FG Wilson (Engineering) Ltd
Old Glenarm Read
Larne, Co. Antrim
Northern Ireland
BT40 1EJ

Modbus Communications Protocol

Implementation of Modbus communication protocol for Access 4000 control panel, including block diagram, description, source code and test results.

Date: 17 September 2000
Author: Colin McCord
Department: Electronic Control
Revision: 1.1

CONTENTS

INTRODUCTION

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”, the data and a check sum.

Access 4000 controller uses RTU transmission mode. The data format is 8 bits, 1 stop bit, and no parity. Baud rate 9600bps. The maximum waiting time for a master to get response it 128ms.

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.

For a detailed introduction into Modbus, including: address allocation tables, supported Modbus functions and exception responses. See report “Modbus Communications on Access 4000”, 05 September 2000, electronic control department (Revision 1.4).

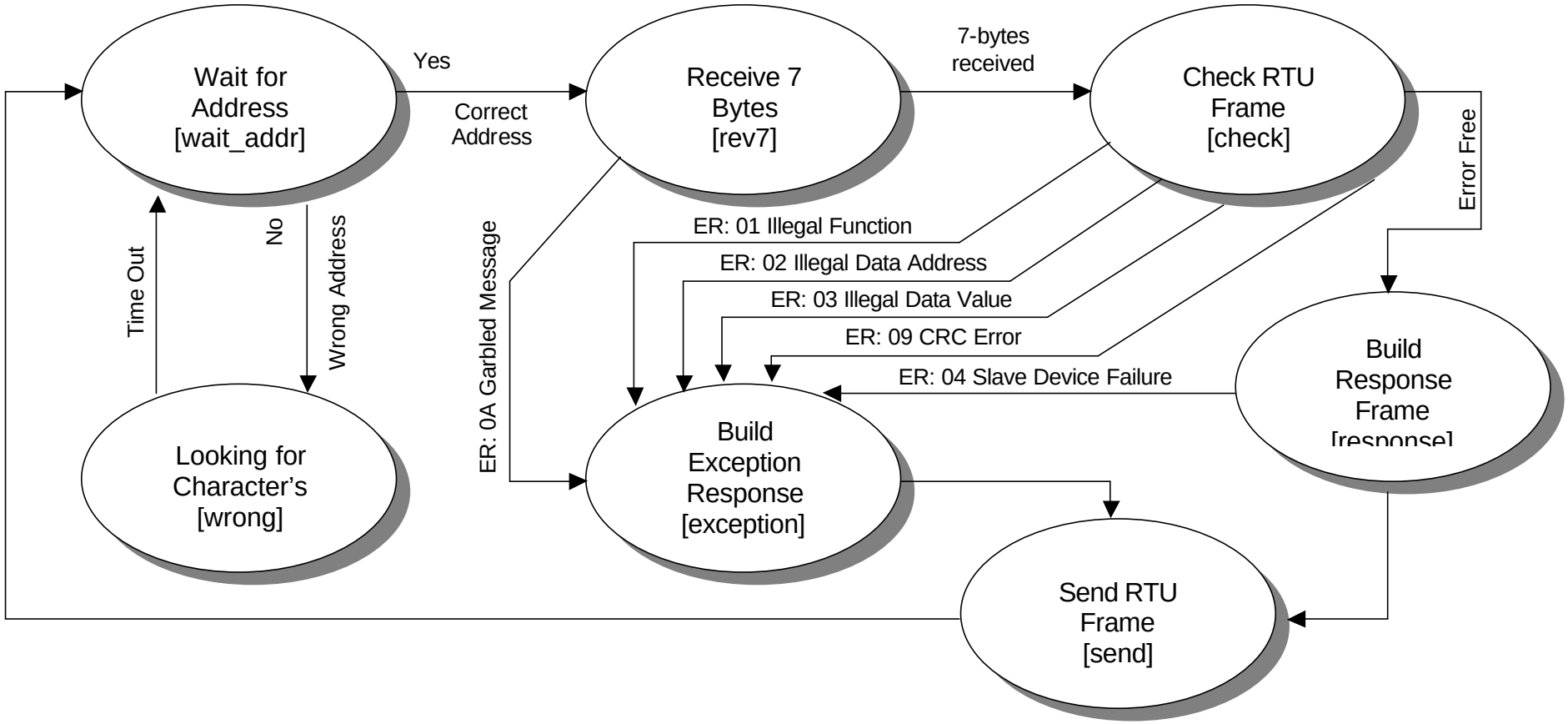
This report concentrates on the construction of the Modbus protocol, including block diagram, detailed description and source code. The Modbus program was tested using a specifically design test program, included in this report is a brief description on how to operate the test program, a number of screen dump’s form actual practical tests and the complete source code.

Tests carried out using a PC version of the Modbus protocol, when this version passed all tests. The Modbus protocol was implemented on the Access 4000 control panel using a hardware simulator to simulate switches, coils, registers etc… Once the Modbus protocol passed this set of tests, the protocol was tested on an Access 4000 panel connected to a generator. Results of all three tests are included in this report.

Address Range and Supported Function Codes

Data Type	Relative Addr	Func Code	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

MODBUS PROTOCOL BLOCK DIAGRAM



MODBUS PROTOCOL

Modbus protocol is a messaging structure, widely used to establish master-slave communication between intelligent devices. Access 4000 controller uses RTU transmission mode, 8-bits, 1 stop bit, and no parity, with a baud rate of 9600bps. The source code for the Access 4000 Modbus communication protocol can be found in "Appendix 1 – Modbus.c".

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

Description of each Function

@port void recept ():

> Interrupt service routine for receiving and sending data. An incoming character causes an interrupt to occur, which calls this function. The incoming characters are stored into an array "buffer[]", if the end of the buffer is reached, the buffer is reset and incoming characters will continue being stored onto the buffer, this time from the start.
>
> When in transmit mode, character's can be transmitted, the entire RTU frame can be transmitted by sending one character at a time to this routine, see modbus() case: send.

Int ReadChr (unsigned char *c)

> Reads a character from the buffer, This is achieved first of all by checking that there is in-fact a character to read, if no character received function returns –1.
>
> If a character change in the buffer is detected, '*c' will now contain the received character. If the end of the buffer is reached the buffer counter will reset and readchr will start's looking for new character's starting at the start of the buffer.

Int wait_chr (char *c)

> Wait's for a character, timeout occurs after 1.5ms. Modbus states that: the entire message frame must be transmitted as a continuous stream. If silent for more than 1.5 character times, timeout occurs.
>
> This function implements this by calling ReadChr(), if ReadChr() returns –1 (no character received) counter is incremented, this process repeats until a character is received or timeout has occurred. For a baud rate of 9600pbs the maximum waiting time is 1.5ms (1.5 X 0.96 = 1.44ms).

If timeout has occurred the function returns TRUE, if not FALSE is returned and *b contains the received character.

Unsigned short CRC16 (unsigned char *message, int length)

A CRC (Cyclical Redundancy Check) calculation is performed on the message contents.

Procedure: -

1. Load a 16-bit 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 register., putting the result in the CRC register.
3. Shift the CRC register one bit to the right, 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: EOR the CRC register with the polynomial value A001 hex.
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 steps 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 content of the CRC register is the CRC value.
8. Swap low & high bytes of CRC, Return CRC.

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

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

See Page 8, "Example CRC Calculation".

Void modbus()

Modbus communication protocol for access 4000.

There are 7 states:

Wait_Addr
Wrong
Rev7
Check

Exception
Response
Send

State is initialised with state = wait_addr.

Wait_Addr: When in this state, every time mobus() is called (every 125ms), the function ReadChr() is called. If no character has been received, nothing happens, and function ends, when next called the above is repeated until an incoming character is detected.

If a character has been received, the received address is compare with the address of the slave device. If equal state is changed to rev7, else state is changed to wrong.

Wrong: Wait's until unwanted message is finished, this is achieved by continuously calling the function wait_chr() until timeout occurs. When timeout occurs the unwanted message has finished and state can be changed back to wait_addr.

A Valid "Access 4000 Modbus message" will never be above 25 character's long, So something is wrong if more than 25 character's is received and timeout will occur. State will be changed back to wait_addr.

The reason why it's necessary to have a max timeout of 25 characters is because if there was a consisted stream of data at 9600bps, with no gaps between bytes (should **never** occur when using Modbus, at least 3.5 character times between RTU frames). The access 4000 panel (without this max timeout) may lockup until the data stream is stopped.

Rev7: Receives 7 bytes, this is achieved by calling the function wait_chr() 7 times and storing incoming characters onto the RTU_frame[]. If timeout occurs at any time state is change to exception (er: 0Ah Garbled Message), else state is changed to check.

Check: Check's that received frame is correct, this is implemented as follows: -

1. Calculate CRC
2. Check for valid function code.
3. Check for valid data address range.
4. Check for Illegal data Value.
5. Compere calculated CRC with received CRC

If anyone of this fails, state = exception and the appropriate error code is stored in ‘er’, else state = response.

Exception: Build’s exception response frame, this is implemented as follows: -

1. Exception Address = Query Address, so RTU_frame[0] does not need to be changed.
2. Function code = query function + 80h.
3. Exception code stored onto the frame.
4. CRC calculated & added to frame.
5. State = send.

Response: Build’s normal response frame, different function codes required different responses so the function code is important. Next state = send.

Func1 or Func2:

Increments a counter through the requested address range. If an address contains a ‘1’, ‘1’ is shifted into a temp 8-bit data register, else ‘0’ is shifted.

