# LC-101 User Manual

## Warranty

All products manufactured by ICP DAS are under warranty regarding defective materials for a period of one year, beginning from the date of delivery to the original purchaser.

## Warning!

ICP DAS assumes no liability for any damage resulting from the use of this product. ICP DAS reserves the right to change this manual at any time without notice. The information furnished by ICP DAS is believed to be accurate and reliable. However, no responsibility is assumed by ICP DAS for its use, nor for any infringements of patents or other rights of third parties resulting from its use.

## Copyright



## Trademarks

Names are used for identification purposes only and may be registered trademarks of their respective companies.

Date: 2012/12/01

# Table of Contents

# Introduction

The LC-101 is an easy-to-use lighting control module that requires no specialist skills to install and operate, and no software is needed in order to control the Digital Output channel.

The LC-101 provides 1 channel for Digital Input (photocouple isolation) and 1 channel for relay output. The output channel is a Form C type relay, while the input channel is based on a sink-type using a wire connection. The input channel can be used to directly control a 1-channel relay ON and OFF sequence without requiring a remote host controller. 4 kV ESD protection and 5000 Vrms intra-module isolation are also provided.

When required, communication with the LC-101 is programmable based on either the DCON or the Modbus RTU protocol, and an added benefit is that different addresses can be set for DCON or Modbus RTU communication via hardware or software configuration.

# 1 Hardware Information

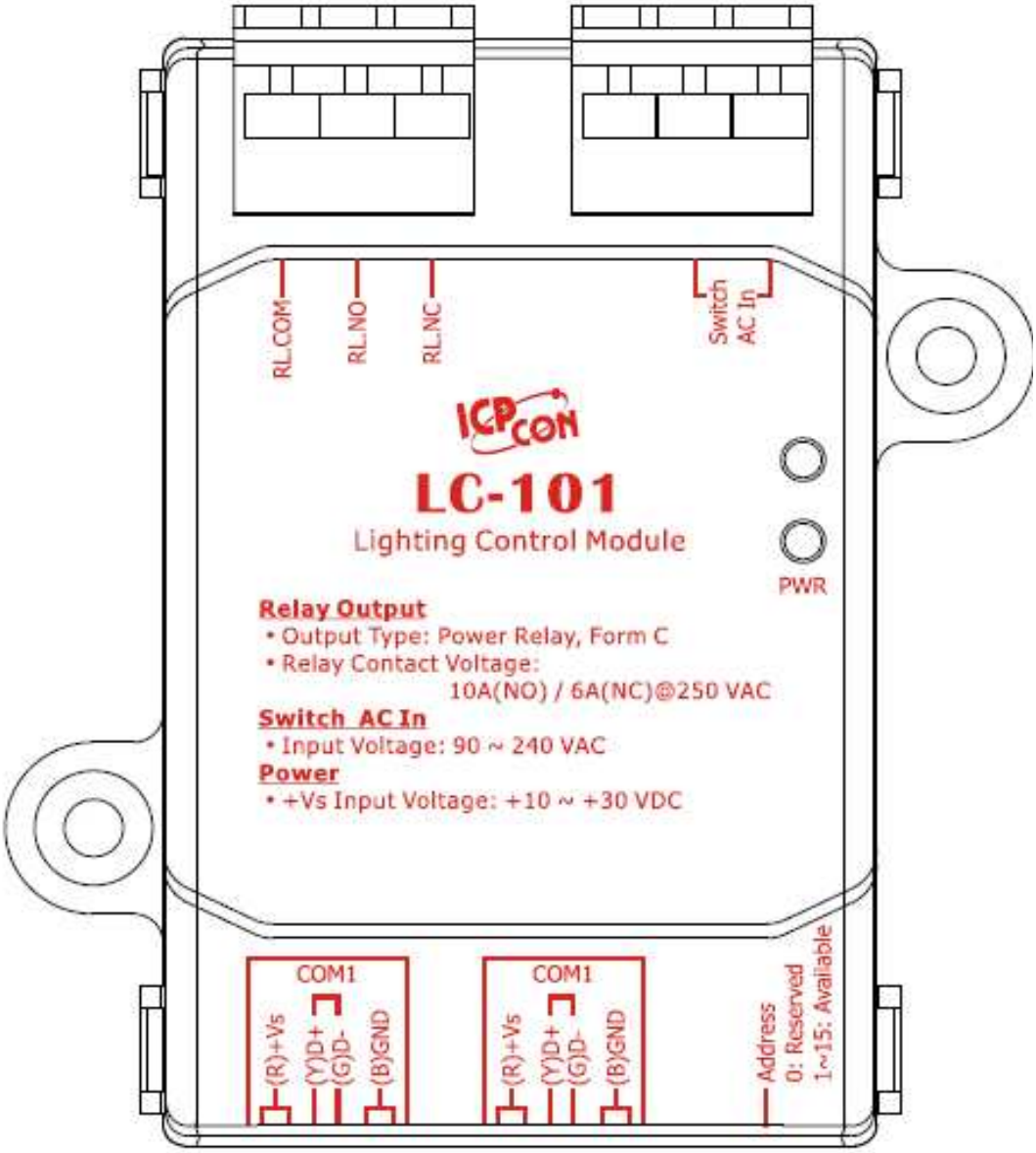
## 1.1 IO Specifications

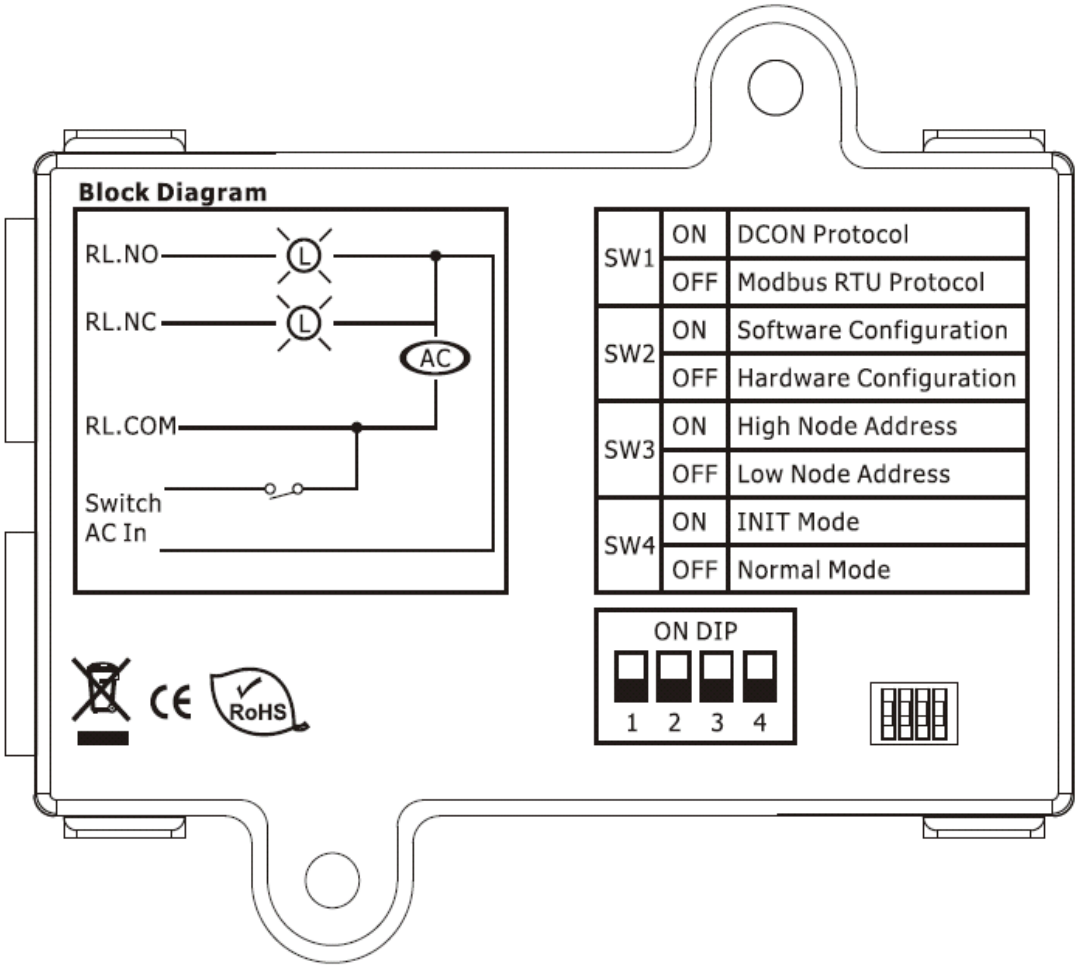
| Digital Input | |
|---|---|
| Input Channels | 1 |
| Type | 90 to 240 $V_{AC}$ |
| On Voltage Level | 85 $V_{AC}$ |
| Off Voltage Level | 60 $V_{AC}$ |
| Input Impedance | 68 KΩ, 1 W |
| Isolation | 5000 Vrms |
| Function | Local and Remote Direct Control Relay ON/OFF and Remote Status Monitoring |
| Relay Output | |
| Output Channels | 1 |
| Type | Power Relay, Form C |
| Operating Voltage | 250 $V_{AC}$ or 30 $V_{DC}$ |
| Max. Load Current | 10 A (NO) / 6A (NC) @ 250 $V_{AC}$ |
| Operating Time | 15 ms Max. |
| Release Time | 5 ms Max. |
| Electrical Life (Resistive load) | 50,000 ops |
| Mechanical Life | 1,000,000 ops at no load (300 ops/minute) |
| Safety Approval | UL/CUL, TÜV |
| Power-on Value | Yes |

## 1.2 System Specifications

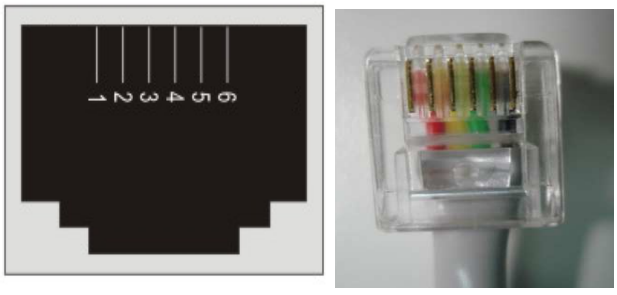
| | |
|---|---|
| Communication | |
| Interface | RS-485 |
| Data Format | N,8,1/O,8,1/E,8,1/N,8,2 |
| Baud Rate | Hardware Configuration: Fixed 9600 bps |
| | Software Configuration: 1200 to 115200 bps |
| Protocol | Modbus RTU or DCON |
| Node Addresses | 32 to 63 for hardware configuration or 0 to 255 for software configuration |
| Connector | RJ-11 |
| LED Indicators | |
| Power | 1 LED as Power Indicator |
| EMS Protection | |
| ESD (IEC 61000-4-2) | ±2 kV Contact for Each Terminal |
| | ±4 kV Air for Random Point |
| EFT (IEC 61000-4-4) | ±2 kV for Power |
| Power Requirements | |
| Input Voltage Range | +10 to +30 $V_{DC}$ |
| Consumption | 0.5 W Max. |
| Connector | RJ-11 |
| Mechanical | |
| Dimensions (W x L x H) | 52 mm x 98 mm x 27 mm |
| Installation | Screw Mounting |
| Environment | |
| Operating Temperature | -25°C to +75°C |
| Storage Temperature | -30°C to +75°C |
| Humidity | 10 to 95% RH, Non-condensing |

