



Date: 22 September 2000
Department: Electronic Control
Revision: 1.5
Apply to: Version 1.19K or later

WARNING

! Improper disclosure of this document may result in unauthorized gen-set operation.

Contents

	Page
1. Introduction	2
2. Address Allocation Tables	5
3. Supported Modbus Functions	9
4. Exception Response	14
5. CRC Calculation	15

1. Introduction

1.1 Modbus Protocol

Modbus protocol is a messaging structure, widely used to establish master-slave communication between intelligent devices. A Modbus message sent from a master to a slave contains the address of the slave, the “command” (e.g. “read register” or “write register”), the data, and a check sum (LRC or CRC).

Since Modbus protocol is just a messaging structure, it is independent of the underlying physical layer. It is traditionally implemented using RS232, RS422, or RS485 over a variety of media.

The Modbus has two transmission modes:

- **ASCII mode:** Each eight-bit byte in a message is sent as 2 ASCII characters.
- **RTU (Remote Terminal Unit) mode:** Each eight-bit byte in a message is sent as two four-bit hexadecimal characters.

The basic structure of a Modbus frame is shown below:

ADDRESS	FUNCTION	DATA	CHECKSUM
---------	----------	------	----------

1.2 Modbus on Access 4000

Access 4000 controller uses RTU transmission mode. The data format is 8 bits, 1 stop bit, and no parity. The maximum waiting time for a master to get response is 128ms.

1.2.1 RTU Framing

In RTU mode, message 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 (shown as T1-T2-T3-T4 in the figure below). The first field then transmitted is the device address.

The allowable characters transmitted for all fields are hexadecimal 0...9, A...F. Networked devices monitor the network bus continuously, including during the silent intervals. When the first field (the address field) is received, each device decodes it to find out if it is the addressed device.

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

The entire message frame must be transmitted as a continuous stream. If a silent interval of more than 1.5 character times occurs before completion of the frame, the receiving device flushes the incomplete message and assumes that the next byte will be the address field of a new message.

Similarly, if a new message begins earlier than 3.5 character times following a previous message, the receiving device will consider it a continuation of the previous message. A typical RTU message frame is given below.

START	ADDRESS	FUNCTION	DATA	CRC CHECK	END
T1-T2-T3-T4	8 bits	8 bits	nx8 bits	16 bits	T1-T2-T3-T4

1.2.2 Address Field

The address field of a message frame contains eight bits. Valid slave device addresses are in the range of 0...247 decimal in Modbus protocol. On Access 4000, the individual

slave by placing the slave address in the address field of the message. When the slave sends its response, it places its own address in this address field of the response to let the master know which slave is responding.

1.2.3 Function Field

The function code field of a message frame contains eight bits. Valid codes are in the range of 1...255 decimal in Modbus. Of these, some codes are supported in Access 4000 controller. These codes will be documented in the following sections.

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 (called an exception response).

1.2.4 Contents of the Data Field

The data field of messages sent from a master to slave devices contains additional information which the slave must use to take the action defined by the function code. This can include items like discrete and register **relative addresses**, the quantity of items to be handled, and the count of actual data bytes in the field.

If no error occurs, the data field of a response from a slave to a master contains the data requested. If an error occurs, the field contains an exception code that the master application can use to determine the next action to be taken.

1.2.5 Contents of the Error Checking Field

When RTU mode is used for character framing, the error checking field contains a 16-bit value implemented as two eight-bit byte. The error check value is the result of a Cyclical Redundancy Check calculation performed on the message contents.

The CRC field is appended to the message as the last field in the message. When this is done, the low-order byte of the field is appended first, followed by the high-order byte. The CRC high-order byte is the last byte to be sent in the message.

The polynomial value for CRC calculation in Access 4000 is A001h.

1.2.6 Address Range and Supported Function Codes

Data Type	Relative Addresses	Function Codes	Description
Coils	00h - 0Fh	01	Read Coil Status
Coils	00h - 0Fh	05	Force Single Coil
Discrete Inputs	00h - 2Fh	02	Read Input Status
Input Registers	00h - 1Fh	04	Read Input Registers
Holding Registers	00h - 20h	03	Read Holding Registers
Holding Registers	00h - 20h	06	Preset Single Register

1.2.7 Setup on Access 4000

The following settings in Configure/System Menu are required for Modbus communication:

UNIT ADDRESS (1-15) - unique address for each slave in the system. The default address is 01. Two slaves with the same address may cause message collision.

