GIGARISE®
嘉升科技有限公司

SD200 Dual Channels Isolated Signal Converter

RS-485 COMMUNICATION INSTRUCTION MANUAL

MODBUS Protocol Reference Guide

1. COMMUNICATION FUNCTIONS

1.1 General

- ☐ The dual channel isolated signal converter ***SD200*** provides a communication function by RS-485 interface, by which it can transmit and receive data to and from host computer,PLC, graphic display panel, etc.

- ☐ The communication system consists of master station and slave station. Up to 247 slave stations can be connected per master station.

- ☐ In order that the master station and slave station can communicate, the format of the transmit/receive data must coincide. For the ***SD200*** series, the format of the communication data is determined by the MODBUS protocol (RTU mode).

- ☐ Please use an RS-232C→RS-485 converter in case of designating a personal computer or other devices which have an RS-232C interface as a master station.

2. SPECIFICATIONS

2.1 Communication Specifications

Item	Specification	
Electrical specification	Based on EIA RS-485	
Transmit system	2-wire, semi-duplicate	
Synchronizing system	Asynchronous mode	
Number connection unit	Up to 247 units	
Transmission distance	500m max	
Transmission speed	4800 / 9600 / 19200 / 38400 selectable	
Data format	Start bit	1 bit
	Data length bit	8 bits
	Parity bit	None
	Stop bit	2 bits
Transmission code	HEX value (MODBUS RTU mode)	
Error detection	CRC-16 bits	

A typical MODBUS protocol character is shown below:

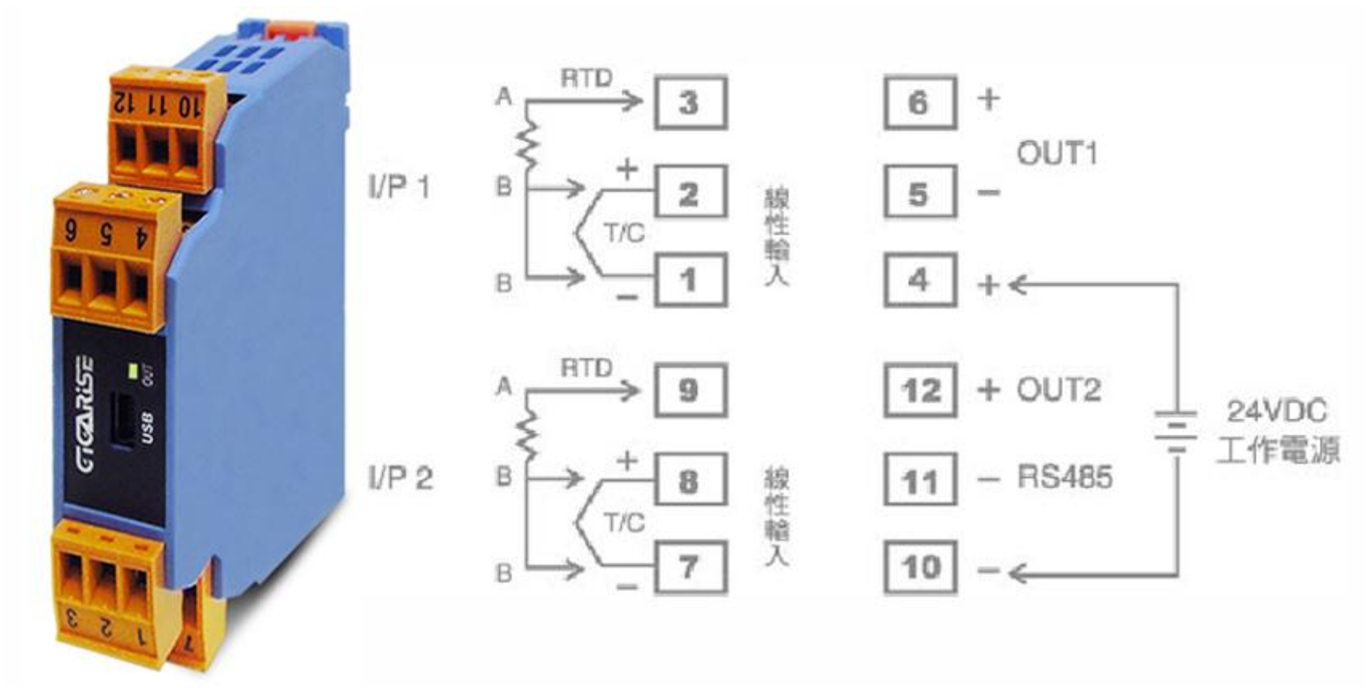
1	2	3	4	5	6	7	8	9	10	11
Start bit	Data bits								Stop bits	

The format (11 bits) for each byte in RTU mode is:
Coding System: 8-bits binary
Bits per byte: 1 start bit.
8 data bits,least significant bit sent first.
2 stop bit.