Once the temp 8-bit data register has been shifted 8 times, it can be stored onto the RTU frame ready for transmission.

If all requested address register have not yet been completed. Temp is reset and the process repeats until all requested address locations have been read.

Once finished, the program check’s for incomplete byte. e.g. temp finished when 8 shifts have taken place, so if only 5 shifts have taken place, 3 zeros will have to be shifted in before the byte can be added to the RTU frame.

CRC calculated & added to RTU frame.

Func3 or Func4:

Adds double byte data values onto the RTU frame with the requested limits. CRC calculated & added to RTU frame.

Func5:

Change's one coil or control switch, if data = FF00h the requested address is changed to logic '1' else change to logic '0'. If slave is unable to carry out the query, er=04 and state will be changed to exception.

Response frame is an echo of query, so no changes to the RTU frame needs to take place.

Func6:

Preset a value into a single register, the high byte of the preset value will be stored into the requested address, the low byte is stored in the next location. If slave is unable to carry out the query, er=04 and state will be changed to exception.

Response frame is an echo of query, so no changes to the RTU frame needs to take place.

Send: Sends RTU frame, this is achieved by looping through every character in the RTU frame with the transmit int enabled.

When finished state is changed back to wait_addr and buffer is cleared.

Default: If state is not any of the above something is wrong, and program will crash, so change state to wait_addr to make sure program break's out of the loop.

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 110 1
```

LSB is ‘1’, >>1 (no. 1) & EOR with A001.
```
>>1:        0111 1111 1111 1110
A001:       1010 0000 0000 0001
NEW CRC:    1101 1111 1111 111 1
```

LSB is ‘1’, >>1 (no. 2) & EOR with A001.
```
>>1:        0110 1111 1111 1111
A001:       1010 0000 0000 0001
NEW CRC:    1100 1111 1111 111 0
```

LSB is ‘0’, >>1 (no. 3).
```
>>1:        0110 0111 1111 111 1
```

LSB is ‘1’, >>1 (no. 4) & EOR with A001.
```
>>1:        0011 0011 1111 1111
A001:       1010 0000 0000 0001
NEW CRC:    1001 0011 1111 111 0
```

LSB is ‘0’, >>1 (no. 5).
```
>>1:        0100 1001 1111 111 1
```

LSB is ‘1’, >>1 (no. 6) & EOR with A001.
```
>>1:        0010 0100 1111 1111
A001:       1010 0000 0000 0001
NEW CRC:    1000 0100 1111 111 0
```

LSB is ‘0’, >>1 (no. 7).
```
>>1:        0100 0010 0111 111 1
```

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
```

TESTING

The Modbus communication protocol (Appendix 1 – Modbus.c) was tested mainly using a test program, which was designed specifically for the task. The source code for this test program can be found in "Appendix 2 – Test.c".

This was implemented in three ways: -

1) PC to Laptop (RS232).
2) PC to Access 4000 Control Panel (RS232 to RS485) using hardware simulator.
3) PC to Access 4000 Control Panel (RS232 to RS485) connected to generator.

PC running the specifically designed Modbus test program in every case. Mbpc.c (Appendix 3) is a PC version of the Modbus communication protocol, running on a laptop. Once the PC version had passed all tests Modbus was implemented on the access 4000 control panel, Modbus.c (Appendix 1) is the access 4000 Modbus communication protocol. Tests on the Access 4000 control panel carried out with hardware simulator to simulate switches, coils, register etc… Then Modbus tests where carried out once more this time on an actual generator.

The Modbus Test Program

When loaded the user has several options: -

```
Modbus Protocol Test Program
----------------------------

Wait for Response Frame   -- hit 1
Build & Transmit Query    -- hit 2
Build Query with bad CRC  -- hit 3
Manual RTU frame with CRC -- hit 4
Manual RTU frame no CRC   -- hit 5
Auto Random blocks - TX   -- hit 6
Simulate Random Noise     -- hit 7
Quit                      -- hit Q
```

1. Wait for Response Frame:

Wait's for a Modbus response frame from a slave device. Once an incoming frame is detected it decodes the frame into its key components (address, function, data, CRC, etc…) and display's them onto the screen, along with the actual time in milliseconds it took the slave to respond. Calculates CRC and compares it with the received one. The user can cancel the function at anytime by pressing [anykey]. Timeout will occur after 65 seconds.

2. Build & Transmit Query:

The user enters (via keyboard) a test RTU frame; CRC is calculated and added to the end of the frame. Frame is then transmitted. The function above "Wait for Response Frame" is now called.

3. Build Query with bad CRC:
Same as "Build & Transmit Query" except that the CRC is not calculated and a bad CRC is added to the end of the frame. The received response should be an exception response with error code 0x09 – CRC error.

4. Manual RTU frame with CRC:
The user enters a value for RTU_Frame[0], RTU_Frame[1], RTU_Frame[2], etc… There's no limit on the frame's size, the user simply enters DF when finished. Once the user is finished entering data, CRC is calculated and added to the end of the frame. This function is mainly used to test exception error 0x0A Garbled message, this is achievement by sending a RTU frame smaller than 8 bytes (including 2 byte CRC).

5. Manual RTU frame no CRC:
Same as "Manual RTU frame with CRC", but without the added CRC.

6. Auto Random blocks – TX:
Transmits "Random RTU frames of Random Size" or "Address & Random data of Random Size" or "Address & function & Random data of Random size" data for ever, until stopped by the user . This is used to test Modbus reliability.

7. Simulate Random Noise:
Simulates random noise, by sending out random bytes over and over again. Either with "no delay" or "random delay" or "predefined delay" between bytes. This is used to test Modbus reliability.

Q. Quit:
Exit from Modbus test program.

Testing Modbus

The following tests will be carried out using the Modbus test program, on both the laptop emulating access 4000 and the access 4000 control panel: -

- Function 1 – Read Coil Status
  - General operation.
  - Exception error: 02 – Illegal Data Address.
  - Exception error: 03 – Illegal Data Value.

- Function 2 – Read Input Status
  - General operation.
  - Exception error: 02 – Illegal Data Address.
  - Exception error: 03 – Illegal Data Value.

- Function 3 – Read Holding Registers
  - General operation.
  - Exception error: 02 – Illegal Data Address.
  - Exception error: 03 – Illegal Data Value.

- Function 4 – Read Input Registers
  - General operation.
  - Exception error: 02 – Illegal Data Address.
  - Exception error: 03 – Illegal Data Value.

- Function 5 – Force Single Coil
  - General operation.
  - Exception error: 02 – Illegal Data Address.
  - Exception error: 03 – Illegal Data Value.

- Function 6 – Preset Single Register
  - General operation.
  - Exception error: 02 – Illegal Data Address.

- Unknown function
  - Exception error: 01 – Illegal Function

- Bad CRC
  - Exception error: 09 – CRC error.

- Short message
  - Exception error: 0A – Garbled Message.

- Modbus reliability
  - Random blocks.
  - Simulated random noise.

- Response Time
  - Check slave response time.

Other Tests carried out (access 4000 control panel only): -

- Function 5 – Force Single Coil
  - Exception error: 04 – Slave device failure

- Function 6 – Preset Single Register
  - Exception error: 04 – Slave device failure

- General operation
  - Every valid control coil
  - Every valid status/Alarm
  - Every valid monitoring register
  - Every valid configuration register

TEST RESULTS – LAPTOP

Function 1 – Read Coil Status

Address allocation table:

Address	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F
Data	1	1	0	0	1	0	1	0	0	0	1	1	1	0	1	1

General operation: Read 00h to 0Fh from slave 1: -

```
Build & Transmit Query
------------------------

Please enter test RTU Query Frame: -
Slave Address    (Hex - 1 byte)  eg   00: 1
Function Code    (Hex - 1 byte)  eg   02: 1

Read Coil Status (1) - Slave 1
Starting Address (Hex - 2 bytes) eg 0001: 0
No. of Points    (Hex - 2 bytes) eg 000A: 10
Calculated CRC (Hex):    3DC6
Total Frame Length (Dec): 8

Sending RTU Frame (HEX): -
1 1 0 0 0 10 3D C6
```

```
Wait for Response Frame
------------------------
Waiting for response (press a key to stop)
Slave Responded in 90ms (MAX 128ms): -

Address:     Slave 1
Function:    Read Coil Status (1)
Byte Count:  2
Data:        53 DC
16-bit CRC:  8495
Cal. CRC     8495
[Message Correct - CRC Matches]

Received RTU Frame (HEX): 1 1 2 53 DC 84 95
```

53h = 0101 0011, CDh = 1101 1100

Exception error: 02 Read 10h to 15h from slave 1: -

```
Build & Transmit Query
------------------------

Please enter test RTU Query Frame: -
Slave Address    (Hex - 1 byte)  eg   00: 1
Function Code    (Hex - 1 byte)  eg   02: 1

Read Coil Status (1) - Slave 1
Starting Address (Hex - 2 bytes) eg 0001: 10
No. of Points    (Hex - 2 bytes) eg 000A: 6
Calculated CRC (Hex):    BDCD
Total Frame Length (Dec): 8

Sending RTU Frame (HEX): -
1 1 0 10 0 6 BD CD
```

```
Wait for Response Frame
------------------------
Waiting for response (press a key to stop)
Slave Responded in 82ms (MAX 128ms): -

Address:     Slave 1
Function:    Warning: Exception Error (81)
Error Code:  Illegal Data Address (2)
16-bit CRC:  C191
Cal. CRC     C191
[Message Correct - CRC Matches]

Received RTU Frame (HEX): 1 81 2 C1 91
```

