Draft

Engineering specification

Modbus on

Telasia : VS18-RS485

Table of contents

1. Revision information

Table 1: Document revision history

Rev.	Date:	Author	Status
P01.00	Mar 22, 2018	RC	Specification revision is based on SenseAir CO2 engine, model S8 by PZ, JE & LN. New version addresses by SenseAir S8 family of sensors.

2. General

Modbus is a simple, open protocol for both PLC and sensors. Details on Modbus can be found on www.modbus.org.

This specification is based on the specification of Modbus implementation on VS18-RS485 CO2 sensors and aims to support backwards compatibility with them. The differences between the Modbus specification [1] and the default implementation in the sensor are listed in this document.

General overview of protocol and implementation in the sensor

Master – slave:
Only master can initiate a transaction. The sensor is a slave and will never initiate communication. The host system initiates transactions to read CO2 value from the corresponding register. The host system shall also check status of the sensor periodically (e.g. every 2 seconds) in order to determine if it is running without faults detected.

Packet identification:
Any message (packet) starts with a silent interval of 3.5 characters. Another silent interval of 3.5 characters marks message end. Silence interval between characters in the message needs to be kept less than 1.5 characters.
Both intervals are from the end of Stop-bit of previous byte to the beginning of the Start-bit of the next byte.

Packet length:
According to the Modbus specification [1], the packet length shall be maximum 255 bytes including address and CRC. We cannot support so large packets. Maximum length of packet (serial line PDU including address byte and 2 bytes CRC) supported by the sensor is **39 bytes**. **Packets of larger size are rejected without any answer from sensor, even if the packet was addressed to the sensor.**

Modbus data model:
There are 4 primary data tables (addressable registers), which may overlay:

- Discrete Input (read only bit).
- Coil (read / write bit).
- Input register (read only 16 bit word, interpretation is up to application).
- Holding register (read / write 16 bit word).

Note: The sensor does not support bitwise access of registers.

Exception responses:
Slave will send answer to the master only in the case of valid message structure. Nevertheless, it can send exception response because of detection of:

- Invalid function code.
- Invalid data address (requested register doesn't exist in given device).
- Invalid data.
- Error in execution of requested function.

Modbus diagnostic counters:
T.B.D.

3. Byte transmission.

RTU transmission mode is the only mode supported by the sensor.

3.1. Byte format:

The format for each byte in RTU mode differs between the sensor default configuration and the description on page 12 of MODBUS over serial line specification [2].

Table 2: Byte format differences

	MODBUS over serial line specification [2]		**Sensor default configuration**
Coding system	8-bit binary		8-bit binary
Bits per byte:	1 start bit		1 start bit
Data bits	8 data bits, least significant bit first		8 data bits, least significant bit first
	1 bit for even parity	No parity bit	NO parity bit
	1 stop bit	2 stop bits	1 stop bit for receiving 2 stop bits at transmission

Implementation of 1 stop bit on receive and 2 stop bits at transmit provides compatibility with masters using both 1 and 2 stop bits.

3.2. Baud rate (data signaling rate)

9600 bps and 19200 bps are required baud rates and required default baud rate according to MODBUS over serial line specification [2], page 20, is 19200 bps.

VS18-RS485 **supports 9600 baud rate only.**

3.3. Physical layer:

The sensor provides RS485 logical levels **A** and **B** lines for serial transmission.

4. Modbus registers on sensor.

The Modbus registers are mapped in memory, both RAM and EEPROM of the sensor. Mapping is interpreted by sensor firmware at command reception.

Presently, the following restrictive decisions are made:

1. Read only and read / write registers are not allowed to overlay.
2. Bit addressable items (i.e. Coils and Discrete inputs) will not be implemented.
3. Only write single register functional codes are implemented. Multiple write functional codes are not planned for implementation.
4. The total number of registers should be limited. Present decision is to limit number of input registers to 32 and number of holding registers to 32.
   Note: the limited buffer space of the sensor puts a limit on how many registers that can be read in one command, currently 8 registers.
5. Larger amount of data should be transferred as file. It is not implemented at the current stage of development.