3. CONNECTION

△ WARNING
For avoiding electric shock and malfunctions, don’t turn on the power supply until all wiring has been completed.

3.1 Terminal Allocation



3.2 Wiring

□ Use twisted pair cables with shield.
Recommended cable: UL2464, UL2448, etc.

□ The total extension length of the cable is up to 500m. A master station and up to 247 units of the ***SD200*** series can be connected per line.

□ Both ends of the cable should be connecting with terminate resistors 100Ω 1/2W.

□ The shield wire of the cable should be grounded at one place on the master station unit side.

4. SETTING OF COMMUNICATION CONDITION

In order that the master station and ***SD200*** devices can correctly communicate, following settings are required.

- All communication condition settings of the master station are the same as those of ***SD200*** series.
- All ***SD200*** devices connected on a line are set to address (ADDR), which are different from each other.

4.1 Set Items

The parameters to be set are shown in the following table.

Parameter	Item	Value at delivery	Setting range	Remarks
BaudRate	Transmission speed	19200	4800/9600/19200/38400	Set the same communication condition to the master station and all slave station.
ID	Slave address	255	1 to 255	Set a different value to each station.

5. MODBUS COMMUNICATION PROTOCOL

5.1 General

The communication system by the MODBUS protocol is that the communication is always started from the master station and a slave station responds to the received message.

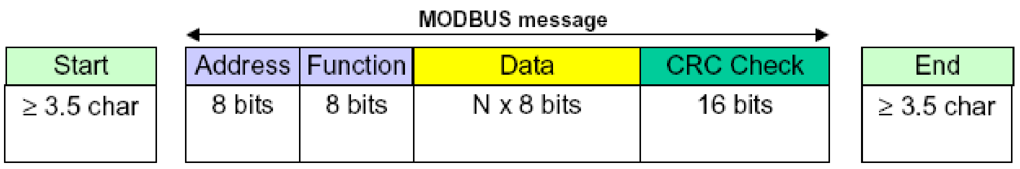
Transmission procedures is as shown below.

1. The master station sends a command message to a slave station.
2. The slave station checks that the address in the received message matches with the own address or not.
3. If matched, the slave station executes the command and sends back the response message.
4. If mismatched, the slave station leaves the command message and wait for the next command message.
5. The master station can individually communicate with any one of slave stations connected on the same line upon setting the address in the command message.

5.2 Composition of Message

Command message and response message consist of 4 fields; Slave Address, Function code, Data and CRC check code. And these are sends in this order. The allowable character transmitted for all fields are hexadecimal 0-9,A-F.

RTU message framing



In the following, each field is explained.

1. **Start**
   In RTU mode, messages start with a silent interval of at least 3.5 character times. This is most easily implemented as a multiple of character times at the baud rate that is being used on the network. The first field then transmitted is the device address.

2. **Address**
   Address is the number specifying a slave station. The individual slave device addresses are in the range of 1-247 decimal. A master addresses a slave by placing the slave address in the address field of the message. When the slave returns its response, it places its own address in this address field of the response to let the master know which slave is responding.
   Address 0 is used for the broadcast address, which all slave stations recognize.
   When the broadcast address (address 0) is applied on the command message, no any response message will be sent from the slave stations.

3. **Function**
   This is a code to designate the function executed at a slave station. When a message is sent from a master to a slave device the function code field tells the slave what kind of action to perform. When the slave responds to the master, it uses the function code field to indicate either a normal response or that some kind of error occurred. For normal response, the slave simply echoes the original function code. For an exception response, the slave returns a code that is equivalent to the original function code with its most-signification bit set to a logic 1.

4. **Data**
   Data are the data required for executing function codes. The composition of data varies with function codes. A data address is assigned to each data in the ***SD200***. For reading/writing the data by communication, designate the data address. Refer to chapter 6 for details.

