Revision 1.2

July 24, 2017

Serial communication protocol for MDC & ADC

COM Protocol Manual

HANTAS

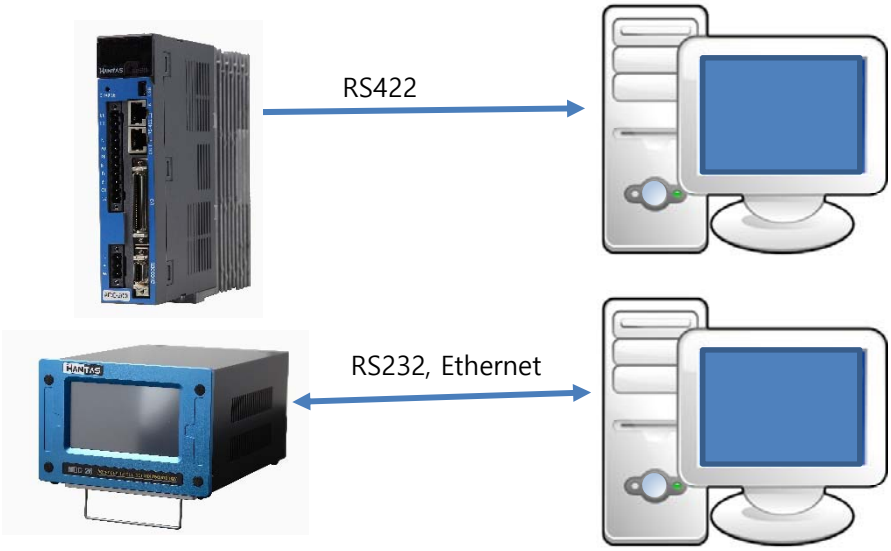
Human And Tightening Assembly Specialist

1 Overview and Communication Specifications

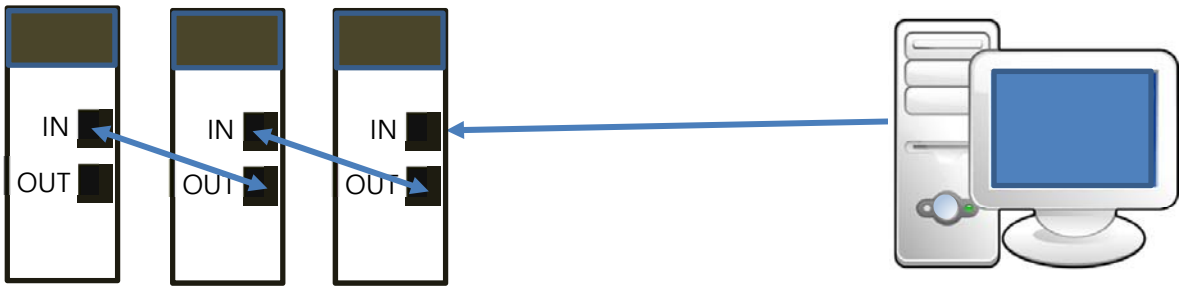
1.1 Overview

MDC, ADC is capable of connecting to the host controller (Handy Loader, HMI, PLC, PC, etc.) through RS232, RS422 serial communication or Ethernet, allowing the user to use such functions as parameter change and data monitoring.

◆ Series Communication Connection using RS232(MDC), RS422(ADC)



◆ Multi-Drop Connection using RS422 (up to 31 controllers max.)



The pins of IN & OUT port are connected by parallel together (1:1), allowing for convenient multi-drop wiring.

If the PC does not have Serial COM port, use good quality of USB to RS422 Serial converter.

USB to RS422 Serial converter.

1.2 Communication Specifications and Connection Diagram

◆ Communication Specifications

Item		Specifications
Communication Standard		ANSI/TIA/EIA-422 Standard
Communication Protocol		MODBUS-RTU (Remote Terminal Unit)
Data Type	Data bit	8 bit
	Stop bit	1 bit
	Parity	None
Synchronous		Asynchronous method
Transmission Speed		9600/19200/38400/57600/115200(MDC)[bps] Speed can be selected at communication speed setting [0x3002]
Transmission Distance		Up to 200 [m]
Power Consumption		Less than 100[㎃]

◆ Modbus-RTU Mode

RTU (Remote Terminal Unit) mode, each 8–bit byte in a message contains two 4–bit hexadecimal characters. The main advantage of this mode is that its greater character density allows better data throughput than ASCII for the same baud rate. Each message must be transmitted in a continuous stream.

