CYBER MODBUS CUSTOMER MANUAL

- MT3741 -

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-3741

Document Description

CYBER MODBUS CUSTOMER

PROTOCOL MANUAL

Prepared by:	Alessio Giovannetti	Data	10/06/2015
Checked by:	Damiano Frigo	Data	10/06/2015
Approved by:	Giacomo Frigo	Data	10/06/2015

Rev.	Description of the changes	Data	Sign
0	New Emission	10/06/2015	Alessio Giovannetti
1	Changed the references to the technical manual from MT1910 and MT1950 to MT3822 and MT3823	09/09/2015	Alessio Giovannetti
2	Fixed the description of holdind register 0x2005 and 0x2006 at page 17	25/02/2016	Alessio Giovannetti
3	Changed description of Input register 0x1002, Added concentration calculation	23/11/2016	Emanuele Mariani
4	Minor fixes	14/03/2017	Emanuele Mariani
5	General revision	31/03/2017	Maurizio Rossi
6	Added example to read holding reg; explained content of reg 0x2014h and 0x200El; added table with gas type; described how to read serial number	20/11/2017	Alessio Giovannetti
7	Insert in new template	05/07/2018	Maurizio Rossi
8	Table update	21/03/2019	Maurizio Rossi
9	Removed COILS section	07/11/2019	Emanuele Mariani

n.e.t. innovative gas sensing

Rev.	Description of the changes	Data	Sign
10	Corrected and updated Sensor type an gas type information of register 0x2014h, updated information to register 0x2007 high	24/01/2020	Alessio Giovannetti
11	Corrected and updated Unit of measure information of register 0x2019L; updated Gas type information register 0x2014H	28/10/2020	Alessio Giovannetti
12	Corrected Concentration calculation	13/04/2021	Emanuele Mariani
13	Removed Sensor Error (0x09) and Heating VGND ERROR (0x0D) from error list	29/07/2021	Alessio Giovannetti
14	Newly added Heating VGND ERROR (0x0D) inside error list	19/10/2021	Alessio Giovannetti
15	Added Concentration calculation example	19/01/2022	Emanuele Mariani
16	Updated sensor's list and Banner	21/03/2022	Alessio Giovannetti
17	Updated threshold and hysteresis information and calculations of registers 0x2001;0x2002;0x2003;0x2004 and updated Concentration calculation	27/04/2023	Alessio Giovannetti
18	Update concentration calculations, threshold, and hysteresis calculations; added new gas codes associated to register 0x2014	04/09/2023	Gianluca Cavallaro
19	Corrected gas name in sensor 0x5D and added new sensors type on register 0x200E and new gasses on register 0x2014	28/05/2024	Andrea Temporali
20	-Modified section 3.4 -Modified section 5.2: • Modified explanation of reading register 0x2000.	05/08/2024	Fraboni Fabio
21	Concentration calculation examples correction Modified section 3.1	13/01/2025	Emanuele Mariani

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 5
1.1 COMMUNICATION PARAMETERS 6
2.0 DEFINITION OF MEMORY AREAS 7
3.0 DATA 9
3.1 FLOATING POINT 10
3.2 STRING VALUES 11
3.3 CRC CALCULATION 11
3.4 CONCENTRATION CALCULATION 12
4.0 SUPPORTED MODBUS FUNCTIONS 13
5.0 REGISTERS AND COILS 13
5.1 INPUT REGISTERS 13
5.2 HOLDING REGISTERS 16
5.3 HOW TO READ INPUT REGISTERS (COMMAND 0X04) 31
5.4 HOW TO READ HOLDING REGISTERS (COMMAND 0X03) 32
5.5 MODBUS FRAME DESCRIPTION 34

1.0 INTRODUCTION

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 CYBER sensors, this protocol has been adapted to give the user a standard protocol for accessing to the resources of the apparatus.
A sub-set of the function codes has been implemented which is provided by the Modbus. Also, some memory locations assume certain functions, if you have to write or read them, allowing to access the values of registers and variable of the equipment.
It's also possible to access directly to the discrete input/output provided by the sensor.
All this allows the CYBER sensor 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 and assume a particular meaning, becomes trivial using the standard function code of the MODBUS to interact with the CYBER sensor.
Exist two typologies of MODBUS protocol on serial line: MODBUS ASCII and MODBUS RTU.
ASCII mode foresees that all characters that carry information between DEVICES, should be converted to ASCII characters 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 problem, 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 the SLAVE encounters a pause of 1.5 characters between a character and the other, the current message is discarded, and it starts again to receive a new message.
In this way, all data can be transmitted without undergoing conversion to ASCII and therefore the numbers of bytes for each frame is considerably reduced and the communication is faster.
It's for this reason that MODBUS RTU protocol has been chosen to be implemented in the CYBER sensor.
The ASCII protocol, although it is required by the MODBUS 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. Those who wish to deepen their knowledge of the protocol can download from the MODICON site the specific of PI-MBUS-300 RevJ which is the standard de facto 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

CYBER sensors communicate through an RS485 serial port in half duplex mode.
The setting parameters of that port are permanent and are the following:

Half Duplex
Baud Rate: 9600
Parameters: N (no parity), 1 (start bit), 8 (data bits), 1 (stop bit).

CYBER is to be considered as a slave (it can never take the initiative to transmit, it can only reply to a query). Due to the Cyber microprocessor's flash and ram limitations, the Modbus protocol is not a full standard protocol.
Most significant differences are on the number of Modbus command implemented, on the maximum numbers of registers that are possible to be read or written with a single multiple command and the capability of protocol to identify a broadcast command.
The address of the slave can be set through a MODBUS protocol command, writing the value on the appropriate registry. The address 0 (zero) in MODBUS identifies a broadcast message, it cannot be assigned to any device. In Cyber sensor, broadcast message aren't implemented in firmware and the address 0 (zero) is used as backdoor address to communicate with the device when master isn't able to known the real address of cyber sensor.

Please refer to manual MT3822 for
Single Board CYBER®
Cyber® Module for flammable, toxic and O2 gas detection
NOTE: TTL serial line
The serial communication output is a TTL serial output and enables
digital communication with a microprocessor system if present, in standard MODBUS.

