Response**

| | | | |
|---|---|---|---|
| 00 | Address | 1 Byte | 1 to 247 |
| 01 | Function code | 1 Byte | 0xC6 |
| 02 | Exception code | 1 Byte | 03: Reserved bits should filled with zero, or an incorrect number of bytes were received |

### 3.4.4 Sub-function 06 (0x06)-Set the communication settings

This sub-function code is used to set the communication protocol settings for a module.

**Request**

| | | | |
|---|---|---|---|
| 00 | Address | 1 Byte | 1 to 247 |
| 01 | Function code | 1 Byte | 0x46 |
| 02 | Sub-function code | 1 Byte | 0x06 |
| 03 | Reserved | 1 Byte | 0x00 |
| 04 | Baud Rate | 1 Byte | Baud Rate code, see Section 1.7 for details. |
| 05 ~ 07 | Reserved | 3 Bytes | 0x00 0x00 0x00 |
| 08 | Mode | 1 Byte | 0: DCON protocol<br>1: Modbus RTU protocol |
| 09 ~ 10 | Reserved | 2 Bytes | 0x00 0x00 |
| 11 | Reserved | 1 Bytes | 0x00 |

**Response**

| | | | |
|---|---|---|---|
| 00 | Address | 1 Byte | 1 to 247 |
| 01 | Function code | 1 Byte | 0x46 |
| 02 | Sub-function code | 1 Byte | 0x06 |
| 03 | Reserved | 1 Byte | 0x00 |
| 04 | Baud Rate | 1 Byte | 0: OK, others: error |
| 05 ~ 07 | Reserved | 3 Bytes | 0x00 0x00 0x00 |
| 08 | Mode | 1 Byte | 0: OK, others: error |
| 09 ~ 10 | Reserved | 2 Bytes | 0x00 0x00 |

**Error Response**

| | | | |
|---|---|---|---|
| 00 | Address | 1 Byte | 1 to 247 |
| 01 | Function code | 1 Byte | 0xC6 |
| 02 | Exception code | 1 Byte | 03: The Baud Rate or the module address is out of range, reserved bits should be filled with zero, or an incorrect number of bytes were received |

### 3.4.5 Sub-function 07 (0x07)-Read the type code

This sub-function code is used to read the type code information for a module.

**Request**

| | | | |
|---|---|---|---|
| 00 | Address | 1 Byte | 1 to 247 |
| 01 | Function code | 1 Byte | 0x46 |
| 02 | Sub-function code | 1 Byte | 0x07 |
| 03 | Reserved | 1 Byte | 0x00 |
| 04 | Channel | 1 Byte | 0x00 ~ 0x03 |

**Response**

| | | | |
|---|---|---|---|
| 00 | Address | 1 Byte | 1 to 247 |
| 01 | Function code | 1 Byte | 0x46 |
| 02 | Sub-function code | 1 Byte | 0x07 |
| 03 | Type code | 1 Byte | The type code, see Section 1.7 for details. |

**Error Response**

| | | | |
|---|---|---|---|
| 00 | Address | 1 Byte | 1 to 247 |
| 01 | Function code | 1 Byte | 0xC6 |
| 02 | Exception code | 1 Byte | 03: The channel number is out of range reserved bits should be filled with zero, or an incorrect number of bytes were received |

### 3.4.6 Sub-function 08 (0x08)-Set the type code

This sub-function code is used to set the type code for a module.

**Request**

| | | | |
|---|---|---|---|
| 00 | Address | 1 Byte | 1 to 247 |
| 01 | Function code | 1 Byte | 0x46 |
| 02 | Sub-function code | 1 Byte | 0x08 |
| 03 | Reserved | 1 Byte | 0x00 |
| 04 | Channel | 1 Byte | 0x00 ~ 0x03 |
| 05 | Type code | 1 Byte | The type code, see Section 1.7 for details. |

**Response**

| | | | |
|---|---|---|---|
| 00 | Address | 1 Byte | 1 to 247 |
| 01 | Function code | 1 Byte | 0x46 |
| 02 | Sub-function code | 1 Byte | 0x08 |
| 03 | Type code | 1 Byte | 0: OK, others: error |

**Error Response**

| | | | |
|---|---|---|---|
| 00 | Address | 1 Byte | 1 to 247 |
| 01 | Function code | 1 Byte | 0xC6 |
| 02 | Exception code | 1 Byte | 03: The type code is out of range, the channel is out of range, reserved bits should be filled with zero, or an incorrect number of bytes were received |

### 3.4.7 Sub-function 32 (0x20)-Read the firmware version information

This sub-function code is used to read the firmware version information for a module.