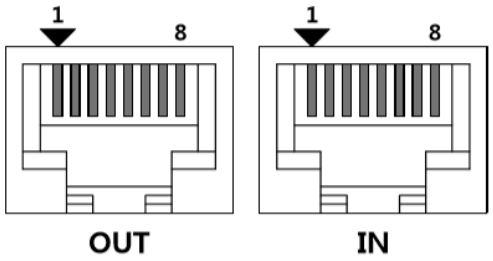
The format for each byte in RTU mode is:

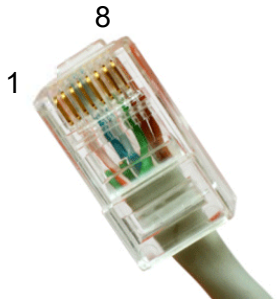
Coding System: 8–bit binary, hexadecimal 0–9, A–F Two hexadecimal characters contained in each 8–bit field of the message

Bits per Byte: 1 start bit 8 data bits, least significant bit sent first 1 bit for even/odd parity; no bit for no parity 1 stop bit if parity is used; 2 bits if no parity

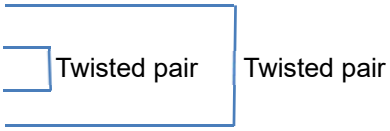
Error Check Field: Cyclical Redundancy Check (CRC)

◆ RS422 communication connector pin details (for ADC)





Pin no.	Description
1	No use
2	No use
3	RXD +
4	TXD -
5	TXD +
6	RXD -
7	No use (Note 1)
8	No use (Note 1)



The pare of TXD+ / TXD- (4,5) and RXD+ / RXD- (3,6) wires should be twisted wiring.

Note 1 : Never use these two pins to others. There is 5V power output for other purpose.

2 Basic Structure of Communication Protocol

ADC complies with the MODBUS-RTU Protocol for communication. For issues not specified in this manual, please see the related standards (Related Standard: Modbus Application Protocol Specification 1.1b, 2006.12.28)

In addition, the transmission (Tx) and reception (Rx) concepts are defined in reference to the host.

◆ Protocol Packet Descriptions

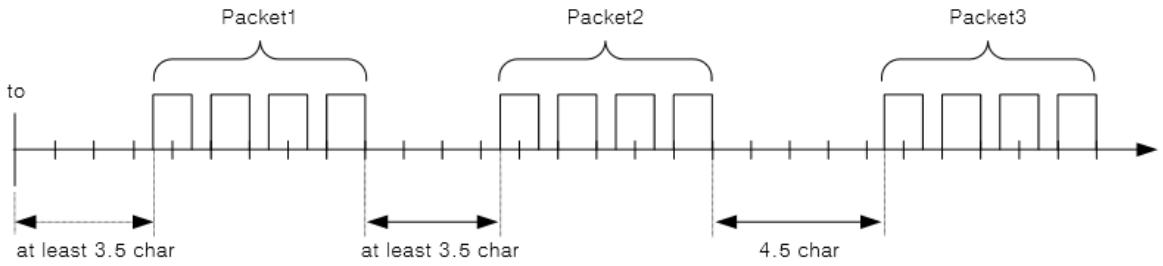
- Modbus RTU Frame Format

Name	Length(bits)	Function
Start	28	at least 3 ½ character times of silence (mark condition)
Address	8	Station Address
Function	8	Indicates the function code, eg read coils / inputs
Data	n * 8	Data + length will be filled depending on the message type
CRC	16	Error checks
End	28	at least 3 ½ character times of silence between frames

Transmission(Query)/Reception(Response) Packet Structure

Maximum length of transmission/reception packet of MODBUS-RTU is 256 Byte. Please make sure the total length of transmission/reception packet does not exceed 256 byte..

To classify packets, MODBUS-RTU Communication Mode requires empty spaces of at least 3.5 characters at the starting point and the end point.



ADC controller provide limit of maximum transmission up to 100 integers.