5. **CRC check**
   This is the code to detect message errors (change in bit) in the signal transmission.
   On the MODBUS protocol (RTU mode), CRC-16 (Cyclical Redundancy Check) is applied. For CRC

calculation method, refer to section 5.5.

6. **End**
   Following the last transmitted character, a similar interval of at least 3.5 character times marks the end of message. A new message can begin after this interval.

5.3 Response of Slave Station

1. Response for normal command
To a relevant message, the slave station creates and sends back a response message, which corresponds to the command message. The composition of message in this case is the same as in section 5.2. Content of the data field depend on the function code. For details, refer to Chapter 6.

2. Response for abnormal command
If contents of a command message have an abnormality (for example, non-actual function code is designated) other than transmission error, the slave station does not execute that command but creates and sends back a response message at error detection.
The composition of response message at error detection is shown on below, the value used for function field is function code of command message plus 80H.

Slave Address	Function code (Function code + 80H)	Error code	CRC check
8 BITS	8 BITS	8 BITS	16 BITS

Error Code	Contents	Description
01	Illegal function	The function code received is not an allowable action for the slave.
02	Illegal data address	The data address received is not an allowable address for the slave.
03	Illegal data value	A value contained in the data field is not an allowable value for the slave.

5.4 Function Code

The listing below shows the function codes supported by the ***SD200*** .

Function code		
Code	Function	Object
03	Read-out	Holding Register
04	Read-out	Input Register
06	Write-in	Holding Register

5.5 Calculation of Error Check Code (CRC-16)

CRC-16 is the 2-bytes (16-bits) error check code. From the top of the message (address) to the end of the data field are calculated.
The slave station calculates the CRC of the received message, and does not respond if the calculated CRC is different from the contents of the received CRC code.

A procedure for generating a CRC is:

1. Load a 16-bits register with FFFF hex (all 1's). Call this the CRC register.
2. Exclusive OR the first 8-bit byte of the message with the low-order byte of the 16-bit CRC registers, putting the result in the CRC register.
3. Shift the CRC register one bit to the right (toward the LSB), Zero-filling the MSB. Extract and examine the LSB.
4. If the LSB was 0: Repeat Step 3.
   If the LSB was 1: Exclusive OR the CRC registers with the polynomial value 0xA001 (1010 0000 0000 0001).
5. Repeat step 3 and 4 until 8 shifts have been performed. When this is done, a complete 8-bit byte will have been processed.
6. Repeat step 2 through 5 for the next 8-bit byte of the message. Continue doing this until all bytes have been processed.
7. The final content of the CRC register is the CRC value.
8. When the CRC is placed into the message, its upper and lower bytes must be swapped as described below.

For example, if the CRC value is 1241 hex (0001 0010 0100 0001):

Addr	Func	Data Count	Data	Data	Data	Data	CRC Lo	CRC Hi
							0x41	0x12

An example of a C language function performing CRC generation is shown on the following pages. All of the possible CRC values are preloaded into two arrays, which are simply indexed as the function increments through the message buffer. One array contains all of the 256 possible CRC values for the high byte of the 16–bit CRC field, and the other array contains all of the values for the low byte.

Indexing the CRC in this way provides faster execution than would be achieved by calculating a new CRC value with each new character from the message buffer.

Note: This function performs the swapping of the high/low CRC bytes internally. The bytes are already swapped in the CRC value that is returned from the function.
Therefore the CRC value returned from the function can be directly placed into the message for transmission.

The function takes two arguments:
unsigned char *puchMsg; A pointer to the message buffer containing binary data to be used for generating the CRC unsigned short usDataLen; The quantity of bytes in the message buffer.