**Request**

| | | | |
|---|---|---|---|
| 00 | Address | 1 Byte | 1 to 247 |
| 01 | Function code | 1 Byte | 0x46 |
| 02 | Sub-function code | 1 Byte | 0x20 |

**Response**

| | | | |
|---|---|---|---|
| 00 | Address | 1 Byte | 1 to 247 |
| 01 | Function code | 1 Byte | 0x46 |
| 02 | Sub-function code | 1 Byte | 0x20 |
| 03 | Major version | 1 Byte | 0x00 ~ 0xFF |
| 04 | Minor version | 1 Byte | 0x00 ~ 0xFF |
| 05 | Reserved | 1 Byte | 0x00 |
| 06 | Build version | 1 Byte | 0x00 ~ 0xFF |

**Error Response**

| | | | |
|---|---|---|---|
| 00 | Address | 1 Byte | 1 to 247 |
| 01 | Function code | 1 Byte | 0xC6 |
| 02 | Exception code | 1 Byte | 03: An incorrect number of bytes were received, reserved bits should be filled with zero |

## 3.4.8 Sub-function 37 (0x25)-Read the channel enabled/disabled status

This sub-function code is used to read whether each channel of a module is enabled or disabled.

**Request**

| | | | |
|---|---|---|---|
| 00 | Address | 1 Byte | 1 to 247 |
| 01 | Function code | 1 Byte | 0x46 |
| 02 | Sub-function code | 1 Byte | 0x25 |

**Response**

| | | | |
|---|---|---|---|
| 00 | Address | 1 Byte | 1 to 247 |
| 01 | Function code | 1 Byte | 0x46 |
| 02 | Sub-function code | 1 Byte | 0x25 |
| 03 | Enabled/disabled status | 1 Byte | 0x00 ~ 0x0F, the enabled/disabled status of each channel, where bit 0 corresponds to channel 0, bit 1 corresponds to channel 1, etc. When the bit is 0, it denotes that the channel is disabled and 1 denotes that the channel is enabled. |

**Error Response**

| | | | |
|---|---|---|---|
| 00 | Address | 1 Byte | 1 to 247 |
| 01 | Function code | 1 Byte | 0xC6 |
| 02 | Exception code | 1 Byte | 03: An incorrect number of bytes were received |

## 3.4.9 Sub-function 38 (0x26)-Set the channel to either enabled or disabled

This sub-function code is used to specify which channels of a module are to be enabled or disabled.

**Request**

| | | | |
|---|---|---|---|
| 00 | Address | 1 Byte | 1 to 247 |
| 01 | Function code | 1 Byte | 0x46 |
| 02 | Sub-function code | 1 Byte | 0x26 |
| 03 | Enabled/disabled settings | 1 Byte | 0x00 ~ 0x0F, the enabled/disabled settings for each channel, where bit 0 corresponds to channel 0, bit 1 corresponds to channel 1, etc. When the bit is 0, it denotes that the channel is to be disabled and 1 denotes that the channel is to be enabled. |

**Response**

| | | | |
|---|---|---|---|
| 00 | Address | 1 Byte | 1 to 247 |
| 01 | Function code | 1 Byte | 0x46 |
| 02 | Sub function code | 1 Byte | 0x26 |
| 03 | Enabled/disabled settings | 1 Byte | 0: OK, others: error. |

**Error Response**

| | | | |
|---|---|---|---|
| 00 | Address | 1 Byte | 1 to 247 |
| 01 | Function code | 1 Byte | 0xC6 |
| 02 | Exception code | 1 Byte | 03: The enabled/disabled settings are out of range, or an incorrect number of bytes were received |

### 3.4.10 Sub-function 41 (0x29)-Read the miscellaneous settings

This sub-function code is used to read the miscellaneous settings for a module.