Modbus-RTU frame structure

Slave address (00 for TCP)	Function Code	Data	CRC (RTU) Low	CRC (RTU) High

◈ Slave address
◈ Function Code
◈ data
◈ CRC

How to Compute the Modbus RTU Message CRC

To ensure message data integrity, it is advisable to implement code that checks for serial port (UART) framing errors in addition to the verifying the message CRC. If the CRC in the received message does not match the CRC calculated by the receiving device, the message should be ignored. The C language code snippet below shows how to compute the Modbus message CRC using bit-wise shift and exclusive OR operations. The CRC is computed using every byte in the message frame except for the last two bytes which comprise the CRC itself.

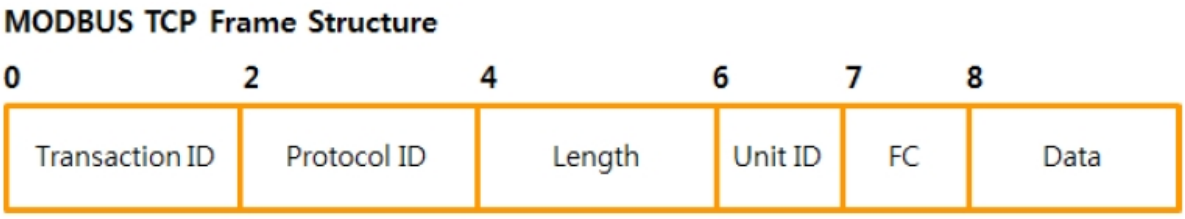
```
// Compute the MODBUS RTU CRC
UInt16 ModRTU_CRC(byte[] buf, int len)
{
  UInt16 crc = 0xFFFF;

  for (int pos = 0; pos < len; pos++) {
    crc ^= (UInt16)buf[pos];          // XOR byte into least sig. byte of crc

    for (int i = 8; i != 0; i--) {    // Loop over each bit
      if ((crc & 0x0001) != 0) {      // If the LSB is set
        crc >>= 1;                    // Shift right and XOR 0xA001
        crc ^= 0xA001;
      }
      else                            // Else LSB is not set
        crc >>= 1;                    // Just shift right
    }
  }
  // Note, this number has low and high bytes swapped, so use it accordingly (or swap bytes)
  return crc;
}
```

[On-line CRC calculation and free library and Help site]

- Modbus TCP Frame Format

Name	Length(byte)	Function
Transaction Identifier	2	For synchronization between messages of server & client
Protocol Identifier	2	Zero for Modbus/TCP
Length Field	2	Number of remaining bytes in this frame
Unit Identifier	1	Slave Address (use 0)
Function	1	Function codes as in other variants
Data	n	Data as response or commands

Format of Modbus/TCP frame is described in the below figure.



◈ byte 0 ~ 1: transaction ID (Transaction Identification)

This means the sequence number of queries and responses. While TCP operates as a master, it is incremented by one in every query (It doesn't matter if this field is set to 0x0000).

◈ byte 2 ~ 3: protocol ID (Protocol Identification)

This means the protocol identification and the value is fixed as 0x0000 for Modbus/TCP

◈ byte 4 ~ 5: length

The value of this means the number of bytes from next byte of length field to the end of the frame.

◈ byte 6: unit ID(Unit Identification)

◈ byte 7: FC (Function Code)

◈ byte 8~ : data depending on function code

2 Function code & message details

Function code	Description	Remark
03 (0x03)	**Read** Holding Register	16bit data (Integer) ex) parameter
04 (0x04)	**Read** Input Register	16bit data ex) monitoring data
06 (0x06)	**Write** Single Register	16bit integer format ex) parameter data
16 (0x10)	**Write** Multi Register	16bit integer format ex) parameter data
17 (0x11)	**Request** Slave ID	
100 (0x64)	***Request*** Graph date for MD	No use (Factory only)
200 (0xc8)	***Request*** Graph date for AD	No use (Factory only)

◆ < **READ** > Function code 03 & 04 details

Function code 03 & 04 is used to read the register as like parameters and alarm in the slave device. The only integer number is allowed.

[Query (Request)]

Slave address (00 for TCP)	Function Code	Start address High	Start address Low	No of address High	No of address Low	CRC (RTU) Low	CRC (RTU) High

Function code consist of one byte. But start address and number of address are consisted by 2 bytes with 4 digits of hexadecimal, starting with first 2 digits for high number, second 2 digits for low number.