CRC Generation Function

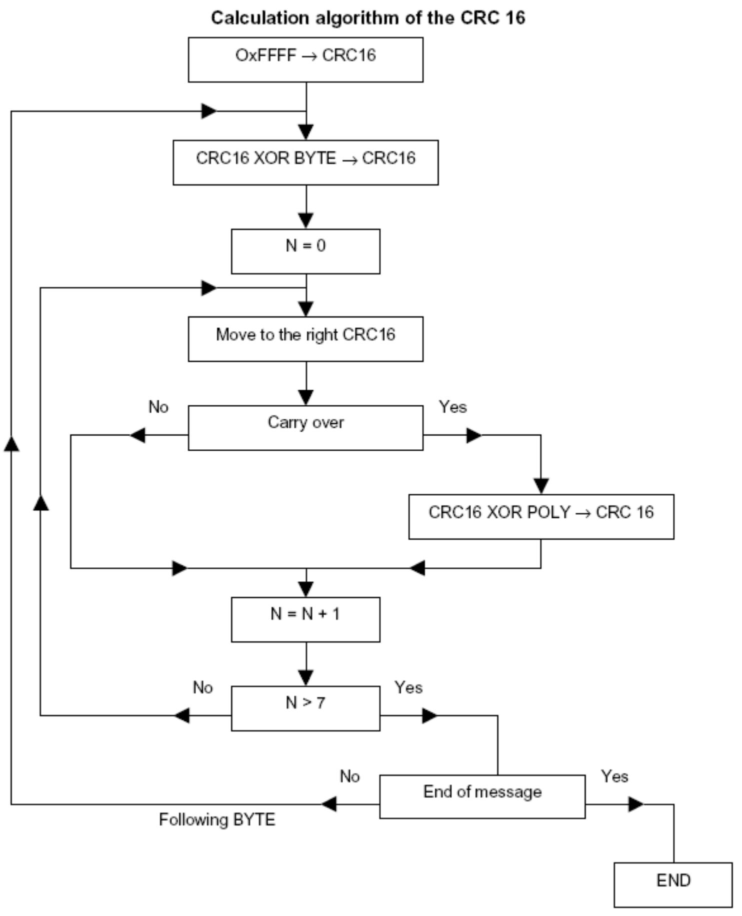
```
unsigned short CRC16 ( puchMsg, usDataLen ) /* The function returns the CRC as a unsigned short type */
unsigned char *puchMsg ; /* message to calculate CRC upon */
unsigned short usDataLen ; /* quantity of bytes in message */
{
unsigned char uchCRCHi = 0xFF ; /* high byte of CRC initialized */
unsigned char uchCRCLo = 0xFF ; /* low byte of CRC initialized */
unsigned uIndex ; /* will index into CRC lookup table */
while (usDataLen--) /* pass through message buffer */
        {
        uIndex = uchCRCLo ^ *puchMsg++ ; /* calculate the CRC */
        uchCRCLo = uchCRCHi ^ auchCRCHi[uIndex] ;
        uchCRCHi = auchCRCLo[uIndex] ;
        }
return (uchCRCHi << 8 | uchCRCLo) ;
}
```

High-Order Byte Table

```
/* Table of CRC values for high–order byte */
static unsigned char auchCRCHi[] = {
0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81,
0x40, 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0,
0x80, 0x41, 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40, 0x00, 0xC1, 0x81, 0x40, 0x01,
0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x01, 0xC0, 0x80, 0x41,
0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40, 0x00, 0xC1, 0x81,
0x40, 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x01, 0xC0,
0x80, 0x41, 0x00, 0xC1, 0x81, 0x40, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x01,
0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40,
0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81,
0x40, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0,
0x80, 0x41, 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40, 0x00, 0xC1, 0x81, 0x40, 0x01,
0xC0, 0x80, 0x41, 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41,
0x00, 0xC1, 0x81, 0x40, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81,
0x40, 0x01, 0xC0, 0x80, 0x41, 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0,
0x80, 0x41, 0x00, 0xC1, 0x81, 0x40, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x01,
0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41,
0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81,
0x40
} ;
```

