IRNET-PRO

MODBUS PROTOCOL
CUSTOMER 'S MANUAL
- MODBUS PROTOCOL MT3071 -
REV. 21

N.E.T. SRL
Via Campania, 5 | 20006 | Pregnana Milanese | Milano | Italy

C.F. e P. IVA (VAT) 03231490966

T +39.02.9354.4190
E info@nenvitech.com
www.nenvitech.com

Technical Manual

N. MT-3071

Document Description

IRNET PRO MODBUS PROTOCOL

CUSTOMER'S MANUAL

Prepared by: Alessio Giovannetti Data 04/06/2013

Checked by: Damiano Frigo Data 05/06/2013

Approved by: Giacomo Frigo Data 06/06/2013

Rev.	Description of the changes	Data
10	Added the 5.1 paragraph and the sentences highlighted in yellow	05/09/2014
11	Added 3.1 CRC calculation paragraph	19/06/2015
12	Added AMP_ERROR and LAMP_ERROR	31/03/2016
13	Added Analogue_4-20mA_ERROR, RFI_ERROR and SW_ERROR	27/07/2016
14	Correct register address UartSpeed, information related and note for calibration gas level Modbus register (present only for data structure starting from 1)	19/10/2016
15	Corrected structure ID response frame and holding register information	24/11/2016
16	General revision	31/03/2017
17	Add some error description	18/05/201
18	Added Reset Coil information	15/06/2017
19	Added lookup table for Modbus frames	19/12/2017
20	Insert in new template	03/07/2018
21	Added SensorLife InputRegister	26/10/2022

N.E.T. has a policy of continuous development and improvement of its products. As such the specification for the device outlined in this manual may be changed without notice.

CONTENTS

1.0 Introduction

The MODBUS standard defines an application layer messaging protocol, positioned at level 7 of the OSI model that provides "client/server" communications between devices connected on different types of buses or networks. It standardizes also a specific protocol on serial line to exchange MODBUS request between a master and one or several slaves.
The objective of this document is to present the MODBUS protocol over serial line, in order to be used by all system designers when they want to implement MODBUS protocol on their serial line products. Thus, this document will facilitate interoperability between devices using the MODBUS protocol.
MODBUS is a protocol request/response type and offers services specified by function codes.
In the sensors IRNET PRO, this protocol was adapted to give the user a standard protocol for accessing to the resources of the apparatus.
A sub-set of the function codes provided by the Modbus protocol has been implemented, some memory locations assume certain functions if you have to write or read them, allowing to the user to be able to access the values of registers and variables of the equipment, as well as enable specific commands such as ZERO CALIBRATION and SPAN CALIBRATION Commands.
Also, it's possible to access directly to the input/output discrete provided by the detector.
All these features allow the detector IRNET PRO to communicate with any equipment that has the MODBUS RTU protocol on board such as gas detection systems, computer or PLC.
Once understood the mechanism of how each memory location is reachable through the protocol, assume a particular meaning, becomes trivial using the standard function code of the MODBUS to interact with the IRNET PRO detector.
MODBUS protocol on serial line, exists in two typologies: MODBUS ASCII and MODBUS RTU.
ASCII mode foresees that all characters that carry information between DEVICES, should be converted to ASCII characters so as to leave control characters to establish the beginning and end of a frame. This implies a remarkable increase of bytes that must be transmitted from a device to another.
To overcome this has been introduced MODBUS RTU.
MODBUS RTU is a binary protocol in which all 256 values or byte carry information. The beginning and end of the frame take place by detecting the timing of pauses between one frame and another and between one character and the other.
If it encounters a pause of 3.5 times the transmission time of a character on the serial line, it means that the frame is terminated and then you can proceed to its analysis.
The slave response occurs after the interpretation of the frame received, however always after a break of at least 3.5 characters.
If it encounters a pause of 1.5 characters between a character and the other instead, the current message is discarded and start again to receive a new message.
In this way, all data can be transmitted without undergoing conversion to ASCIII and therefore the numbers of bytes for each frame is considerably reduced and communication is faster.
It's for this reason that MODBUS RTU protocol has been chosen to be implemented in the IRNET PRO.
The ASCII protocol, although it is required by the specification, has not been implemented due to problems with internal resources to the equipment.
It is beyond the scope of this manual to explain exhaustively the protocol itself. Who wish to deepen their knowledge of the said protocol, can download from the MODICON site the specifics of PI-MBUS-300 RevJ which are the standard defacto of the protocol.
There is also a website www.modbus.org in which are discussed issues related to the MODBUS protocol and there is also a useful list of interesting links to MODBUS resources.