Exception error: 03 Read 0h to 15h from slave 1: -

```
Build & Transmit Query
------------------------

Please enter test RTU Query Frame: -
Slave Address    (Hex - 1 byte)  eg   00: 1
Function Code    (Hex - 1 byte)  eg   02: 1

Read Coil Status (1) - Slave 1
Starting Address (Hex - 2 bytes) eg 0001: 0
No. of Points    (Hex - 2 bytes) eg 000A: 16
Calculated CRC (Hex):    BDC4
Total Frame Length (Dec): 8

Sending RTU Frame (HEX): -
1 1 0 0 0 16 BD C4
```

```
Wait for Response Frame
------------------------
Waiting for response (press a key to stop)
Slave Responded in 64ms (MAX 128ms): -

Address:     Slave 1
Function:    Warning: Exception Error (81)
Error Code:  Illegal Data Value (3)
16-bit CRC:  51
Cal. CRC     51
[Message Correct - CRC Matches]

Received RTU Frame (HEX): 1 81 3 0 51
```

Function 2 – Read Input Status

Address allocation table:

Address	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
Data	1	0	1	1	0	0	1	1	1	0	0	0	1	1	1	1
Address	10	11	12	13	14	15	16	17	18	19	1A	1B	1C	1D	1E	1F
Data	0	0	0	0	1	1	1	1	1	0	0	0	0	0	1	1
Address	20	21	22	23	24	25	26	27	28	29	2A	2B	2C	2D	2E	2F
Data	1	1	1	1	0	0	0	0	0	0	1	0	1	0	1	1

General operation: Read 00h to 2Fh from slave 1:

```
Build & Transmit Query
-----------------------

Please enter test RTU Query Frame: -
Slave Address    (Hex - 1 byte)  eg   00: 1
Function Code    (Hex - 1 byte)  eg   02: 2

Read Input Status (2) - Slave 1
Starting Address (Hex - 2 bytes) eg 0001: 0
No. of Points    (Hex - 2 bytes) eg 000A: 30
Calculated CRC (Hex):    781E
Total Frame Length (Dec): 8

Sending RTU Frame (HEX): -
1 2 0 0 0 30 78 1E
```

```
Wait for Response Frame
------------------------
Waiting for response (press a key to stop)
Slave Responded in 87ms (MAX 128ms): -

Address:     Slave 1
Function:    Read Input Status (2)
Byte Count:  6
Data:        CD F1 F0 C1 F D4
16-bit CRC:  EA22
Cal. CRC     EA22
[Message Correct - CRC Matches]

Received RTU Frame (HEX): 1 2 6 CD F1 F0 C1
F D4 EA 22
```

CDh = 1100 1101, F1h = 1111 0001, F0h = 1111 0000, C1h = 1100 0001, 0Fh = 0000 1111, D4h = 11010100

Exception error: 02 Read 35h to 36h from slave 1: -

```
Build & Transmit Query
-----------------------

Please enter test RTU Query Frame: -
Slave Address    (Hex - 1 byte)  eg   00: 1
Function Code    (Hex - 1 byte)  eg   02: 2

Read Input Status (2) - Slave 1
Starting Address (Hex - 2 bytes) eg 0001: 35
No. of Points    (Hex - 2 bytes) eg 000A: 2
Calculated CRC (Hex):    E9C5
Total Frame Length (Dec): 8

Sending RTU Frame (HEX): -
1 2 0 35 0 2 E9 C5
```

```
Wait for Response Frame
------------------------
Waiting for response (press a key to stop)
Slave Responded in 47ms (MAX 128ms): -

Address:     Slave 1
Function:    Warning: Exception Error (82)
Error Code:  Illegal Data Address (2)
16-bit CRC:  C161
Cal. CRC     C161
[Message Correct - CRC Matches]

Received RTU Frame (HEX): 1 82 2 C1 61
```

Exception error: 03 Read 00h to 30h from slave 1:

```
Build & Transmit Query
-----------------------

Please enter test RTU Query Frame: -
Slave Address    (Hex - 1 byte)  eg   00: 1
Function Code    (Hex - 1 byte)  eg   02: 2

Read Input Status (2) - Slave 1
Starting Address (Hex - 2 bytes) eg 0001: 0
No. of Points    (Hex - 2 bytes) eg 000A: 31
Calculated CRC (Hex):    B9DE
Total Frame Length (Dec): 8

Sending RTU Frame (HEX): -
1 2 0 0 0 31 B9 DE
```

```
Wait for Response Frame
------------------------
Waiting for response (press a key to stop)
Slave Responded in 81ms (MAX 128ms): -

Address:     Slave 1
Function:    Warning: Exception Error (82)
Error Code:  Illegal Data Value (3)
16-bit CRC:  A1
Cal. CRC     A1
[Message Correct - CRC Matches]

Received RTU Frame (HEX): 1 82 3 0 A1
```

Function 3 – Read Holding Registers

Address allocation table, 00h to 20h, table empty all ‘0’s.

General operation: Read 00h to 09h from slave 1:

```
Build & Transmit Query
-----------------------

Please enter test RTU Query Frame: -
Slave Address    (Hex - 1 byte)  eg   00: 1
Function Code    (Hex - 1 byte)  eg   02: 3

Read Holding Registers (3) - Slave 1
Starting Address (Hex - 2 bytes) eg 0001: 0
No. of Points    (Hex - 2 bytes) eg 000A: A
Calculated CRC (Hex):    C5CD
Total Frame Length (Dec): 8

Sending RTU Frame (HEX): -
1 3 0 0 0 A C5 CD
```

```
Wait for Response Frame
------------------------
Waiting for response (press a key to stop)
Slave Responded in 101ms (MAX 128ms): -

Address:     Slave 1
Function:    Read Holding Registers (3)
Byte Count:  20
Data:        00 00 00 00 00 00 00 00 00 00
16-bit CRC:  A367
Cal. CRC     A367
[Message Correct - CRC Matches]

Received RTU Frame (HEX): 1 3 14 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 A3 67
```

Using function 6, 00h = ABCD, 05h = DDEE, 09h = EEEE

General operation: Read 00h to 09h from slave 1:

```
Build & Transmit Query
-----------------------

Please enter test RTU Query Frame: -
Slave Address    (Hex - 1 byte)  eg   00: 1
Function Code    (Hex - 1 byte)  eg   02: 3

Read Holding Registers (3) - Slave 1
Starting Address (Hex - 2 bytes) eg 0001: 0
No. of Points    (Hex - 2 bytes) eg 000A: A
Calculated CRC (Hex):    C5CD
Total Frame Length (Dec): 8
```

```
Wait for Response Frame
------------------------
Waiting for response (press a key to stop)
Slave Responded in 97ms (MAX 128ms): -

Address:     Slave 1
Function:    Read Holding Registers (3)
Byte Count:  20
Data:        ABCD 00 00 00 00 DDEE 00 00 00
             EEEE
16-bit CRC:  5103
Cal. CRC     5103
```

Exception error: 02 Read 30h from slave 1: -

```
Build & Transmit Query
-----------------------

Please enter test RTU Query Frame: -
Slave Address    (Hex - 1 byte)  eg   00: 1
Function Code    (Hex - 1 byte)  eg   02: 3

Read Holding Registers (3) - Slave 1
Starting Address (Hex - 2 bytes) eg 0001: 30
No. of Points    (Hex - 2 bytes) eg 000A: 1
Calculated CRC (Hex):    8405
Total Frame Length (Dec): 8

Sending RTU Frame (HEX): -
1 3 0 30 0 1 84 5
```

```
Wait for Response Frame
------------------------
Waiting for response (press a key to stop)
Slave Responded in 44ms (MAX 128ms): -

Address:     Slave 1
Function:    Warning: Exception Error (83)
Error Code:  Illegal Data Address (2)
16-bit CRC:  C0F1
Cal. CRC     C0F1
[Message Correct - CRC Matches]

Received RTU Frame (HEX): 1 83 2 C0 F1
```

Exception error: 03 Read 20h & 21h from slave 1:

```
Build & Transmit Query
-----------------------

Please enter test RTU Query Frame: -
Slave Address    (Hex - 1 byte)  eg   00: 1
Function Code    (Hex - 1 byte)  eg   02: 3

Read Holding Registers (3) - Slave 1
Starting Address (Hex - 2 bytes) eg 0001: 20
No. of Points    (Hex - 2 bytes) eg 000A: 2
Calculated CRC (Hex):    C5C1
Total Frame Length (Dec): 8

Sending RTU Frame (HEX): -
1 3 0 20 0 2 C5 C1
```

```
Wait for Response Frame
------------------------
Waiting for response (press a key to stop)
Slave Responded in 44ms (MAX 128ms): -

Address:     Slave 1
Function:    Warning: Exception Error (83)
Error Code:  Illegal Data Value (3)
16-bit CRC:  131
Cal. CRC     131
[Message Correct - CRC Matches]

Received RTU Frame (HEX): 1 83 3 1 31
```

Function 4 – Read Input Registers

Address allocation table:

Address	0		1		2		3		4		5		6		7	
Data	40	41	42	43	44	45	46	47	48	49	4A	4B	4C	4D	4E	4F
Address	8		9		A		B		C		D		E		F	
Data	50	51	52	53	54	55	56	57	58	59	5A	5B	5C	5D	5E	5F
Address	10		11		12		13		14		15		16		17	
Data	60	61	62	63	64	65	66	67	68	69	6A	6B	6C	6D	6E	6F
Address	18		19		1A		1B		1C		1D		1E		1F	
Data	70	71	72	73	74	75	76	77	78	79	7A	7B	7C	7D	7E	7F

General operation: Read 00h to 09h from slave 1:

```
Build & Transmit Query
-----------------------

Please enter test RTU Query Frame: -
Slave Address    (Hex - 1 byte)  eg   00: 1
Function Code    (Hex - 1 byte)  eg   02: 4

Read Input Registers (4) - Slave 1
Starting Address (Hex - 2 bytes) eg 0001: 0
No. of Points    (Hex - 2 bytes) eg 000A: A
Calculated CRC (Hex):    700D
Total Frame Length (Dec): 8

Sending RTU Frame (HEX): -
1 4 0 0 0 A 70 D
```

```
Wait for Response Frame
------------------------
Waiting for response (press a key to stop)
Slave Responded in 67ms (MAX 128ms): -

Address:     Slave 1
Function:    Read Input Registers (4)
Byte Count:  20
Data:        4041 4243 4445 4647 4849 4A4B
             4C4D 4E4F 5051 5253
16-bit CRC:  C743
Cal. CRC     C743
[Message Correct - CRC Matches]

Received RTU Frame (HEX): 1 4 14 40 41 42 43
44 45 46 47 48 49 4A 4B 4C 4D 4E 4F 50 51 52
53 C7 43
```

Exception error: 02 Read 30h from slave 1: -

```
Build & Transmit Query
-----------------------

Please enter test RTU Query Frame: -
Slave Address    (Hex - 1 byte)  eg   00: 1
Function Code    (Hex - 1 byte)  eg   02: 4

Read Input Registers (4) - Slave 1
Starting Address (Hex - 2 bytes) eg 0001: 30
No. of Points    (Hex - 2 bytes) eg 000A: 1
Calculated CRC (Hex):    31C5
Total Frame Length (Dec): 8

Sending RTU Frame (HEX): -
1 4 0 30 0 1 31 C5
```

```
Wait for Response Frame
------------------------
Waiting for response (press a key to stop)
Slave Responded in 108ms (MAX 128ms): -

Address:     Slave 1
Function:    Warning: Exception Error (84)
Error Code:  Illegal Data Address (2)
16-bit CRC:  C2C1
Cal. CRC     C2C1
[Message Correct - CRC Matches]

Received RTU Frame (HEX): 1 84 2 C2 C1
```

Exception error: 03 Read 1Fh & 20h from slave 1:

```
Build & Transmit Query
-----------------------

Please enter test RTU Query Frame: -
Slave Address    (Hex - 1 byte)  eg   00: 1
Function Code    (Hex - 1 byte)  eg   02: 4

Read Input Registers (4) - Slave 1
Starting Address (Hex - 2 bytes) eg 0001: 1F
No. of Points    (Hex - 2 bytes) eg 000A: 2
Calculated CRC (Hex):    400D
Total Frame Length (Dec): 8

Sending RTU Frame (HEX): -
1 4 0 1F 0 2 40 D
```

```
Wait for Response Frame
------------------------
Waiting for response (press a key to stop)
Slave Responded in 54ms (MAX 128ms): -

Address:     Slave 1
Function:    Warning: Exception Error (84)
Error Code:  Illegal Data Value (3)
16-bit CRC:  301
Cal. CRC     301
[Message Correct - CRC Matches]

Received RTU Frame (HEX): 1 84 3 3 1
```

Function 5 – Force Single Coil

Address allocation table:

Address	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F
Data	1	1	0	0	1	0	1	0	0	0	1	1	1	0	1	1

General operation: Change 00h to 0 from slave 1:

```
Build & Transmit Query
----------------------

Please enter test RTU Query Frame: -
Slave Address    (Hex - 1 byte)  eg   00: 1
Function Code    (Hex - 1 byte)  eg   02: 5

Force Single Coil (5) - Slave 1
Coil Address     (Hex - 2 bytes) eg 0001: 0
Force Data         (Hex) eg FF00 or 0000: 0
Calculated CRC (Hex):    CDCA
Total Frame Length (Dec): 8

Sending RTU Frame (HEX): -
1 5 0 0 0 0 CD CA
```

```
Wait for Response Frame
-----------------------
Waiting for response (press a key to stop)
Slave Responded in 16ms (MAX 128ms): -

Address:     Slave 1
Function:    Force Single Coil (5)
Coil Addre:  00
Force Data:  00
16-bit CRC:  CDCA
Cal. CRC     CDCA
[Message Correct - CRC Matches]

Received RTU Frame (HEX): 1 5 0 0 0 0 CD CA
```

Check location 00h has changed to 0, using function 1:

```
Build & Transmit Query
----------------------

Please enter test RTU Query Frame: -
Slave Address    (Hex - 1 byte)  eg   00: 1
Function Code    (Hex - 1 byte)  eg   02: 1

Read Coil Status (1) - Slave 1
Starting Address (Hex - 2 bytes) eg 0001: 0
No. of Points    (Hex - 2 bytes) eg 000A: 1
Calculated CRC (Hex):    FDCA
Total Frame Length (Dec): 8

Sending RTU Frame (HEX): -
1 1 0 0 0 1 FD CA
```

```
Wait for Response Frame
-----------------------
Waiting for response (press a key to stop)
Slave Responded in 27ms (MAX 128ms): -

Address:     Slave 1
Function:    Read Coil Status (1)
Byte Count:  1
Data:        0
16-bit CRC:  5188
Cal. CRC     5188
[Message Correct - CRC Matches]

Received RTU Frame (HEX): 1 1 1 0 51 88
```

Exception error: 02 change address 10h to logic 1 (FF00h) - slave 1: -

```
Build & Transmit Query
----------------------

Please enter test RTU Query Frame: -
Slave Address    (Hex - 1 byte)  eg   00: 1
Function Code    (Hex - 1 byte)  eg   02: 5

Force Single Coil (5) - Slave 1
Coil Address   (Hex - 2 bytes) eg 0001: 10
Force Data       (Hex) eg FF00 or 0000: FF00
Calculated CRC (Hex):    8DFF
Total Frame Length (Dec): 8

Sending RTU Frame (HEX): -
1 5 0 10 FF 0 8D FF
```

```
Wait for Response Frame
-----------------------
Waiting for response (press a key to stop)
Slave Responded in 48ms (MAX 128ms): -

Address:     Slave 1
Function:    Warning: Exception Error (85)
Error Code:  Illegal Data Address (2)
16-bit CRC:  C351
Cal. CRC     C351
[Message Correct - CRC Matches]

Received RTU Frame (HEX): 1 85 2 C3 51
```

Exception error: 03 change address 00h to logic ? (1234h) - slave 1:

```
Build & Transmit Query
----------------------

Please enter test RTU Query Frame: -
Slave Address    (Hex - 1 byte)  eg   00: 1
Function Code    (Hex - 1 byte)  eg   02: 5

Force Single Coil (5) - Slave 1
Coil Address   (Hex - 2 bytes) eg 0001: 0
Force Data       (Hex) eg FF00 or 0000: 1234
Calculated CRC (Hex):    C0BD
Total Frame Length (Dec): 8

Sending RTU Frame (HEX): -
1 5 0 0 12 34 C0 BD
```

```
Wait for Response Frame
-----------------------
Waiting for response (press a key to stop)
Slave Responded in 114ms (MAX 128ms): -

Address:     Slave 1
Function:    Warning: Exception Error (85)
Error Code:  Illegal Data Value (3)
16-bit CRC:  291
Cal. CRC     291
[Message Correct - CRC Matches]

Received RTU Frame (HEX): 1 85 3 2 91
```

Function 6 – Preset Single Coil

Address allocation table: 00h to 20h – table empty

General Operation: Preset register 00h to ABCDh: -

```
Build & Transmit Query
------------------------

Please enter test RTU Query Frame: -
Slave Address    (Hex - 1 byte)  eg   00: 1
Function Code    (Hex - 1 byte)  eg   02: 6

Preset Single Register (6) - Slave 1
Register Addr  (Hex - 2 bytes) eg 0001: 0
Preset Data    (Hex - 2 bytes) eg 000A: ABCD
Calculated CRC (Hex):    376F
Total Frame Length (Dec): 8

Sending RTU Frame (HEX): -
1 6 0 0 AB CD 37 6F
```

```
Wait for Response Frame
------------------------
Waiting for response (press a key to stop)
Slave Responded in 10ms (MAX 128ms): -

Address:     Slave 1
Function:    Preset Single Register (6)
Reg. Addre:  00
Preset Data: abcd
16-bit CRC:  376F
Cal. CRC     376F
[Message Correct - CRC Matches]

Received RTU Frame (HEX): 1 6 0 0 AB CD 37
6F
```

Also: 20h to FEDCh, 18h to CCDDh, 10h to 1234h, 08h to AABBh

Check, if change is taking place, screen dump of address allocation table: -

```
Configuration (80 - C0)
ABCD 00 00 00 00 00 00 00 AABB 00 00 00 00 00 00 00 1234 00 00 00 00 00 00 00 CCDD 00
00 00 00 00 00 00 FEDC
```