## 1.3 Pin Assignments






RJ-11 Connector

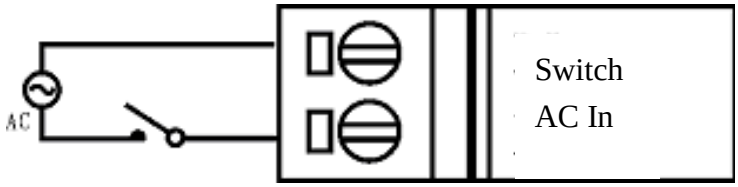



| Pin | Description | |
|---|---|---|
| 1 | +VS | Power Input Voltage (+10 $V_{DC}$ to +30 $V_{DC}$) |
| 2 | +VS | Power Input Voltage (+10 $V_{DC}$ to +30 $V_{DC}$) |
| 3 | DATA+ | RS-485 Serial Communication Interface |
| 4 | DATA- | |
| 5 | GND | Ground |
| 6 | GND | Ground |

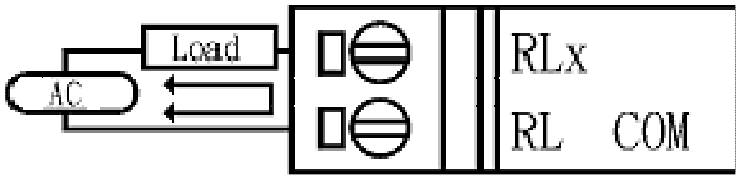
## 1.4 Wire Connections

### DIO Wire Connections

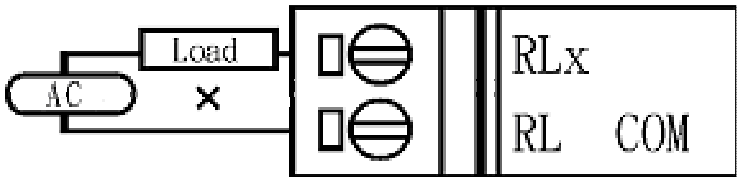
DI



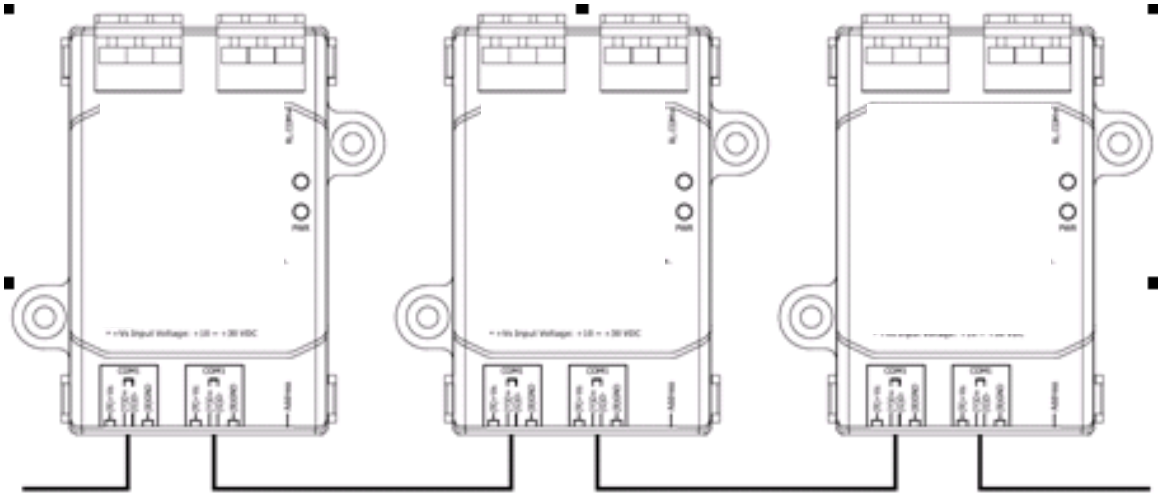

Relay Output ON




Relay Output OFF

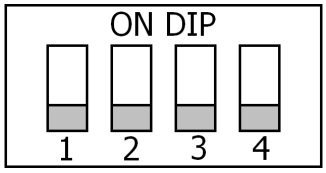



### Power and Communication



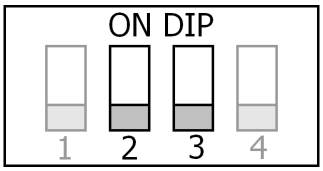
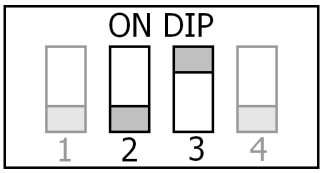
RS-485 and power input daisy chain using RJ-11 connectors

## 1.5 DIP Switch Configuration




| | | |
|---|---|---|
| SW1 | ON | DCON Protocol |
| | OFF | Modbus RTU Protocol |
| SW2 | ON | Software Configuration |
| | OFF | Hardware Configuration |
| SW3 | ON | Node Address (High) |
| | OFF | Node Address (Low) |
| SW4 | ON | INIT Mode |
| | OFF | Normal Mode |

**Address Settings via Hardware Configuration**

| | | |
|---|---|---|
|  |  | 0 to F for Address 32 - 47 (Node Address - Low) |
| |  | 0 to F for Address 48 - 63 (Node Address High) |



### 1.5.1 INIT Mode

When the LC-101 is powered on with DIP switch SW4 in the ON position, the module will be set to INIT Mode. In this mode, the position of DIP switches SW1-SW3 and the Address settings switch will be ignored and the LC-101 module will use the fixed configuration parameters listed below.

| | |
|---|---|
| **Protocol:** | DCON |
| **Address:** | 00 |
| **Baud Rate:** | 9600 bps |
| **Data Format:** | N,8,1 |

In this mode, the relevant commands can be used to change the configuration, and the new settings will be saved to the EEPROM.

### 1.5.2 Hardware Configuration Mode

When the LC-101 is powered on with both the SW4 and the SW2 DIP switches in the OFF position, the module will be set to Hardware Configuration Mode. In this mode, the following configuration parameters are used.

| | |
|---|---|
| **Protocol:** | Dependent on the position of DIP switch SW1 |
| **Address:** | Refer to the "Address Settings via Hardware Configuration" table above |
| **Baud Rate:** | Fixed at 9600 bps |
| **Data Format:** | Fixed to N,8,1 |

In this mode, any software command related to configuration will be ignored when using the Modbus RTU protocol, or will return an error when using the DCON protocol.

### 1.5.3 Software Configuration Mode

When the LC-101 is powered on with DIP switch SW4 in the OFF position and DIP switch SW2 in the ON position, the module will be set to Software Configuration Mode. In this mode, the configuration parameters to be used will be retrieved from the EEPROM. The default configuration parameters stored in the EEPROM is:

| | |
|---|---|
| **Protocol:** | Modbus RTU |
| **Address:** | 01 (0x01) |
| **Baud Rate:** | 9600 bps |
| **Data Format:** | N,8,1 |

In this mode, the relevant commands can be used to change the configuration parameters, and the new settings will be saved to the EEPROM.

## 1.6 Software Configuration Tables

### Baud Rate Settings (CC)

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

### Type Code Settings (TT)

For the LC-101, the Type Code is fixed to 40 and cannot be changed.

### Data Format Settings (FF)

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| CU | CS | Reserved | | | | | |

| Key | Description |
|---|---|
| CS | Checksum Settings<br>0: Disabled<br>1: Enabled |
| CU | Counter Update:<br>0: The counter is updated when there is a falling edge in the input signal.<br>1: The counter is updated when there is a rising edge in the input signal. |

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

## 1.7 Digital Input/Output Data Format for the DCON Protocol

The data format for the response to the **$AA4**, **$AA6** and **$AALS** commands is:

(First Value)(Second Value)00

The data format for the response to the **@AA** command is:

(First Value)(Second Value)

**Note**: Both the First Value and the Second Value are in the form of two hexadecimal digits.

| Module | First Value | | Second Value | |
|---|---|---|---|---|
| LC-101 | DO0 | 00 - 01 | DI0 | 00 - 01 |

# 2 DCON Protocol

All communication with the LC-101 consists of commands generated by the Host and responses transmitted by the LC-101 module. Each module has a unique ID number that is used for addressing purposes and is stored in non-volatile memory. The module ID number is set to 01 by default and can be changed by sending a user command. All commands to the modules contain the ID number as the address, meaning that only the addressed module will respond.

**Command Format:**

| Delimiter Character | Module Address | Command | Checksum | CR |
|---|---|---|---|---|

**Response Format:**

| Delimiter Character | Module Address | Data | Checksum | CR |
|---|---|---|---|---|

CR = End of command character, carriage return (0x0D), used to end a frame.

Note: All characters should be in upper case.