**Request**

| | | | |
|---|---|---|---|
| 00 | Address | 1 Byte | 1 to 247 |
| 01 | Function code | 1 Byte | 0x46 |
| 02 | Sub-function code | 1 Byte | 0x29 |

**Response**

| | | | |
|---|---|---|---|
| 00 | Address | 1 Byte | 1 to 247 |
| 01 | Function code | 1 Byte | 0x46 |
| 02 | Sub-function code | 1 Byte | 0x29 |
| 03 | Miscellaneous settings | 1 Byte | Bit 7: Filter settings<br>0: 60 Hz rejection<br>1: 50 Hz rejection<br>Bit 6: Reserved<br>Bit 5: Mode settings<br>0: Normal mode<br>1: Fast mode<br>Bits 4 ~ 0: Reserved |

**Error Response**

| | | | |
|---|---|---|---|
| 00 | Address | 1 Byte | 1 to 247 |
| 01 | Function code | 1 Byte | 0xC6 |
| 02 | Exception code | 1 Byte | 03: An incorrect number of bytes were received |

## 3.4.11 Sub-function 42 (0x2A)-Write the miscellaneous settings

This sub-function code is used to set the miscellaneous settings for a module.

**Request**

| | | | |
|---|---|---|---|
| 00 | Address | 1 Byte | 1 to 247 |
| 01 | Function code | 1 Byte | 0x46 |
| 02 | Sub-function code | 1 Byte | 0x2A |
| 03 | Miscellaneous settings | 1 Byte | Bit 7: Filter settings<br>0: 60 Hz rejection<br>1: 50 Hz rejection<br>Bit 6: Reserved<br>Bit 5: Mode settings<br>0: Normal mode<br>1: Fast mode<br>Bits 4 ~ 0: Reserved |

**Response**

| | | | |
|---|---|---|---|
| 00 | Address | 1 Byte | 1 to 247 |
| 01 | Function code | 1 Byte | 0x46 |
| 02 | Sub-function code | 1 Byte | 0x2A |
| 03 | Miscellaneous settings | 1 Byte | 0: OK, others: error |

**Error Response**

| | | | |
|---|---|---|---|
| 00 | Address | 1 Byte | 1 to 247 |
| 01 | Function code | 1 Byte | 0xC6 |
| 02 | Exception code | 1 Byte | 03: Reserved bits should be filled with zero, or an incorrect number of bytes were received |

## 3.5. Address Mappings

The address mappings are as follows:

| Address | Description | Attribute |
|---|---|---|
| 00001 ~ 00004 | Digital output | R/W |
| 00065 ~ 00069 | Digital input latch high | R/W |
| 00073 ~ 00076 | Digital output latch high | R/W |
| 00097~ 00101 | Digital input latch low | R/W |
| 00105 ~ 00108 | Digital output latch low | R/W |
| 00129 ~ 00132 | Digital output safe value | R/W |
| 00193 ~ 00196 | Digital output power-on value | R/W |
| 00257 | Communication protocol<br>0: DCON, 1:Modbus RTU | R/W |
| 00259 | Filter settings, 0: 60 Hz rejection, 1: 50 Hz rejection | R/W |
| 00260 | Host Watchdog mode<br>0: The same as I-7000 series module<br>1: AO and DO commands can be used to clear the Host Watchdog timeout status | R/W |
| 00261 | Enables or disables the Host Watchdog<br>0: disabled, 1: enabled | R/W |
| 00262 | Enables or disables the alarm function<br>0: disabled, 1: enabled | R/W |
| 00263 | Alarm mode<br>0: momentary, 1: latch | R/W |
| 00264 | Clear DI/O latch, write 1 to clear | W |
| 00265 | Clear all DI counters, write 1 to clear | W |
| 00269 | Modbus data format<br>0: hexadecimal<br>1: engineering | R/W |
| 00270 | Host Watchdog timeout status. Write 1 to clear the Host Watchdog timeout | R/W |