Maps of registers (All registers are 16 bit word) are summarized in Table 3 and Table 4. Associated number is Modbus register number: Register address is calculated as (register number -1)

Table 3 : Input Registers

IR#	#	Name																
IR1	0	MeterStatus	DI 16	DI 15	DI 14	DI 13	DI 12	DI 11	DI 10	DI 9	DI 8	DI 7	DI 6	DI 5	DI 4	DI 3	DI 2	DI 1
			DI 1 - Fatal error DI 2 - Offset regulation error DI 3 - Algorithm Error DI 4 - Output Error DI 5 - Self diagnostics error DI 6 - Out Of Range DI 7 - Memory error DI 8 - Reserved [1] DI 9 - Reserved [1] DI 10 - Reserved [1] DI 11 - Reserved [1] DI 12 - Reserved [1] DI 13 - Reserved [1] DI 14 - Reserved [1] DI 15 - Reserved [1] DI 16 - Reserved [1]															
IR2	1	AlarmStatus	DI 16	DI 15	DI 14	DI 13	DI 12	DI 11	DI 10	DI 9	DI 8	DI 7	DI 6	DI 5	DI 4	DI 3	DI 2	DI 1
			DI 17 - Reserved [1] DI 18 - Reserved [1] DI 19 - Reserved [1] DI 20 - Reserved [1] DI 21 - Reserved [1] DI 22 - Reserved [1] DI 23 - Reserved [1] DI 24 - Reserved [1] DI 25 - Reserved [1] DI 26 - Reserved [1] DI 27 - Reserved [1] DI 28 - Reserved [1] DI 29 - Reserved [1]															

<table>
<tr><td></td><td></td><td></td><td colspan="16">DI 30 - Reserved [1]
DI 31 - Reserved [1]
DI 32 - Reserved [1]</td></tr>
<tr><td>IR3</td><td>2</td><td>Output Status</td><td>DI 16</td><td>DI 15</td><td>DI 14</td><td>DI 13</td><td>DI 12</td><td>DI 11</td><td>DI 10</td><td>DI 9</td><td>DI 8</td><td>DI 7</td><td>DI 6</td><td>DI 5</td><td>DI 4</td><td>DI 3</td><td>DI 2</td><td>DI 1</td></tr>
<tr><td></td><td></td><td></td><td colspan="16">DI 33 - Alarm Output status (inverted due to Open Collector)
DI 34 - PWM Output status (1 means full output)
DI 35 - Reserved [1]
DI 36 - Reserved [1]
DI 37 - Reserved [1]
DI 38 - Reserved [1]
DI 39 - Reserved [1]
DI 40 - Reserved [1]
DI 41 - Reserved [1]
DI 42 - Reserved [1]
DI 43 - Reserved [1]
DI 44 - Reserved [1]
DI 45 - Reserved [1]
DI 46 - Reserved [1]
DI 47 - Reserved [1]
DI 48 - Reserved [1]</td></tr>
<tr><td>IR4</td><td>3</td><td>Space CO2</td><td colspan="16">Space CO2</td></tr>
<tr><td>IR5</td><td>4</td><td></td><td colspan="16">Reserved for Space Temp, returns "illegal data address" exception</td></tr>
<tr><td>IR6</td><td>5</td><td></td><td colspan="16">Reserved, returns "illegal data address" exception</td></tr>
<tr><td>IR7</td><td>6</td><td></td><td colspan="16">Reserved, returns "illegal data address" exception</td></tr>
<tr><td>IR8</td><td>7</td><td></td><td colspan="16">Reserved, returns "illegal data address" exception</td></tr>
<tr><td>IR9</td><td>8</td><td></td><td colspan="16">Reserved, returns "illegal data address" exception</td></tr>
<tr><td>IR10</td><td>9</td><td></td><td colspan="16">Reserved, returns "illegal data address" exception</td></tr>
<tr><td>IR11</td><td>10</td><td></td><td colspan="16">Reserved, returns "illegal data address" exception</td></tr>
<tr><td>IR12</td><td>11</td><td></td><td colspan="16">Reserved, returns "illegal data address" exception</td></tr>
<tr><td>IR13</td><td>12</td><td></td><td colspan="16">Reserved, returns "illegal data address" exception</td></tr>
<tr><td>IR14</td><td>13</td><td></td><td colspan="16">Reserved, returns "illegal data address" exception</td></tr>
<tr><td>IR15</td><td>14</td><td></td><td colspan="16">Reserved, returns "illegal data address" exception</td></tr>
<tr><td>IR16</td><td>15</td><td></td><td colspan="16">Reserved, returns "illegal data address" exception</td></tr>
<tr><td>IR17</td><td>16</td><td></td><td colspan="16">Reserved, returns "illegal data address" exception</td></tr>
<tr><td>IR18</td><td>17</td><td></td><td colspan="16">Reserved, returns "illegal data address" exception</td></tr>
<tr><td>IR19</td><td>18</td><td></td><td colspan="16">Reserved, returns "illegal data address" exception</td></tr>
<tr><td>IR20</td><td>19</td><td></td><td colspan="16">Reserved, returns "illegal data address" exception</td></tr>
<tr><td>IR21</td><td>20</td><td></td><td colspan="16">Reserved, returns "illegal data address" exception</td></tr>
<tr><td>IR22</td><td>21</td><td>PWM Output[2]</td><td colspan="16">PWM Output[2]</td></tr>
<tr><td>IR23</td><td>22</td><td></td><td colspan="16">Reserved, returns "illegal data address" exception</td></tr>
</table>