Exception error: 02 change address 40h to 4321h - slave 1: -

```
Build & Transmit Query
------------------------

Please enter test RTU Query Frame: -
Slave Address    (Hex - 1 byte)  eg   00: 1
Function Code    (Hex - 1 byte)  eg   02: 6

Preset Single Register (6) - Slave 1
Register Addr  (Hex - 2 bytes) eg 0001: 40
Preset Data    (Hex - 2 bytes) eg 000A: 4321
Calculated CRC (Hex):    7936
Total Frame Length (Dec): 8

Sending RTU Frame (HEX): -
1 6 0 40 43 21 79 36
```

```
Wait for Response Frame
------------------------
Waiting for response (press a key to stop)
Slave Responded in 26ms (MAX 128ms): -

Address:     Slave 1
Function:    Warning: Exception Error (86)
Error Code:  Illegal Data Address (2)
16-bit CRC:  C3A1
Cal. CRC     C3A1
[Message Correct - CRC Matches]

Received RTU Frame (HEX): 1 86 2 C3 A1
```

Exception error: 02 change address 21h to AAAAh - slave 1: -

```
Build & Transmit Query
------------------------

Please enter test RTU Query Frame: -
Slave Address    (Hex - 1 byte)  eg   00: 1
Function Code    (Hex - 1 byte)  eg   02: 6

Preset Single Register (6) - Slave 1
Register Addr  (Hex - 2 bytes) eg 0001: 21
Preset Data    (Hex - 2 bytes) eg 000A: AAAA
Calculated CRC (Hex):    271F
Total Frame Length (Dec): 8

Sending RTU Frame (HEX): -
1 6 0 21 AA AA 27 1F
```

```
Wait for Response Frame
------------------------
Waiting for response (press a key to stop)
Slave Responded in 37ms (MAX 128ms): -

Address:     Slave 1
Function:    Warning: Exception Error (86)
Error Code:  Illegal Data Address (2)
16-bit CRC:  C3A1
Cal. CRC     C3A1
[Message Correct - CRC Matches]

Received RTU Frame (HEX): 1 86 2 C3 A1
```

Unknown Function

Exception error: 01 – Illegal Function (7): -

```
Build & Transmit Query
-----------------------

Please enter test RTU Query Frame: -
Slave Address    (Hex - 1 byte)  eg   00: 1
Function Code    (Hex - 1 byte)  eg   02: 7

Unknown Function Code (7) - Slave 1
Starting Address (Hex - 2 bytes) eg 0001: 1
No. of Points    (Hex - 2 bytes) eg 000A: 2
Calculated CRC (Hex):    640B
Total Frame Length (Dec): 8

Sending RTU Frame (HEX): -
1 7 0 1 0 2 64 B
```

```
Wait for Response Frame
------------------------
Waiting for response (press a key to stop)
Slave Responded in 91ms (MAX 128ms): -

Address:     Slave 1
Function:    Warning: Exception Error (87)
Error Code:  Illegal Function (1)
16-bit CRC:  8230
Cal. CRC     8230
[Message Correct - CRC Matches]

Received RTU Frame (HEX): 1 87 1 82 30
```

Exception error: 01 – Illegal Function (21): -

```
Build & Transmit Query
-----------------------

Please enter test RTU Query Frame: -
Slave Address    (Hex - 1 byte)  eg   00: 1
Function Code    (Hex - 1 byte)  eg   02: 21

Unknown Function Code (21) - Slave 1
Starting Address (Hex - 2 bytes) eg 0001: 1
No. of Points    (Hex - 2 bytes) eg 000A: 1
Calculated CRC (Hex):    2DCD
Total Frame Length (Dec): 8

Sending RTU Frame (HEX): -
1 21 0 1 0 1 2D CD
```

```
Wait for Response Frame
------------------------
Waiting for response (press a key to stop)
Slave Responded in 30ms (MAX 128ms): -

Address:     Slave 1
Function:    Warning: Exception Error (A1)
Error Code:  Illegal Function (1)
16-bit CRC:  9850
Cal. CRC     9850
[Message Correct - CRC Matches]

Received RTU Frame (HEX): 1 A1 1 98 50
```

Bad CRC

Exception error: 09 – CRC error: -

```
Build Query with bad CRC
-------------------------

Please enter test RTU Query Frame: -
Slave Address    (Hex - 1 byte)  eg   00: 1
Function Code    (Hex - 1 byte)  eg   02: 1

Read Coil Status (1) - Slave 1
Starting Address (Hex - 2 bytes) eg 0001: 0
No. of Points    (Hex - 2 bytes) eg 000A: 10
False CRC (Hex):         ABCD
Total Frame Length (Dec): 8

Sending RTU Frame (HEX): -
1 1 0 0 0 10 AB CD
```

```
Wait for Response Frame
------------------------
Waiting for response (press a key to stop)
Slave Responded in 96ms (MAX 128ms): -

Address:     Slave 1
Function:    Warning: Exception Error (81)
Error Code:  CRC error (9)
16-bit CRC:  8056
Cal. CRC     8056
[Message Correct - CRC Matches]

Received RTU Frame (HEX): 1 81 9 80 56
```

Exception error: 09 – CRC error: -

```
Build Query with bad CRC
-------------------------

Please enter test RTU Query Frame: -
Slave Address    (Hex - 1 byte)  eg   00: 1
Function Code    (Hex - 1 byte)  eg   02: 5

Force Single Coil (5) - Slave 1
Coil Address   (Hex - 2 bytes) eg 0001: FFFF
Force Data       (Hex) eg FF00 or 0000: AAAA
False CRC (Hex):         ABCD
Total Frame Length (Dec): 8

Sending RTU Frame (HEX): -
1 5 FF FF AA AA AB CD
```

```
Wait for Response Frame
------------------------
Waiting for response (press a key to stop)
Slave Responded in 21ms (MAX 128ms): -

Address:     Slave 1
Function:    Warning: Exception Error (85)
Error Code:  CRC error (9)
16-bit CRC:  8296
Cal. CRC     8296
[Message Correct - CRC Matches]

Received RTU Frame (HEX): 1 85 9 82 96
```

Short Message

Exception error: 0A – Garbled Message: -

```
Manual RTU frame with added CRC
--------------------------------
Enter RTU_frame in byte's enter 'DF' when
finished
RTU_frame[0] (e.g 01) = 1
RTU_frame[1] (e.g 01) = 1
RTU_frame[2] (e.g 01) = 12
RTU_frame[3] (e.g 01) = 34
RTU_frame[4] (e.g 01) = 56
RTU_frame[5] (e.g 01) = df
Calulated CRC (Hex):     2EC7
Total Frame Length (Dec): 7

Sending RTU Frame (HEX): -
1 1 12 34 56 2E C7
```

```
Wait for Response Frame
------------------------
Waiting for response (press a key to stop)
Slave Responded in 45ms (MAX 128ms): -

Address:     Slave 1
Function:    Warning: Exception Error (81)
Error Code:  Garbled Message (A)
16-bit CRC:  C057
Cal. CRC     C057
[Message Correct - CRC Matches]

Received RTU Frame (HEX): 1 81 A C0 57
```

Exception error: 0A – Garbled Message: -

```
Manual RTU frame with added CRC
--------------------------------
Enter RTU_frame in byte's enter 'DF' when
finished
RTU_frame[0] (e.g 01) = 1
RTU_frame[1] (e.g 01) = 4
RTU_frame[2] (e.g 01) = 0
RTU_frame[3] (e.g 01) = 80
RTU_frame[4] (e.g 01) = df
Calulated CRC (Hex):     41B9
CRC_LOW = B9, CRC_High = 41
Total Frame Length (Dec): 6

Sending RTU Frame (HEX): -
1 4 0 80 41 B9
```

```
Wait for Response Frame
------------------------
Waiting for response (press a key to stop)
Slave Responded in 56ms (MAX 128ms): -

Address:     Slave 1
Function:    Warning: Exception Error (84)
Error Code:  Garbled Message (A)
16-bit CRC:  C307
Cal. CRC     C307
[Message Correct - CRC Matches]

Received RTU Frame (HEX): 1 84 A C3 7
```

Exception error: 0A – Garbled Message: -

```
Manual RTU frame no CRC
------------------------
Enter RTU_frame in byte's enter 'DF' when
finished
RTU_frame[0] (e.g 01) = 1
RTU_frame[1] (e.g 01) = DF
Total Frame Length (Dec): 1

Sending RTU Frame (HEX): -
1
```

```
Wait for Response Frame
------------------------
Waiting for response (press a key to stop)
Slave Responded in 90ms (MAX 128ms): -

Address:     Slave 1
Function:    Warning: Exception Error (80)
Error Code:  Garbled Message (A)
16-bit CRC:  C1C7
Cal. CRC     C1C7
[Message Correct - CRC Matches]

Received RTU Frame (HEX): 1 80 A C1 C7
```

Modbus Reliability

Auto random blocks: -

```
Auto Random blocks - TX
------------------------

Transmits Random RTU_frames of Random Size      -- hit 1
Address & Random data of Random Size            -- hit 2
Address & function & Random data of Random size -- hit 3
```

Program left running for 30 min in each mode above, the Modbus program did not crash.

Simulated random noise: -

```
Simulate Random Noise
----------------------
No delay between bytes                  -- Hit 1
Random delay (0 to 100ms) between bytes -- hit 2
User sets delay in milliseconds         -- hit 3
Cancel - Return to Main Menu            -- hit 4
```

Program left running for 30 min in each mode above, the Modbus program did not crash.

