Technical Support
Bulletin Nr.18 – Modbus Tips

Contents

- *Definitions*
- *Implemented commands*
- *Examples of commands or frames*
- *Calculating the logical area*
- *Reading a signed variable*
- *Example of commands supported by Ech200B controllers*
- *Example of commands supported by Ert400B controllers*

Definitions

The section that follows provides a description of the main terms used in this document in order to enable readers to fully understand and follow the text of this bulletin.

Binary number: a binary number is a quantity expressed in a base 2 notation, which substantially means that the number is represented by using two characters only (0 , 1). This numerical system is generally used by processors that interpret numbers using a logical signal (on , off) [Fig1].

Decimal number: the decimal system is simply the metrical system. It expresses a quantity using 10 characters or a base 10 notation (0,1,2….8,9). This system is designed to simplify mathematical calculations.

Hexadecimal number: in this system quantities are expressed using 16 characters or a base 16 notation, which can be particularly useful to handle very large numbers. This system is generally used for programming applications. Its format is (0,1,2,….,9,A,B,C,D,E,F). Hexadecimal numbers are generally preceded by H (for example HFF) or 0x (for example OxFF).

Bit: is the abbreviation of binary digit and is used to define an amount of information. A bit represents the minimum amount that can be identified by a processor and a logical state (True\False; On\Off, etc) [Fig1].

Most/least significant bit: the most significant bit or MBS is the first item on the left of a number. It represents the item that has more "weight" in terms of absolute value. The least significant bit is instead the first bit on the right and represents the items that has less "weight" in terms of absolute value.

Byte: a byte represents the number of bits required to define an alphanumerical character. Each character is represented by a sequence of 8 bits [Fig1].

Nibble: this unit of measurement is used to express a group of 4 bits, that is half a byte. It is generally used to describe single hex digit. The values ranging from 0000 to 1111 can therefore be expressed using the range of units from H0 to HF [Fig1].

Word: unit of measurement with a length of 16 bits or 2 bytes. On controllers that use the ModBus-RTU protocol, every piece of information requested or transmitted is expressed with a length in words or multiples of this unit [Fig1].

Server: system that requests information or sends commands to the devices to which it is connected by means of a physical bus; for example a EIA\TIA-485-A network (also called RS485 network). Servers are also sometimes called masters or parent systems.

Client: device connected to the data bus that replies to specific requests received by the server. Clients are also sometimes called slaves or sibling systems.

Frame: a frame is a complete message exchanged by a server and client.

Broadcast: a broadcast message is not directed to a specific recipient but to all systems in listening mode. Broadcast messages do not require a reply.

Parity check: it is used to prevent transmission faults originating from transient noise. The parity parameter has to be defined both for the server and client. This method enables to add to each transmitted byte (8 bits) an additional parity bit. If an ODD parity (odd) condition occurs for each byte transmitted, an additional 1 o 0 bit is added so that the number of high logical values is high.
Example: let's suppose we wish to send byte 11011011, which contains an even number (6 bits) of high logical values (1). If we specify that the parity check is ODD, it is possible to add a bit to the high logical value so that the total increases from 6 to 7 bits with a high value.

Start/stop bit: it is generally sufficient to configure at least the stop bits because each server already defines 1 start bit. These two bits are used to specify the start and end of the data packet transmitted in serial asynchronous mode. This type of communication transmits the bits in sequence so that the recipient knows where the data packet starts and ends. Sending a bit means setting the serial line to a high logical value (1) or maintaining it at a low logical value (0) for a specific interval of time. This interval of time defines the transfer rate or baud rate of the serial line.
Example: a RS485 serial line with a baud rate of 9600bits\sec (that is typically used on all Eliwell controllers) is able to transfer 9,600 bits per second, which means that each bit has a typical length of approximately 0.1ms (1 sec\ 9600 bits).
The transfer mechanism can be defined as follows: the server sets the serial line to the high logical value for the interval of time that corresponds to one bit (start bit), then sends 8 bits that form the data byte, and ends the transfer with the parity bit. After this the serial line maintains the high setting for an additional interval of time of 1\2 bits (1 or 2 stop bits).

Time-out: another parameter that needs to be defined for servers is the timeout, which represents the interval of time during which the systems waits for a reply from the client before starting another task. Timeout varies according to the baud rate and the actual time required by the client to process the request and formulate a reply.
The data packet transfer time of a serial line with a baud rate of 9600bits\sec is:

$$(0.1 \text{ X } 11) \text{ X } 20 = 20\text{mS}$$

Where 0.1 is the time required to define a single bit, 11 is the typical number of bits that have to be transferred for each single byte (1 start bit + 8 data bits + 1 parity bit + 1 stop bit), and 20 is the number of bytes of a generic message, which could be larger depending on the amount of data requested/sent.
This means that the server requires 20 ms to send the message and almost the same or more time to send the reply.

Thus, timeout must always be set taking into account the following:

Time required to send the request + Time required to process the request + Time required to receive a reply + Waiting time

The timeout of Eliwell controllers is generally 200mS.