IR24	23		Reserved, returns "illegal data address" exception
IR25	24		Reserved, returns "illegal data address" exception
IR26	25		Sensor Type ID High[3]
IR27	26		Sensor Type ID Low[3]
IR28	27		Memory Map version
IR29	28		FW version Main.Sub[4]
IR30	29		Sensor ID High[5]
IR31	30		Sensor ID Low[5]
IR32	31		Reserved, returns "illegal data address" exception

[1] – Reserved DIs return 0.
[2] – 0x3FFF represents 100% output. Refer to sensor model's specification for voltage at 100% output.
[3] – IR26 low byte + IR27 contains Sensor Type ID 3-bytes value.
[4] – IR29 high byte is FW Main revision, low byte – FW Sub revision.
[5] – IR30 + IR31 – 4-bytes Sensor's Serial Number.

Table 4: Holding Registers

HR#	**#**	**Name**																
HR1	0	Acknowledgement register	DI 16	DI 15	DI 14	DI 13	DI 12	DI 11	DI 10	DI 9	DI 8	DI 7	DI 6	DI 5	DI 4	DI 3	DI 2	DI 1
			CI 1 - Reserved [6] CI 2 - Reserved [6] CI 3 - Reserved [6] CI 4 - Reserved [6] CI 5 - Reserved [6] CI 6 - CO2 background calibration has been performed CI 7 - CO2 nitrogen calibration has been performed CI 8 - Reserved [6] CI 9 - Reserved [6] CI 10 - Reserved [6] CI 11 - Reserved [6] CI 12 - Reserved [6] CI 13 - Reserved [6] CI 14 - Reserved [6] CI 15 - Reserved [6] CI 16 - Reserved [6]															
HR2	1	Special Command Register[7]	Command						Parameter									
			0x7C						0x6 - CO2 background calibration 0x7 - CO2 zero calibration									
HR3	2		Reserved, returns "illegal data address" exception															
HR4	3		Reserved, returns "illegal data address" exception															

HR5	4		Reserved, returns "illegal data address" exception
HR6	5		Reserved, returns "illegal data address" exception
HR7	6		Reserved, returns "illegal data address" exception
HR8	7		Reserved, returns "illegal data address" exception
HR9	8		Reserved, returns "illegal data address" exception
HR10	9		Reserved, returns "illegal data address" exception
HR11	10		Reserved, returns "illegal data address" exception
HR12	11		Reserved, returns "illegal data address" exception
HR13	12		Reserved, returns "illegal data address" exception
HR14	13		Reserved, returns "illegal data address" exception
HR15	14		Reserved, returns "illegal data address" exception
HR16	15		Reserved, returns "illegal data address" exception
HR17	16		Reserved, returns "illegal data address" exception
HR18	17		Reserved, returns "illegal data address" exception
HR19	18		Reserved, returns "illegal data address" exception
HR20	19		Reserved, returns "illegal data address" exception
HR21	20		Reserved, returns "illegal data address" exception
HR22	21		Reserved, returns "illegal data address" exception
HR23	22		Reserved, returns "illegal data address" exception
HR24	23		Reserved, returns "illegal data address" exception
HR25	24		Reserved, returns "illegal data address" exception
HR26	25		Reserved, returns "illegal data address" exception
HR27	26		Reserved, returns "illegal data address" exception
HR28	27		Reserved, returns "illegal data address" exception
HR29	28		Reserved, returns "illegal data address" exception
HR30	29		Reserved, returns "illegal data address" exception
HR31	30		Reserved, returns "illegal data address" exception
HR32	31	ABC Period	ABC Period in hours[8]