## An Overview of the DCON Command Set

| General Command Sets | | | |
|---|---|---|---|
| **Command** | **Response** | **Description** | **Section** |
| %AANNTTCCFF | !AA | Sets the Configuration of the Module | 2.1 |
| #** | No Response | Sends the Synchronized Sampling Command | 2.2 |
| #AA00(Data) | > | Sets the Value for all Digital Output Channels | 2.3 |
| #AA0A(Data) | > | Sets the Value for all Digital Output Channels | 2.4 |
| #AA10DD | > | Sets the Digital Output for a Single Channel | 2.5 |
| #AAA0DD | > | Sets the Digital Output for a Single Channel | 2.6 |
| #AAN | !AA(Data) | Reads the Digital Input Counter Value for a Specific Channel | 2.7 |
| $AA2 | !AANNTTCCFF | Reads the Configuration of the Module | 2.8 |
| $AA4 | !S(Data) | Reads the Synchronized Data | 2.9 |
| $AA5 | !AAS | Reads the Reset Status of the Module | 2.10 |
| $AA6 | !(Data) | Reads the Status of the Digital Input/Output Channels | 2.11 |
| $AAC | !AA | Clears the Status of the Latched Digital Input Channels | 2.12 |
| $AACN | !AA | Clears the Digital Input Counter | 2.13 |
| $AAF | !AA(Data) | Reads the Firmware Version of the Module | 2.14 |
| $AALS | !(Data) | Reads the Status of the Latched Digital Input | 2.15 |
| $AAM | !AA(Data) | Reads the Name of the Module | 2.16 |
| $AAP | !AASC | Reads the Communication Protocol | 2.17 |
| $AAPN | !AA | Sets the Communication Protocol | 2.18 |
| @AA | >(Data) | Reads the Status of the Digital Input/Output Channels | 2.19 |
| @AA(Data) | > | Sets the Value for all Digital Output Channels | 2.20 |
| $AALC0N | !AA | Coordinates the operation status between the Digital Input and the Digital Output | 2.21 |
| $AALC1 | !AAN | Reads whether or not the operation status between the Digital Input and the Digital Output is | 2.22 |

| | | | |
|---|---|---|---|
| | | coordinated | |
| $AALC2NNNN | !AA | Sets the Active Delay Time for the Digital Output | 2.23 |
| $AALC3 | !AANNNN | Reads the Active Delay Time for the Digital Output | 2.24 |
| ~AAD | !AAVV | Reads whether the Digital Input/Output is active or inactive | 2.25 |
| ~AADVV | !AA | Sets the Digital Input/Output to Active | 2.26 |
| ~AA4P | !AA(Data) | Reads the Digital Output Power-on Value | 2.27 |
| ~AA5P | !AA | Sets the Digital Output Power-on Value | 2.28 |
| ~AARD | !AA(Data) | Reads the Response Delay Time | 2.29 |
| ~AARDVV | !AA | Sets the Response Delay Time | 2.30 |

## 2.1 %AANNTTCCFF

### Description:

This command is used to set the configuration of a specified 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** | The Type code, which should be set to 40 for DIO modules |
| **CC** | The new Baud Rate, see Section 1.6 for details. The INIT* pin must be connected to the ground pin in order to change Baud Rates. For modules using frame ground, this is achieved by moving the rear slide switch to the INIT position. |
| **FF** | The command used to set the counter update direction and the checksum (see Section 1.6). The INIT* pin must be connected to the ground pin in order to change the checksum settings. For modules using frame ground, this is achieved by moving the rear slide switch to the INIT position. |

### Response:

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

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

| | |
|---|---|
| **!** | Delimiter character to indicate that the command was valid |
| **?** | Delimiter character to indicate that the command was invalid. If an attempt is made to change the **Baud Rate** or **Checksum** settings without first connecting the INIT* pin to the ground pin or without switching the rear slide switch to the INIT position, the module will return a response indicating that the command was invalid. |
| **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: %0102400600 Response: !02

Changes the address of module 01 to 02. The module returns a response indicating that the command was valid and includes the new address of the module.

Command: %0101200A00 Response: ?01

Changes the Baud Rate of module 01 to 115200bps. The module returns a response indicating that the command was invalid, because it is not in INIT* mode.

Command: %0101200A00 Response: !01

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

**Related Commands:**

Section 2.8 $AA2

**Related Topics:**

Section 1.6 Software Configuration Tables

**Notes:**

Changes to the address and counter update direction settings take effect immediately after a valid command is received. Changes to the Baud Rate and checksum settings take effect on the next power-on reset.

## 2.2 #**

### Description:

This command is used to instruct all modules to sample their input values and store the data 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 2.9 for details.

### Examples:

Command: #** Response: There is no response to this command.

Sends the synchronized sampling command to all modules.

Command: $014 Response: !10F0000

Reads the synchronized sampling data and the module returns a response indicating that the command was valid. The status byte of the response is 1, which means that it is the first time the synchronized sampling data has been read since the previous #** command was received.

Command: $014 Response: !00F0000

Reads the synchronized sampling data and the module returns a response indicating that the command was valid. The status byte of the response is 0, which means that it is **NOT** the first time the synchronized sampling data has been read since the previous #** command was received.

### Related Commands:

Section 2.9 $AA4

## 2.3 #AA00(Data)

**Description:**

This command is used to set the Digital Output value for channel RL1 of a specified module.

**Syntax:**

**#AA00(Data)[CHKSUM](CR)**

| | |
|---|---|
| **#** | Delimiter character |
| **AA** | The address of the module to be set in hexadecimal format (00 to FF) |
| **00** | The command to set the Digital Output value for channel RL1 |
| **(Data)** | A two-digit hexadecimal value, where bit 0 corresponds to channel RL1. When the bit is 0, it denotes that the Digital Output channel is set to OFF, and 1 denotes that the Digital Output channel is set to ON. |

**Response:**

Valid Command: **>[CHKSUM](CR)**

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

Ignored Command: **![CHKSUM](CR)**

| | |
|---|---|
| **>** | Delimiter character to indicate that the command was valid |
| **?** | Delimiter character to indicate that the command was invalid |
| **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: #020001 Response: >

Sets channel RL1 of module 02 to ON, and the module returns a response indicating that the command was valid.

Command: #020002 Response: ?AA

Attempts to set channel RL2 of module 02 to ON, but the module returns a response

indicating that the command was invalid because channel RL2 does not exist.

**Related Commands:**

Section 2.4 #AA0A(Data), Section 2.5 #AA10DD, Section 2.6 #AAA0DD

**Related Topics:**

Section 1.6 Software Configuration Tables

## 2.4 #AA0A(Data)

**Description:**

This command is used to set the Digital Output value for channel RL1 of a specified module.

**Syntax:**

**#AA0A(Data)[CHKSUM](CR)**

| | |
|---|---|
| **#** | Delimiter character |
| **AA** | The address of the module to be set in hexadecimal format (00 to FF) |
| **0A** | The command to set the Digital Output value for channel RL1 |
| **(Data)** | A two-digit hexadecimal value, where bit 0 corresponds to channel RL1. When the bit is 0, it denotes that the Digital Output channel is set to OFF, and 1 denotes that the Digital Output channel is set to ON. |

**Response:**

Valid Command: **>[CHKSUM](CR)**

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

Ignored Command: **![CHKSUM](CR)**

| | |
|---|---|
| **>** | Delimiter character to indicate that the command was valid |
| **?** | Delimiter character to indicate that the command was invalid |
| **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: #020A01 Response: >

Sets channel RL1 of module 02 to ON, and the module returns a response indicating that the command was valid.

Command: #020A02 Response: ?AA

Attempts to set channel RL2 of module 02 to ON, but the module returns a response indicating that the command was invalid because channel RL2 does not exist.

**Related Commands:**

Section 2.3 #AA00(Data), Section 2.5 #AA10DD, Section 2.6 #AAA0DD

**Related Topics:**

Section 1.6 Software Configuration Tables

## 2.5 #AA10DD

**Description:**

This command is used to set the Digital Output value for a single channel of a specified module.

**Syntax:**

**#AA10DD[CHKSUM](CR)**

| | |
|---|---|
| **#** | Delimiter character |
| **AA** | The address of the module to be set in hexadecimal format (00 to FF) |
| **1** | The command to set the Digital Output for a single channel |
| **0** | Specifies the relay output channel to be set, zero based |
| **DD** | The command to set the relay output channel:<br>00: Sets the relay output channel to OFF<br>01: Sets the relay output channel to ON |

**Response:**

Valid Command: **>[CHKSUM](CR)**

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

Ignored Command: **![CHKSUM](CR)**

| | |
|---|---|
| **>** | Delimiter character to indicate that the command was valid |
| **?** | Delimiter character to indicate that the command was invalid |
| **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: #021001 Response: >

Sets channel RL1 of module 02 to ON, and the module returns a response indicating that the command was valid.

Command: #021101 Response: ?AA

Attempts to set channel RL2 of module 02 to ON, but the module returns a response

indicating that the command was invalid because channel RL2 does not exist.

### Related Commands:

Section 2.3 #AA00(Data), Section 2.4 #AA0A(Data), Section 2.6 #AAA0DD

### Related Topics:

Section 1.6 Software Configuration Tables

## 2.6 #AAA0DD

### Description:

This command is used to set the Digital Output value for a single channel of a specified module.

### Syntax:

**#AAA0DD[CHKSUM](CR)**

| | |
|---|---|
| **#** | Delimiter character |
| **AA** | The address of the module to be set in hexadecimal format (00 to FF) |
| **A** | The command to set the Digital Output value for a single channel |
| **0** | Specifies the relay output channel to be set, zero based |
| **DD** | The command to set the relay output channel:<br>00: Sets the relay output channel to OFF.<br>01: Sets the relay output channel to ON. |

### Response:

Valid Command: **>[CHKSUM](CR)**

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

Ignored Command: **![CHKSUM](CR)**

| | |
|---|---|
| **>** | Delimiter character to indicate that the command was valid |
| **?** | Delimiter character to indicate that the command was invalid |
| **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: #02A001 Response: >

Sets channel RL1 of module 02 to ON, and the module returns a response indicating that the command was valid.

Command: #02A101 Response: ?AA

Attempts to set channel RL2 of module 02 to ON, but the module returns a response

indicating that the command was invalid because channel RL2 does not exist.

### Related Commands:

Section 2.3 #AA00(Data), Section 2.4 #AA0A(Data), Section 2.5 #AA10DD

### Related Topics:

Section 1.6 Software Configuration Tables

## 2.7 #AAN

### Description:

This command is used to read the Digital Input counter value from a specific channel 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 Command: **!AA(Data)[CHKSUM](CR)**

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

| | |
|---|---|
| **!** | Delimiter character to indicate that the command was valid. |
| **?** | Delimiter character to indicate that the command was invalid. |
| **AA** | The address of the responding module in hexadecimal format (00 to FF) |
| **(Data)** | A five-digit decimal value representing the Digital Input counter data from the specified channel (00000 to 65535) |

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: #030 Response: !0300103

Reads the Digital Input counter value from channel 0 of module 03 and returns a response indicating that the command was valid, with a counter value of 103.

Command: #029 Response: ?02

Attempts to read the Digital Input counter value from channel 9 of module 02, but the module returns a response indicating that the command was invalid because channel 9 does not exist.