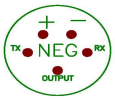
1.1 Communication Parameters

IRNET PRO sensor communicate through a LOGIC LEVEL serial port in half duplex mode.
The setting parameters of that port are permanent and are the following:

Half Duplex
Baud Rate:
4800
Parameters:
N (no parity),
1 (start bit),
8 (data bits),
1 (stop bit).

IRNET PRO is to be considered as a slave (it can never take the initiative to transmit, it can only reply to a query).
The address of the slave can be set through a MODBUS protocol, writing the value on the appropriate registry. The address 0 (zero) in MODBUS identifies a broadcast message, it cannot be assigned to any device.

serial port Connector
The serial line is present on the Rx/Tx pins, the pinout of the IRNET I/O connector is the following:



2.0 Definition of memory areas

The data model of the MODBUS protocol consists of 4 memory main zones defined within each device.
Each one of these zones has a specific meaning and can be achieved through specific function code. Such areas are:

NAME	AREA	OBJECT TYPE	ACCESS TYPE	COMMENTS
COILS	0	SINGLE BIT	READ / WRITE	THESE DATA CAN BE MODIFIED BY THE APPLICATION
UNUSED	1	SINGLE BIT	READ	NOT USED ON IRNET PRO
INPUT REGISTERS	3	16-BIT-WORD	READ	THESE DATA ARE PROVIDED BY THE SYSTEM I/O
HOLDING REGISTERS	4	16-BIT-WORD	READ / WRITE	THESE DATA CAN BE MODIFIED BY THE APPLICATION

According to the specifics these areas may also be overlapped each another. There is no need to be divided. As for IRNET PRO, in these areas have been mapped working registers, User variables, and control commands.
Let us see now in broad terms how the various memory areas have been used in order to adapt the protocol to IRNET PRO device. Subsequently all of the instructions here quoted will be discussed in deep way.

***Discrete Inputs** memory zone has not been used because there are no instruction in the IRNET PRO that can relay on the way to operate of the Discrete Inputs area.*

*In the **Input Register** memory area have been mapped all registers that can relay to the operating way of the Input Register zone*
Being that MODBUS provides 16bit registers while IRNET PRO has registers and variables both at 32bit and at 16 bit, access to this area of memory must always take place by accessing 2 consecutive registers, (in the case of 32bit float variables) or by accessing one register in the case of 16 variables unit at 16 bit.
In case of instructions GO and GOR the function code 10h must be used by setting the simultaneous transmission of 2 registers.
*In the **Holding Register** memory zone have been mapped all registers of the device.*
Also, in this case the access to this memory area must always take place by accessing 2 consecutive registers, (in the case of 32bit float variables) or by accessing one register in the case of 16 variables unit at 16 bit.

Access to the variables or to the float registers may also take place with two 16 bit communications (although not recommended).
In this case is necessary to consider the following:

In the reading phase, the data could change during the first of the two communications. If the sequence of reading lower-upper part is not correctly completed or if the data changes during communications, there might be false records. This can be avoided by using the 03h reading function code set with a reading of two registers.

In the writing phase, it's needed to write the lower part of the 32 bit data (register or variable) than proceed with the writing of the upper part. If the sequence of writing lower-upper part is not correctly completed there can be entries with false data. Even in this case the problem can be simplified by using the 10h function code with 2 registers set in writing.

3.0 Data

The serial communication occurs through the transmission of 16 bit binary words (word). Data are of 2 types: floating point [float] (consisting of 2 words) and string (formed by an array of n byte (8bit).

The microprocessor used presents an organization of the internal memory under little endian format, meaning that the less significant byte is located in the lowest byte of the memory and the most significant in the highest byte.
Big--endian and little-endian are two different methods used by computers to store in the memory data of larger size than the byte (es. word, dword, qword).