Please refer to manual MT3823 for
4-20mA CYBER®
Cyber® Transmitter for flammable, toxic and oxygen gas detection using catalytic, electrochemical cell and IR technology
Cyber ® Head – Transmitter enclosed in ATEX certified head

NOTE: RS485 serial line
The serial communication output is a standard industrial RS485 and enables digital communication with a microprocessor system if present, in standard MODBUS.

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	0x4000	single Bit	Read / Write	These data can be modified by the application
Input Registers	0x1000 0x3000	16-bit-word	Read Only	These data are provided by the system I/O
Holding Registers	0x2000	16-bit-word	Read / Write	These data can be modified by the application

According to the specifications these areas may be also overlapped each another. There is no need to be divided. In these areas of the CYBER sensor 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 CYBER equipment. Subsequently will be discussed in deep way all the instructions here quoted.

In the **Coils** zone, have been mapped instructions for:

TO DO AN ACTION LIKE
For example: TO RESET SENSOR,

These instructions have no parameters to be set up and do not return values, it's enough to set the related bit to perform the related instruction.

Reading the addresses related to these statements, if the returned value is 1, the function results still active, otherwise, if the returned value is 0, the function is complete.

In the **Input Register** memory area have been mapped all registers that can relay to the way of operating of the Input Register zone

In the **Holding Register** memory zone have been mapped all registers of the equipment.

Being that the MODBUS provides 16bit registers while CYBER has registers and variables both at 32bit and at 16 bit, access to this memory area must always take place by accessing 2 consecutive registers, in the case of 32bit variables, or by accessing one register in the case of 16bit variables.
For the instructions GO and GOR the function code 10h must be used by setting the simultaneous transmission of 2 registers.

Access to the variables or to the float or int32 registers may also take place with two 16 bit communications (although not recommended). But is necessary to consider the following:

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

During the writing phase, it is necessary 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 completed correctly 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 or int32] (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 big endian format, meaning that the less significant byte is located in the highest byte of the memory and the most significant in the lowest byte.
Big-endian and little-endian are two different methods used by computers to store in the memory data of larger size than one 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 this 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)

3.1 Floating point

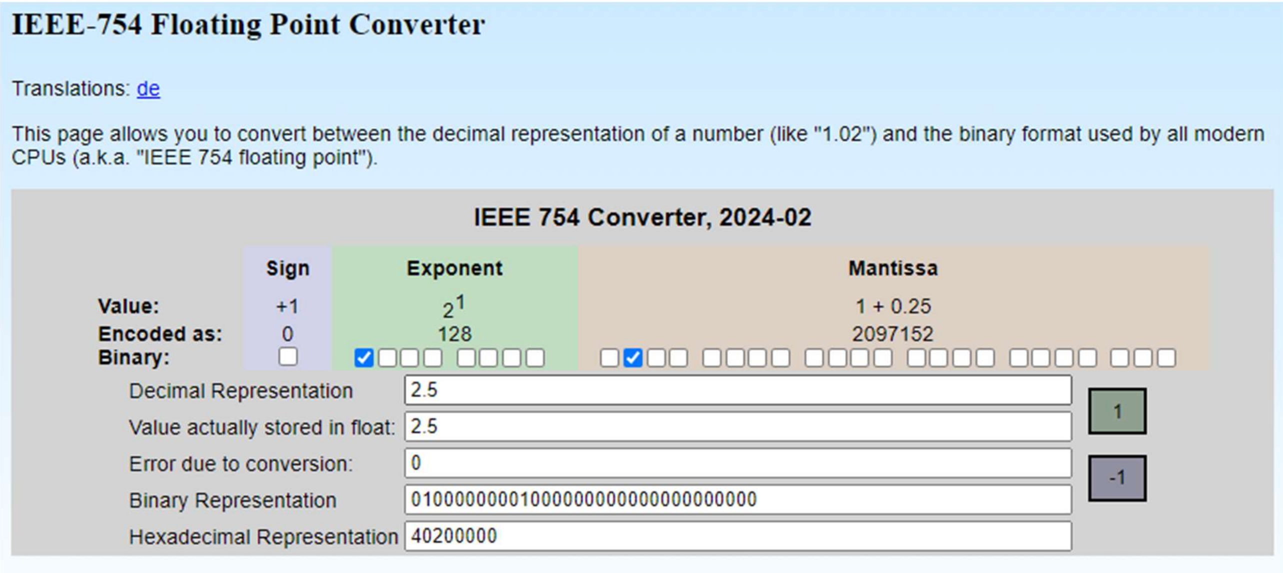
The values of floating point follow the specific IEEE 754 with 32bit floating point standard.

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 word will have to be asked, then the registers 4 0000 – 4 0005 will be read.

At https://www.h-schmidt.net/FloatConverter/IEEE754.html you can find a simple form for IEEE-754 analysis, in order to simplify float to int32 conversion.



3.2 String values

MSB		LSB	
CCCCCCCC : CCCCCCCC		CCCCCCCC : CCCCCCCC	
Character[a] 0	Character[a] 1	Character[b] 2	Character[b] 3
WORD A		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 will be used.

3.3 CRC Calculation

In order to compute MODBUS CRC-16m (big 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 litte-endian).

3.4 Concentration calculation

The following calculations allow to obtain the value of concentration:

There are three methods to calculate the gas concentration.

The first method described is the following:
FS full= (ROUND ((value in 0x2000 *418) / 4096))
Concentration = (value in 0x1002 * FS full / 418) /10^ value of bit 4-5 in 0x2007 (Hi byte).

The second method described is the following:
Real FS = (ROUND ((value in 0x2000 *418) / 4096)) /10^ value of bit 4-5 in 0x2007 (Hi byte).
Concentration = (value in 0x1002 * Real FS / 418).

The third method described is the following:
If **Full-Scale Cyber > 5000**:
Real FS Eng = 0x201B / 10^ value of bit 4-5 in 0x2007 (Hi byte).
Concentration = ROUND((value in 0x1002 * Real FS Eng / 418)).