Low-Order Byte Table

```
/* Table of CRC values for low–order byte */
static char auchCRCLo[] = {
0x00, 0xC0, 0xC1, 0x01, 0xC3, 0x03, 0x02, 0xC2, 0xC6, 0x06, 0x07, 0xC7, 0x05, 0xC5, 0xC4,
0x04, 0xCC, 0x0C, 0x0D, 0xCD, 0x0F, 0xCF, 0xCE, 0x0E, 0x0A, 0xCA, 0xCB, 0x0B, 0xC9, 0x09,
0x08, 0xC8, 0xD8, 0x18, 0x19, 0xD9, 0x1B, 0xDB, 0xDA, 0x1A, 0x1E, 0xDE, 0xDF, 0x1F, 0xDD,
0x1D, 0x1C, 0xDC, 0x14, 0xD4, 0xD5, 0x15, 0xD7, 0x17, 0x16, 0xD6, 0xD2, 0x12, 0x13, 0xD3,
0x11, 0xD1, 0xD0, 0x10, 0xF0, 0x30, 0x31, 0xF1, 0x33, 0xF3, 0xF2, 0x32, 0x36, 0xF6, 0xF7,
0x37, 0xF5, 0x35, 0x34, 0xF4, 0x3C, 0xFC, 0xFD, 0x3D, 0xFF, 0x3F, 0x3E, 0xFE, 0xFA, 0x3A,
0x3B, 0xFB, 0x39, 0xF9, 0xF8, 0x38, 0x28, 0xE8, 0xE9, 0x29, 0xEB, 0x2B, 0x2A, 0xEA, 0xEE,
0x2E, 0x2F, 0xEF, 0x2D, 0xED, 0xEC, 0x2C, 0xE4, 0x24, 0x25, 0xE5, 0x27, 0xE7, 0xE6, 0x26,
0x22, 0xE2, 0xE3, 0x23, 0xE1, 0x21, 0x20, 0xE0, 0xA0, 0x60, 0x61, 0xA1, 0x63, 0xA3, 0xA2,
0x62, 0x66, 0xA6, 0xA7, 0x67, 0xA5, 0x65, 0x64, 0xA4, 0x6C, 0xAC, 0xAD, 0x6D, 0xAF, 0x6F,
0x6E, 0xAE, 0xAA, 0x6A, 0x6B, 0xAB, 0x69, 0xA9, 0xA8, 0x68, 0x78, 0xB8, 0xB9, 0x79, 0xBB,
0x7B, 0x7A, 0xBA, 0xBE, 0x7E, 0x7F, 0xBF, 0x7D, 0xBD, 0xBC, 0x7C, 0xB4, 0x74, 0x75, 0xB5,
0x77, 0xB7, 0xB6, 0x76, 0x72, 0xB2, 0xB3, 0x73, 0xB1, 0x71, 0x70, 0xB0, 0x50, 0x90, 0x91,
0x51, 0x93, 0x53, 0x52, 0x92, 0x96, 0x56, 0x57, 0x97, 0x55, 0x95, 0x94, 0x54, 0x9C, 0x5C,
0x5D, 0x9D, 0x5F, 0x9F, 0x9E, 0x5E, 0x5A, 0x9A, 0x9B, 0x5B, 0x99, 0x59, 0x58, 0x98, 0x88,
0x48, 0x49, 0x89, 0x4B, 0x8B, 0x8A, 0x4A, 0x4E, 0x8E, 0x8F, 0x4F, 0x8D, 0x4D, 0x4C, 0x8C,
0x44, 0x84, 0x85, 0x45, 0x87, 0x47, 0x46, 0x86, 0x82, 0x42, 0x43, 0x83, 0x41, 0x81, 0x80,
0x40
};
```

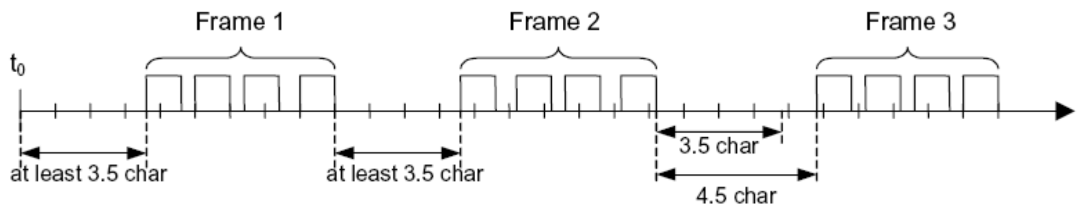


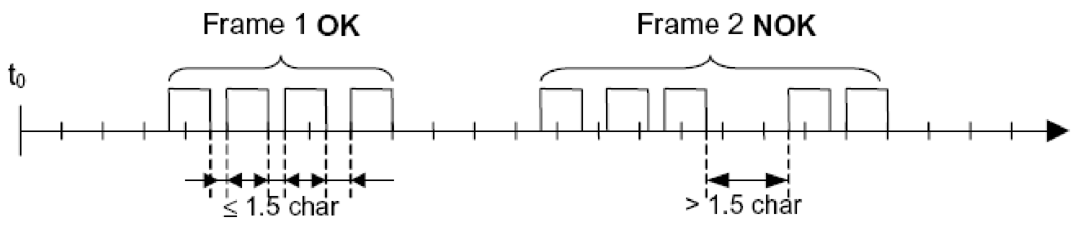
5.6 Transmission Control Procedure

1. Transmission procedure of master station

Slave Address	Function Code	Data	CRC
1 byte	1 byte	0 up to 252 byte(s)	2 bytes CRC Low CRC Hi

RTU Message Frame





The master station must proceed to a communication upon conforming to the following items.