[OK Response]

Slave address (00 for TCP)	Function Code	No of byte	Data #1 High	Data #1 Low	... x n data	CRC (RTU) Low	CRC (RTU) High

The number of data is consisted by 2 bytes with 4 digits of hexadecimal. So the total number of data is equal to 1/2 of the number of byte.

[Abnormal Respons]

Slave address (00 for TCP)	Function code +0x080	Error code	CRC (RTU) Low	CRC (RTU) High

By adding 0x080 to the function code, it response any abnormal or wrong message

[Address for parameters]

Refer the appendix A for all address details for parameters

1 – 250 : Parameter address

1001 – 1250 : Parameter value range MIN

2001 – 2250 : Parameter value range MAX

[Example message, Query & Response]

To read the data of parameter 1 and 2, which is torque & speed value of Preset #1

(03 : Query)

Slave address (00 for TCP)	03	00	02	00	02	CRC Low	CRC High

Start address : 0002 (hex) = 2 (dec)

Number of address : 0002 (hex) = 2 (dec)

Read (Function code 03) data of two addresses (number of address 0002) from the address starting from address #2 (0002)

(03 : Response)

Slave address (00 for TCP)	Function Code	No of byte	Data #1 High	Data #1 Low	... x n data	CRC Low	CRC High

Slave address (00 for TCP)	03	04	01	0F	03	21	CRC Low	CRC High

Data value of 1st address : 010F (hex) = 271 (dec) ← torque value of Preset #1

Date value of 2nd address : 03E8(hex) = 1000 (dec) ← Torque limit value 10.00%

[Address for monitoring]

Refer the appendix A for all address details for monitoring

3100 – 3199 : Alarm data

3200 – 3299 : Data updated by event (Start, F/L, Preset change, Torque up)

3300 – 3399 : Real-time data

◆ < **WRITE** > Function code 06 : writing parameters

Function code 06 is used to WRITE the parameter value in each register. The only integer number is allowed.

(Query)

Slave address (00 for TCP)	Function Code	Address High	Address Low	Date High	Data Low	CRC Low	CRC High

(OK Response)

Slave address (00 for TCP)	Function Code	Address High	Address Low	Date High	Data Low	CRC Low	CRC High

It provides the echo response on the query (request) after writing data in register.

- Refer the appendix A for all address details for writing.

- For frequent torque parameter changing, there are register addresses from 261 to 267 for torque of preset #1 7. These data are saved in RAM of the controller for quick and temporary use. It can save the life time of EEPROM from frequent erase and writing. These memory are disappeared when the power is off.

- Address for the REMOTE CONTROL via serial COM.

Description	Address	Data
Alarm reset	4000	1
Driver operation Lock	4001	0 : Driver Unlock 1 : Lock both For & Rev 2 : Lock Reverse 3 : Lock Forward
Real-time monitoring	4002	No use (Factory only)
Remote start	4003	0 : Stop 1 : Start
Preset no. change	4004	1 – 15 : Preset #1 – 15 16 : Multi sequence A 17 : Multi sequence B
Forward / Reverse rotation	4005	0 : Fastening (Forward) 1 : Loosening (Reverse)
Firmware upgrade	4500	No use (Factory only)
Initialize to factory setting of the controller parameters	170	77

◆ < **WRITE - Multiple parameters** > Function code 16 : writing multiple parameters

!!! ADC controller ONLY !!!

Function code 16 is used to WRITE the multiple parameters in multiple registers. The only integer number is allowed.

For remote control, use the function code 06.

It is recommended not over 20 parameters for writing at once. Max data is limited within 200 bytes.

(Query)

Slave address (00 for TCP)	Function code	Start address High	Start address Low	Number of address High	Number of address Low	Byte count (address no x 2)	Date High	Date Low	CRC (RTU) Low	CRC (RTU) High

(OK Response)

Slave address (00 for TCP)	Function Code	Start address High	Start address Low	Number of address High	Number of address Low	CRC (RTU) Low	CRC (RTU) High

[Example message, Query & Response]

To write the data of parameter 2 and 3, which is torque & torque limit value of Preset #1 as below

Torque : 15.00 Torque limit : 10% (10.00)

Query

01 10 00 02 00 03 04 05 DC 03 E8 B2 EF

Response

01 10 00 02 00 02 E0 08

◆ < **REQUEST** > Function code 17 : Slave device information

Function code 17 is used to read the slave device information about ID no, controller model, screwdriver model, serial no and firmware version.