*ROUND is the rounding function to the closest integer value.
*FS full is the full-scale value expressed as integer value (full scale multiplied 10^ value of bit 4-5 in 0x2007 (Hi byte)).
*Real FS is the full-scale value expressed with the unit of measure of the product
*Real FS Eng is the full-scale value expressed in case of the full-scale of cyber is greater than 5000

EXAMPLE 1:

Sensor specifications:
Full scale: 20,00
Unity of measure: ppm

value in 0x2000 = 0x4C8E = 19598
value in 0x1002 = 0x013D = 317
value of bit 4-5 in 0x2007 (Hi byte) = 0x02 = 2

FS full= (ROUND ((19598*418) / 4096)) = 2000
Concentration = (317 * 2000 / 418) /10^ value of bit 4-5 in 0x2007 (Hi byte) = 15,16ppm
(Value of bit 4-5 in 0x2007 (Hi byte) contains number of digits after the comma; in this case 2 decimal)

EXAMPLE 2 (full-scale with 2 decimals):

Sensor specifications:
Full scale: 20,00
Unity of measure: ppm

value in 0x2000 = 0x4C8E = 19598
value in 0x1002 = 0x013D = 317
value of bit 4-5 in 0x2007 (Hi byte) = 0x02 = 2

Real FS = (ROUND ((19598 *418) / 4096)) /10^ value of bit 4-5 in 0x2007 (Hi byte) = 20
Concentration = (317 * 20 / 418) = 15,16ppm

(Value of bit 4-5 in 0x2007 (Hi byte) contains number of digits after the comma; in this case 2 decimal).
EXAMPLE 3 (full-scale with 0 decimals):

Sensor specifications:
Full scale: 10000
Unity of measure: ppm

value in 0x201B = 0x2710 =10000
value in 0x1002 = 0x013D = 317
value of bit 4-5 in 0x2007 (Hi byte) = 0x00 = 0

Real FS Eng = 10000 / 10^0 = 10000
Concentration = ROUND((317 * 10000 / 418)) = **7583** ppm

4.0 SUPPORTED MODBUS FUNCTIONS

Supported functions
The CYBER sensor supports the following functions:

2 03h Read Holding Registers
3 04h Read Input Registers

* Additional functions as Zero Calibration, Span Calibration and writing commands are available on request. Please refer to manual MT3778

Following are provided the addresses of the MODBUS registers and the related correspondence with the functions of the CYBER sensor.

5.0 REGISTERS AND COILS

5.1 Input Registers

Modbus address	Type	R / W	Register	Description	Measurement unit	Quantity of registers
0x1000	Uint16 16bit	r	Raw sensor reading	Sensor value in point gas related to gross zero	--	1
0x1001	Uint16 16bit	r	Voltage input reading	Voltage input value	--	1
0x1002	--	r	Sensor reading	Sensor value in point gas	--	1
0x1003h 0x1003L	Uint16 16bit	r	Byteh errsts Bytel stsch1	Error Status	See description See description	1
0x1004	Uint16 16bit	r	Zero420 DAC	Zero DAC value		1

0x3000h 0x3000l	Uint16 16bit	r	Byteh Firmware Version Major bytel Firmware Version Revision	Firmware Version Major Firmware version minor Version Revision		1
0x3001 H 0x30011	Uint16 16bit	r	Byteh Firmware Build Date DD bytel Firmware Build Date MM	Firmware Build Date DD Firmware Build Date MM	--	1
0x3002	Uint16 16bit	r	Firmware Build Date YYYY	Firmware Build Date YYYY	--	1
0x3003	--	r	INTERNAL USE ONLY	Do not change	--	1

0x1000H Input Registers

Register 0x1000

• ADC converter's signal value in points uPc (0÷1024).
Vcc_uPC = 5v
Signal in V = ((Signal in points * Vcc_uPC) / 1024)

Register 0x1001

• Power supply value, in points.
Power supply mv = ((Power supply in points *(5.337/100)) + 0.74)

Register 0x1002

• Sensor reading.

Register 0x1003 low

Actual state of detector
Warning code encrypted (BIT VALUE):

NO_ALARM	0x00
FAULT	0x01
THRESHOLD1	0x02
THRESHOLD 2	0x04
THRESHOLD 3	0x08
OVERRANGE	0x10
Internal Use	0x20
Internal Use	0x40
Internal Use	0x80

If the bits from 0÷4 are equal to zero, the detector is in the normal operating state

Register 0x1003 high

Indicates the error state of detector
error code encrypted:

NO_ERRORS	0x00
HEATING	0x01

E2PROM_ERROR	0x02
FLASH_ERROR	0x03
RAM_ERROR	0x04
VCC_ERROR	0x05
VGND_ERROR	0x08
E2PROM_CKSM_ERROR	0x0A
CHANGE_SENSOR	0x0B
ANALOG_OUT_ERROR_4_20ma	0x0C
Heating VGND ERROR	0x0D
ADC_ERROR	0x0E

Register 0x1004
• Value for generating the 4mA of the current output (4...20mA) in points.
Internal use only

0x3000H Input Registers
Register 0x3000 high

Version Number use this sintax **X.yyz**

X is version major	(Version Major **X**.yyz	Es. **1**.14A)
YY is version minor	(Version Minor x.**Y**yz	Es. 1.**14**A)
Z is the version revision	(Version Revision x.YY**Z**	Es. 1.14**A**)

• Version number major (x)
Es. oxo1 => major 1

Register 0x3000 low
• Version number minor (yy).
Es. Version 14 => 0x0D = 14 dec

(z) is facultative
Version 1.14A => 0x8C hex = 140 dec => 14- 0 => 14 =14 and 0 = A

Register 0x3001 high
• Firmware date day (dd).

Register 0x3001 low
• Firmware date month (mm).

Register 0x3002
• Firmware date year (yyyy)

Register 0x3003
•Internal use only.