1-1. Before sending a command message, at least 3.5 character times silent interval must be provided.
1-2. For sending, the interval between bytes of a command message must be below 1.5 character times.
1-3. Within 1.5 character times after sending a command message, the receiving status is posted.
1-4. Provide 3.5 character times vacant status between the end of response message reception and beginning for next command message sending (same as in 1-1).
1-5. For ensuring the safety, make a confirmation of the response message and make an arrangement so as to provide 3 or more retries in case of no response, error occurrence, etc.

2. Description

(1). Detection of the message frame
Since the communication system uses the 2-write RS-485 interface, there may be 2 statuses on a line below.
(a) Vacant status (no data on line)
(b) Communication status (data is existing)
Devices connected on the line are initially at a receiving status and monitoring the line. When 1.5 character times or more vacant status has appeared on the line, the end of preceding frame is assumed and, within following 1.5 character times, a receiving status is posted. When data appears on the line, devices receive it while 1.5 character times more vacant status is detected again, and the end of that frame is assumed. Data, which appeared on the line from the first 3.5 character times or more vacant status to the next 3.5 character times or more vacant status is fetched as one frame.
1-1. 3.5 character times or more vacant status precedes the command message sending.
1-2. Interval between bytes of 1 command message must be smaller than 1.5 character times.

(2). Response of ***SD200***
After a frame detection (1.5 character times or more vacant status), ***SD200*** carries out-processing with that frame as a command message. If the command message is destined to the own station, a response message is returned. Its processing time is 1 to 10ms (depends on contents of command message). After sending a command message, therefore, the master station must observe the following.
1-3. Receiving status is posted within1.5 character times after sending a command message.

6. DETAILS OF MESSAGE

Read-out of Word Data [Function Code: 03]

Read the contents of holding registers in the slave.

Broadcast is not possible.

1. Message composition

Command message composition

Address	Function Code	Starting Address	Word Number*	CRC-16	
01~FF	03	0xxx	0001~007D	Low-order byte	High-order byte
1 byte	1 byte	2 byte	2 bytes	2 bytes	

* Maximum word number = 7E

Response message composition

Address	Function Code	Byte Number *	Word Data	CRC-16	
01 ~ FF	03	02~FC		Low-order byte	High-order byte
1 byte	1 byte	1 bytes	N bytes	2 bytes	

* Byte number = Word number × 2

2. Message transmission (example)

The following show an example of reading the Input signal type (0000) from address No.1.

Command message composition

Address	Function Code	Starting Address	Word Number	CRC-16
01	03	0000	0001	840A

Response message composition

Address	Function Code	Byte Number	Word Data	CRC-16
01	03	02	0001	7984

Read-out of Read-Only Word Data [Function Code: 04]

Read the contents of input registers (1000~1002) in the slave.

Broadcast is not possible.

1. Message composition

Command message composition

Address	Function Code	Starting Address	Word Number	CRC-16	
01~FF	04	1xxx	0001~007D	Low-order byte	High-order byte
1 byte	1 byte	2 bytes	2 bytes	2 bytes	

Response message composition

Address	Function Code	Byte Number	Word Data	CRC-16	
01 ~ FF	04	02~FC		Low-order byte	High-order byte
1 byte	1 byte	1 byte	N bytes	2 bytes	

* Byte number = Word number × 2

2. Message transmission (example)

The following show an example of reading the Measuring Value of channel 1 (PV1=27.0) from address No.1.