The difference between the two systems is given by the order in which the byte constituting the data to be stored are memorized:
Big-endian is the memorizing that start from the most significant byte to end with the lower significant one; it's used by Motorola processor and by protocol used in Internet.
Little-endian is the memorizing that start from the lower significant byte to end with the most significant one it's used by Intel processors.

This differentiation does not concern the bit position inside the byte (in which case we speak of bit order) or the position of the characters in a string. Instead it's important in the interpretation (or decoding) of the multi-byte encoding of string of characters (such as: encoding UTF-16 of the standard unicode).
The big-endian order, which has been chosen as the standard order in several standard protocols used on the internet, is therefore also called network byte.

IN THE CASE OF A WORD (16 BIT), THE HEXADECIMAL NUMBER 0X0123 WILL BE STORED AS:

LITTLE ENDIAN		BIG ENDIAN	
0X23	0X01	0X01	0X23
BYTE: 0	1	0	1

IN THE CASE OF A DWORD (32 BIT), THE HEXADECIMAL NUMBER 0X01234567 WILL BE STORED AS:

LITTLE ENDIAN				BIG ENDIAN			
0X67	0X45	0X23	0X01	0X01	0X23	0X45	0X67
BYTE: 0	1	2	3	0	1	2	3

(In the examples the value 0x01 is the most significant byte)

FLOATING POINT

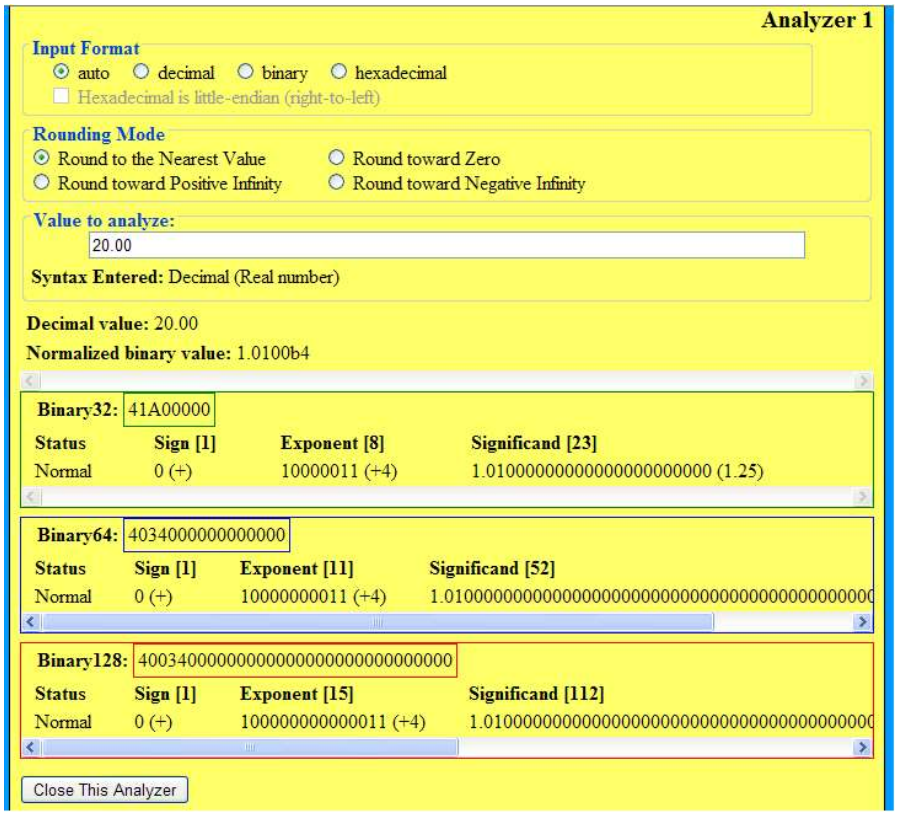
The values of floating point follow the specific IEEE 754 with 32 bit floating point standard.