5.2 Holding Registers

Modbus address	Type	R/W	Register	Description	Measurement unit	Quantity of registers
0x2000	Uint16 16bit	R	fsch1	Full scale		1
0x2001	Uint16 16bit	R/W	S1ch1	Threhsold1		1
0x2002	Uint16 16bit	R/W	S2ch1	Threhsold2		1
0x2003	Uint16 16bit	R/W	S3ch1	Threhsold3		1
0x2004	Uint16 16bit	R/W	Isch1	Hysteresis		1
0x2005	--	R	INTERNAL USE ONLY	Do not change		1
0x2006	--	R/W	Kappa	Multiplication constant for normalizing readout		1
0x2007h 0x2007l	Uint16 16bit	R	Measure unit INTERNAL USE ONLY	Number of decimals and thresholds parameter Do not change		1
0x2008	Uint16 16bit	R/W	Zero value	Zero value		1
0x2009h 0x2009l	Uint16 16bit	R	Sn0 INTERNAL USE ONLY Sn1 INTERNAL USE ONLY	Serial number (letter)		1
0x200Ah 0x200AL	Uint16 16bit	R	Sn2 INTERNAL USE ONLY Sn3 INTERNAL USE ONLY	Serial number		1

Modbus address	Type	R/W	Register	Description	Measurement unit	Quantity of registers
0x200Bh 0x200Bl	--	R	INTERNAL USE ONLY INTERNAL USE ONLY	Do not change Do not change		1
0x200Ch 0x200Cl	--	R	INTERNAL USE ONLY INTERNAL USE ONLY	Do not change Do not change		1
0x200D	--	R	INTERNAL USE ONLY	Do not change		1
0x200Eh 0x200El	Uint16 16bit	R	INTERNAL USE ONLY Numsens INTERNAL USE ONLY	Do not change Sensor type Do not change		1
0x200Fh 0x200Fl	--	R	INTERNAL USE ONLY INTERNAL USE ONLY	Do not change Do not change		1
0x2010h 0x2010l	--	R	INTERNAL USE ONLY INTERNAL USE ONLY	Do not change Do not change		1
0x2011h 0x2011l	--	R	INTERNAL USE ONLY INTERNAL USE ONLY	Do not change Do not change		1
0x2012h 0x2012l	--	R	INTERNAL USE ONLY INTERNAL USE ONLY	Do not change Do not change		1
0x2013	--	R	INTERNAL USE ONLY	Do not change		1
0x2014h 0x2014l	Uint16 16bit	R	Typegas INTERNAL USE ONLY flags	Do not change		1

Modbus address	Type	R/W	Register	Description	Measurement unit	Quantity of registers
0x2015	--	R	INTERNAL USE ONLY	Do not change		1
0x2016	--	R	INTERNAL USE ONLY	Do not change		1
0x2017h 0x2017l	Uint16 16bit	R/W	INTERNAL USE ONLY netid	Do not change RS485 Address		1
0x2018h 0x2018l	--	R	INTERNAL USE ONLY INTERNAL USE ONLY	Do not change Do not change		1
0x2019h 0x2019l	Uint16 16bit	R	Sensortech uom	Sensor technology Unit of measure		1
0x201A	Sint16	R	INTERNAL USE ONLY	Do not change		1
0x201B	Uint16 16bit	R	Full scale eng.	Reg. 0x2000 engineering		1

Register 0x2000.
• To calculate the Full scale after reading of register:
Real FS = (ROUND ((value in 0x2000 *418) / 4096)) /10^ value of bit 4-5 in 0x2007 (Hi byte).

*ROUND is the rounding function to the closest integer value.

NB.
If the full-scale of your CYBER is greater than **5000,** the calculation of the **Real FS** is as follows:
Real FS = 0x201B / 10^ value of bit 4-5 in 0x2007 (Hi byte).

Register 0x2001.
• Threshold 1 after reading of register:
Threshold 1 = ROUND ((value in ***0x2001 reg*** * Real Fs) / 418)

EXAMPLE:
value in 0x2001 = 21.
Real Fs = 100
Threshold_1_percentage of sensor’s full scale = ROUND ((21 * Real Fs) / 418) = 5%

• Threshold 1 value to be written in register:
Value to be written in register 0x2001 = ROUND (Threshold_1_percentage of sensor’s full scale * 418 / Real FS)
(100% FS = 418pnt)

Value to be written in register 0x2001 = ROUND (5 * 418 / 100) = 21pnt = 0x15h (in case Real FS=100)

For negative threshold reg 2007h (bit1 = 1) complement the value of each bit to 1 of the content of the register

Register 0x2002.
• Threshold 2 after reading of register:
Threshold 2 = ROUND ((value in ***0x2002 reg*** * Real Fs) / 418)

EXAMPLE:
value in 0x2002 = 42.
Real Fs = 100
Threshold_2_percentage of sensor's full scale = ROUND ((42* Real Fs) / 418) =10 %

• Threshold 2 value written in register:
Value to be written in register 0x2002= ROUND (Threshold_1_percentage of sensor's full scale * 418 / Real FS)
(100% FS = 418pnt)

Value to be written in register 0x2002 = ROUND (10*418 / 100) = 42pnt = 0x2Ah (in case Real FS=100)

For negative threshold reg 2007h (bit1 = 1) complement the value of each bit to 1 of the content of the register

Register 0x2003.
• Threshold 3 after reading of register:
Threshold 3 = ROUND ((value in ***0x2003 reg*** * Real Fs) / 418)

EXAMPLE:
value in 0x2003 = 63.
Real Fs = 100
Threshold_3_percentage of sensor's full scale = ROUND ((63 * Real Fs) / 418) =15 %

• Threshold 3 value written in register
Value to be written in register 0x2003= ROUND (Threshold_1_percentage of sensor's full scale * 418 / Real FS)
(100% FS = 418pnt)

Value to be written in register 0x2003 = ROUND (15*418 / 100) = 63pnt = 0x3Fh (in case Real FS=100)

For negative threshold reg 2007h (bit1 = 1) complement the value of each bit to 1 of the content of the register

Register 0x2004.
• Hysteresis after reading of register:
Hysteresis = ROUND ((value in ***0x2004 reg*** * Real Fs) / 418)

EXAMPLE:
Value in 0x2004=4
Real Fs = 100
Hysteresis= ROUND ((4*100)/418) =1

• Hysteresis value written in register:
Hysteresis = 1% of FS
Value to be written in register 0x2004= ROUND (Hysteresis * 418 / Real FS)
(in case Real FS=100)
Value to be written in register 0x2004= ROUND (1 * 418 / 100) = 4pnt = 0x04h (in case Real FS=100)

Register 0x2005.
• Internal use only.