DATA SPEED (9600, 14400, 19200, 32768, or 38400) - communication bus data speed. The default speed is 9600. The master and slaves should use the same speed.

COMMS TYPE (GenAccess, or Modbus) - communication protocol type. The default protocol is GenAccess which is used with GenAccess software. Communication using Modbus protocol won't work if Modbus is not selected.

2. Address Allocation Tables

Controls (**Function 01** - Read Coil Status; **Function 05** - Force Single Coil)

Buffer Address	Relative Address	Data Type	Scaling	Description
00	00	Unsigned char	----	Reserved
01	01	Unsigned char	----	Start
02	02	Unsigned char	----	Stop
03	03	Unsigned char	----	Emergency Stop
04	04	Unsigned char	----	Reset Alarm
05	05	Unsigned char	----	Reserved
06	06	Unsigned char	----	Reserved
07	07	Unsigned char	----	Reserved
08	08	Unsigned char	----	Reserved
09	09	Unsigned char	----	Reserved
0A	0A	Unsigned char	----	Reserved
0B	0B	Unsigned char	----	Reserved
0C	0C	Unsigned char	----	Reserved
0D	0D	Unsigned char	----	Reserved
0E	0E	Unsigned char	----	Reserved
0F	0F	Unsigned char	----	Reserved

Note:

1. The controls START, STOP, and EMERGENCY STOP work under the condition that the AUTO switch is at ON position. Otherwise, device failure response will be received.
2. When START is set to ON, the STOP and EMERGENCY STOP will be set automatically to OFF. Likewise, when STOP or EMERGENCY STOP is set to ON, the other two will be set to OFF.
3. Trying to set START, STOP, EMERGENCY STOP, or RESET ALARM to zero will get a device failure response.
4. When setting RESET ALARM to ON, the slave will clear all active alarms and reset this switch to OFF. So RESET ALARM is normally read as OFF.

Status/Alarms (**Function 02** - Read Input Status)

Buffer Address	Relative Address	Data Type	Scaling	Description (1 = ON, 0 = OFF)
10	00	Unsigned char	----	Auto Switch
11	01	Unsigned char	----	General Alarm
12	02	Unsigned char	----	Reserved
13	03	Unsigned char	----	Reserved
14	04	Unsigned char	----	Heart beat
15	05	Unsigned char	----	Remote start
16	06	Unsigned char	----	Output relay 1
17	07	Unsigned char	----	Output relay 2
18	08	Unsigned char	----	Approaching low oil pressure

19	09	Unsigned char	----	Approaching high engine temperature
1A	0A	Unsigned char	----	Low coolant temperature
1B	0B	Unsigned char	----	Low battery voltage
1C	0C	Unsigned char	----	High battery voltage

Status/Alarms (continued) (Function 02 - Read Input Status)

Buffer Address	Relative Address	Data Type	Scaling	Description
1D	0D	Unsigned char	----	Battery charger fail
1E	0E	Unsigned char	----	Not in auto
1F	0F	Unsigned char	----	Fail to start
20	10	Unsigned char	----	Under frequency
21	11	Unsigned char	----	Over frequency
22	12	Unsigned char	----	Under voltage
23	13	Unsigned char	----	Over voltage
24	14	Unsigned char	----	Over current
25	15	Unsigned char	----	Over speed
26	16	Unsigned char	----	Emergency stop
27	17	Unsigned char	----	High engine temperature
28	18	Unsigned char	----	Low oil pressure
29	19	Unsigned char	----	Spare fault 1
2A	1A	Unsigned char	----	Spare fault 2
2B	1B	Unsigned char	----	Spare fault 3
2C	1C	Unsigned char	----	Spare fault 4
2D	1D	Unsigned char	----	Reserved
2E	1E	Unsigned char	----	Reserved
2F	1F	Unsigned char	----	Reserved
30	20	Unsigned char	----	Reserved
31	21	Unsigned char	----	Reserved
32	22	Unsigned char	----	Reserved
33	23	Unsigned char	----	Reserved
34	24	Unsigned char	----	Reserved
35	25	Unsigned char	----	Reserved
36	26	Unsigned char	----	Reserved
37	27	Unsigned char	----	Reserved
38	28	Unsigned char	----	Reserved
39	29	Unsigned char	----	Reserved
3A	2A	Unsigned char	----	Reserved
3B	2B	Unsigned char	----	Reserved
3C	2C	Unsigned char	----	Reserved
3D	2D	Unsigned char	----	Reserved
3E	2E	Unsigned char	----	Reserved
3F	2F	Unsigned char	----	Reserved