**Related Commands:**

Section 2.13 #AACN

## 2.8 $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 Command: **!AATTCCFF[CHKSUM](CR)**

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

| | |
|---|---|
| **!** | Delimiter character to indicate that the command was valid |
| **?** | Delimiter character to indicate that the command was invalid |
| **AA** | The address of the responding module in hexadecimal format (00 to FF) |
| **TT** | The Type code for the module, which should be 40 for DIO modules |
| **CC** | The Baud Rate for the module. See Section 1.6 for details. |
| **FF** | The checksum and counter update direction settings of the module. See Section 1.6 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: $012 Response: !01400600

Reads the configuration of module 01 and returns a response indicating that the command was valid, and showing that the Type code is set to 40, the Baud Rate is 9600 bps, the Checksum is Disabled and the counter update direction is Falling Edge.

Command: $032 Response: ?03

Attempts to read the configuration of module 03, but returns a response indicating that the command was invalid because module 03 does not exist.

**Related Commands:**

Section 2.1 %AANNTTCCFF

**Related Topics:**

Section 1.5 DIP Switch Configuration

Section 1.6 Software Configuration Tables

## 2.9 $AA4

### Description:

This command is used to read the synchronized sampling data that was stored when the last #** command 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 synchronized sampling data |

### Response:

Valid Command: **!S(Data)[CHKSUM](CR)**

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

| | |
|---|---|
| **!** | Delimiter character to indicate that the command was valid |
| **?** | Delimiter character to indicate that the command was invalid |
| **AA** | The address of the responding module in hexadecimal format (00 to FF) |
| **S** | The status of the synchronized sampling data:<br>0: This is **NOT** the first time that the data has been read<br>1: This is the first time that the data has been read |
| **(Data)** | The synchronized sampling data. See Section 1.6 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: $014 Response: ?01

Attempts to read the synchronized sampling data for module 01, but returns a response indicating that the command was invalid because the Synchronized Sampling Command, #**, was not sent in advance.

Command: #** Response: There is no response to this command.

Sends the synchronized sampling command to all modules.

Command: $014 Response: !1000F00

Reads the synchronized sampling data for module 01. The module returns a response indicating that the command was valid containing the synchronized sampling data, and sets the status byte to 1 to signify that this is the first time that the synchronized sampling data has been read.

Command: $014 Response: !0000F00

Reads the synchronized sampling data for module 01. The module returns a response indicating that the command was valid containing the synchronized sampling data, and sets the status byte to 0 to signify that the synchronized sampling data has been read.

Command: $034 Response: ?03

Attempts to read the synchronized sampling data for module 03, but returns a response indicating that the command was invalid because module 03 does not exist.

**Related Commands:**

Section 2.2 #**

**Related Topics:**

Section 1.6 Software Configuration Tables

## 2.10 $AA5

**Description:**

This command is used to read the reset status for 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 of the module

**Response:**

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

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

**!** Delimiter character to indicate that the command was valid

**?** Delimiter character to indicate that the command was invalid

**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 $AA5 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 for module 01 and returns a response indicating that the command was valid, and 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 for module 01 and returns a response indicating that the command was valid, and that there has been no module reset since the last $AA5 command was sent.

Command: $035 Response: ?03

Attempts to read the reset status for module 03, but returns a response indicating that the command was invalid because module 03 does not exist.

**Related Commands:**

None

## 2.11 $AA6

**Description:**

This command is used to read the status of both the Digital Input and Digital Output channels of a specified module.

**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 status of the Digital Input and Digital Output channels |

**Response:**

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

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

| | |
|---|---|
| **!** | Delimiter character to indicate that the command was valid |
| **?** | Delimiter character to indicate that the command was invalid |
| **AA** | The address of the responding module in hexadecimal format (00 to FF) |
| **(Data)** | The status of the Digital Output and Digital Input channels represented by a four-digit hexadecimal value followed by 00. The first two digits represent the status of the Digital Output channels and the second two represent the status of the Digital Input channels. See Section 1.7 for more 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: $026 Response: !010100

Reads the status of the Digital Output and Digital Input channels for module 02 and returns a response indicating that the command was valid and that the current Digital Output value is 01 and the current Digital Input value is 01 denoting that both the Digital Output and Digital Input channels are ON.

Command: $036 Response: ?03

Attempts to read the status of the Digital Output and Digital Input channels for module 03 and returns a response indicating that the command was invalid because module 03 does not exist.

### Related Commands:

Section 2.19 @AA

### Related Topics:

Section 1.6 Software Configuration Tables, Section 1.7 Digital Input/Output Data Format Settings

## 2.12 $AAC

**Description:**

This command is used to clear the status of the latched Digital Input and Digital Output channels of 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 status of the latched Digital Input and Digital Output channels |

**Response:**

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

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

| | |
|---|---|
| **!** | Delimiter character to indicate that the command was valid |
| **?** | Delimiter character to indicate that the command was invalid |
| **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: $01L0 Response: !010100

Reads the status of the low latched Digital Output and Digital Input channels of module 01 and returns a response indicating that the command was valid, with a value of 0101, which denotes that the latched values for the both the Digital Output and Digital Input channels have recently been set to ON.

Command: $01C Response: !01

Clears the status of the latched Digital Input channels of module 01 and returns a response indicating that the command was valid.

Command: $01L0 Response: !000000

Reads the status of the low latched Digital Output and Digital Input channels of module 01 and returns a response indicating that the command was valid, with a value of 0000, which denotes that the status of all low latched Digital Output and Digital Input channels has been cleared.

Command: $03C Response: ?03

Attempts to clear the status of the latched Digital Input channels of module 03 and returns a response indicating that the command was invalid because module 03 does not exist.

### Related Commands:

Section 2.15 $AALS

### Related Topics:

None

### Notes:

The status of both the low and the high latched Digital Output and Digital Input channels will be cleared when using this command.

## 2.13 $AAC0

**Description:**

This command is used to clear the Digital Input counter for a specific channel of a specified module.

**Syntax:**

**$AACN[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 counter |
| **0** | The channel to be cleared, zero based |

**Response:**

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

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

| | |
|---|---|
| **!** | Delimiter character to indicate that the command was valid |
| **?** | Delimiter character to indicate that the command was invalid |
| **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: #030 Response: !0300103

Reads the counter data from channel 1 of module 03 and returns a response indicating that the command was valid, and that the counter value is 103.

Command: $03C0 Response: !03

Clears the counter value for channel 1 of module 03 and returns a response indicating that the command was valid.

Command: #032 Response: !0300003

Reads the counter data from channel 2 of module 03 and returns a response indicating that the command was valid, and that the counter value is 3.

Command: #039 Response: ?03

Attempts to read the counter data from channel 9 of module 03 and returns a response indicating that the command was invalid because channel 9 does not exist.

**Related Commands:**

Section 2.7 #AAN

## 2.14 $AAF

### Description:

This command is used to read the firmware version of 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 |

### Response:

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

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

| | |
|---|---|
| **!** | Delimiter character to indicate that the command was valid |
| **?** | Delimiter character to indicate that the command was invalid |
| **AA** | The address of the responding module in hexadecimal format (00 to FF) |
| **(Data)** | A string indicating the firmware version 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: $01F Response: !0101.00

Reads the firmware version of module 01, and returns a response indicating that the command was valid, and that the firmware is version 01.00.

Command: $03F Response: ?03

Attempts to read the firmware version of module 03 and returns a response indicating that the command was invalid because module 03 does not exist.

## 2.15 $AALS

### Description:

This command is used to read the status of the latched Digital Output and Digital Input channels 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 latched channels |
| **S** | The status to be read:<br>0: Reads the status of the low latched channels<br>1: Reads the status of the high latched channels |

### Response:

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

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

| | |
|---|---|
| **!** | Delimiter character to indicate that the command was valid |
| **?** | Delimiter character to indicate that the command was invalid |
| **AA** | The address of the responding module in hexadecimal format (00 to FF) |
| **(Data)** | The status of the latched Digital Output and Digital Input channels, represented by a four-digit hexadecimal value followed by 00. See Section 1.6 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: $01L0 Response: !010100

Reads the status of the low latched Digital Output and Digital Input channels of module 01 and returns a response indicating that the command was valid, with a value of 0101 denoting that the latched values for the both the Digital Output and Digital Input channels have recently been set to ON.

Command: $01C Response: !01

Clears the status of the latched Digital Output and Digital Input channels of module 01 and returns a response indicating that the command was valid.

Command: $01L0 Response: !000000

Reads the status of the low latched Digital Output and Digital Input channels of module 01 and returns a response indicating that the command was valid, with a value of 0000 denoting that the latched value has recently been set to ON .

Command: #03C Response: ?03

Attempts to clear the status of the latched Digital Input channels of module 03 and returns a response indicating that the command was invalid because module 03 does not exist.

### Related Commands:

Section 2.12 $AAC

### Related Topics:

Section 1.6 Software Configuration Tables

## 2.16 $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 set in hexadecimal format (00 to FF) |
| **M** | The command to read the name of the module |

### Response:

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

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

| | |
|---|---|
| **!** | Delimiter character to indicate that the command was valid |
| **?** | Delimiter character to indicate that the command was invalid |
| **AA** | The address of the responding module in hexadecimal format (00 to FF) |
| **(Data)** | A string indicating 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: $02M Response: !02LC101

Reads the name of module 02 and returns a response indicating that the command was valid, and that the name of the module is “LC-101”.

Command: $03M Response: ?03

Attempts to read the name of module 03 and returns a response indicating that the command was invalid because module 03 does not exist.

## 2.17 $AAP

**Description:**

This command is used to read the communication protocol information for 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 information

**Response:**

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

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

**!** Delimiter character to indicate that the command was valid

**?** Delimiter character to indicate that the command was invalid

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

**S** The protocols supported by the module:

0: Only the DCON protocol is supported

1: Both the DCON and Modbus RTU protocols are supported

**C** The current protocol that is saved in the EEPROM that will be used at the next power-on reset:

0: The protocol saved in the EEPROM is DCON

1: The protocol saved 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: $01P Response: !0110

Reads the communication protocol information for module 01 returns a response indicating that the command was valid, with a value of 10, which denotes that the module supports both the DCON and Modbus RTU protocols and that the protocol that will be used at the next power-on reset is DCON.

Command: $03P Response: ?03

Attempts to read the communication protocol information for module 03 and returns a response indicating that the command was invalid because module 03 does not exist.

**Related Commands:**

Section 2.18 $AAPN

**Related Topics:**

Section 1.5 DIP Switch Configuration

## 2.18 $AAPN

**Description:**

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

**Syntax:**

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

| | |
|---|---|
| **$** | Delimiter character |
| **AA** | The address of the module to be read in hexadecimal format (00 to FF) |
| **P** | The command to set the communication protocol |
| **N** | The protocol to be used:<br>0: DCON Protocol<br>1: Modbus RTU Protocol |

Note that before using this command, the rear slide switch must be in the INIT position. The new protocol information will be saved in the EEPROM and will become effective after the next power-on reset.