MSB		LSB	
SEEEEEEEE	:	MMMMMMMM	:
EMMMMMMM		MMMMMMMM	
WORD A		WORD B	

S : sign of the number 0 positive 1 negative
E : exponent at 8 bit
M : mantissa of the number 23 bit

Example reading values in float:
To read 3 Holding register expressed in float starting from the register Modbus 4000, 6 words will have to be asked, then the registers 4 0000 – 4 0005.

At http://babbage.cs.qc.cuny.edu/IEEE-754/ you can find a simple form for IEEE-754 Analysis, in order to simplify float to int32 conversion.



STRING VALUES

MSB	LSB
CCCCCCCC : CCCCCCCC	CCCCCCCC : CCCCCCCC
CHARACTER[A]0 CHARACTER[A]1 WORD A	CHARACTER[B]2 CHARACTER[B]3 WORD B

C : bit in Character[x]

Example:
To read a 4bytes long string from the holding register area starting from the register Modbus 4 00BA 2 word will have to be asked, then the registers 4 00ba – 4 00bb must be read

3.1 CRC Calculation

In order to compute MODBUS CRC-16 (little endian) in a CRC calculation tool use the following parameters:

- *Type: CRC-16*
- *Initial value: FFFF (hex)*
- *Polynomial: 8005 (hex)*
- *XOR Out: 0*
- *Reflection In: ON*
- *Reflection Out: ON*

At http://www.lammertbies.nl/comm/info/crc-calculation.html you can find a simple tool to perform MODBUS CRC-16 calculation (note that the result is in big endian).

4.0 Supported Modbus functions

Supported functions
The IRNET PRO supports the following functions:

03h Read Holding Registers
04h Read Input Registers
Reset Device Coil (0x00D0)
Reset Device Coil bytes Frame:
[Address, Coil Command, Coil High, Coil Low, Value High, Value Low, CRC Low, CRC High]
Example for Reset Device Frame for address 0x01:
[0x01, 0x05, 0x00, 0xD0, 0x00, 0x00, 0xCC, 0x33]

Request:

Binary	Hex	Decimal	Purpose
00000001	0x01	1	Sensor Address
00000101	0x05	5	Function Code (Write Single Coil)
00000000	0x00	0	Output Address High
11010000	0xD0	208	Output Address Low
00000000	0x00	0	Output Value High
00000000	0x00	0	Output Value Low
11001100	0xCC	204	CRC Low
00110011	0x33	51	CRC High

Response:

Binary	Hex	Decimal	Purpose
00000001	0x01	1	Sensor Address
00000101	0x05	5	Function Code (Write Single Coil)
00000000	0x00	0	Output Address High
11010000	0xD0	208	Output Address Low
00000000	0x00	0	Output Value High
00000000	0x00	0	Output Value Low
11001100	0xCC	204	CRC Low
00110011	0x33	51	CRC High

** Additional functions as Zero Calibration and Span Calibration are available on request.*

Following are provided the addresses of the MODBUS registers and the related correspondence with the functions of the IRNET PRO.

5.0 Input and Holding Registers

5.1 Data structure concept

From firmware version 3.X.X Net has introduced the structure concept, in order to maintain compatibility respect older firmware's version and capability to introduce new functionality (request from customers) and from the natural evolution of the product. Customers are able to read structure id of sensor using READ_INPUT_REGISTER (0x04) modbus command, from register location 0x0400 (only one register)

To read the structure id, user must send the command:

READ_INPUT_REGISTER (0x04) for register 0x0400

0x01 0x04 0x04 0x00 0x00 0x01 CRC

where:
0x01 is the sensor address
0x04 is the function to read input register
0x0400 is the Modbus address to read the structure id
0x0001 is the quantity of register to read structure id

The string received back from the sensor as answer to the above command will be:
0x01 0x04 0x02 0xXX 0xXX CRC

where:
0x01 is the address
0x04 is the function for read input register
0x02 is the byte count, or rather number of byte concerning the input register
0xXXXX is the UNIT16 register containing the id of the structure

if sensor respond with a modbus error code

0x01 0x84 0x02 CRC

where:
0x01 is the address
0x084 is the command error code for read input register

0x02 is the Modbus error (Illegal Data Address) is to be intended that sensor is using first type of data structure n°0

Actual value of data structure is 1 (ONE). All new functionality of data structure 1 will be detailed below.