Monitoring (Function 04 - Read Input Registers)

Buffer Address	Relative Address	Data Type	Scaling	Description
40	00	Word	----	Water temperature
42	01	Word	x10	Oil pressure
44	02	Word	x10	Battery voltage

46	03	Word	----	Hours run
48	04	Word	----	Starts
4A	05	Word	x10	Rpm
4C	06	Word	----	Phase A voltage
4E	07	Word	----	Phase B voltage
50	08	Word	----	Phase C voltage
52	09	Word	x10	Line AB voltage

Monitoring (continued) (**Function 04** - Read Input Registers)

Buffer address	Relative address	Data Type	Scaling	Description
54	0A	Word	x10	Line BC voltage
56	0B	Word	x10	Line CA voltage
58	0C	Word	x10	Phase A current
5A	0D	Word	x10	Phase B current
5C	0E	Word	x10	Phase C current
5E	0F	Word	x10	Frequency
60	10	Word	x10	Power factor
62	11	Word	----	Total kW
64	12	Word	----	Total KVA
66	13	Word	----	Total kVAr
68	14	Word	----	Total kWh, high word
6A	15	Word	----	Total kWh, low word
6C	16	Word	----	Fuel level
6E	17	Word	----	Reserved
70	18	Word	----	Reserved
72	19	Word	----	Reserved
74	1A	Word	----	Reserved
76	1B	Word	----	Reserved
78	1C	Word	----	Reserved
7A	1D	Word	----	Reserved
7C	1E	Word	----	Reserved
7E	1F	Word	----	Reserved

Note:

1. The value range for fuel level is from 0x80 to 0x8A corresponding to 0 - 10. If this option is disabled the value is always zero.
2. Fuel level measurement requires fuel level sender and simple loom change.

Configuration (**Function 03** - Read Holding Registers; **Function 06** - Preset Single Register)

Buffer Address	Relative Address	Data Type	Scaling	Description	Range (dec) Min - Max
80	00	Word	----	crank cut-out RPM	1 - 999
82	01	Word	----	crank duration time(sec)	1 - 99
84	02	Word	----	crank delay (rest period in sec)	1 - 99
86	03	Word	----	crank repeats (number of tries)	1 - 10
88	04	Word	----	voltage high setpoint	100 - 15500
8A	05	Word	----	voltage high time delay	0 - 99
8C	06	Word	----	voltage low setpoint	60 - 15500
8E	07	Word	----	voltage low time delay	0 - 99
90	08	Word	----	frequency high setpoint	50 - 70
92	09	Word	----	frequency high time delay	0 - 99
94	0A	Word	----	frequency low setpoint	40 - 60
96	0B	Word	----	frequency low time delay	0 - 99

98	0C	Word	----	overspeed setpoint	1000 - 4200
9A	0D	Word	----	overcurrent setpoint	1 - 9999
9C	0E	Word	----	overcurrent time delay	0 - 99
9E	0F	Word	x10	battery voltage high setpoint	120 - 360
A0	10	Word	----	battery voltage high time delay	0 - 99

Configuration (**Function 03** - Read Holding Registers; **Function 06** - Preset Single Register)