Screen dump of Addr & func & Rand data of Rand size (add = 1, func = 1): -

```
                              ++¼oO£¦ ¦_É÷+_ñ¦¼=_ +b?? Yrd»ºDê
 5· + _ù+÷ô+u 8P__! .P ¦_       -++o·= ½??ÿ¦£âc-{¡ì_ë^+xÆ __î _/0-_p¦++-.+ e[µ"_  -R;
            + +]C$-u'4¿ï-4+-_3@T*o@¶\<±_??$++f(;__ó` +-¶_% ¦+ûèù«__ewéâí
                              #•g£O+6æºP??ºc^+Q¦ >_ +
                             ÷¦^»+ä÷Ñ+¦_ƒ+--??£•w•- ƒ.ê`.
  7};_•yB ~_P+òM_H__+ K 9??ª§_ô 6+ò+¡ë+<ÇO_+ÜçK+ö ¦¦¦'Le+Ep_¬D**k_--¤iïº + _/¦wQ
               ù ___Çs)np +__ü*»a&_k q'| -++-G öà$cB¶V-+.¦¶Q__-_è{v

                              co_dÅ _¢Mó+_??î96H ä¢¦v_Bê
   ¦ !±+§-å{+ _p+-¤f!-p_4+??sé--¦-+<-+\]g_<S+ tE¦+;nö(è}I èí_+D°_¶_Ob+JA +c ] ¥
    «± +kû|ì$<Ä` f_â_`_-_y-Öh£¼_j+[_M?9ïó??7_¦Uòµ£ä¢-±+`_@-+èe+Fs{( êjí__Çì·)w_+{
                              -¦Y²Hánòü J÷}½µ-++??
  ½H-_Y!. ë\[äzI_Q8¬ ??¦•G_ _+kJ_]buG_òsP_û cLÜàè4ª ç_#÷òr++5å` _ ¦_+_ c<tz-_s_Sm-
_ P+-++ä á¦¶5&ßg___æ % [•ì¼÷1ö+if1 -+K •·5ò,êh-Æòc¦A+\ ÑÉ9ë_t •p:_,Æi ??+§`W¬+2è7??
-       _vƒ++!' ¦Æ_ âG +¿"Ssªí_ @sFNzz_äQ¦ Ñà -G+äP3ç_»¦+#_ t¦7_Å___¿Hüç: «J_q!_+|¦
  ??+ óí¼_ +çXê_á_Æ+_++__I+÷__+[½1+r¼c_7°+_+Å _^ó-_ü-:Äx_++ § ¡ ~_±zkC§ <¤û -+++ä-h
                             ôb«{¿@_ ^¦â» /+éR_ò__3bå-_ùm5
  á*¦ñ=)rìC7á6 6é+_:a-krª}_.+B+æ+MÑ°_}b¿§-•<-t-ä??Åì¶_ ë+Ä3- ¦§_+·:+¦&pƒ7æ¦UVC"¢ -¦
                                   O7¦Ü   -
                             å÷u+J)ü _-Ç_¦¬_F•_ß,_
```

Each block starts with '??' (01h 01h).

Screen dump of random noise with random delay: -

```
º¡_jï»·§ì{3++?9Pü} ¼q??§¦++¬? •+@{»LI./_CñG$ ök·$t;ÿ+_)_4$ -}>P»d~k+[_NÅ- ¦é>W_
 ?waP§x4¦6 \Æ__ì :_ ,ùv MÄ+ ï=+T_°Kèµzj++++ƒíû
_ÑY_Oxyz4»A ö_
]¢5" m- d: ¦2û² -
ß¦ 7Jf;Xvig•c¥éJnÖS-'_R+.¦0^_ë  - ¦ß ¦+eA·m}?*_x`h¦-}H)ÄK1 -_+-59\-ÇIÄ¶iHVÄ*
¬-++_¦.5m[én[+iJ)½+p+Æb à +y(M= Mmt
L_Æ?•_&î¬¦¶_!û1¦+l-0+C@O-ìàg_ä{+ +öcfp;9;~+_
EæS¦½_+z8kzI__J;:mW- zw -pT_+n+D_-¦üP«î$+·îêë[§#+4+ô_%~hm9PZÑ¦-£+_ _Y;O+¦_
 w_µ%Ñ,öÄ -O;r¬úè8_{    -Æ6[--X6c¿-L·|e±¦[_Ö&Ñ_+ 7æ· ¶jâ_ [_¿J_níí_77-c•b+#_m @
Qn_c_+_mäyúp§ 4ÿ__Q*5-¦-¶(Kº%W+Pÿ$P_ì¦d$¥_í _?+}U bÜæ__Âl-_°0ùyqP¦ W
Hc¦»x_&Dün.ñ:äpZ ù+cdPª+i_¦%ä+++O•K~ _l- ¿¬ y+k_+æZ¦_ú`N_M -\ƒ° °-Aª
-+vZ ì+¶ï ·+"__f·   a_è?_Az úó[a+òô¶ÿ¦"Ñ~¡ô9ù Dºµìì}_ÉQ+¢'ù ü_ÿÿ_@nNX-+ñ]++½ =nP
 _ÄQ+ .ò=¦F_m++&T z¿~¦àW_1î__)e=Å ÉÅ+ñæó à° -ú|+-•+¤§ _m|-!z__-__Ö_^e¦YO +Cb}$»
æ k•-»ï+Ñ++-n3+¿_¦     ÿ¦ì¦Üû½-Üê&' 7A¦°è+ +â(+k+_A}ûJ¼ ú--d)C=~¦ÿ½¦_^
 j\_BníF)á•Ü  _¥_$+_¦ª -_ ++n¦Y_¦-^W )D+pôô×¦_S_,G-}1 [+§_w4_`æ6+1+<îz+ ö+_É_
ƒ ¥h\ ·Öª5ù2£+ezXô_ -èFæ_*#¶-I¦++ôq¢`1ò|•ä+J_+@_H^B_ z+-°E geèlAb_+P5ñ+å¦bx4eAf5
v_-²-'U6Ç S§nz ódifîT_Æ+
:ù_;bZ ¼~l\.L_+ ²_U_m_ÜU8 @-F>ÖuU?ñJè_-_¥o,Öñ`µYï_JµMRl
```

Wrong Slave

Read Coil Status 00h to 0Fh – Slave 2: -

```
Build & Transmit Query
-----------------------

Please enter test RTU Query Frame: -
Slave Address    (Hex - 1 byte)  eg   00: 2
Function Code    (Hex - 1 byte)  eg   02: 1

Read Coil Status (1) - Slave 2
Starting Address (Hex - 2 bytes) eg 0001: 0
No. of Points    (Hex - 2 bytes) eg 000A: 10
Calculated CRC (Hex):    3DF5
Total Frame Length (Dec): 8

Sending RTU Frame (HEX): -
2 1 0 0 0 10 3D F5
```

```
Wait for Response Frame
------------------------
Waiting for response (press a key to stop)

Operation Aborted - Timeout
```

Time Response

For the entire test results above, slave responded within 128ms, with an average response of 61.9ms, min response of 10ms and a max response of 114ms.

TEST RESULTS – ACCESS 4000

All test carried out for the PC version of Modbus ("Test Results – Laptop" pages 12 to 20), where repeated when implemented on Access 4000 control panel. All tests were successful, results have been summarised in the following tables: -

Function 1 – Read Coil Status

	Description	Slave 1	T	Slave 2	T	Pass / Fail
Norm	Read 00h to 0Fh	04h 00h	4	04h 00h	16	Pass
Norm	Read 02h	01h	21	01h	44	Pass
ER: 2	Read 10h	Er: 2	18	Er: 2	91	Pass
ER: 2	Read 10h to FFFFh	Er: 2	36	Er: 2	91	Pass
ER: 3	Read 00h to 10h	Er: 3	3	Er: 3	41	Pass
ER: 3	Read 0Fh to FFFFh	Er: 3	81	Er: 3	57	Pass
	Average Response Time	26.667 ms		56.67 ms		

Function 2 – Read Input Status

	Description	Slave 1	T	Slave 2	T	Pass / Fail
Norm	Read 00h to 2Fh	0 0 0 0 0 0	71	0 0 0 0 0 0	30	Pass
Norm	Read 04h "Heart Beat"	00h	43	01h	30	Pass
ER: 2	Read 30h	Er: 2	16	Er: 2	78	Pass
ER: 2	Read 30h to FFFFh	Er: 2	93	Er: 2	44	Pass
ER: 3	Read 2Fh and 30h	Er: 3	12	Er: 3	25	Pass
ER: 3	Read 00h to FFFFh	Er: 3	82	Er: 3	106	Pass
	Average Response Time	52.833 ms		52.1667 ms		

Function 3 – Read Holding Registers

	Description	Slave 1	T	Slave 2	T	Pass / Fail
Norm	Read 0Fh "bat hi set volt"	92h = 14.6 V	104	92h = 14.6 V	50	Pass
Norm	Read 00h to 03h	50h 5h 7h 3h	9	111h 7h 7h 3h	4	Pass
ER: 2	Read 21h	Er: 2	58	Er: 2	18	Pass
ER: 2	Read 21h to FFFFh	Er: 2	101	Er: 2	46	Pass
ER: 3	Read 20h and 21h	Er: 3	102	Er: 3	8	Pass
ER: 3	Read 00h to FFFFh	Er: 3	21	Er: 3	34	Pass
	Average Response Time	65.83 ms		26.667 ms		