5.2 INPUT REGISTERS

MODBUS ADDRESS	TYPE	R / W	REGISTER	DESCRIPTION	MEASUREMENT UNIT	QUANTITY OF REGISTERS
0x0000	FLOAT 32BIT	R	RESERVED	RESERVED	-	2
0x0002	FLOAT 32BIT	R	RESERVED	RESERVED	-	2
0x0004	FLOAT 32BIT	R	TEMPERATURE	SENSOR TEMPERATURE	KELVIN	2
0x0006	FLOAT 32BIT	R	CONCENTRATION	TARGET GAS CONCENTRATION	CONCENTRATION PPM, %VOL,	2
0x0008	UINT16 16BIT	R	ERROR	SENSOR ERROR STATE	SEE DESCRIPTION	1
0x0009	UINT16 16BIT	R	WARNING	SENSOR WARNING STATE	SEE DESCRIPTION	1
0x000A	UINT16 16BIT	R	FIRMWARE VERSION MAJOR	FIRMWARE VERSION MAJOR	-	1
0x000B	UINT16 16BIT	R	FIRMWARE VERSION MINOR	FIRMWARE VERSION MINOR	-	1
0x000C	UINT16 16BIT	R	FIRMWARE VERSION REVISION	FIRMWARE VERSION REVISION	-	1
0x000D	UINT16 16BIT	R	FIRMWARE BUILD DATE DD	FIRMWARE BUILD DATE DD	-	1
0x000E	UINT16 16BIT	R	FIRMWARE BUILD DATE MM	FIRMWARE BUILD DATE MM	-	1
0x000F	UINT16 16BIT	R	FIRMWARE BUILD DATE YYYY	FIRMWARE BUILD DATE YYYY	-	1
0x001E	UINT32 32BIT	R	SENSOR LIFE DD	SENSOR LIFE DD	-	2
0x0400	UINT16 16BIT	R	DATA STRUCTURE ID *	DATA STRUCTURE USED INSIDE THE SENSOR	-	1

*Note: *0x0400 is available only with data structure n° 1 or higher*

5.3 HOLDING REGISTERS

MODBUS ADDRESS	TYPE	R / W	REGISTER	DESCRIPTION	MEASUREMENT UNIT	QUANTITY OF REGISTERS
0x0016	UINT16 16BIT	R	MODBUS DEVICE ADDRESS (NOTE)		1..240	1
0x0017	UINT16 16BIT	R	LOT DD		DAY	1
0x0018	UINT16 16BIT	R	LOT MM		MONTH	1
0x0019	UINT16 16BIT	R	LOT YYYY		YEAR	1
0x0026	STRING 8BYTES	R	GAS CYLINDER			4
0x002A	STRING 4BYTES	R	SERIAL NUMER CODE			2
0x002C	STRING 8BYTES	R	SERIAL NUMBER			4
0x0030	STRING 4BYTES	R	LOT CODE			2
0x0032	STRING 8BYTES	R	BUILD DATE			4
0x0036	STRING 8BYTES	R	TARGET GAS			4
0x003C	STRING 4BYTES	R	MEASUREMENT UNIT			2
0x003E	FLOAT 32BIT	R	FULL SCALE VALUE			2
0x0200	FLOAT 32BIT	R/W	CALIBRATION GAS LEVEL*			2
0x007C	UINT16 16BIT	R	UART SPEED*			1

Note: Sensor's address of each sensor is set at "1" as default.