**Response:**

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

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

| | |
|---|---|
| **!** | Delimiter character to indicate that the command was valid |
| **?** | Delimiter character to indicate that the command was invalid |
| **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 to be used for module 01 to Modbus RTU, but returns a response indicating that the command was invalid because the module is not in INIT mode.

Command: $01P1 Response: !01

Sets the communication protocol to be used for module 01 to Modbus RTU and returns a response indicating that the command was valid.

**Related Commands:**

Section 2.17 $AAP

**Related Topics:**

Section 1.5 DIP Switch Configuration

## 2.19 @AA

**Description:**

This command is used to read the status of both the Digital Output and Digital 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 Command: **>(Data)[CHKSUM](CR)**

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

| | |
|---|---|
| **>** | Delimiter character to indicate that the command was valid |
| **?** | Delimiter character to indicate that the command was invalid |
| **AA** | The address of the responding module in hexadecimal format (00 to FF) |
| **(Data)** | The status of the Digital Output and Digital Input channels represented by a four-digit hexadecimal value. The first two digits represent the status of the Digital Output channels and the second two represent the status of the Digital Input channels. See Section 1.7 for more 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: @02 Response: >0101

Reads the status of the Digital Output and Digital Input channels for module 02 and returns a response indicating that the command was valid, and that the current Digital Output value is 01 and the current Digital Input value is 01 denoting that both the Digital Output and Digital Input channels are ON..

Command: @03 Response: ?03

Attempts to read the status of the Digital Output and Digital Input channels for module 03 and returns a response indicating that the command was invalid because module 03 does not exist.

**Related Commands:**

Section 2.11 $AA6, Section 2.20 @AA(Data)

**Related Topics:**

Section 1.7 Digital Input/Output Data Format Settings

## 2.20 @AA(Data)

**Description:**

This command is used to set the value for all Digital Output channels of a specified module.

**Syntax:**

**@AA(Data)[CHKSUM](CR)**

| | |
|---|---|
| **@** | Delimiter character |
| **AA** | The address of the module to be set in hexadecimal format (00 to FF) |
| **(Data)** | A single-digit hexadecimal value representing the data to be written to the Digital Output channels, where bit 0 of the value corresponds to channel RL1. When the bit is 0, it denotes that the Digital Output channel is set to OFF, and 1 denotes that the Digital Output channel is set to ON. |

**Response:**

Valid Command: **>[CHKSUM](CR)**

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

Ignored Command: **![CHKSUM](CR)**

| | |
|---|---|
| **>** | Delimiter character to indicate that the command was valid |
| **?** | Delimiter character to indicate that the command was invalid |
| **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: @021 Response: **>**

Sets channel RL1 of module 02 to ON, and the module returns a response indicating that the command was valid.

Command: #020A02 Response: ?02

Attempts to set channel RL2 of module 02 to ON, but the module returns a response indicating that the command was invalid because channel RL2 does not exist.

### Related Commands:

Section 2.3 #AA00(Data), Section 2.4 #AA0A(Data), Section 2.5 #AA10DD, Section 2.6 #AAA0DD, Section 2.19 @AA

### Related Topics:

Section 1.7 Digital Input/Output Data Format Settings

### Notes:

This command is only applicable to modules that include Digital Output channels.

## 2.21 $AALC0N

**Description:**

This command is used to coordinate the operation status between the Digital Input and the Digital Output for a specified module.

**Syntax:**

**$AALC0N[CHKSUM](CR)**

| | |
|---|---|
| **$** | Delimiter character |
| **AA** | The address of the module to be set in hexadecimal format (00 to FF) |
| **LC0** | The command to coordinate the operation status between the Digital Input and the Digital Output |
| **N** | The command to set the operating status<br>0: The ON/OFF sequence of the Digital Output will **NOT** be coordinated when the status of the Digital Input is changed<br>1: The ON/OFF sequence of the Digital Output will be coordinated when the status of the Digital Input is changed |

**Note**:

When the status of the Digital Input is changed, the Digital Output will be set to either ON or OFF depending on the current status, i.e., if the Digital Output is currently set to ON, it will be set to OFF when the status of the Digital Input changes, and vice versa.

**Response:**

Valid Command: **>AA[CHKSUM](CR)**

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

| | |
|---|---|
| **!** | Delimiter character to indicate that the command was valid |
| **?** | Delimiter character to indicate that the command was invalid |
| **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: $01LC01 Response: !01

Coordinates the operation between the Digital Input and the Digital Output of module 01, and the module returns a response indicating that the command was valid.

**Related Commands:**

Section 2.22 $AALC1

## 2.22 $AALC1

**Description:**

This command is used to read whether the operation between the Digital Input and the Digital Output for a specified module is coordinated.

**Syntax:**

**$AALC1[CHKSUM](CR)**

| | |
|---|---|
| **$** | Delimiter character |
| **AA** | The address of the module to be read in hexadecimal format (00 to FF) |
| **LC1** | The command to read whether the operation between the Digital Input and the Digital Output is coordinated |

**Response:**

Valid Command: **>AAN[CHKSUM](CR)**

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

Ignored Command: **!AA[CHKSUM](CR)**

| | |
|---|---|
| **!** | Delimiter character to indicate that the command was valid |
| **?** | Delimiter character to indicate that the command was invalid |
| **AA** | The address of the responding module in hexadecimal format (00 to FF) |
| **N** | The operation status<br>0: The ON/OFF sequence of the Digital Output is NOT coordinated when the status of the Digital Input is changed<br>1: The ON/OFF sequence of the Digital Output is coordinated when the status of the Digital Input is changed |

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: $01LC1 Response: !011

Reads whether the operation between the Digital Input and the Digital Output is coordinated and the module returns a response indicating that the command was valid with a value of 1 meaning that the sequence of the Digital Output will be coordinated when the status of the Digital Input is changed.

**Related Commands:**

Section 2.21 $AALC0N

## 2.23 $AALC2NNNN

**Description:**

This command is used to set the Active Delay Time for the Digital Output of a specified module.

**Syntax:**

**$AALC2NNNN[CHKSUM](CR)**

| | |
|---|---|
| **$** | Delimiter character |
| **AA** | The address of the module to be set in hexadecimal format (00 to FF) |
| **LC2** | The command to set the Active Delay Time for the Digital Output |
| **NNNN** | A four-digit hexadecimal value representing the Active Delay Time in milliseconds. The maximum delay time is 0x0BB8 (3000 milliseconds). |

**Response:**

Valid Command: **>AA[CHKSUM](CR)**

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

Ignored Command: **!AA[CHKSUM](CR)**

| | |
|---|---|
| **!** | Delimiter character to indicate the command was valid |
| **?** | Delimiter character to indicate the command was invalid |
| **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: $01LC203E8 Response: !01

Sets the Active Delay Time for the Digital Output of module 01 to 0x03E8 (1000 milliseconds) and the module returns a response indicating that the command was valid. The Digital Output will be active 1000 milliseconds after the module is powered on.

Command: $01LC20BB9 Response: ?01

Attempts to set the Active Delay Time for the Digital Output of module 01 to 0x0BB9 (3001 milliseconds), but the module returns a response indicating that the command was invalid because the value for the Active Delay Time was not within the valid

range.

**Related Commands:**

Section 2.24 $AALC3

## 2.24 $AALC3

**Description:**

This command is used to read the Active Delay Time for the Digital Output of a specified module.

**Syntax:**

**$AALC3[CHKSUM](CR)**

| | |
|---|---|
| **$** | Delimiter character |
| **AA** | The address of the module to be read in hexadecimal format (00 to FF) |
| **LC3** | The command to read the Active Delay Time for the Digital Output |

**Response:**

Valid Command: **>AANNNN[CHKSUM](CR)**

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

| | |
|---|---|
| **!** | Delimiter character to indicate that the command was valid |
| **?** | Delimiter character to indicate that the command was invalid |
| **AA** | The address of the responding module in hexadecimal format (00 to FF) |
| **NNNN** | A four-digit hexadecimal value representing the Active Delay Time in milliseconds |

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: $01LC20BB8 Response: !01

Sets the Active Delay Time for the Digital Output of module 01 to 0x0BB8 (3000 milliseconds) and the module returns a response indicating that the command was valid. The Digital Output will be active 3000 milliseconds after the module is powered on.

Command: $01LC3 Response: !010BB8

Reads the Active Delay Time for the Digital Output of module 01 and returns a response indicating that the command was valid, with a value of 0BB8 meaning that the Active Delay Time is 3000 milliseconds.

**Related Commands:**

Section 2.23 $AALC2NNNN

## 2.25 ~AAD

**Description:**

This command is used to read whether the Digital Input/Output signal for a specified module is active or inactive.

**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 whether the Digital Input/Output is active or inactive

**Response:**

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

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

**!** Delimiter character to indicate that the command was valid

**?** Delimiter character to indicate that the command was invalid

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

**VV** A two-digit hexadecimal value representing the status of the Digital Input/Output signal

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| Reserved | | | | | | OAS | IAS |

| Key | Description |
|---|---|
| OAS | Specifies the status of the Digital Output signal<br>0: an output value of 0 indicates that the relay is inactive<br>an output value of 1 indicates that the relay is active<br>1: an output value of 0 indicates that the relay is active<br>an output value of 1 indicates that the relay is inactive |
| IAS | Specifies the status of the Digital Input signal<br>0: an input value of 0 indicates that the voltage is high<br>an input value of 1 indicates that there is no signal or the voltage is low<br>1: an input value of 0 indicates that there is no signal or the voltage is low<br>an input value of 1 indicates that the voltage is high |

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: ~02D03 Response: !02

Sets the Digital Input/Output signals for module 02 to 03, which denotes that the Digital Output channels are in inactive mode, and returns a response indicating that the command was valid,.

Command: ~02D Response: !0203

Reads the status of the Digital Input/Output signals for module 02 and returns a response indicating that the command was valid, with a value of 03, which denotes that the Digital Output channels are in inactive mode.

**Related Commands:**

Section 2.26 ~AADVV

## 2.26 ~AADVV

**Description:**

This command is used to set the Digital Input/Output signal for a specified module to active or inactive.