|  |  |  |
|---|---|---|
|  | status |  |
| 00271 | AI filter format<br>0: normal, 1: fast | R/W |
| 00272 | Write 1 to load the factory calibration parameters | W |
| 00273 | Reset status<br>0: not the first the status has been read after being powered on<br>1: the first time the status has been read after being powered on | R |
| 00274 | Count on overflow<br>0: disable<br>1: enable | R/W |
| 00289 ~ 00292 | Low alarm status of channels 0 to 3, write 1 to clear | R/W |
| 00305 ~ 00308 | High alarm status of channels 0 to 3, write 1 to clear | R/W |
| 00513 ~ 00517 | Digital input counter channel 0 to 4, write 1 to clear | W |
| 10033 ~ 10037 | Digital input status for channels 0 to 4 | R |
| 30001 ~ 30004 | Analog input value for channels 0 to 3 | R |
| 30097 ~ 30101 | Digital input counter for channels 0 to 4 | R |
| 40225 ~ 40228 | High alarm value | R/W |
| 40233 ~ 40236 | Low alarm value | R/W |
| 40257 ~ 40260 | Type code for channels 0 to 3 | R/W |
| 40481 | Firmware version (low word) | R |
| 40482 | Firmware version (high word) | R |
| 40483 | The name of the module (low word) | R |
| 40484 | The name of the module (high word) | R |
| 40485 | Module address, valid range: 0x1 ~ 0xF7 | R |
| 40486 | Bits 5:0 Baud Rate, 0x0A<br>Bits 7:6 Reserved | R |
| 40488 | Response delay time, 0~30, in ms | R/W |

|  |  |  |
|---|---|---|
| 40489 | Host Watchdog timeout value, 0 ~ 255, in 0.1s | R/W |
| 40490 | Channel enabled/disabled | R/W |
| 40492 | Host Watchdog timeout count, write 0 to clear | R/W |

Note:
The command to load the factory calibration parameters takes about 3 seconds to be processed. The next command should not be sent before this time has elapsed.

## 3.6. Engineering Data Format Table

The Modbus protocol supports engineering data format and the type code information is as follows.

| Type Code | Analog Input Type | -F.S. | +F.S. |
|---|---|---|---|
| 07 | +4 to +20<br>mA | 4000 | 20000 |
| 08 | -10 to +10<br>V | -10000 | 10000 |
| 09 | -5 to +5<br>V | -5000 | 5000 |
| 0A | -1 to +1<br>V | -10000 | 10000 |
| 0B | -500 to +500<br>mV | -5000 | 5000 |
| 0C | -150 to +150<br>mV | -15000 | 15000 |
| 0D | -20 to +20<br>mA | -20000 | 20000 |
| 1A | 0 to +20<br>mA | 0 | 20000 |

The under-range value is –32768 and the over-range value is +32767. For hexadecimal data format, please refer to Section 1.7 for details.

# 4. Troubleshooting

If you are having difficulty using the M-7002 module, here are some suggestions that may help. If you cannot find the answers you need in this guide, contact ICP DAS Product Support.

## 4.1. Communicating with the module

If you attempt to communicate with the module and receive no response, first check the following:

- □ Ensure that the supplied power is within the range of +10 to +30 $V_{DC}$. If the supplied power is sufficient, then the power LED should be on.
- □ Ensure that the RS-485 converter provides the bias. The RS-485 converters manufactured by ICP DAS all provide the bias. If the RS-485 converter does not provide the bias and the PCB version of the M-7002 is 3.01 and later, then you can refer to section 1.5 to open the cover to adjust the jumper to enable the RS-485 bias.
- □ When the module receives a command, the power LED will be set to "off". The power LED will be shown as "on" after the module responds. This method can be used to check whether the module has received a command sent from the host.
- □ If possible, use another device that is known to be functional to check whether the host can communicate with the device through the same network.
- □ If the host is a PC installed with a Windows operating system, then execute the DCON Utility to determine whether the module can be found. The DCON Utility can be downloaded from the ICP DAS website at http://www.icpdas.com. Documentation for the DCON Utility can be found in the "**Getting Started For I-7000 Series Modules**" manual.
- □ Set the module to the "INIT mode" and communicate with the module using the following settings: address 00 and DCON protocol. See Section 1.7 for more details related to configuration settings.