**UART SPEED and Calibration Gas Level are available only with data structure n° 1 or higher*

Following are the Warning code encrypted (BIT VALUE):

Warning	Code	Description
NO_WARNING	0x00	(no warning, functioning ok)
WARMUP_WARNING	0x01	(warm-up)
INVALID_ACTIVE_WARNING	0x02	(active not included within the functional limits expected)
INVALID_REFERENCE_WARNING	0x04	(reference not included within the functional limits expected)
INVALID_TEMPERATURE_WARNING	0x08	(TEMP not included within the functional limits expected)
INVALID_READINGS_WARNING	0x10	(active and reference signals changed too fast. This condition can happen in case of fast gas flow rate transient, fast temperature changes and presence of radio frequency,interferences. In case this flag is active then gas concentration is freezed)
INVALID_ACTIVERMS_WARNING	0x20	(WRONG VALUES on the active channel. Active signal is too low)
INVALID_REFERENCERMS_WARNING	0x40	(WRONG VALUES on the reference channel;.Reference signal is too low)
HW_TEST_WARNING	0x80	(HW test in progress. It is performed once per day)

Following are the error code encrypted:

ERROR	Code	Description
NO_ERROR	0x00	(no error, functioning ok)
UNUSED	0x01	
E2PROM_CKSM_ERROR	0x02	(Error on e2prom check sum calculation)
FLASH_CKSM_ERROR	0x03	(Error on flash check sum calculation)
RAM_ERROR	0x04	(Error in the internal RAM of CPU, checked once per day)
VDD_ERROR	0x05	(error in the internal voltage supply)
I2C_ERROR	0x06	(This error happens if it is occurred a writing or communication error to I2C hardware)
UNUSED	0x07	
SPI_ERROR	0x08	(This error happens if it is occurred a writing or communication error to SPI)
VREF_ERROR	0x09	(Error to the internal voltage reference)
DAC_ERROR	0x0A	(Error in initialization or writing of the internal DAC)
UNUSED	0x0B	
ANALOGUE_4-20MA_ERROR	0x0C	(Error to the analogue output signal, not correlated to digital concentration)
UNUSED	0x0D	
ADC_ERROR	0x0E	(Error in the initialization or reading of one ADC channel)
SW_ERROR	0x0F	(Error caused in case of internal wrong calculation or active or reference signals could be to a wrong value)
VIN_ERROR	0x10	(External power supply voltage out of range)
FLASH_READ_ERROR	0x11	(Error during flash reading)
FLASH_WRITE_ERROR	0x12	(Error during flash writing)
FLASH_ERASE_ERROR	0x13	(Error during flash erasing)
E2PROM_WRITE_ERROR	0x14	(Error during e2prom writing)
UNUSED	0x15	
RFI_ERROR	0x16	(error generated in case invalid readings warning is continuously present for five minutes. It can be caused by anomalous variation of active or reference signals)
VBG_ERROR	0x17	(Error of internal band gap measurement)
LAMP_ERROR	0x18	(Error caused when active and reference signals are too low.)
AMP_ERROR	0x19	(Error caused when active or reference signals are too low)

Here are following the Uart speed encrypted data:

UART_MODBUS_4800 BAUD	*0*
UART_MODBUS_9600 BAUD	*1*
UART_MODBUS_19200 BAUD	*2*
UART_MODBUS_38400 BAUD	*3*

THE DEFAULT VALUE OF THE UART SPEED IS 38400 BAUD FOR IRNET20 mm and IRNET20 mm LOW POWER VERSION.

THE DEFAULT VALUE OF THE UART SPEED IS 4800 BAUD FOR IRNET32 mm AND IRNET32 mm FOR REFRIGERANT GASES

THE UART SPEED IS A PARAMETER THAT CAN'T BE MODIFIED BY THE USER.

6.0 EXAMPLE OF QUERY

6.1READ INPUT REGISTERS

To read the concentration, user must send the command:

READ_INPUT_REGISTER (0x04) for register 0x0006

(Note: Concentration register is a FLOAT register at 32 bit, the reading must be done on 2 registers starting from position 0x0006)

E.g.:
The command to read the concentration is:

0x01 0x04 0x00 0x06 0x00 0x02 CRC

where:
0x01 is the sensor address
0x04 is the function to read input register
0x0006 is the Modbus address to start to read the concentration
0x0002 is the quantity of register to read (Concentration register is a float 32 bit)

The string received back from the sensor as answer to the above command will be:

0x01 0x04 0x04 0xXX 0xXX 0xXX 0xXX CRC

where:
0x01 is the address
0x04 is the function for read input register
0x04 is the byte count, or rather number of byte concerning the input register
0xXXXXXXXX is the value of Concentration in Little Endian mode. (2 Register, in float type mode)

During the concentration polling, it's better to read also WARNING and ERRORS registers.