**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 Digital Input/Output to active or inactive

**VV** A two-digit hexadecimal value representing the status of the Digital Input/Output

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| Reserved | | | | | | OAS | IAS |

| Key | Description |
|---|---|
| OAS | Specifies the status of the Digital Output signal<br>0: an output value of 0 indicates that the relay is inactive<br>an output value of 1 indicates that the relay is active<br>1: an output value of 0 indicates that the relay is active<br>an output value of 1 indicates that the relay is inactive |
| IAS | Specifies the status of the Digital Input signal<br>0: an input value of 0 indicates that the voltage is high<br>an input value of 1 indicates that there is no signal or the voltage is low<br>1: an input value of 0 indicates that there is no signal or the voltage is low<br>an input value of 1 indicates that the voltage is high |

**Response:**

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

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

**!** Delimiter character to indicate that the command was valid

**?** Delimiter character to indicate that the command was invalid

**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: ~02D03 Response: !02

Sets the Digital Input/Output signals for module 02 to 03, which denotes that the Digital Output channels are in inactive mode, and returns a response indicating that the command was valid,.

Command: ~02D Response: !0203

Reads the status of the Digital Input/Output signals for module 02 and returns a response indicating that the command was valid, with a value of 03, which denotes that the Digital Output channels are in inactive mode.

**Related Commands:**

Section 2.25 ~AAD

## 2.27 ~AA4P

**Description:**

This command is used to read the Digital Output power-on value for a specified module.

**Syntax:**

**~AA4P[CHKSUM](CR)**

| | |
|---|---|
| **~** | Delimiter character |
| **AA** | The address of the module to be read in hexadecimal format (00 to FF) |
| **4P** | The command to read the Digital Output power-on value |

**Response:**

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

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

| | |
|---|---|
| **!** | Delimiter character to indicate that the command was valid |
| **?** | Delimiter character to indicate that the command was invalid |
| **AA** | The address of the responding module in hexadecimal format (00 to FF) |
| **(Data)** | The Digital Output power-on 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.

**Examples:**

Command: ~035P Response: !03

Sets the current Digital Output value as the power-on value for module 03, and returns a response indicating that the command was valid.

Command: ~034P Response: !030100

Reads the Digital Output power-on value and the Digital Output safe value for module 03 and returns a response indicating that the command was valid, with a value of 0100, which denotes that the Digital Output power-on value is 01.

**Related Commands:**

Section 2.28 ~AA5P

## 2.28 ~AA5P

**Description:**

This command is used to set the current Digital Output value as the power-on value for a specified module.

**Syntax:**

**~AA5P[CHKSUM](CR)**

| | |
|---|---|
| **~** | Delimiter character |
| **AA** | The address of the module to be set in hexadecimal format (00 to FF) |
| **5P** | The command to set the current Digital Output value as the power-on value |

**Response:**

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

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

| | |
|---|---|
| **!** | Delimiter character to indicate that the command was valid |
| **?** | Delimiter character to indicate that the command was invalid |
| **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: @031 Response: >

Sets Digital Output channel RL1 of module 03 to ON, and the module returns a response indicating that the command was valid.

Command: ~035P Response: !03

Sets the current Digital Output value as the power-on value for module 03, and returns a response indicating that the command was valid.

**Related Commands:**

Section 2.27 ~AA4P

## 2.29 ~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 Command: **!AA(Data)[CHKSUM](CR)**

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

| | |
|---|---|
| **!** | Delimiter character to indicate that the command was valid |
| **?** | Delimiter character to indicate that the command was invalid |
| **AA** | The address of the responding module in hexadecimal format (00 to FF) |
| **(Data)** | A two-digit hexadecimal value representing the Response Delay Time. The valid range is 00 to 1E in 1 ms intervals. |

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: ~03RD1E Response: !03

Sets the Response Delay Time for module 03 to 1E (30 ms), and returns a response indicating that the command was valid.

Command: ~03RD Response: !0300

Reads the Response Delay Time for module 03 and returns a response indicating that the command was valid, with a value of 00 (0 ms).

### Related Commands:

Section 2.30 ~AARDVV

## 2.30 ~AARDVV

**Description:**

This command is used to set the Response Delay Time for a specified module.

**Syntax:**

**~AARDvv [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. |
| **VV** | A two-digit hexadecimal value representing the Response Delay Time in milliseconds. The valid range is 00 to 1E in 1 ms intervals. |

**Response:**

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

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

| | |
|---|---|
| **!** | Delimiter character to indicate that the command was valid |
| **?** | Delimiter character to indicate that the command was invalid |
| **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:~03RD1E Response:!03

Sets the Response Delay Time for module 03 to 1E (30 ms), and returns a response indicating that the command was valid.

Command:~03RD Response:!0300

Attempts to set the Response Delay Time for module 03 to 1F (31 ms), but the module returns a response indicating that the command was invalid because the Response Delay Time was not within the valid range.

**Related Commands:**

Section 2.29 ~AARD

# 3 Modbus RTU Protocol

The Modbus protocol was originally developed for Modicon controllers by Modicon Inc. Detailed information related to the Modbus RTU protocol can be found at http://www2.schneider-electric.com/sites/corporate/en/products-services/automation-control/automation-control.page. You can also visit http://www.modbus.org for more valuable information.

The LC-101 module supports the Modbus RTU protocol, with communication Baud Rates ranging from 1200 bps to 115200 bps. The parity, data bits and stop bits are fixed as no parity, 8 data bits and 1 stop bit. The following Modbus functions are supported.

| Function Code | Description | Section |
|---|---|---|
| 0x01 | Reads the Coils | 3.1 |
| 0x02 | Reads the Discrete Inputs | 3.2 |
| 0x03 | Reads Multiple Registers | 3.3 |
| 0x04 | Reads Multiple Input Registers | 3.4 |
| 0x05 | Writes a Single Coil | 3.5 |
| 0x06 | Writes a Single Register | 3.6 |
| 0x0F | Writes Multiple Coils | 3.7 |
| 0x10 | Writes Multiple Registers | 3.8 |
| 0x46 | Reads/writes the Module Settings | 3.9 |

If the function specified in the message is not supported, then the module responds as below. Note that the address mapping for the Modbus protocol is Base 0.

**Error Response**

| Byte | Description | Length (in Bytes) | Value |
|---|---|---|---|
| 00 | Address | 1 | 1 to 247 |
| 01 | Function Code | 1 | Function Code + 0x80 |
| 02 | Exception Code | 1 | 01 |

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

## Modbus Address Mapping

<table>
<tr><th>Address</th><th>Description</th><th>Attribute</th></tr>
<tr><td>00001, 10001</td><td>Reads the current status of the Digital Output or sets the Digital Output to either active or inactive</td><td>R/W</td></tr>
<tr><td>00161, 10161</td><td>Reads/sets the Digital Output Power-on Value</td><td>R/W</td></tr>
<tr><td>00257, 10257</td><td>Reads/sets the Communication Protocol<br>0: DCON<br>1: Modbus RTU</td><td>R/W</td></tr>
<tr><td>00264</td><td>Clears the latched Digital Input and Digital Output channels. Write 1 to clear.</td><td>W</td></tr>
<tr><td>00274, 10274</td><td>Enables or disables the Digital Output ON and OFF sequence when the status of the Digital Input is changed.<br>0: Disabled<br>1: Enabled</td><td>R/W</td></tr>
<tr><td>00513</td><td>Clears the Digital Input Counter</td><td>W</td></tr>
<tr><td>10032</td><td>Digital Input Channel</td><td>R</td></tr>
<tr><td>10064</td><td>Reads the Status of the High Latched Digital Input and Digital Output Channels</td><td>R</td></tr>
<tr><td>10096</td><td>Reads the Status of the Low Latched Digital Input and Digital Output Channels</td><td>R</td></tr>
<tr><td>10273</td><td>Reads the Reset Status<br>0: This is NOT the first time the module has been read since being powered on<br>1: This is the first time the module has been read since being powered on</td><td>R</td></tr>
<tr><td>30001</td><td>Reads the Digital Input Counter Value</td><td>R</td></tr>
<tr><td>40481-40482</td><td>Reads the Firmware Version</td><td>R</td></tr>
<tr><td>40483-40484</td><td>Reads the Name of the Module</td><td>R</td></tr>
<tr><td>30485, 40485</td><td>Reads/sets the Module address. The valid range is 1 to 247.</td><td>R/W</td></tr>
<tr><td>40486</td><td>Reads/sets the Baud Rate and the Data Format<br>Bits 5:0 (Baud Rate)<br><table><tr><td>Code</td><td>0x03</td><td>0x04</td><td>0x05</td><td>0x06</td></tr><tr><td>Baud</td><td>1200</td><td>2400</td><td>4800</td><td>9600</td></tr><tr><td>Code</td><td>0x07</td><td>0x08</td><td>0x09</td><td>0x0A</td></tr><tr><td>Baud</td><td>19200</td><td>38400</td><td>57600</td><td>115200</td></tr></table></td><td>R/W</td></tr>
</table>

| | | |
|---|---|---|
| | Baud Rate, valid range: 0x03 to 0x0A<br>Bits 7:6 (Data Format)<br>00: no parity, 1 stop bit<br>01: no parity, 2 stop bits<br>10: even parity, 1 stop bit<br>11: odd parity, 1 stop bit | |
| 30488, 40488 | Reads/sets the Response Delay Time in milliseconds. The valid range is 0 to 30 ms | R/W |
| 30498, 40498 | Reads/sets the Digital Output Active Delay Time in milliseconds. The valid range: 0 to 0xBB8. | R/W |

Modbus RTU Function Description:
(0xxxx): 0x05, 0x0F Function Code
(1xxxx): 0x01 Function Code
(3xxxx): 0x06, 0x10 Function Code
(4xxxx): 0x03 Function Code

## 3.1 01 (0x01) Read Coils

This function code is used to read the value at addresses 0xxxx and 1xxxx.

**Request**

| Byte | Description | Length (in Bytes) | Value |
|---|---|---|---|
| 00 | Address | 1 | 1 to 247 |
| 01 | Function Code | 1 | 0x01 |
| 02 - 03 | Starting Address | 2 | Refer to the Modbus Address Mapping Table for details. |
| 04 - 05 | Number of Addresses Requested | 2 | 0x0001 to 0x0001 + *N |

*N = Number of channels requested

**Response**

| Byte | Description | Length (in Bytes) | Value |
|---|---|---|---|
| 00 | Address | 1 | 1 to 247 |
| 01 | Function Code | 1 | 0x01 |
| 02 | Byte Count | 1 | *N |
| 03 | Value of the Requested Address | *N | |

*N = (Number of channels requested / 8)

**Error Response**

| Byte | Description | Length (in Bytes) | Value |
|---|---|---|---|
| 00 | Address | 1 | 1 to 247 |
| 01 | Function Code | 1 | 0x81 |
| 02 | Exception Code | 1 | Refer to the Modbus standard for more details. |

## 3.2 02 (0x02) Read Discrete Inputs

This function code is used to read the value at address 1xxxx.