Register 0x2006.
• Calibration coefficient multiplication factor in points.

Register 0x2007 low (byte low)
• Internal use only.

Register 0x2007 high (byte high)
• Bit 0 => 0 = 60% thresholds no limit – 1 = 60% thresholds limit
• Bit 1 => 0 = rising threshold – 1 = dropping threshold
• Bit 2 => Not used
• Bit 3 => Not used
• Bit 4÷5 => Indicates the measurement's number of decimals

Value 0 = No decimals	bit 4=0 and bit 5=0
Value 1 = 1 decimal	bit 4=1 and bit 5=0
Value 2 = 2 decimals	bit 4=0 and bit 5=1
Value 3 = 3 decimals	bit 4=1 and bit 5=1

Bit 6÷7 => Not used

Register 0x2008.
• Zero from calibration, in points.

Register 0x2009 low (byte low)
• Serial number (low) Uint8 8bit.

Register 0x2009 high (byte high)
• Serial number (middle low) in ASCII.

Register 0x200A low (byte low)
• Serial number (middle high) Uint8 8bit.

Register 0x200A high (byte high)
• Serial number (high) Uint8 8bit.

Reading Serial Number Example
Expected result: Returns serial number K71234
Structure of the request frame and expected frame as described in point '5.5 Reading holding register' of this document.
Frame structure: [adr, cmd, pos_l, pos_h, N_bytes_l, N_bytes_h, crc_l, crc_h]
Frame request: [0x00, 0x03, 0x20, 0x09, 0x00, 0x02, 0x1E, 0x18]
Frame expected: [0x00, 0x03, 0x04, 0x07, 0x4B, 0x22, 0x0C, 0x83, 0x34]
Serial Number Registers: [0x07, 0x4B], [0x22, 0x0C]
Decode bytes as explained above: [7, K], [34, 12]
Swap bytes of each register: [K, 7], [12,34]
Obtain expected result: K71234

Register 0x200B low (byte low)
• Internal use only.

Register 0x200B high (byte high)
• Internal use only.

Register 0x200C low (byte low)
• Internal use only.

Register 0x200C high (byte high)
• Internal use only.

Register 0x200D.
• Internal use only.
Register 0x200E low (byte low)
• Sensor types configured in the configuration file.
Possible values in this register are between 0x01 and 0x45.

Register 0x200E high (byte high)
• Internal use only.

Register 0x2013.
• Internal use only.

Register 0x2014h.
• Gas type.
• Valuex = Gas types configured in the configuration file.

In the next pages are indicated gas type related to the sensor used inside FW version equal to or greater than 2.13.
To identify gas type, it is necessary to also read sensor type register.
In case of old FW version (2.11) corrected association to the sensor type is not obtained looking at the row indicated inside the register of the sensor but at this row= value of the register (expressed in hexadecimal) -0x02 it means that in case inside a sensor with FW version 2.11 I'm reading Sensor type=0x30 it is not indicating a sensor for Nitric oxide but it is related to the row 0x2E that means Sulphur dioxide.
At the same time, in case of old FW version (2.11) inside the gas type register associated to the Sulphur dioxide sensor (row 0x2E) it will not be present a value equal to 0x01 but a value equal to= value inside table+ 0x01. In other words, inside old FW, Hydrogen gas associated to sensor 0x0A is not indicated by 0x0D but with 0x0E.

Content of register 0x200E low (sensor type)	Content of register 0x2014 high (gas type)	Gas Type
sensor 0x09	0x00	Methane
	0x00	Methane
	0x01	Acetic acid
	0x02	Acetone
	0x03	Ammonia
	0x04	Butane
	0x05	Butyl-acetate

sensor 0x0A	0x06	Cyclo-hexane
	0x07	Cyclo-pentane
	0x08	Dioxane
	0x09	Ethane
	0x0A	Ethanol
	0x0B	Ethyl-acetate
	0x0C	Ethylene
	0x0D	Hydrogen
	0x0E	Iso-butane
	0x0F	Pentane
	0x10	Iso-propyl-alcool
	0x11	Metanolo
	0x12	Metyletilkhetone
	0x13	Iso-butyl-alcool
	0x14	N-butyl-alcool
	0x15	N-heptane
	0x16	N-hexane
sensor 0x0A	0x17	N-propanol
	0x18	Propane
	0x19	Iso-octane
	0x1A	Nonane
	0x1B	Decane
	0x1C	Propylene
	0x1D	Styrene monomer
	0x1E	Toluene
	0x1F	Iso-pentane
	0x20	Xylene
	0x21	Gasoline vapours
	0x22	GPL
	0x23	Ethylene oxide
	0x24	Heptane
	0x25	Di-ethylether
	0x26	Xilolo
	0x27	N-Octane
	0x28	Carbon Monoxide
Sensor 0x0B	0x00	Nitric-oxide
sensor 0x0C	0x00	Carbon-monoxide
sensor 0x0D	0x00	Sulphur-dioxide

Sensor 0x0E	0x00	Sulphur-dioxide
sensor 0x0F	0x00	Hydrogen
sensor 0x10	0x00	Nitrogen-dioxide
sensor 0x11	0x00	Chlorine
Sensor 0x12	0x00	Carbon-monoxide
Sensor 0x13	0x00	Oxygen
sensor 0x14	0x00	Chlorine
sensor 0x15	0x00	Ammonia
sensor 0x16	0x00	Oxygen
sensor 0x17	0x00	Oxygen
sensor 0x18	0x00	Carbon-monoxide
sensor 0x19	0x00	Hydrogen Sulphide
sensor 0x1A	0x00	Methane
	0x01	LPG
	0x02	Gasoline Vapours
	0x03	Butane
	0x04	Propane
sensor 0x1B	0x00	Butane
sensor 0x1C	0x00	Ethyl ether
	0x01	Methane
sensor 0x1D	0x00	Ethyl-Alchool
sensor 0x1E	0x00	Ozone