[6] – Reserved CIs return 0.
[7] – Special Command Register is write-only.
[8] – Writing to ABC_Period zero value suspends ABC function. ABC samples and ABC time counting do not losed. To resume ABC function with prior ABC samples and ABC time write to ABC_Period non-zero value.

5. Serial line frame and addressing.

5.1. Serial line frame

Modbus over serial line specification [2] distinguishes Modbus Protocol PDU and Modbus serial line PDU in the following way (RTU mode only is under consideration):

Modbus serial line PDU

Address field (1 byte)	Function Code	Data	CRC (Hi and Low)
	Modbus PDU		

5.2. Addressing rules

Addressing rules are summarized in the table:

Address	Modbus over serial line V1.0	*SenseAir®* *S8* Sensor
0	Broadcast address	No broadcast commands currently implemented
From 1 to 247	Slave individual address	Slave individual address
From 248 to 253	Reserved	Nothing[1)]
254	Reserved	“Any sensor”
255	Reserved	Nothing[1)]

Notes:

1. “Nothing” means that sensor doesn’t recognise Modbus serial line PDUs with this address as addressed to the sensor. Sensor does not respond.
2. “Any sensor” means that any sensor with any slave individual address will recognise serial line PDUs with address 254 as addressed to them. They will respond. So that this address is forproduction / test purposes only. It must not be used in the installed network.
   This is a violation against the Modbus specification [1].

5.3. Broadcast address

Modbus specification [1] requires execution of all write commands in the broadcast address mode.

Current status for the sensor:
Only one broadcast command, reset sensor, is planned but not implemented yet.

Draft

6. Bus timing.

Parameter	Min	Typ	Max	Units
Response time-out			180	msec
Turnaround delay			TBD	msec

"Response time-out" is defined to prevent master (host system) from staying in "Waiting for reply" state indefinitely. Refer to page 9 of MODBUS over serial line specification [2].

For slave device "Response time-out" represents maximum time allowed to take by "processing of required action", "formatting normal reply" and "normal reply sent" alternatively by "formatting error reply" and "error reply sent", refer to the slave state diagram on page 10 of the document mentioned above.

"Turn-around delay" is defined in MODBUS over serial line specification [2] as delay respected by Master after broadcast command in order to allow any slave to process the current request before sending a new one.

7. Function codes descriptions (PUBLIC).

Description of exception responses.

If the PDU of the received command has wrong format:

No Response PDU, (sensor doesn't respond)

If Function Code isn't equal to any implemented function code:

Exception Response PDU,

Function code	1 byte	Function Code + 0x80
Exception code = ***Illegal Function***	1 byte	0x01

If one or more of addressed Registers is not assigned (register is reserved or Quantity of registers is larger than maximum number of supported registers):

Exception Response PDU,

Function code	1 byte	Function Code + 0x80
Exception code = ***Illegal Data Address***	1 byte	0x02

7.1. 01 (0x01) Read Coils (one bit read / write registers).

Not implemented.

7.2. 02 (0x02) Read Discrete Inputs (one bit read only registers).

Not implemented.

7.3. 03 (0x03) Read Holding Registers (16 bits read / write registers).

Refer to Modbus specification [1].
Quantity of Registers is limited to 32.

Address of Modbus Holding Registers for 1-command reading is limited in range 0x0000..0x001F.

Request PDU

Function code	1 byte	**0x03**
Starting Address Hi	1 byte	Address Hi
Starting Address Lo	1 byte	Address Lo
Quantity of Registers Hi	1 byte	Quantity Hi
Quantity of Registers Lo	1 byte	Quantity Lo

Response PDU

Function code	1 byte	**0x03**
Byte Count	1 byte	2 x N*
Register Value	N* x 2 bytes	

* N = Quantity of Registers

If Address>0x001F or (Address + Quantity)>0x0020:

Exception Response PDU.