**Request**

| Byte | Description | Length (in Bytes) | Value |
|---|---|---|---|
| 00 | Address | 1 | 1 to 247 |
| 01 | Function Code | 1 | 0x02 |
| 02 - 03 | Starting Address | 2 | 0x0020 to 0x003F |
| 04 - 05 | Number of Addresses Requested | 2 | 0x0001 to 0x0001 + *N |

*N = Number of channels requested

**Response**

| Byte | Description | Length (in Bytes) | Value |
|---|---|---|---|
| 00 | Address | 1 | 1 to 247 |
| 01 | Function Code | 1 | 0x02 |
| 02 | Byte Count | 1 | *N |
| 03 | Value of the Requested Address | *N | |

*N = (Number of channels requested / 8)

**Error Response**

| Byte | Description | Length (in Bytes) | Value |
|---|---|---|---|
| 00 | Address | 1 | 1 to 247 |
| 01 | Function Code | 1 | 0x82 |
| 02 | Exception Code | 1 | Refer to the Modbus standard for more details. |

## 3.3 03 (0x03) Read Multiple Registers

This function code is used to read the value at addresses 3xxxx and 4xxxx.

**Request**

| Byte | Description | Length (in Bytes) | Value |
|---|---|---|---|
| 00 | Address | 1 | 1 to 247 |
| 01 | Function Code | 1 | 0x03 |
| 02 - 03 | Starting Address | 2 | Refer to the Modbus Address Mapping Table for details. |
| 04 - 05 | Number of Addresses Requested | 2 | 0x0001 to 0x0001 + *N |

*N = Number of channels requested

**Response**

| Byte | Description | Length (in Bytes) | Value |
|---|---|---|---|
| 00 | Address | 1 | 1 to 247 |
| 01 | Function Code | 1 | 0x03 |
| 02 | Byte Count | 1 | *N x 2 |
| 03 - | Value of the Requested Address | *N x 2 | . |

*N = Number of channels requested

**Error Response**

| Byte | Description | Length (in Bytes) | Value |
|---|---|---|---|
| 00 | Address | 1 | 1 to 247 |
| 01 | Function Code | 1 | 0x83 |
| 02 | Exception Code | 1 | Refer to the Modbus standard for more details. |

## 3.4 04 (0x04) Read Multiple Input Registers

This function code is used to read the value at address 4xxxx.

**Request**

| Byte | Description | Length (in Bytes) | Value |
|---|---|---|---|
| 00 | Address | 1 | 1 to 247 |
| 01 | Function Code | 1 | 0x04 |
| 02 - 03 | Starting Address | 2 | Refer to the Modbus Address Mapping Table for details. |
| 04 - 05 | Number of Addresses Requested | 2 | 0x0001 to 0x0001 + *N |

*N = Number of channels requested

**Response**

| Byte | Description | Length (in Bytes) | Value |
|---|---|---|---|
| 00 | Address | 1 | 1 to 247 |
| 01 | Function Code | 1 | 0x04 |
| 02 | Byte Count | 1 | *N x 2 |
| 03 - | Value of the Requested Address | *N x 2 | |

*N = Number of channels requested

**Error Response**

| Byte | Description | Length (in Bytes) | Value |
|---|---|---|---|
| 00 | Address | 1 | 1 to 247 |
| 01 | Function Code | 1 | 0x84 |
| 02 | Exception Code | 1 | Refer to the Modbus standard for more details. |

## 3.5 05 (0x05) Write Single Coil

This function code is used to write a value to address 0xxxx.

**Request**

| Byte | Description | Length (in Bytes) | Value |
|---|---|---|---|
| 00 | Address | 1 | 1 to 247 |
| 01 | Function Code | 1 | 0x05 |
| 02 - 03 | Starting Address | 2 | Refer to the Modbus Address Mapping Table for details. |
| 04 - 05 | Requested Value | 2 | A value of 0xFF00 will set the output to ON. A value of 0x0000 will set it to OFF. All other values are invalid and will not affect the coil. |

**Response**

| Byte | Description | Length (in Bytes) | Value |
|---|---|---|---|
| 00 | Address | 1 | 1 to 247 |
| 01 | Function Code | 1 | 0x05 |
| 02 - 03 | Requested Address | 2 | The value is the same as bytes 02 and 03 of the Request |
| 04 - 05 | Value of the Requested Address | 2 | The value is the same as bytes 04 and 05 of the Request |

**Error Response**

| Byte | Description | Length (in Bytes) | Value |
|---|---|---|---|
| 00 | Address | 1 | 1 to 247 |
| 01 | Function Code | 1 | 0x85 |
| 02 | Exception Code | 1 | Refer to the Modbus standard for more details. |

## 3.6 05 (0x06) Write Single Register

This function code is used to write a value to address 3xxxx.

**Request**

| Byte | Description | Length (in Bytes) | Value |
|---|---|---|---|
| 00 | Address | 1 | 1 to 247 |
| 01 | Function Code | 1 | 0x05 |
| 02 - 03 | Starting Address | 2 | Refer to the Modbus Address Mapping Table for details. |
| 04 - 05 | Write Value | 2 | |

**Response**

| Byte | Description | Length (in Bytes) | Value |
|---|---|---|---|
| 00 | Address | 1 | 1 to 247 |
| 01 | Function Code | 1 | 0x06 |
| 02 - 03 | Requested Address | 2 | The value is the same as bytes 02 and 03 of the Request |
| 04 - 05 | Value of the Requested Address | 2 | The value is the same as bytes 04 and 05 of the Request |

**Error Response**

| Byte | Description | Length (in Bytes) | Value |
|---|---|---|---|
| 00 | Address | 1 | 1 to 247 |
| 01 | Function Code | 1 | 0x86 |
| 02 | Exception Code | 1 | Refer to the Modbus standard for more details. |

## 3.7 15 (0x0F) Write Multiple Coils

This function code is used to write a value to address 0xxxx.

**Request**

| Byte | Description | Length (in Bytes) | Value |
|---|---|---|---|
| 00 | Address | 1 | 1 to 247 |
| 01 | Function Code | 1 | 0x0F |
| 02 - 03 | Starting Address | 2 | Refer to the Modbus Address Mapping Table for details. |
| 04 - 05 | Number of Addresses Requested | 2 | 0x0001 to 0x0001 + *N |
| 06 | Byte Count | 1 | *N/8 |
| 07 | Write Value | 1 | A bit corresponds to a channel. If the bit is 1, it denotes that the channel that was set is ON. If the bit is 0, it denotes that the channel that was set is OFF. |

*N = Number of channels requested

**Response**

| Byte | Description | Length (in Bytes) | Value |
|---|---|---|---|
| 00 | Address | 1 | 1 to 247 |
| 01 | Function Code | 1 | 0x0F |
| 02 - 03 | Starting Address | 2 | The value is the same as byte 02 and 03 of the Request |
| 04 - 05 | Number of Addresses Requested | 2 | The value is the same as byte 04 and 05 of the Request |

**Error Response**

| Byte | Description | Length (in Bytes) | Value |
|---|---|---|---|
| 00 | Address | 1 | 1 to 247 |
| 01 | Function Code | 1 | 0x8F |
| 02 | Exception Code | 1 | Refer to the Modbus standard for more details. |

## 3.8 16 (0x10) Write Multiple Registers

This function code is used to write a value to address 0xxxx.

**Request**

| Byte | Description | Length (in Bytes) | Value |
|---|---|---|---|
| 00 | Address | 1 | 1 to 247 |
| 01 | Function Code | 1 | 0x0F |
| 02 - 03 | Starting Address | 2 | Refer to the Modbus Address Mapping Table for details. |
| 04 - 05 | Number of Addresses Requested | 2 | 0x0001 to 0x0001 + *N |
| 06 | Byte Count | 1 | *N x 2 |
| 07 | Write Value | *N x 2 | |

*N = Number of channels requested

**Response**

| Byte | Description | Length (in Bytes) | Value |
|---|---|---|---|
| 00 | Address | 1 | 1 to 247 |
| 01 | Function Code | 1 | 0x010 |
| 02 - 03 | Starting Address | 2 | The value is the same as byte 02 and 03 of the Request |
| 04 - 05 | Number of Addresses Requested | 2 | The value is the same as byte 04 and 05 of the Request |

**Error Response**

| Byte | Description | Length (in Bytes) | Value |
|---|---|---|---|
| 00 | Address | 1 | 1 to 247 |
| 01 | Function Code | 1 | 0x91 |
| 02 | Exception Code | 1 | Refer to the Modbus standard for more details. |

## 3.9 70 (0x46) Read/Write Module Settings

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

| Sub-function Code | Description | Section |
|---|---|---|
| 00 (0x00) | Reads the Name of the Module | 3.9.1 |
| 04 (0x04) | Sets the Module Address | 3.9.2 |
| 05 (0x05) | Reads the Communication Settings | 3.9.3 |
| 06 (0x06) | Sets the Communication Settings | 3.9.4 |
| 32 (0x20) | Reads the Firmware Version | 3.9.5 |
| 33 (0x21) | Sets the Digital Input Counter Edge | 3.9.6 |
| 34 (0x22) | Reads the Digital Input Counter Edge Settings Value | 3.9.7 |
| 39 (0x27) | Sets the Digital Output Power-on Value | 3.9.8 |
| 40 (0x28) | Reads the Digital Output Power-on Value | 3.9.9 |

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

**Error Response**

| Byte | Description | Length (in Bytes) | Value |
|---|---|---|---|
| 00 | Address | 1 | 1 to 247 |
| 01 | Function Code | 1 | 0xC6 |
| 02 | Exception Code | 1 | Refer to the Modbus standard for more details. |

### 3.7.1 Sub-function 00 (0x00) Read Module Name

This sub-function code is used to read the name of the LC-101 module.

**Request**

| Byte | Description | Length (in Bytes) | Value |
|---|---|---|---|
| 00 | Address | 1 | 1 to 247 |
| 01 | Function Code | 1 | 0x46 |
| 02 | Sub-function Code | 1 | 0x00 |

**Response**

| Byte | Description | Length (in Bytes) | Value |
|---|---|---|---|
| 00 | Address | 1 | 1 to 247 |
| 01 | Function Code | 1 | 0x46 |
| 02 | Sub-function Code | 1 | 0x00 |
| 03 - 06 | Module Name | 4 | 0x4C 0x43 0x01 0x01<br>(LC-101) |

**Error Response**

| Byte | Description | Length (in Bytes) | Value |
|---|---|---|---|
| 00 | Address | 1 | 1 to 247 |
| 01 | Function Code | 1 | 0xC6 |
| 02 | Exception Code | 1 | Refer to the Modbus standard for more details. |

### 3.7.2 Sub-function 04 (0x04) Write Module Address

This sub-function code is used to set the address of the LC-101 module.