## 4.2. Reading Data

If the data read from the input channel is not correct, first check the following:

- □ Ensure that the type code and data format settings are correct. The type code is set by using the $AA7CiRrr command, see Section 2.12 for details. The data format is set by using the %AANNTTCCFF command (see Section 2.1). If you are using the Modbus RTU protocol, the type code is set by using sub-function 08h of the function 46h.
- □ If the voltage read by the module is incorrect, then it may be because the calibration parameters stored in the non-volatile memory are corrupted. You can calibrate the module by yourself, but be sure to read Section 1.6 for details before performing any calibration. Use the $AAS1 command to reload the factory calibration parameters, see Section 2.21 for details.

# 5. Appendix

## 5.1. INIT Mode

Each I-7000 and M-7000 module has a built-in EEPROM that can be used to store configuration information, such as the module address, Type Code, and Baud Rate, etc. Occasionally, the configuration of a module may be forgotten and there may be no visual indications of the configuration of the module. It is difficult to communicate with the module when the configuration of the module is unknown. To help avoid this problem, the I-7000 and M-7000 series has a special mode called “INIT mode”. When the module is powered on in “INIT mode” the configuration of the module is reset to the default settings shown below, allowing it to be operated as normal.

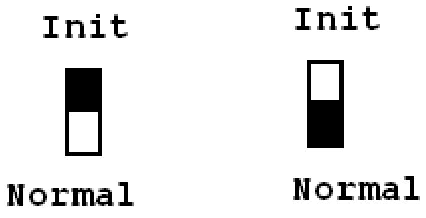
1. Address: 00
2. Baud Rate: 9600 bps
3. No checksum
4. Protocol: DCON

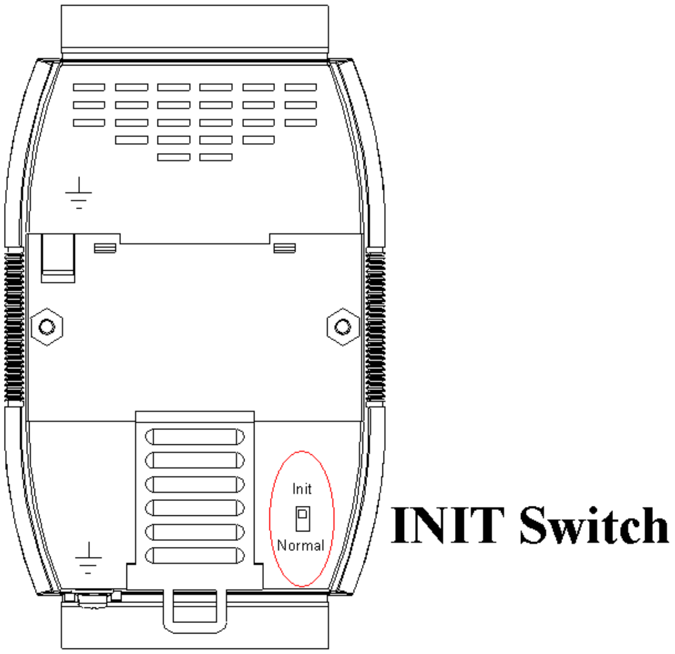
The configuration information stored in the EEPROM is not changed and can be read by sending the $AA2(CR) command at 9600 bps.

There are also other commands that require the module to set to INIT mode before being used. They are:

1. %AANNTTCCFF, which is used when changing the Baud Rate and checksum settings. See Section 2.1 for details.
2. $AAPN, see Section 2.20 for details.

Originally, INIT mode was accessed by connecting the INIT* terminal to the GND terminal. Newer I-7000 and M-7000 modules have an INIT switch located on the rear of the module to allow easier access to INIT mode. For these modules, INIT mode is accessed by sliding the INIT switch to the Init position, as shown below.