Function code	1 byte	**0x83**
Exception code = ***Illegal Data Address***	1 byte	0x02

If Quantity=0 or Quantity>8:

Exception Response PDU.

Function code	1 byte	**0x83**
Exception code = ***Illegal Data Value***	1 byte	0x03

7.4. 04 (0x04) Read Input Registers (16 bits read only registers).

Refer to Modbus specification [1].
Quantity of Registers is limited to 32.

Address of Modbus Input Registers for 1-command reading is limited in range 0x0000..0x001F.

Request PDU

Function code	1 byte	**0x04**
Starting Address Hi	1 byte	Address Hi
Starting Address Lo	1 byte	Address Lo
Quantity of Registers Hi	1 byte	Quantity Hi
Quantity of Registers Lo	1 byte	Quantity Lo

Response PDU

Function code	1 byte	**0x04**
Byte Count	1 byte	2 x N*
Register Value	N* x 2 bytes	

* N = Quantity of Registers

If Address>0x001F or (Address + Quantity)>0x0020:

Exception Response PDU,

Function code	1 byte	**0x84**
Exception code = ***Illegal Data Address***	1 byte	0x02

If Quantity=0 or Quantity>8:

Exception Response PDU,

Function code	1 byte	**0x84**
Exception code = ***Illegal Data Value***	1 byte	0x03

7.5. 05 (0x05) Write Single Coil (one bit read / write register).

Not implemented.

7.6. 06 (0x06) Write Single Register (16 bits read / write register).

Refer to Modbus specification [1].

Address of Modbus Holding Registers for 1-command reading/writing is limited in range 0x0000..0x001F.

Request PDU

Function code	1 byte	**0x06**
Starting Address Hi	1 byte	Address Hi
Starting Address Lo	1 byte	Address Lo
Register Value Hi	1 byte	Value Hi
Register Value Lo	1 byte	Value Lo

Response PDU (is an echo of the Request)

Function code	1 byte	**0x06**
Starting Address Hi	1 byte	Address Hi
Starting Address Lo	1 byte	Address Lo
Register Value Hi	1 byte	Value Hi
Register Value Lo	1 byte	Value Lo

If Address>0x001F:

Exception Response PDU,

Function code	1 byte	**0x86**
Exception code = ***Illegal Data Address***	1 byte	0x02

7.7. 15 (0x0F) Write Multiple Coils (one bit read / write registers).

Not implemented.

7.8. 16 (0x10) Write Multiple Registers (16 bits read / write register).

Not implemented.

7.9. 20 (0x14) Read File record.

Not implemented.

7.10. 21 (0x15) Write File record.

Not implemented.

7.11. 22 (0x16) Mask Write Register (16 bits read / write register).

Not implemented.

7.12. 23 (0x17) Read / Write Multiple Registers (16 bits read / write register).

Not implemented.

43 / 14 (0x2B / 0x0E) Read Device Identification.

Not implemented.

Sensor Type ID, sensor Serial Number can be read through Input Registers. (see Table 5 “Input Registers compatibility” for details).

8. References

- [1] MODBUS Application Protocol Specification V1.1b
- [2] MODBUS over serial line specification and implementation guide V1.02

9. Appendix A: Application examples

Prerequisites for the application examples:

1. A single slave (sensor) is assumed (address “any sensor” is used).
2. Values in <..> are hexadecimal.

CO2 read sequence:

The sensor is addressed as “Any address” (0xFE).
We read CO2 value from IR4 using “Read input registers” (function code 04). Hence, Starting address will be 0x0003 (register number-1) and Quantity of registers 0x0001. CRC calculated to 0xC5D5 is sent with low byte first.
We assume in this example that by sensor measured CO2 value is 400ppm*.

Sensor replies with CO2 reading 400ppm (400 ppm = 0x190 hexadecimal).