**Request**

| Byte | Description | Length (in Bytes) | Value |
|---|---|---|---|
| 00 | Address | 1 | 1 to 247 |
| 01 | Function Code | 1 | 0x46 |
| 02 | Sub-function Code | 1 | 0x04 |
| 03 | New Address | 1 | 1 to 247 |
| 04 - 06 | Reserved | 3 | 0x00 0x00 0x00 |

**Response**

| Byte | Description | Length (in Bytes) | Value |
|---|---|---|---|
| 00 | Address | 1 | 1 to 247 |
| 01 | Function Code | 1 | 0x46 |
| 02 | Sub-function Code | 1 | 0x04 |
| 03 | New Address | 1 | 0: OK<br>Others: Error |
| 04 - 06 | Reserved | 3 | 0x00 0x00 0x00 |

**Error Response**

| Byte | Description | Length (in Bytes) | Value |
|---|---|---|---|
| 00 | Address | 1 | 1 to 247 |
| 01 | Function Code | 1 | 0xC6 |
| 02 | Exception Code | 1 | Refer to the Modbus standard for more details. |

### 3.7.3 Sub-function 05 (0x05) Read Communication Settings

This sub-function code is used to read the communication protocol settings for the LC-101 module.

**Request**

| Byte | Description | Length (in Bytes) | Value |
|---|---|---|---|
| 00 | Address | 1 | 1 to 247 |
| 01 | Function Code | 1 | 0x46 |
| 02 | Sub-function Code | 1 | 0x05 |
| 03 | Reserved | 1 | 0x00 |

**Response**

| Byte | Description | Length (in Bytes) | Value |
|---|---|---|---|
| 00 | Address | 1 | 1 to 247 |
| 01 | Function Code | 1 | 0x46 |
| 02 | Sub-function Code | 1 | 0x05 |
| 03 | Reserved | 1 | 0x00 |
| 04 | Baud Rate | 1 | Refer to the Baud Rate Settings table 錯誤! 尚未指定書籤名稱。below for details. |
| 05 - 07 | Reserved | 3 | 0x00 0x00 0x00 |
| 08 | Mode | 1 | 0: DCON Protocol<br>1: Modbus RTU Protocol |
| 09 - 10 | Reserved | 2 | 0x00 0x00 |

**Note**: This information is the data saved in the EEPROM and will be used for the next power-on reset. It is **NOT** the currently used settings.

**Baud Rate Settings:**

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

**Error Response**

| Byte | Description | Length (in Bytes) | Value |
|---|---|---|---|
| 00 | Address | 1 | 1 - 247 |
| 01 | Function Code | 1 | 0xC6 |
| 02 | Exception Code | 1 | Refer to the Modbus standard for more details. |

### 3.7.4 Sub-function 06 (0x06) Write Communication Settings

This sub-function code is used to configure the communication protocol for the LC-101 module.

**Request**

| Byte | Description | Length (in Bytes) | Value |
|---|---|---|---|
| 00 | Address | 1 | 1 - 247 |
| 01 | Function Code | 1 | 0x46 |
| 02 | Sub-function Code | 1 | 0x06 |
| 03 | Reserved | 1 | 0x00 |
| 04 | Baud Rate | 1 | Refer to the Baud Rate Settings table for details. |
| 05 - 07 | Reserved | 3 | 0x00 0x00 0x00 |
| 08 | Mode | 1 | 0: DCON Protocol<br>1: Modbus RTU Protocol |
| 09 - 10 | Reserved | 2 | 0x00 0x00 |

**Response**

| Byte | Description | Length (in Bytes) | Value |
|---|---|---|---|
| 00 | Address | 1 | 1 - 247 |
| 01 | Function Code | 1 | 0x46 |
| 02 | Sub-function Code | 1 | 0x06 |
| 03 | Reserved | 1 | 0x00 |
| 04 | Baud Rate | 1 | 0: OK<br>Others: Error |
| 05 - 07 | Reserved | 3 | 0x00 0x00 0x00 |
| 08 | Mode | 1 | 0: OK<br>Others: Error |
| 09 - 10 | Reserved | 2 | 0x00 0x00 |

**Note**: The new Baud Rate and Protocol settings will only become effective after the next power-on reset.

**Error Response**

| Byte | Description | Length (in Bytes) | Value |
|---|---|---|---|
| 00 | Address | 1 | 1 - 247 |
| 01 | Function Code | 1 | 0xC6 |
| 02 | Exception Code | 1 | Refer to the Modbus standard for more details. |

### 3.7.5 Sub-function 32 (0x20) Read Firmware Version

This sub-function code is used to read the firmware version information for the LC-101 module.

**Request**

| Byte | Description | Length (in Bytes) | Value |
|---|---|---|---|
| 00 | Address | 1 | 1 - 247 |
| 01 | Function Code | 1 | 0x46 |
| 02 | Sub-function Code | 1 | 0x20 |

**Response**

| Byte | Description | Length (in Bytes) | Value |
|---|---|---|---|
| 00 | Address | 1 | 1 - 247 |
| 01 | Function Code | 1 | 0x46 |
| 02 | Sub-function Code | 1 | 0x20 |
| 03 | Major Version | 1 | 0x00 - 0xFF |
| 04 | Minor Version | 1 | 0x00 - 0xFF |
| 05 | Build Version | 1 | 0x00 - 0xFF |

**Error Response**

| Byte | Description | Length (in Bytes) | Value |
|---|---|---|---|
| 00 | Address | 1 | 1 - 247 |
| 01 | Function Code | 1 | 0xC6 |
| 02 | Exception Code | 1 | Refer to the Modbus standard for more details. |

### 3.7.6 Sub-function 33 (0x21) Write Digital Input Counter Edge Settings

This sub-function code is used to set the Digital Input counter edge value for the LC-101 module.

**Request**

| Byte | Description | Length (in Bytes) | Value |
|---|---|---|---|
| 00 | Address | 1 | 1 to 247 |
| 01 | Function Code | 1 | 0x46 |
| 02 | Sub-function Code | 1 | 0x21 |
| 03 | Edge Setting Value | 1 | *0x00 - 0x0F |

* 0 = Falling Edge, 1 = Rising Edge. For example, 0x03 denotes that the counters for channels 0 and 1 are set to rising edge and those for channels 2 and 3 are set to falling edge.

**Response**

| Byte | Description | Length (in Bytes) | Value |
|---|---|---|---|
| 00 | Address | 1 | 1 to 247 |
| 01 | Function Code | 1 | 0x46 |
| 02 | Sub-function Code | 1 | 0x21 |
| 03 | Edge Setting Value | 1 | 0: OK<br>Others: Error |

**Error Response**

| Byte | Description | Length (in Bytes) | Value |
|---|---|---|---|
| 00 | Address | 1 | 1 to 247 |
| 01 | Function Code | 1 | 0xC6 |
| 02 | Exception Code | 1 | Refer to the Modbus standard for more details. |

### 3.7.7 Sub-function 34 (0x22) Read Digital Input Counter Edge Settings

This sub-function code is used to read the Digital Input counter edge value for the LC-101 module.

**Request**

| Byte | Description | Length (in Bytes) | Value |
|---|---|---|---|
| 00 | Address | 1 | 1 to 247 |
| 01 | Function Code | 1 | 0x46 |
| 02 | Sub-function Code | 1 | 0x22 |

**Response**

| Byte | Description | Length (in Bytes) | Value |
|---|---|---|---|
| 00 | Address | 1 | 1 to 247 |
| 01 | Function Code | 1 | 0x46 |
| 02 | Sub-function Code | 1 | 0x22 |
| 03 | Edge Setting Value | 1 | *0x00 - 0x0F |

*0 = Falling Edge, 1 = Rising Edge. For example, 0x03 denotes that the counters for channels 0 and 1 are set to rising edge and those for channels 2 and 3 are set to falling edge.

**Error Response**

| Byte | Description | Length (in Bytes) | Value |
|---|---|---|---|
| 00 | Address | 1 | 1 to 247 |
| 01 | Function Code | 1 | 0xC6 |
| 02 | Exception Code | 1 | Refer to the Modbus standard for more details. |

### 3.7.8 Sub-function 39 (0x27) Write Power-on Value

This sub-function code is used to set the power-on value for the LC-101 module.

**Request**

| Byte | Description | Length (in Bytes) | Value |
|---|---|---|---|
| 00 | Address | 1 | 1 to 247 |
| 01 | Function Code | 1 | 0x46 |
| 02 | Sub-function Code | 1 | 0x27 |
| 03 | Power-on Value | 1 | *0x00-0xFF |

*0x00~0x0F for M-7060/M-7060D

0x00~0x7F for M-7067/M-7067D

**Response**

| Byte | Description | Length (in Bytes) | Value |
|---|---|---|---|
| 00 | Address | 1 | 1 to 247 |
| 01 | Function Code | 1 | 0x46 |
| 02 | Sub-function Code | 1 | 0x28 |
| 03 | Power-on Value | 1 | 0: OK<br>Others: Error |

**Error Response**

| Byte | Description | Length (in Bytes) | Value |
|---|---|---|---|
| 00 | Address | 1 | 1 to 247 |
| 01 | Function Code | 1 | 0xC6 |
| 02 | Exception Code | 1 | Refer to the Modbus standard for more details. |

### 3.7.9 Sub-function 40 (0x28) Read Power-on Value

This sub-function code is used to read the power-on value for the LC-101 module.

**Request**

| Byte | Description | Length (in Bytes) | Value |
|---|---|---|---|
| 00 | Address | 1 | 1 to 247 |
| 01 | Function Code | 1 | 0x46 |
| 02 | Sub-function Code | 1 | 0x28 |

**Response**

| Byte | Description | Length (in Bytes) | Value |
|---|---|---|---|
| 00 | Address | 1 | 1 to 247 |
| 01 | Function Code | 1 | 0x46 |
| 02 | Sub-function Code | 1 | 0x28 |
| 03 | Power-on Value | 1 | *0x00 - 0xFF |

*0x00~0x0F for M-7060/M-7060D

0x00~0x7F for M-7067/M-7067D

**Error Response**

| Byte | Description | Length (in Bytes) | Value |
|---|---|---|---|
| 00 | Address | 1 | 1 to 247 |
| 01 | Function Code | 1 | 0xC6 |
| 02 | Exception Code | 1 | Refer to the Modbus standard for more details. |