Buffer Address	Relative Address	Data Type	Scaling	Description	Range Min - Max
A2	11	Word	x10	battery voltage low setpoint	90 - 240
A4	12	Word	----	battery voltage low time delay	0 - 99
A6	13	Word	x10	battery charger fail setpoint	90 - 280
A8	14	Word	----	battery charger fail time delay	0 - 99
AA	15	Word	x10	appr low oil pressure setpoint	10 - 90
AC	16	Word	----	appr high engine temperature setpoint	40 - 105
AE	17	Word	----	low coolant temperature setpoint	20 - 99
B0	18	Word	----	fault protection timer delay	0 - 99
B2	19	Word	----	remote start time delay ON (sec)	0 - 99
B4	1A	Word	----	remote start time delay OFF (min)	0 - 99
B6	1B	Word	----	Reserved	
B8	1C	Word	----	Reserved	
BA	1D	Word	----	Reserved	
BC	1E	Word	----	Reserved	
BE	1F	Word	----	Reserved	
C0	20	Word	----	Reserved	

Note:

1. Each time, only one register can be set.
2. If the value is out of the range, a device failure response will be received.
3. For those parameters with scaling factor, the received or transmitted value is 10xReal value.

3. Supported Modbus Functions

Function Code 01 – Read Coil Status

This function allows the host to obtain the ON/OFF status of one or more logic coils in the target device. With Access 4000, this function is used to read controls. The valid relative address is from 00 to 0F and each address contains either 1 (ON) or 0(OFF).

Query Message

The query message specifies the starting coil and quantity of coils to be read. Here is an example of a request to read coils 01h - 0Ah from slave 02.

Byte	Contents	Example (Hex)
1	Slave Address	02
2	Function	01
3	Starting Address Hi	00
4	Starting Address Lo	01
5	No. of Points Hi	00
6	No. of Points Lo	0A
7	CRC Lo	ED
8	CRC Hi	FE

Response message

The coil status in the response message is packed as one coil per bit of the data field. Status is indicated as: 1 = ON; 0 = OFF. The Least Significant Bit (LSB) of the first data byte contains the coil addressed in the query. Other coils follow toward the high order end of this byte, and from 'low order to high order' in subsequent bytes.

Byte	Contents	Example (Hex)
1	Slave Address	02
2	Function	01
3	Byte count	02
4	Data (address 08 – 01)	02
5	Data(address 10 – 09)	00
6	CRC Lo	FC
7	CRC Hi	9C

Function Code 02 – Read Digital Input Status

This function enables the host to read one or more discrete inputs in the target device. With Access 4000, it can be used to read status or alarms from a generator set. The relative addresses are allocated from 00 to 2F. Each address contains either 1(OFF) or 0(ON).

Query message

The data field of the request frame consists of the relative address of the first discrete input followed by the number of discrete inputs to be read. Here is an example of a request to read inputs 00h – 15h from slave 02:

Byte	Contents	Example (Hex)
1	Slave Address	02
2	Function	02
3	Starting address Hi	00
4	Starting address Lo	00
5	No. of points Hi	00
6	No. of points Lo	16
7	CRC Lo	F9
8	CRC Hi	F7

Response message

The data field of the response frame consists of a count of the discrete input data bytes followed by that many bytes of discrete input data.

The discrete-input data bytes are packed with one bit for the status of each consecutive discrete input (1=ON, 0=OFF). The LSB of the first discrete input data byte conveys the status of the first input read. If the number of discrete inputs read is not an even multiple of eight, the last data byte will be padded with zeros on the high end. The low order bit of the first byte of the response from the slave contains the first addressed digital input.

Byte	Contents	Example(Hex)
1	Slave address	02
2	Function	02
3	Byte count	03
4	Data (address 7 – 0)	10
5	Data (address F - 8)	00
6	Data (address 15 – 10)	00
7	CRC Lo	79
8	CRC Hi	B8

Function Code 03 – Read Holding Registers

This function allows the host to obtain the contents of one or more (**maximum 10**) holding registers in the target device.

With Access 4000 this function is used to read the configuration set points. The relative addresses are allocated from 00h to 20h. Each address contains a 16 bit data. The maximum registers to be read in each time is ten, ie., the number of points should be less than 11.

Query Message

The query message specifies the starting register and the quantity of registers to be read. Here is an example of request to read register 00h and 01h from slave device 03:

Byte	Contents	Example (Hex)
1	Slave address	03
2	Function	03
3	Starting Address Hi	00
4	Starting Address Lo	00
5	No. of points Hi	00
6	No. of points Lo	02
7	CRC Lo	C5
8	CRC Hi	E9

Response message

The data field of the response frame consists of a count of the register data bytes followed by that many bytes of holding register data.