## 5.2. Dual Watchdog Operation

**Dual Watchdog = Module Watchdog + Host Watchdog**

The Module Watchdog is a hardware reset circuit that monitors the operating status of the module. While working in harsh or noisy environments, the module may be shut down by external signals. The Watchdog circuit allows the module to work continuously without disruption.

The Host Watchdog is a software function that monitors the operating status of the host. Its purpose is to prevent problems due to network/communication errors or host malfunctions. When a Host Watchdog timeout occurs, the module will reset all outputs to a safe state in order to prevent the controlled target from performing any erroneous operations.

I-7000 series modules include an internal Dual Watchdog, making the control system more reliable and stable.

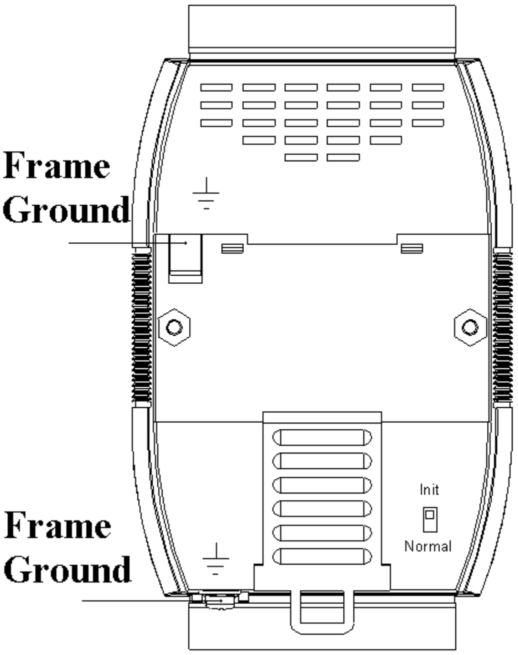
For more information regarding the Dual Watchdog, please refer to Chapter 5 of the “Getting Started For M-7002 Modules” manual that can be downloaded from the ICP DAS website

## 5.3. Frame Ground

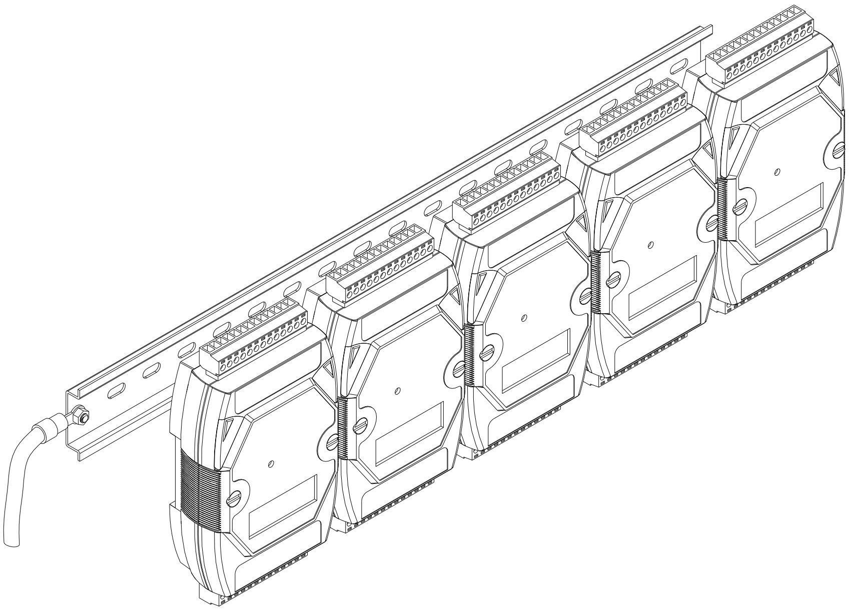
Electronic circuits are constantly vulnerable to Electrostatic Discharge (ESD), which becomes worse in a continental climate area. Some I-7000 and M-7000 modules feature a new design for the frame ground, which provides a path for bypassing ESD, allowing enhanced static protection (ESD) capabilities and ensures that the module is more reliable.

Either of the following options will provide better protection for the module:

1. If the module is DIN-Rail mounted, connect the DIN-Rail to the earth ground. This is because the DIN-Rail is in contact with the upper frame ground, as shown in the figure below.
2. Alternatively, connect the lower frame ground terminal to a wire and connect the wire to the earth ground, as shown in the figure below.