Command message composition

Address	Function Code	Starting Address	Word Number	CRC-16
01	04	1000	0001	350A

Response message composition

Address	Function Code	Byte Number	Word Data	CRC-16
01	04	02	010E	3964

Write-in of Word Data (1 word) [Function Code: 06]

1. **Message composition**

Command message composition

Address	Function	Starting Address	Word Data	CRC-16	
1 ~ FF	06	0xxx	0xxx	Low-order byte	High-order byte
1 byte	1 byte	2 bytes	2 bytes	2 bytes	

Response message composition

Address	Function	Starting Address	Word Data	CRC-16	
1 ~ FF	06	0xxx	0xxx	Low-order byte	High-order byte
1 byte	1 byte	2 bytes	2 bytes	2 bytes	

2. **Message transmission (example)**

The following show an example of writing the Input signal type to address No.1.

Command message composition

Address	Function Code	Starting Address	Word Data	CRC-16
01	06	0000	0001	89DD

Response message composition

Address	Function Code	Starting Address	Word Data	CRC-16
01	06	0000	0001	480A

7. ADDRESS MAP AND DATA FORMAT

7.1 Data Format

The MODBUS protocol used in ***SD200*** is RTU (Remote Terminal Unit) mode.

Transmitted data is “numeric value” and not “ASCII code”.

7.2 Data Address Map

■ **Table of Holding Registers : Function code [03,06] Word data (read-out/write-in)**

Data Address	**Parameter**	**Range**
0	Input signal type	0~12 ,refer to detail of register 0
1	Unit	0~1,refer to detail of register 1
2	Lower limit of measuring range	Refer to detail of register 0
3	Upper limit of measuring range	Refer to detail of register 0
4	Lower limit of Math. function	-30000~30000
5	Upper limit of Math. function	-30000~30000
6	Offset correction of PV1	-10%~10% of Max. input range

7	Offset correction of PV2	-10%~10% of Max. input range
8	Output direction	0~1,refer to detail of register 8
9	Fault signal on sensor break	0~1,refer to detail of register 9
10	Zero adjustment of output 1	0~32767
11	Span adjustment of output 1	0~32767
12	Zero adjustment of output 2	0~32767
13	Span adjustment of output 2	0~32767
14	Constant A of Math. function	-128~127
15	Constant B of Math. function	-128~127
16	Constant C of Math. function	-30000~30000 (0 is not available)
17	Math. Function	0~1,refer to detail of register 17
18	Square-root function	0~1,refer to detail of register 18
19	Address for RS485 communication (ID)	1~255
20	Baud rate	0~3,refer to detail of register 20

Detail of register 0

Code	Input Signal Type	Max. Range
0	Thermocouple J type	-50 ~1000 ℃
1	Thermocouple K type	-50 ~1370 ℃
2	Thermocouple T type	-200 ~400 ℃
3	Thermocouple E type	-50 ~960 ℃
4	Thermocouple B type	250 ~1750 ℃
5	Thermocouple R type	-50 ~1750 ℃
6	Thermocouple S type	-50 ~1750 ℃
7	Thermocouple N type	-50 ~1300 ℃
8	Thermocouple C type	-50 ~1800 ℃
9	mV	-60.00 ~60.00 **mV**
10	PT100	-200 ~600 ℃
11	mA	0.00~24.00 **mA**
12	V	-10.000~10.000 **V**

Detail of register 1

Code	Unit
0	℃
1	℉

Detail of register 8

Code	Output Direction
0	Direct
1	Revers

Detail of register 9

Code	Fault Signal On Sensor Break
0	Downscale
1	Upscale

Detail of register 17

Code	Math. Function
0	Off
1	On

Detail of register 18

Code	Squre-Root Function
0	Off
1	On

Detail of register 20

Code	Baud Rate
0	4800
1	9600
2	19200
3	38400

■ **Table of Input Registers : Function code [04] Word data (read-out only)**

Data Address	Parameter
1000	Measuring value of channel 1 (PV1)
1001	Measuring value of channel 2 (PV2)
1002	Calculation value of Math. Function (PV3)

Please be aware that the reading of PV1, PV2 and PV3 have different resolution depended on the input signal type.

For temperature input (thermocouple & Pt100), the reading will have 1-digit decimal. so the reading will be 270 for 27 degree.

For 0~24mA current input, the reading will have 3-digit decimal. so the reading will be 4000 for 4 mA.

For -50~50mV input, the reading will have 2-digit decimal. so the reading will be 5000 for 50mV.

For -10~10V input, the reading will have 3-digit decimal. so the reading will be 10000 for 10V.