Byte	Contents	Example (Hex)
1	Slave Address	03
2	Function	03
3	Byte Count	04
4	Data Hi (address 00)	01
5	Data Lo (address 00)	2C
6	Data Hi (address 01)	00
7	Data Lo (address 01)	01
8	CRC Lo	08
9	CRC Hi	06

Function Code 04 – Read Input Registers

This function allows the host to obtain the contents of one or more (**maximum 10**) input registers in the target device.

With Access 4000 this function is used to read monitoring parameters. The relative addresses are allocated from 00h to 1Fh. Each address contains a 16 bit data. The maximum registers to be read in each time is ten, ie., the number of points in data field should be less than 11.

Query Message

The data field of the request frame consists of the relative address of the first input register followed by the number of registers to be read. Here is an example of request to read register 02h and 03h from slave device 03:

Byte	Contents	Example (Hex)
1	Slave address	03
2	Function	04
3	Starting Address Hi	00
4	Starting Address Lo	02
5	No. of points Hi	00
6	No. of points Lo	02
7	CRC Lo	01
8	CRC Hi	E9

Response message

The data field of the response frame consists of a count of the register data bytes followed by that many bytes of input register data.

Byte	Contents	Example (Hex)
1	Slave Address	03
2	Function	04
3	Byte Count	04
4	Data Hi (address 06)	00
5	Data Lo (address 06)	7F
6	Data Hi (address 07)	00
7	Data Lo (address 07)	05
8	CRC Lo	29
9	CRC Hi	9F

Function Code 05 – Force Single Coil

This function allows the host to alter the ON/OFF status of a single logic coil in the target device. With Access 4000 the function is used to change the control switches. The relative addresses are allocated from 00h to 0Fh.

Query Message

The data field of the request frame consists of the relative address of the coil followed by the desired status for that coil. A status value FF00h (65280) will activate the coil, while a status value of 0000h will deactivate it. Any other status value is illegal.

Here is an example of a request to force coil 04 (Reset Alarm) ON in slave device 01.

Byte	Contents	Example (Hex)
1	Slave Address	01
2	Function	05
3	Coil Address Hi	00
4	Coil Address Lo	04
5	Force Data Hi	FF
6	Force Data Lo	00
7	CRC Lo	CD
8	CRC Hi	FB

Response Message

The normal response is an echo of the query, returned after the coil state has been forced. Otherwise an exception response will be returned.

Byte	Contents	Example (Hex)
1	Slave address	01
2	Function	05
3	Coil Address Hi	00
4	Coil Address Lo	04
5	Force Data Hi	FF
6	Force Data Lo	00
7	CRC Lo	CD
8	CRC Hi	FB

Function Code 06 – Preset Single Register

The function enables the host to alter the contents of a single holding register in the target device. With Access 4000, It is used to change configuration set points. The relative addresses are allocated from 00h to 20h.

Query message

The data field of the request frame consists of the relative address of the holding register followed by the new value to be written to that register (most-significant byte first). Here is an example of request to preset register 10h to 0C00h in slave device 02.

Byte	Contents	Example (Hex)
1	Slave Address	02
2	Function	06
3	Register Address Hi	00
4	Register Address Lo	10
5	Preset Data Hi	0C
6	Preset Data Lo	00
7	CRC Lo	88
8	CRC Hi	39

Response Message

The normal response is an echo of the query, returned after the register contents have been set. Otherwise an exception response will be returned.

Byte	Contents	Example (Hex)
1	Slave Address	02
2	Function	06
3	Register Address Hi	00
4	Register Address Lo	10
5	Preset Data Hi	0C

6	Preset Data Lo	00
7	CRC Lo	88
8	CRC Hi	39

4. Exception Responses

When a master device sends a query to a slave device, three possible events may occur,

- 1). Normal response;
- 2). Slave does not receive the query, no response is returned and the master will eventually process a time out condition;
- 3). Slave receives the query, but detects an error, the slave will return an exception response.