sensor 0x1F	0x00	Carbon Monoxide
sensor 0x20	0x00	Methane
	0x01	Propane
	0x02	N-Butane
	0x03	N-Pentane
	0x04	N-Hexane
	0x05	N-Heptane
	0x06	N-Octane
	0x07	Metanol
	0x08	Ethanol
	0x09	Iso-Propanol
	0x0A	Carbon monoxide
	0x0B	Acetone
	0X0C	Metyletilkhetone
	0x0D	Toluene
	0x0E	Ethyl-Acetate
	0x0F	Hydrogen
	0x10	Ammonia
	0x11	Unleaded Gasoline
	0x12	Ethylene
sensor 0x21	0x00	Methane
	0x01	Propane
	0x02	N-Butane
	0x03	N-Pentane
	0x04	N-Hexane
	0x05	N-Heptane
	0x06	N-Octane
	0x07	Methanol
	0x08	Ethanol
	0x09	Iso-propanol
	0x0A	Carbon-monoxide
	0x0B	Acetone
	0X0C	Metyletilkhetone
	0x0D	Toluene
	0x0E	Ethyl-acetate
	0x0F	Hydrogen
	0x10	Ammonia
	0x11	Unleaded gasoline
	0x12	Ethylene
	0x13	Ammonia
	0x00	Methane
	0x01	Propane
	0x02	N-Butane
	0x03	N-Pentane
	0x04	N-Hexane
	0x05	N-Heptane
	0x06	N-Octane

sensor 0x22	0x07	Methanol
	0x08	Ethanol
	0x09	Iso-propanol
	0x0A	Carbon-monoxide
	0X0B	Acetone
	0X0C	Metyletilkhetone
	0x0D	Toluene
	0x0E	Ethyl-acetate
	0x0F	Hydrogen
	0x10	Ammonia
	0x11	Unleaded gasoline
	0x12	Ethylene
sensor 0x23	0x00	Hydrogen
	0x01	N-Pentane
	0x02	N-Hexane
	0x03	N-Heptane
	0x04	N-Octane
	0x05	Methanol
	0x06	Ethanol
	0x07	Iso-Propanol
	0x08	Carbon-monoxide
	0x09	Acetone
	0x0A	Metyletilkhetone
	0x0B	Toluene
	0X0C	Ethyl-acetate
	0x0D	Ammonia
	0x0E	Unleaded gasoline
	0x0F	Ethylene
sensor 0x24	0x00	Ammonia
	0x01	Propane
	0x02	N-Butane
	0x03	N-Pentane
	0x04	N-Hexane
	0x05	N-Heptane
	0x06	N-Octane
	0x07	Methanol
	0x08	Ethanol
	0x09	Iso-propanol
	0x0A	Carbon-monoxide
	0x0B	Acetone
	0X0C	Metyletilkhetone
	0x0D	Toluene
	0x0E	Ethyl-acetate
	0x0F	Hydrogen
	0x10	Unleaded gasoline
	0x11	Ethylene
sensor 0x25	0X00	Methane

sensor 0x26	0X00	Methane
sensor 0x27	0X00	Propane
sensor 0x28	0X00	Propane
sensor 0x29	0x00	Ammonia
sensor 0x2A	0x00	Nitrogen-dioxide
sensor 0x2B	0x00	R404a
	0x01	R134a
	0x02	R1234yf
	0x03	R125
	0x04	R507
	0x05	SF6
	0x06	R32
	0x07	R452b
	0x08	R407a
	0x09	R1234ze
	0x0A	R449a
	0x0B	R22
	0x0C	R1234zd
	0x0D	R513a
	0x0E	C3H8 (0-2,1%vol)
	0x0F	R454B
	0x10	R448a
sensor 0x2C	0x00	Carbon dioxide
	0x01	Methane (0-4,4 %Vol)
	0x02	Methane (0-5 %Vol)
	0x03	Propane (0-1,7 %Vol)
	0x04	Propane (0-2,1 %Vol)
	0x05	N-Butane (0-1,4 %Vol)
	0x06	Ethanol (0-3,1 %Vol)
	0x07	Ethyl-acetate (0-2 %Vol)
	0x08	Heptane (0-0,85 %Vol)
	0x09	Hexane (0-1 % Vol)
	0x0A	Isopropanol (0-2 %Vol)
	0x0B	Methane (0-100 %Vol)
sensor 0x2D	0x00	Oxygen

sensor 0x2E	0x00	Sulphur-dioxide
sensor 0x2F	0x00	Sulphur-dioxide
sensor 0x30	0x00	Nitric-oxide
sensor 0x31	0x00	Formaldehyde
sensor 0x32	0x00	Oxygen
sensor 0x33	0x00	Sulphur dioxide
sensor 0x34	0x00	Carbon monoxide
sensor 0x35	0x00	Carbon monoxide
sensor 0x36	0x00	Hydrogen-sulphide
sensor 0x38	0x00	Ammonia
sensor 0x39	0x00	Ammonia
sensor 0x3A	0x00	Ammonia
sensor 0x3B	0x00	Methane
sensor 0x3C	0x00	Methane
	0x01	LPG
sensor 0x3D	0x00	Sulphur Dioxide
sensor 0x3E	0x00	Carbon Monoxide
sensor 0x3F	0x00	Hydrogen Sulphide
sensor 0x40	0x00	Carbon dioxide
	0x01	Methane (0-4,4% Vol)
	0x02	Methane (0-5% Vol)
	0x03	Propane (0-1,7% Vol)
	0x04	Propane (0-2,1% Vol)
	0x05	Butane (0-1,6% Vol)

	0x06	Pentane (0-1,1% Vol)
	0x07	Acetone (0-2,5 %Vol)
	0x08	Ethanol (0-3,1 % Vol)
	0x09	Butane (0-1,4 % Vol)
	0x0A	Toluene (0-1 % Vol)
	0x0B	R32 (0-14,4% Vol)
	0x0C	Cyclopentane (0-1,4% Vol)
	0x0D	Iso-butane (0-1,3%vol)
	0x0E	Propylene
	0x0F	Ethane
	0x10	N-butane (0-1,8% Vol)
sensor 0x41	0x00	Carbon-dioxide
sensor 0x42	0x00	Methane
	0x01	Propane
	0x02	N-butane
	0x03	N-pentane
	0x04	N-hexane
	0x05	N-heptane
	0x06	N-octane
	0x07	Methanol
	0x08	Ethanol (0-3,3% Vol)
	0x09	Iso-propanol
	0x0A	Carbon-monoxide
	0x0B	Acetone
	0x0C	Metyletilkhetone
	0x0D	Toluene
	0x0E	Ethyl-acetate
	0x0F	Hydrogen
	0x10	Ammonia
	0x11	Unleaded gasoline
	0x12	Ethylene
	0x13	Benzene
	0x14	Acetic Acid
	0x15	Cyclo-hexane
	0x16	Cyclo-pentane
	0x17	Iso-butane
	0x18	Iso-octane
	0x19	Styrene
	0x1A	Propylene
	0x1B	Xylene