Request:

BINARY	HEX	DECIMAL	PURPOSE
00000001	0x01	1	SENSOR ADDRESS
00000100	0x04	4	FUNCTION CODE (READ INPUT REGISTERS)
00000000	0x00	0	STARTING ADDRESS HIGH
00000110	0x06	6	STARTING ADDRESS LOW
00000000	0x00	0	QUANTITY OF REGISTERS HIGH
00000010	0x02	2	QUANTITY OF REGISTERS LOW
10010001	0x91	145	CRC LOW
11001010	0xCA	202	CRC HIGH

Response (valid):

BINARY	HEX	DECIMAL	PURPOSE
00000001	0x01	1	SENSOR ADDRESS
00000100	0x04	4	FUNCTION CODE (READ INPUT REGISTERS)
00000100	0x04	4	BYTE COUNT
00000000	0x00	0	REGISTER LOW, BYTE LOW *
00000000	0x00	0	REGISTER LOW, BYTE HIGH *
10100000	0xA0	160	REGISTER HIGH, BYTE LOW *
11000001	0xC1	193	REGISTER HIGH, BYTE HIGH *
01000010	0x42	66	CRC LOW
00010100	0x14	20	CRC HIGH

* 0x00 0x00 0xA0 0xC1 corresponds to a gas concentration of -20

Response (Error):

BINARY	HEX	DECIMAL	PURPOSE
00000001	0x01	1	SENSOR ADDRESS
10010000	0x90	144	ERROR CODE
00000001	0x01	1	EXCEPTION CODE
10001101	0x00	141	CRC LOW
11000000	0x00	192	CRC HIGH

6.2READ HOLDING REGISTERS

To read the calibration gas level, user must send the command:

READ_HOLDING_REGISTER (0x03) for register 0x0200

(Note: Calibration gas level register is a FLOAT register at 32 bit, the reading must be done on 2 registers starting from position 0x0200)

E.g.:
The command to read the calibration gas level is:

0x01 0x03 0x02 0x00 0x00 0x02 CRC

where:
0x01 is the sensor address
0x03 is the function to read holding registers
0x0200 is the Modbus address to start to read the calibration gas level
0x0002 is the quantity of register to read (Calibration gas level register is a float 32 bit)

The string received back from the sensor as answer to the above command will be:

where;
0x01 is the address
0x03 is the function for read holding register
0x04 is the byte count, or rather number of byte concerning the holding register
0xXXXXXXXX is the value of Calibration gas level in Little Endian mode. (2 Register, in float type mode)

Request:

BINARY	HEX	DECIMAL	PURPOSE
00000001	0x01	1	SENSOR ADDRESS
00000011	0x03	3	FUNCTION CODE (READ HOLDING REGISTERS)
00000010	0x02	2	STARTING ADDRESS HIGH
00000000	0x00	0	STARTING ADDRESS LOW
00000000	0x00	0	QUANTITY OF REGISTERS HIGH
00000010	0x02	2	QUANTITY OF REGISTERS LOW
11000101	0xC5	197	CRC LOW
10110011	0xB3	179	CRC HIGH

Response (valid):

BINARY	HEX	DECIMAL	PURPOSE
00000001	0x01	1	SENSOR ADDRESS
00000011	0x03	3	FUNCTION CODE (READ HOLDING REGISTERS)
00000100	0x04	4	BYTE COUNT
00000000	0x00	0	REGISTER LOW, BYTE LOW
00000000	0x00	0	REGISTER LOW, BYTE HIGH
10100000	0x48	160	REGISTER HIGH, BYTE LOW
01000001	0x42	65	REGISTER HIGH, BYTE HIGH
01000010	0x4C	66	CRC LOW
00000011	0x02	3	CRC HIGH

Response (Error):

BINARY	HEX	DECIMAL	PURPOSE
00000001	0x01	1	SENSOR ADDRESS
10010000	0x90	144	ERROR CODE
00000001	0x01	1	EXCEPTION CODE
10001101	0x00	141	CRC LOW
11000000	0x00	192	CRC HIGH