Master Transmit:
<FE> <04> <00> <03> <00> <01> <D5> <C5>

Slave Reply:
<FE> <04> <02> <01> <90> <AC> <D8>

** Note that some future models in the VS18-RS485 family of sensors may have a different scale factor on the ppm reading. The reading on these models is divided by 10 (i.e. when ambient CO2 level is 400ppm the sensor will transmit the number 40). In this example the reply from one of these models would be 40 (= 0x28 hexadecimal).*

Sensor status read sequence:

The sensor is addressed as “Any address” (0xFE).
We read status from IR1 using “Read input registers” (function code 04). Hence, Starting address will be 0x0000 (register number-1) and Quantity of registers 0x0001. CRC calculated to 0xC525 is sent with low byte first.

Sensor replies with status 0.

Master Transmit:
<FE> <04> <00> <00> <00> <01> <25> <C5>

Slave Reply:
<FE> <04> <02> <00> <00> <AD> <24>

Sensor status and CO2 read sequence:

The sensor is addressed as “Any address” (0xFE).
Here we read both status and CO2 in one command by reading IR 1 to 4 using “Read input registers” (function code 04). Hence, Starting address will be 0x0000 (register number-1) and Quantity of registers 0x0004. CRC calculated to 0xC6E5 is sent with low byte first.
We assume in this example that by sensor measured CO2 value is 400ppm*.

Sensor replies with status=0 and CO2 value 400ppm (0x190 hexadecimal).

Master Transmit:
<FE> <04> <00> <00> <00> <04> <E5> <C6>

Slave Reply:
<FE> <04> <08> <00> <00> <00> <00> <00> <00> <01> <90> <16> <E6>
| Status | | CO2 |

***** Note that some future models in the VS18-RS485 family of sensors may have a different scale factor on the ppm reading. The reading on these models is divided by 10 (i.e. when ambient CO2 level is 400ppm the sensor will transmit the number 40). In this example the reply from one of these models would be 40 (= 0x28 hexadecimal).

Background calibration sequence:

The sensor is addressed as “Any address” (0xFE).

1. Clear acknowledgement register by writing 0 to HR1. Starting address is 0x0000 and Register value 0x0000. CRC calculated as 0xC59D is sent with low byte first.

Master Transmit:
<FE> <06> <00> <00> <00> <00> <9D> <C5>

Slave Reply:
<FE> <06> <00> <00> <00> <00> <9D> <C5>

2. Write command to start background calibration. Parameter for background calibration is 6 and for nitrogen calibration is 7. We write command 0x7C with parameter 0x06 to HR2.
   Starting address is 0x0001 and Register value 0x7C06. CRC calculated as 0xC76C is sent with low byte first.

Master Transmit:
<FE> <06> <00> <01> <7C> <06> <6C> <C7>

Slave Reply:
<FE> <06> <00> <01> <7C> <06> <6C> <C7>

3. Wait at least 2 seconds for standard sensor with 2 sec lamp cycle.
4.
5. Read acknowledgement register. We use function 3 “Read Holding register” to read HR1. Starting address is 0x0000 and Quantity of registers is 0x0001. CRC calculated as 0x0590 is sent with low byte first.

Master Transmit:
<FE> <03> <00> <00> <00> <01> <90> <05>

Slave Reply:
<FE> <03> <02> <00> <20> <AD> <88>

Check that bit 5 (CI6) is 1. It is an acknowledgement of that the sensor has performed the calibration operation. The sensor may skip calibration; an example of a reason for this could be unstable signal due to changing CO2 concentration at the moment of the calibration request.

Read ABC parameter, ABC_PERIOD:

One of the ABC parameters, ABC_PERIOD, is available for modification as it is mapped as a holding register. This example shows how to read ABC_PERIOD by accessing HR32.

The sensor is addressed as “Any address” (0xFE).
Read current setting of ABC_PERIOD by reading HR32. We use function code 03 “Read Holding registers”. Starting address is 0x001f and Quantity of Registers 0x0001. CRC calculated as 0xC3A1 is sent with low byte first.

Master Transmit:
<FE> <03> <00> <1F> <00> <01> <A1> <C3>

Slave Reply:
<FE> <03> <02> <00> <B4> <AC> <27>

In the slave reply we can see:

Address = 0xFE	
Function code = 0x03	
Byte count = 0x02	- We read 2 bytes (1 register of 16 bits)
Register value = 0x00B4	- 0xB4 hexadecimal = 180 decimal; 180 hours / 24 equals 7,5 days.
CRC = 0x27AC	- CRC sent with low byte first

Disable ABC function:

We can disable the ABC function by setting ABC_PERIOD to 0.