	0x1C	LPG
	0x1D	Ethanol (0-3,1% Vol)
	0x1E	Methane (0-4,4%vol)
	0x1F	Diesel Vapors
	0x20	Hydrogen
	0x21	Ethane (0-3% Vol)
sensor 0x43	0x00	Carbon-monoxide
sensor 0x44	0x00	Ozone
sensor 0x45	0x00	Oxygen
sensor 0x46	0x00	Nitric-oxide
sensor 0x47	0x00	Carbon monoxide
sensor 0x48	0x00	Hydrogen
sensor 0x49	0x00	Hydrogen cyanide
sensor 0x4A	0x00	Formaldehyde
sensor 0x4B	0x00	Chlorine Dioxide
sensor 0x4C	0x00	Hydrochloric Acid
sensor 0x4D	0x00	Ethylene
sensor 0x4E	0x00	Diborane
sensor 0x4F	0x00	Nitric oxide
sensor 0x50	0x00	Hydrogen Sulphide
sensor 0x51	0x00	Acetylene
sensor 0x52	0x00	Chlorine
sensor 0x53	0x00	R32
	0x01	R410a (0-1% Vol)
	0x02	R1234ze
	0x03	R134a

sensor 0x54	0x00	Hydrogen Sulphide
sensor 0x55	0x00	Hydrogen Peroxide
sensor 0x56	0x00	Hydrogen Sulphide
sensor 0x57	0x00	Nitrogen Dioxide
sensor 0x58	0x00	Sulphur Dioxide
sensor 0x59	0x00	Ammonia
sensor 0x5A	0x00	Ammonia
sensor 0x5B	0x00	Ozone
sensor 0x5C	0x00	Nitrogen Dioxide
sensor 0x5D	0x00	Hydrogen Sulphide
sensor 0x5E	0x00	Hydrogen
sensor 0x5F	0x00	Hydrogen Peroxide
sensor 0x60	0x00	Ethylene
sensor 0x61	0x00	Hydrogen Sulphide
sensor 0x62	0x00	Hydrogen
sensor 0x63	0x00	Ethylene Oxide
sensor 0x64	0x00	Carbon Monoxide

Register 0x2014l.
• Enabling optional routines.

• Bit0
o Internal use only.

• Bit1
o Internal use only.

• Bit2
o Internal use only.

• Bit3
o Internal use only.

• Bit4
o 0 = Locked OVERRANGE condition
o 1 = Non-Locked OVERRANGE condition

• Bit5
o Internal use only.

Register 0x2017l.
• Identifier value on RS485 line 1…255.
Every device reply, apart from its own identifier, also to the “0” identifier (which, unlike suggested by the MODBUS protocol, is used not for broadcasting, but used as fault identifier, in order to have access to a detector’s data, which otherwise cannot be identified)

Register 0x2018h.
• Internal use only.

Register 0x2018l.
• Internal use only.

Register 0x2019h.
• Sensor technology
bit0-3-> sensor technology
0 - NOT DEFINED
1 - EC (ELECTROCHEMICAL CELL)
2 - PEL (PELLISTOR)
3 - IR (INFRARED)
4 - O2 (OXYGEN)
5 - SEM (SEMICONDUCTOR)
>6 RESERVED

Register 0x2019l.
• Readout measure unit

0	LFL%
1	Not used.
2	% vol
3	ppm
4	%vol of O2 (used in Oxygen sensors)
5-15	RESERVED

5.3 How to Read Input Registers (command 0x04)

This function code is used to read from 1 to 125 contiguous input register in a remote device.
The Request PDU specifies the starting register address and the number of registers 1-16 are addresses as 0-15.
The register data in the response message are packed as two bytes per register, with the binary contents right justified within each byte. For each register, the first byte contains the high order bits and the second contains the low order bits.

Request

Function code	1 Byte	**0x04**
Starting Address	2 Bytes	0x0000 to 0xFFFF
Quantity of Input Registers	2 Bytes	0x0001 to 0x007D

Response

Function code	1 Byte	**0x04**
Byte count	1 Byte	2 x **N***
Input Registers	**N*** x 2 Bytes	

*N = Quantity of Input Registers

Error

Error code	1 Byte	**0x84**
Exception code	1 Byte	01 or 02 or 03 or 04

Example of Read input registers:

Request 00 04 10 00 00 05 xx yy

00 backdoor address or Cyber address
04 function code
10 00 First register to be read.
00 05 number of registers to be read.
xx yy CRC

Response 00 04 0A 01 5E 00 CD 00 F9 00 2E 00 4B xx yy

00 backdoor address or Cyber address
04 function code
0A number of Bytes
01 5E Value inside register 0x1000 (RAW DATA)
00 CD Value inside register 0x1001 (Vcc INPUT)
00 F9 Sensor value in point gas
00 2E Value inside register 0x1003 (Error (high byte) / Status (low byte))
00 4B INTERNAL USE ONLY
xx yy CRC

5.4 How to Read Holding Registers (command 0x03)

This function code is used to read the contents of a contiguous block of holding registers in a remote device. The Request PDU specifies the starting register address and the number of registers. In the PDU Registers are addressed starting at zero. Therefore, registers numbered 1-16 are addressed as 0-15.
The register data in the response message are packed as two bytes per register, with the binary contents right justified within each byte. For each register, the first byte contains the high order bits and the second contains the low order bits.