(Query)

Slave address (00 for TCP)	Function Code (17)	CRC (RTU) Low	CRC (RTU) High

(Response)

Slave address (00 for TCP)	Function Code	No. of byte	ID High	ID Low	Controller model High	Controller model Low	Screw driver model High	Screw driver model Low	Ver. High	Ver. Low	S/N 4	S/N 3	S/N 2	S/N 1	CRC (RTU) Low	CRC (RTU) High

◆ **Error code** for abnormal response

If there are wrong function code or communication failure by protocol (parity, LRC, CRC..etc.), there will be no response. The master will show " TIME OUT " error.

If the query contains wrong function code or address, the function code + 0x80 will be responsed together with the following error code in data registry.

Error code	Description
0x01	No defined function code or wrong function code
0x02	Wrong address or no existing address
0x03	Data length over the capacity
0x07	Wrong CRC value in query
0x0C	Over the number of byte
0x0E	Range of data is not available

[Example message, Query & Response]

To read the 5 parameter data starting from 564 to 568

(01 : Query)

Slave address (00 for TCP)	01	02	34	00	05	CRC Low	CRC High

Start address : 0234 (hex) = 564 (dec)

Number of address : 0005 (hex) = 5 (dec)

Function code 01 is not defined. → Function code error

Parameter from 564 to 568 are not existing. → No existing address

(01 + 80 : Response)

Slave address (00 for TCP)	81	02	CRC Low	CRC High

Function code (01) + 0x80 = 81

Error code for wrong data address = 02

◆ < **REQUEST** > Function code 100 (0x64) : monitoring data output for MDC
Function code 200 (0x64) : monitoring data output for ADC

*** This function is exceptional from Modbus protocol.

Function code 100 is used to response with the monitoring data.

To request the monitoring data, use the function code 06 with data on the below address.

Parameter and data for monitoring

Address	Description	Data
4100	Monitoring curve data output	0 : Disable, 1 : Enable To keep monitoring ON, repeat request with 1(Enable) in every 10 seconds.
4101	Channel 1 monitoring data	1:torque, 2:current, 3:speed, 4:angle, 5:speed command, 6:current command (mA), 7: Sung Angle
4102	Channel 2 monitoring data	0: disable, 1:torque, 2:current, 3:speed, 4:angle, 5:speed command, 6:current command (mA), 7: Sung Angle
4103	Sampling time	1 : 5ms, 2 : 10ms, 3 : 15ms
4104	Option 1	1 : fastening, 2: loosening, 3 : both

Request the desired data on the above parameters in the address 4101, 4102, 4103 and 4104. And repeat the request data “1” (Enable) on the parameter address 4100 in every 10 seconds to keep data output ON. Otherwise the data output will be OFF

For example, to monitor the curve data of torque at channel 1, speed at channel 2,

Address 4101 : 1 (torque at channel 1)

Address 4102 : 2 (speed at channel 2)

(Query)

Slave address (00 for TCP)	Function Code 06	Address High	Address Low	Date High	Data Low	CRC Low	CRC High

(Response)

Slave address (00 for TCP)	Function Code 100	No of byte High	No of byte Low	Data #1 High	Data #1 Low	... x n data	CRC Low	CRC High

The order of response data

1 : Channel 1

2 : Channel 2

3 : Sampling time

4 : Option 1

5 : Number of data

6 : Fastening time

7 : Target torque

8 : Converted torque

9 : Speed

10 : A1 angle

11 : A2 angle

12 : Sung Angle

13 : Error code

14 : Number of Screw Count

15 : Status (1 : Complete, 0 : others)

16 – 415 (max) : monitoring data (not fixed number) / maximum 400 data for two channels.

If the monitoring data is for only channel 1, all from 16^{th} data are included in Channel 1.

If two channels are selected, the data for channel 2 comes later the data of channel 1.

For example, total number of data are 400, the first 200 data are for channel 1. The rest 200 data are for channel 2

16^{th} – 215^{th} : Data for channel 1

216^{th} – 415^{th} : Data for channel 2