Function 4 – Read Input Registers

	Description	Slave 1	T	Slave 2	T	Pass / Fail
Norm	Read 02h "battery volt"	7Fh = 12.7V	61	82h = 13V	45	Pass
Norm	Read 00h to 05h	E8 0 7F 0 16 0	95	E8 0 83 0 6 0	86	Pass
ER: 2	Read 20h	Er: 2	72	Er: 2	4	Pass
ER: 2	Read 20h to FFFFh	Er: 2	63	Er: 2	82	Pass
ER: 3	Read 1Fh and 20h	Er: 3	21	Er: 3	94	Pass
ER: 3	Read 00h to 1Fh (max 10)	Er: 3	27	Er: 3	5	Pass
	Average Response Time	56.5 ms		52.1667 ms		

Function 5 – Force Single Coil

	Description	Slave 1	T	Slave 2	T	Pass / Fail
Norm	Set 01h (FF00h)	Er: 4	56	Er: 4	96	Pass

Norm	Clear 03h (0000h)	Er: 4	6	Er: 4	24	Pass
ER: 2	Set 10h (FF00h)	Er: 2	54	Er: 2	3	Pass
ER: 2	Clear FFFFh (0000h)	Er: 2	5	Er: 2	4	Pass
ER: 3	Set 03h to FFFFh	Er: 3	86	Er: 3	78	Pass
ER: 3	Set 02h to 1234h	Er: 3	56	Er: 3	31	Pass
ER: 4	Clear 00h (0000h)	Er: 4	108	Er: 4	24	Pass
ER: 4	Set 00h (FF00h)	Er: 4	89	Er: 4	35	Pass
	Average Response Time	57.5 ms		36.5 ms		

Auto is off, as expected received exception error 4 – slave device failure during normal operation.

Function 6 – Preset Single Register

	Description	**Slave 1**	**T**	**Slave 2**	**T**	**Pass / Fail**
Norm	Change 00h to 321h	echo	76	echo	113	Pass
Norm	Change 01h to 54h	echo	85	echo	7	Pass
Norm	Change 03h to 5h	echo	81	echo	45	Pass
ER: 2	Change 21h to 1234h	Er: 2	48	Er: 2	61	Pass
ER: 2	Change FFFFh to FFFFh	Er: 2	96	Er: 2	27	Pass
ER: 2	Change 40h to 4321h	Er: 2	37	Er: 2	26	Pass
	Average Response Time	70.5 ms		46.5 ms		

Illegal Function

	Description	**Slave 1**	**T**	**Slave 2**	**T**	**Pass / Fail**
ER: 1	Function 7	E: 87 ER: 1	36	E: 87 ER: 1	94	Pass
ER: 1	Function 10	E: 90 ER: 1	30	E: 90 ER: 1	65	Pass
ER: 1	Function 35	E: B5 ER: 1	104	E: B5 ER: 1	27	Pass
	Average Response Time	56.6667 ms		62 ms		

Bad CRC

	Description	**Slave 1**	**T**	**Slave 2**	**T**	**Pass / Fail**
ER: 9	Function 1 00h to FFh	E: 81 ER: 9	69	E: 81 ER: 9	46	Pass
ER: 9	Function 2 00h to FFh	E: 82 ER: 9	84	E: 82 ER: 9	73	Pass
ER: 9	Function 25 00h to FFh	E: A5 ER: 9	79	E: A5 ER: 9	63	Pass
	Average Response Time	77.333 ms		60.667 ms		

Short Message

	Description	**Slave 1**	**T**	**Slave 2**	**T**	**Pass / Fail**
ER: A	RTU = 01 02 03	E: 82 ER: A	64	No Response	∞	Pass
ER: A	RTU = 02 03 06	No Response	∞	E: 83 ER: A	74	Pass
ER: A	RTU = 01	E: 80 ER: A	15	No Response	∞	Pass
ER: A	RTU = 02	No Response	∞	E: 80 ER: A	27	Pass
	Average Response Time	39.5 ms		50.5 ms		

Modbus Reliability

Description	**Access 4000 (Engine Off)**	**Access 4000 (Engine on)**	**Pass / Fail**
Random RTU_frames of Random Size	No effect	No effect	Pass
Addr (1 & 2) & Random data of Random Size	No effect	No effect	Pass
Addr(1,2) & fun(1-6) & Rand. data of Rand. size	No effect	No effect	Pass
Random Noise: No delay between bytes	No effect	No effect	Pass
Random Noise: Random delay between bytes	No effect	No effect	Pass
Random Noise: Delay = 3,5,10,15 ms	No effect	No effect	Pass

Program left running for 30 min in each mode above, the Modbus & Access 4000 control panel did not crash.

Response Time

For the entire test results above, both slaves responded within 128ms, with slave 1 having an average response of 55.9ms and slave 2 having an average response of 49.3ms, with a combined average response time of 52.6ms.

Simulator & Generator

The following set of tables summarises tests which were carried out on the Access 4000 control panel, using a hardware simulator to simulate Coils, Registers, etc… which was later repeated on a real generator: -

General Operation – Control Coils (Function 1 & 5)

Addr	Description	Access 4000 (Sim & Gen)	Modbus read (before)	Modbus read (after)	Pass / Fail
01h	Start	Engine Started	Stop=1, other=0	Start=1, other=0	Pass
02h	Stop	Engine Stopped	Start=1, other=0	Stop=1, other=0	Pass
03h	Emergency stop	Emer. shutdown	Start=1, other=0	E_st=1, other=0	Pass
04h	Reset Alarm	Alarm reset	E_st=1, other=0	E_st=1, other=0	Pass

Some alarms (e.g. “over frequency” & “over voltage”) will shut down the engine if activated, if this situation occurs, start (01h) & stop (02h) will both equal ‘1’. The alarms are cleared using Reset Alarm (04h); once cleared the engine can be restarted.

General Operation – Status/Alarms (Function 2)

Addr	Description	Produce	Results	Pass / Fail
00h	Auto switch	Manual switch at off Manual switch at auto	00h = ‘0’, 00h = ‘1’	Pass
01h	General Alarm	No alarms activated, Alarms activated	01h = ‘0’, 01h = ‘1’	Pass
04h	Heart Beat	Read a couple of times.	04h = ‘0’, 04h = ‘1’ 04h = ‘1’, 04h = ‘0’	Pass
05h	Remote Start	Engine Stop, Remote start (func 5)	05h = ‘0’, 05h = ‘1’,	Pass
06h	Output Relay 1	Check normal operation	06h = ‘1’ 06h = ‘0’	Pass
07h	Output Relay 2	Check normal operation	07h = ‘0’ 07h = ‘1’	Pass
08h	Approaching low oil pressure	Change configuration, to activate alarm	Before config. 08h = ‘0’ After config. 08h = ‘1’	Pass
09h	Approaching high engine temperature	Change configuration, to activate alarm	Before config. 09h = ‘0’ After config. 09h = ‘1’	Pass
0Ah	Low coolant temp	Not installed	0Ah = ‘0’	?
0Bh	Low battery voltage	Change configuration, to activate alarm	Before config. 0Bh = ‘0’ After config. 0Bh = ‘1’	Pass
0Ch	High battery voltage	Change configuration, to activate alarm	Before config. 0Ch = ‘0’ After config. 0Ch = ‘1’	Pass
0Dh	Battery charger fail	Change configuration, to activate alarm	Before config. 0Dh = ‘0’ After config. 0Dh = ‘1’	Pass
0Eh	Not in auto	Change configuration, to activate alarm	Before config. 0Eh = ‘0’ After config. 0Eh = ‘1’	Pass
0Fh	Fail to start	Tamper with generator, so it fails to start	Before Tamp. 0Fh = ‘0’ After Tamp. 0Fh = ‘1’	Pass

10h	Under frequency	Change configuration, to activate alarm	Before config. 10h = '0' After config. 10h = '1'	Pass
11h	Over frequency	Change configuration, to activate alarm	Before config. 11h = '0' After config. 11h = '1'	Pass
12h	Under voltage	Change configuration, to activate alarm	Before config. 12h = '0' After config. 12h = '1'	Pass
13h	Over voltage	Change configuration, to activate alarm	Before config. 13h = '0' After config. 13h = '1'	Pass
14h	Over current	Not installed	14h = '0'	?
15h	Over speed	Change configuration, to activate alarm	Before config. 15h = '0' After config. 15h = '1'	Pass
16h	Emergency stop	Hit large red button, or function 5 address 3	Before Estop 16h = '0' After Estop 16h = '1'	Pass
17h	High engine temperature	Hotwire, to cause fault to occur	Before alarm 17h = '0' After alarm 17h = '1'	Pass
18h	Low oil pressure	Hotwire, to cause fault to occur	Before alarm 18h = '0' After alarm 18h = '1'	Pass
19h	Spare fault 1	Hotwire, to cause fault to occur	Before alarm 19h = '0' After alarm 19h = '1'	Pass
1Ah	Spare fault 2	Hotwire, to cause fault to occur	Before alarm 1Ah = '0' After alarm 1Ah = '1'	Pass
1Bh	Spare fault 3	Hotwire, to cause fault to occur	Before alarm 1Bh = '0' After alarm 1Bh = '1'	Pass
1Ch	Spare fault 4	Disabled	1Ch = '0'	Pass

General Operation – Configuration register (Function 3)