Request

Function code	1 Byte	**0x03**
Starting Address	2 Bytes	0x0000 to 0xFFFF
Quantity of Registers	2 Bytes	1 to 125 (0x7D)

Response

Function code	1 Byte	**0x03**
Byte count	1 Byte	2 x **N***
Register value	**N*** x 2 Bytes	

*N = Quantity of Registers

Error

Error code	1 Byte	**0x83**
Exception code	1 Byte	01 or 02 or 03 or 04

Example of Read Holding registers.

Request 00 03 20 00 00 1A xx yy

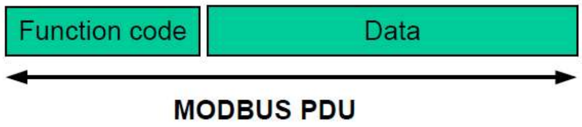
00 backdoor address or Cyber address
03 function code
20 00 First register to be read.
00 1A number of registers to be read.
xx yy CRC

Response 00 03 34 03 D4 00 2A 00 54 01 25 00 08 10 00 16 98 00 00 00 0D 00 54 46 1F 05 0F 00 0D 66 4F 00 42 00 54 46 1F 05 0F 00 0D 00 0D 01 24 00 26 00 5C 00 01 70 70 03 00 xx yy

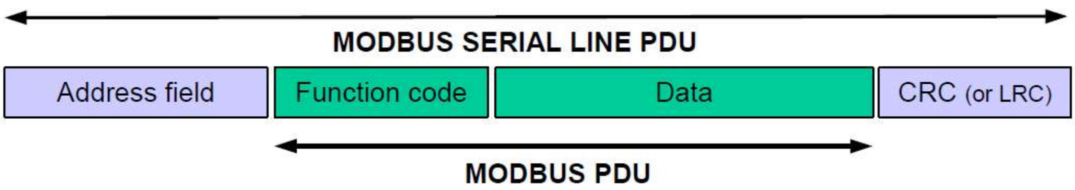
00 backdoor address or Cyber address
03 function code
34 number of Bytes
03 D4 Full scale (reg 0x2000, value in ‘points’)
00 2A Threhsold1 (value in ‘points’)
00 54 Threhsold2 (value in ‘points’)
01 25 Threhsold3 (value in ‘points’)
00 08 Hysteresis
10 00 INTERNAL USE ONLY
16 98 Kappa
0-0 00 Number of decimals -Threshold Parameters / INTERNAL USE ONLY (reg 0x2007)
00 0D Zero (reg 0x2008, value in ‘points’)
00 54 Serial number
46 1F Serial number
05 0F INTERNAL USE ONLY / INTERNAL USE ONLY
00 0D INTERNAL USE ONLY / INTERNAL USE ONLY
66 4F INTERNAL USE ONLY
00 42 INTERNAL USE ONLY / INTERNAL USE ONLY
00 54 INTERNAL USE ONLY / INTERNAL USE ONLY
46 1F INTERNAL USE ONLY / INTERNAL USE ONLY
05 0F INTERNAL USE ONLY / INTERNAL USE ONLY
00 0D INTERNAL USE ONLY / INTERNAL USE ONLY
00 0D INTERNAL USE ONLY
01 24 Gas type (high byte) / Flags (low byte)
00 26 INTERNAL USE ONLY
00 5C INTERNAL USE ONLY
00 01 INTERNAL USE ONLY / RS485 Address (low byte)
70 70 INTERNAL USE ONLY / INTERNAL USE ONLY
03 00 Sensor type (high byte) / Unit of measurement (low byte)
xx yy CRC

5.5 MODBUS frame description

The MODBUS over serial line V1.02 application protocol defines a simple Protocol Data Unit (PDU) independent of the underlying communication layers:



The mapping of MODBUS protocol on a specific bus or network introduces some additional fields on the PDU. The client that initiates a MODBUS transaction builds the MODBUS PDU, and then adds fields in order to build the appropriate communication PDU.



We can write the frame in order to read concentration register from cyber head:
ALL INPUT REGISTERS (0x1000) NEED TO BE READ AT THE SAME TIME, IS NOT POSSIBLE TO READ SINGLE INPUT REGISTER SELECTIVELY.

request:

address field	netid (RS485 ADDRESS)
function code	0x04H
starting address	0x1000
quantity of input registers	0x0005
crc	**CRC High byte, CRC low byte**

response:

address field	netid (RS485 ADDRESS)
functioncode	0x04H
byte count	0x0A
input registers	
Crc	**CRC High byte, CRC low byte**

at 0x1002 register you need to apply the following formula:

register 0x1002
• Signal's zero shift value "delta" in points (0÷418)

ALL INPUT REGISTERS (0x1000),

function code	0x04H
starting address	0x1000
quantity of input registers	0x0005

NEED TO BE READ AT THE SAME TIME, IS NOT POSSIBLE TO READ SINGLE INPUT REGISTER SELECTIVELY.

Modbus address	Type	R / W	Register	Description	Measurement unit	Quantity of registers
0x1000	Uint16 16bit	R	Raw sensor reading	Sensor value in point gas related to gross zero	--	1
0x1001	Uint16 16bit	R	Voltage input reading	Voltage input value	--	1
0x1002	--	R	Sensor Reading	Do not change	--	1
0x1003h 0x1003L	Uint16 16bit	R	Byteh errsts Bytel stsch1	error status	See description See description	1
0x1004	Uint16 16bit	R	Zero420 dac	Zero dac value		1

ALL INPUT REGISTERS (0x3000)

function code 0x04H
starting address 0x3000
quantity of input registers 0x0004

NEED TO BE READ AT THE SAME TIME, IS NOT POSSIBLE TO READ SINGLE INPUT REGISTER SELECTIVELY.

Modbus address	Type	R / W	Register	Description	Measurement unit	Quantity of registers
0x3000h 0x3000l	Uint16 16bit	R	Byteh Firmware Version Major bytel Firmware Version Revision	Firmware Version Major Firmware version minor Version Revision		1
0x3001 H 0x30011	Uint16 16bit	R	Byteh Firmware Build Date DD bytel Firmware Build Date MM	Firmware Build Date DD Firmware Build Date MM	--	1
0x3002	Uint16 16bit	R	Firmware Build Date YYYY	Firmware Build Date YYYY	--	1
0x3003	--	R	INTERNAL USE ONLY	Do not change	--	1