The exception response message has two fields that differentiate it from a normal response:

Function field: exception response function code = query function + 80h

Data field: return the exception codes defined below,

Code	Name	Meaning
01	Illegal Function	The function received is not an allowable action for the addressed slave
02	Illegal Data Address	The address referenced in the data field is not an allowable address for the addressed slave
03	Illegal Data Value	The amount of data requested from the slave was too large for the slave to return in a single response
04	Slave device failure	The slave can not perform the function requested by the master
09	CRC error	The calculated CRC is different from the received CRC
0A	Garbled Message	The data received by the slave is too short to be a valid modbus message

Here is an example with exception response:

Query Message

Byte	Contents	Example value
1	Slave Address	0A
2	Function	01
3	Starting Address Hi	01
4	Starting Address Lo	00
5	Number of Points Hi	00
6	Number of Points Lo	0A
7	CRC Lo	BC
8	CRC Hi	8A

Exception Response

Byte	Contents	Example Value
1	Slave Address	0A
2	Function	81
3	Exception Code	02
4	CRC Lo	B0
5	CRC Hi	53

5. CRC Calculation

The procedure

The Cyclic Redundancy Checking (CRC) is a popular and effective error checking mechanism. A procedure for generating CRC is:

1. Load a 16-bit register with FFFFh, and 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 register, 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): Do nothing - proceed to step 5
(if the LSB was 1): Exclusive OR the CRC register with the polynomial value A001h.
5. Repeat steps 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 the bytes have been processed.
7. The final contents 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, i.e. the low order byte is transmitted first, followed by the high order byte.

Example CRC calculation

Message Length: 1
Message [0]: 0000 0010

Initialised CRC with all '1's
CRC = 1111 1111 1111 1111

Exclusive OR the first 8-byte of the message with the low order byte of the 16-bit CRC register, putting result into the CRC.

CRC: 1111 1111 1111 1111
Message: 0000 0000 0000 0010
NEW CRC: 1111 1111 1111 1101

LSB is '1', >>1 (no. 1) & EOR with A001.
>>1: 0111 1111 1111 1110
A001: 1010 0000 0000 0001
NEW CRC: 1101 1111 1111 1111

LSB is '1', >>1 (no. 2) & EOR with A001.
>>1: 0110 1111 1111 1111
A001: 1010 0000 0000 0001
NEW CRC: 1100 1111 1111 1110

LSB is '0', >>1 (no. 3).
>>1: 0110 0111 1111 1111

LSB is '1', >>1 (no. 4) & EOR with A001.
>>1: 0011 0011 1111 1111
A001: 1010 0000 0000 0001
NEW CRC: 1001 0011 1111 1110

LSB is '0', >>1 (no. 5).
>>1: 0100 1001 1111 1111

LSB is '1', >>1 (no. 6) & EOR with A001.
>>1: 0010 0100 1111 1111
A001: 1010 0000 0000 0001
NEW CRC: 1000 0100 1111 1110

LSB is '0', >>1 (no. 7).

>>1: 0100 0010 0111 1111

LSB is '1', >>1 (no. 8) & EOR with A001.

>>1: 0010 0001 0011 1111

A001: 1010 0000 0000 0001

NEW CRC: 1000 0001 0011 1110 = 0x813E

Swap high & Low Bytes.

Cal CRC: 16001 decimal or 3E81 Hex

A sample 'C' program

```

Unsigned short CRC16(unsigned char *message, int length)
{
    unsigned short    CRC=0xFFFF, /* CRC initialized with '1's */
                     i,           /* loop control */
                     temp;         /* For swap */

    while(length--)
    {
        CRC=CRC^(*message++);
        for(I=0;I++<8;)
        {
            if(CRC&0x0001) CRC=(CRC>>1) ^ 0xA001;
            else CRC>>=1;
        }
    }
    temp=CRC&0xFF; /* temporary storage for swap */
    CRC=((CRC>>8)&0xFF)+(temp<<8); /* swap high and low byte */

    return (CRC); /* return 16-bit CRC value */
}
    
```

Test case:

Slave	Function	Addr Hi	Addr Lo	Points Hi	Points Lo	CRC Lo	CRC Hi
03	01	00	01	00	0A	EC	2F