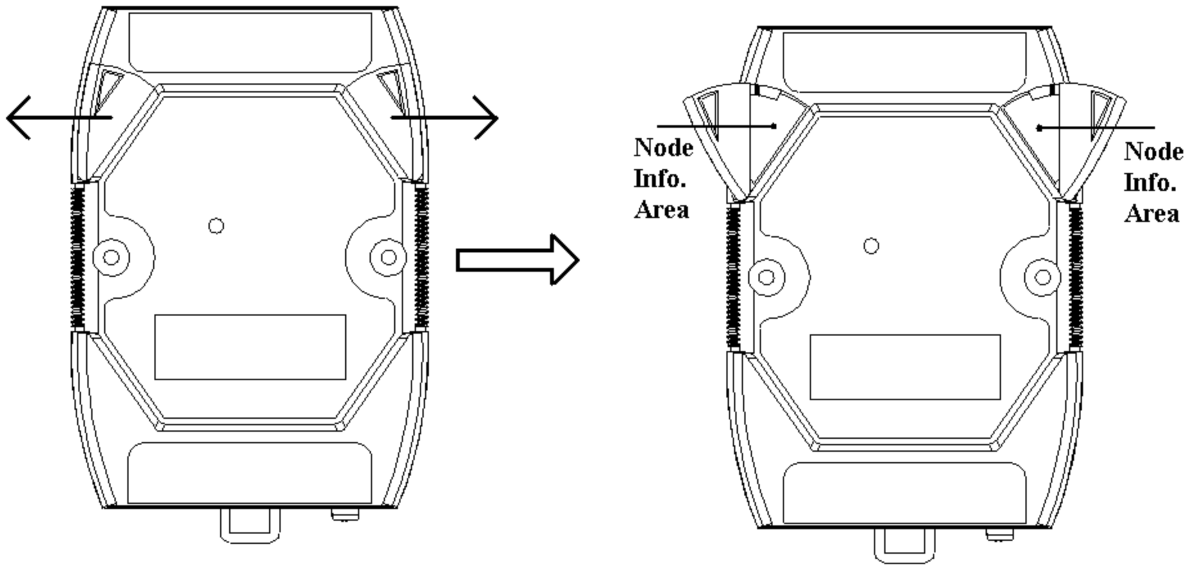


New DIN-Rail models are available that can easily be connected to the earth ground. Each rail is made of stainless steel, which is stronger than those made of aluminum. There is a screw at one end and a ring terminal is included, as shown in the figure below.



## 5.4. Node Information Area

Each I-7000 and M-7000 module has a built-in EEPROM that can be used to store configuration information, such as the module address, Type Code, and Baud Rate, etc. One minor drawback is that there may be no visual indications of the configuration of the module. Newer I-7000 and M-7000 modules include node information areas that are protected by a cover, as shown below, and can be used to make a written record of the node information, such as module address and Baud Rate, etc. To access the node information areas, first slide the covers outward, as shown in the figure below.



## 5.5. Reset Status

The reset status of a module is set when the module is powered on, or when the module is reset by the Module Watchdog, and is cleared after responding to the first $AA5 command. This can be used to check whether the module has recently been reset. If the response from the $AA5 command indicates that the reset status has been cleared, it means that the module has not been reset since the last $AA5 command was sent. If the response from the $AA5 command indicates that the reset status is set and it is not the first time an $AA5 command has been sent, it means that the module has been reset and the digital output value has been changed to either  the default power-on value or the safe value.

Revision History

| Revision | Date | Changes Made |
|---|---|---|
| 1.2 | 2020/4/1 | Add section 1.5 and modify section 4.1 to include RS-485 bias resistor. |
| 1.3 | 2022/9/19 | Modify examples of section 2.45 and 2.46 |
| 1.4 | 2022/10/31 | Modify section 1.8 hex format range of type code 0D |
| 1.5 | 2024/02/17 | Add sections 2.58 and 2.59 to add ~AADT and ~AADTE commands. Modify section 3.5 to add Modbus register 00274 |