The sensor is addressed as “Any address” (0xFE).

We use function code 06 “Write Single Register” to write to HR32. Register address is 0x001f, register value 0x0000. CRC calculated as 0x03AC is sent with low byte first.

Master transmit:
<FE> <06> <00> <1F> <00> <00> <AC> <03>

Slave reply:
<FE> <06> <00> <1F> <00> <00> <AC> <03>

We can see the reply which is an echo of the transmitted sequence.

Enable ABC function:

We can enable the ABC function by setting ABC_PERIOD to some value other than 0. In this example we set it to 7.5 days.

The sensor is addressed as “Any address” (0xFE).

We use function code 06 “Write Single Register” to write to HR32. Register address is 0x001f, register value 0x00B4 (7.5 days * 24 hours = 180; 180 in hexadecimal format is 0xB4). CRC calculated as 0x74AC is sent with low byte first.

Master transmit:

<FE> <06> <00 <1F> <00> <B4> <AC> <74>

Slave reply:
<FE> <06> <00> <1F> <00> <B4> <AC> <74>

We can see the reply which is an echo of the transmitted sequence.

RS485 Network System of VS18-RS485 sensors

Unless otherwise specified or indicated, all VS18-RS485 has a default network address (or ID) at 104 (0x68).

Changing network address

In order to change network sensor address, the sequence should be;

1. Write new address
   Master send: <68> <41> <00> <20> <01> <new address> <crc> <crc>
   Sensor reply: <68> <41> <crc> <crc>

2. Write "Copy to EEPROM" (permanently store ID into EEPROM)
   Master send: <new address> <41> <00> <60> <01> <02> <crc> <crc>
   Sensor reply: <new address> <41> <crc> <crc>

3. Reset sensor (optional step)
   Master send: <new address> <41> <00> <60> <01> <FF> <crc> <crc>
   Sensor reply: <new address> <41> <crc> <crc>

   Note : the sequence assumes only one sensor on the network at time of changing

As an example, if the network sensors 104 (0x68) is to be changed to 110 (0x6E), the exact sequence should be as follows;

1. Write new address
   Master send: <68> <41> <00> <20> <01> <6E> <B5> <4A>
   Sensor reply: <68> <41> <EE> <40>

2. Write "Copy to EEPROM" (permanently store ID into EEPROM)
   Master send: <6E> <41> <00> <60> <01> <02> <B4> <D5>
   Sensor reply: <6E> <41> <ED> <E0>

3. Reset sensor
   Master send: <6E> <41> <00> <60> <01> <FF> <75> <54>
   Sensor reply: <6E> <41> <ED> <E0>

Another example, if the network sensors 112 (0x70) is to be changed to 105 (0x69), the exact sequence should be as follows;

1. Write new address
   Master send: <70> <41> <00> <20> <01> <69> <F7> <4B>
   Sensor reply: <70> <41> <E4> <40>

2. Write "Copy to EEPROM" (permanently store ID into EEPROM)
   Master send: <69> <41> <00> <60> <01> <02> <B5> <62>
   Sensor reply: <69> <41> <EF> <D0>

3. Reset sensor
   Master send: <69> <41> <00> <60> <01> <FF> <74> <E3>
   Sensor reply: <69> <41> <EF> <D0>

Reading CO2 data from network CO2 sensors:

The sensor is addressed at 110 (0x6E).
We assume in this example that by sensor measured CO2 value is 600ppm*.

Sensor replies with CO2 reading 600ppm (600 ppm = 0x0258 hexadecimal).

Master Transmit:
<6E> <04> <00> <03> <00> <01> <C8> <95>

Slave Reply:
<6E> <04> <02> <02> <58> <6D> <A3>

Another example

The sensor is addressed at 111 (0x6F).
We assume in this example that by sensor measured CO2 value is 750ppm*.

Sensor replies with CO2 reading 750ppm (750 ppm = 0x02EE hexadecimal).

Master Transmit:
<6F> <04> <00> <03> <00> <01> <C9> <44>

Slave Reply:
<6F> <04> <02> <02> <EE> <D1> <D5>