Addr	Description	Modbus Read (Fun3)	Modbus Write (Fun6)	Modbus Read (Fun3)	Pass / Fail
00h	Crank cut-out RPM	12Ch = 300	23Dh	23Dh	Pass
01h	Crank duration time	07h = 7	0Bh	0Bh	Pass
02h	Crank delay	07h = 7	1Dh	1Dh	Pass
03h	Crank repeats	03h = 3	04h	04h	Pass
04h	Voltage high setpoint	1C9 = 457	2D8h	2D8h	Pass
05h	Voltage high time delay	05h = 5	12h	12h	Pass
06h	Voltage low setpoint	E0h = 224	DAh	DAh	Pass
07h	Voltage low time delay	05h = 5	0Ch	0Ch	Pass
08h	Frequency high setpoint	37h = 55	3Fh	3Fh	Pass
09h	Frequency high delay	05h = 5	03h	03h	Pass
0Ah	Frequency low setpoint	2Dh = 45	31h	31h	Pass
0Bh	Frequency low time delay	05h = 5	1Fh	1Fh	Pass
0Ch	Overspeed setpoint	79Eh = 1950	876h	876h	Pass
0Dh	Overcurrent setpoint	4AFh = 1199	4BCh	4BCh	Pass
0Eh	Overcurrent time delay	00h = 0	0Ah	0Ah	Pass
0Fh	Battery voltage hi setpoint	92h = 146	8Fh	8Eh	Fail
10h	Battery voltage hi delay	05h = 5	12h	12h	Pass
11h	Battery voltage lo setpoint	5Ah = 90	5Fh	5Fh	Pass
12h	Battery voltage lo delay	10h = 10	15h	15h	Pass
13h	Batt. charger fail setpoint	5Ah = 90	61h	60h	Fail
14h	Battery charger fail delay	0Ah = 10	22h	22h	Pass
15h	Appr low oil pres. setpoint	1Eh = 30	2Bh	2Ah	Fail
16h	Appr hi eng temp setpoint	5Fh = 95	43h	43h	Pass
17h	Low coolant temp setpoint	1Eh = 30	18h	18h	Pass
18h	Fault protection delay	0Ah = 10	32h	32h	Pass
19h	Remote start delay ON	00h = 0	05h	05h	Pass
1Ah	Remote start delay OFF	00h = 0	07h	07h	Pass

General Operation – Monitoring register (Function 4)

Addr	Description	Access 4000 LCD	Modbus Read	Access 4000 LCD	Modbus Read	Pass / Fail
00h	Water temperature	54°C	36h = 54	66°C	42h = 66	Pass
01h	Oil pressure	5.5 B	36h = 5.4	6.9 B	45h = 6.9	Pass
02h	Battery voltage	11.9 V	77h = 11.9	----------	----------	Pass
03h	Hours run	0 Hours	00h = 0	2 Hours	02h = 2	Pass
04h	Starts	31	1Fh = 31	88	58h = 88	Pass
05h	RPM	1524 RPM	3B82h = 1523.4	----------	----------	Pass
06h	Phase A voltage	124 V	7Ch = 124	242 V	F2h = 242	Pass
07h	Phase B voltage	123 V	7Bh = 123	239 V	EFh = 239	Pass
08h	Phase C voltage	123 V	7Bh = 123	245 V	F5h = 245	Pass
09h	Line AB voltage	0 V	04h = 0.4	400 V	F95h = 399	Pass
0Ah	Line BC voltage	0 V	06h = 0.6	401 V	FAD = 401	Pass
0Bh	Line CA voltage	0 V	04h = 0.4	402 V	FB6 = 402	Pass
0Ch	Phase A current	25 A	FAh = 25 A	20.3 A	CBh = 20.3	Pass
0Dh	Phase B current	25.3 A	FDh = 25.3	20.5 A	CDh = 20.5	Pass
0Eh	Phase C current	24.9 A	F9h = 24.9	19.5 A	C3h = 19.5	Pass
0Fh	Frequency	00 Hz	00h = 0	50 Hz	1F6 = 50.2	Pass
10h	Power factor	0	00h = 0	1	0Ah = 1	Pass
11h	Total kW	0	00h = 0	12	0Ch = 12	Pass
12h	Total KVA	0	00h = 0	12	0Ch = 12	Pass
13h	Total kVAr	0	00h = 0	1	00h = 0	Fail
15h	Total kWh, hi word	0	00h = 0	----------	----------	Pass
16h	Total kWh, lo word	29	1Dh = 29	----------	----------	Pass
17h	Fuel level	Disabled	00h = 0	----------	----------	Pass

General Operation – Configuration register (Function 6)

Addr	Description	Range	>max	<min	max	min	Pass / Fail
00h	Crank cut-out RPM	1-999	Er: 4	Er: 4	echo	echo	Pass
01h	Crank duration time	1-99	Er: 4	Er: 4	echo	echo	Pass
02h	Crank delay	1-99	Er: 4	Er: 4	echo	echo	Pass
03h	Crank repeats	1-10	Er: 4	Er: 4	echo	echo	Pass
04h	Voltage high setpoint	100-15500	Er: 4	Er: 4	echo	echo	Pass
05h	Voltage high time delay	0-99	Er: 4	Er: 4	echo	echo	Pass
06h	Voltage low setpoint	60 - 15500	Er: 4	Er: 4	echo	echo	Pass
07h	Voltage low time delay	0-99	Er: 4	Er: 4	echo	echo	Pass
08h	Frequency high setpoint	50-70	Er: 4	Er: 4	echo	echo	Pass
09h	Frequency high delay	0-99	Er: 4	Er: 4	echo	echo	Pass
0Ah	Frequency low setpoint	40-60	Er: 4	Er: 4	echo	echo	Pass
0Bh	Frequency low time delay	0-99	Er: 4	Er: 4	echo	echo	Pass
0Ch	Overspeed setpoint	1000-4200	Er: 4	Er: 4	echo	echo	Pass
0Dh	Overcurrent setpoint	1-9999	Er: 4	Er: 4	echo	echo	Pass
0Eh	Overcurrent time delay	0-99	Er: 4	Er: 4	echo	echo	Pass
0Fh	Battery voltage hi setpoint	120-360	Er: 4	Er: 4	echo	echo	Pass
10h	Battery voltage hi delay	0-99	Er: 4	Er: 4	echo	echo	Pass
11h	Battery voltage lo setpoint	90-240	Er: 4	Er: 4	echo	echo	Pass
12h	Battery voltage lo delay	0-99	Er: 4	Er: 4	echo	echo	Pass
13h	Batt. charger fail setpoint	90-280	Er: 4	Er: 4	echo	echo	Pass
14h	Battery charger fail delay	0-99	Er: 4	Er: 4	echo	echo	Pass
15h	Appr low oil pres. setpoint	10-90	Er: 4	Er: 4	echo	echo	Pass
16h	Appr hi eng temp setpoint	40-105	Er: 4	Er: 4	echo	echo	Pass
17h	Low coolant temp setpoint	20-99	Er: 4	Er: 4	echo	echo	Pass
18h	Fault protection delay	0-99	Er: 4	Er: 4	echo	echo	Pass
19h	Remote start delay ON	0-99	Er: 4	Er: 4	echo	echo	Pass

1Ah	Remote start delay OFF	0-99	Er: 4	Er: 4	echo	echo	Pass

Known Minor Bug – Configuration OFh – Battery Voltage high Setpoint

Write (Func 6)	Read (Func 3)	P / F	Write (Func 6)	Read (Func 3)	P / F
92h	92h	P	97h	96h	F
93h	92h	F	98h	97h	F
94h	93h	F	99h	98h	F
95h	94h	F	9Ah	99h	F
96h	96h	P	9Bh	9Bh	P

Changing battery voltage high setpoint may result in an error of 0.1V, this error occurs during the process which converts integer (X10) to floating point number and back again.

Known Minor Bug – Configuration 11h – Battery Voltage low Setpoint

Write (Func 6)	Read (Func 3)	P / F	Write (Func 6)	Read (Func 3)	P / F
5Ah	5Ah	P	5Fh	5Fh	P
5Bh	5Ah	F	60h	5Fh	F
5Ch	5Bh	F	61h	60h	F
5Dh	5Ch	F	62h	61h	F
5Eh	5Dh	F	63h	62h	F

Changing battery voltage low setpoint may result in an error of 0.1V, this error occurs during the process which converts integer (X10) to floating point number and back again.

Known Minor Bug – Configuration 13h – Battery Charger Fail Setpoint

Write (Func 6)	Read (Func 3)	P / F	Write (Func 6)	Read (Func 3)	P / F
60h	5Fh	F	65h	64h	F
61h	60h	F	66h	65h	F
62h	61h	F	67h	66h	F
63h	62h	F	68h	67h	F
64h	64h	P	69h	69h	P

Changing battery charger fail setpoint may result in an error of 0.1V, this error occurs during the process which converts integer (X10) to floating point number and back again.

Known Minor Bug – Configuration 15h – Appr low oil pressure setpoint

Write (Func 6)	Read (Func 3)	P / F	Write (Func 6)	Read (Func 3)	P / F
2Bh	2Ah	F	30h	2Fh	F
2Ch	2Bh	F	31h	30h	F
2Dh	2Dh	P	32h	32h	P
2Eh	2Dh	F	33h	32h	F
2Fh	2Eh	F	34h	33h	F

Changing Appr low oil pressure setpoint may result in an error of 0.1, this error occurs during the process, which converts integer (X10) to floating point number and back again.