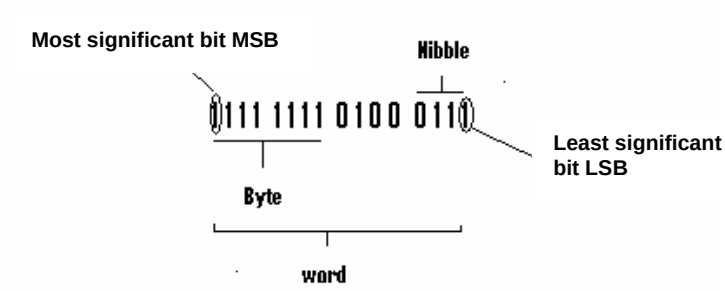
ModBus-RTU: in RTU transfer mode, data is exchanged in binary format, which means that each data byte contains a coded communication byte. The start of the message is preceded by a pause of 3.5 bits. When the client detects a pause of 1.5 bits, it assumes that a new message is being received and therefore deletes the reception buffer. This means that the message needs to be transmitted in continuous mode.

ModBus-ASCII:, in ASCII transfer mode, all messages are coded with hex values, which in turn represent a readable ASCII character. The characters used range from 0 to 9 and from A to F. 2 bytes of each data byte are used for transfer purposes because each character can only define 4 communication bits in the hex system. In this type of transfer mode, it is necessary to send specific characters to mark the start and end of the frame: ASCII character " **:** " is used as start character, while "Cr|LF" is used to end the message. The advantage of this method lies in the fact that it enables to extend the intervals between the transfer of single bytes to a maximum of 1 second.

Physical area: the RAM memory is divided into several sectors called areas that have a real index. However, users do not need this information to access single resources.

Logical area: the firmware manages the reallocation of the RAM's physical sectors in order to optimize the processing tasks. Logical addresses are used by users to access the desired resources.

(fig1) Example of binary number:



Implemented commands

Eliwell controllers only use some of the ModBus protocol commands, that is commands 3 and 16 expressed in decimal format or 0x03 and 0x10 in hexadecimal format.
Commands 3 and 16 have been implemented in all models of Eliwell controllers that use the ModBus-RTU protocol. Some controllers also support commands 43 and 4, or 0x2B and 0x04

Command 0x03: this command enables to read multiple consecutive registers (holding registers). The block defined by the protocol as holding register contains data that can be changed by the application program and that has a length of 16 bits, that is 1 word.

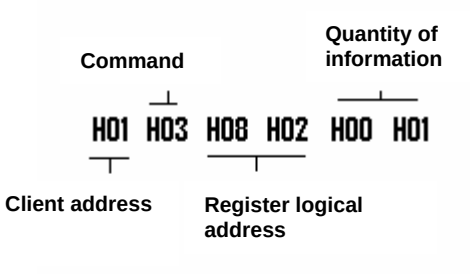
Command 0x04: this command enables to read multiple inputs provided by a I/O system. In systems in which memory is organized by blocks, resources are accessed through the use of different commands. All Eliwell controllers connected to a network operate as clients and do not support this command because all resources are allocated to a single memory block and managed as holding registers. The only controller that supports this command is XT Pro, because it is able to operate as client when connected to a network or as master, thus requesting information and sending commands.

Command 0x10: this command enables to perform writing operations on several and contiguous registers (holding registers). As in the case above, the length of data is equivalent to 16 bits or 1 word.

Command 0x2B/0x0E: this command enables to read the description or ID of the client device. More specifically, it enables to acquire information like the manufacturer ID and product code. The command is constituted by several subcommands. Some are mandatory for manufacturers who support command 43 (0x2B), while others are optional.
Mandatory subcommands are 0x00, 0x01,0x02. These return in sequence the manufacturer ID, product ID and version. Eliwell controllers that support this command return INVENSYS as manufacturer ID, Firmware mask as product ID and release as version.

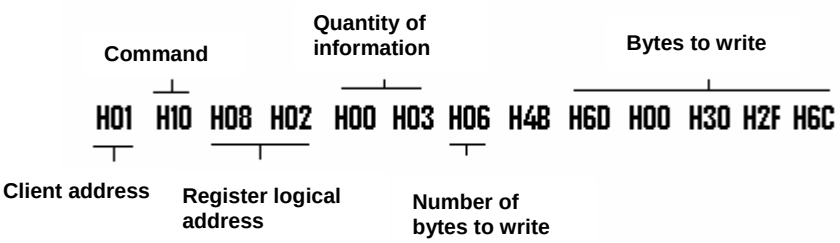
Examples of commands or frames
Let's take as example a read message send by the server to the client with address 1.
It is useful to remember that in the ModBus protocol, address 0 is allocated to the broadcasting message.

Example of a register read command (0x03):



The first byte defines the address of the device (H01), while the second one defines the read command (H03)
The third and fourth bytes define in sequence the high and low sections of the word that contains the address of the register that needs to be read.
The fifth and sixth bytes respectively define the high and low sections of the word that specifies the amount of information. If we start from logical address 0x08 0x02, we only need to read one register.
If the sixth byte had been 0x03 instead of 0x01 we would have had to read 3 contiguous addresses if we started with address 0x08 0x02.

Example of a register write command (0x10)



The first byte identifies the address of the client to which the message is addressed, as in the former case, while the second byte defines the command. The length of both is equivalent to one byte.
The third and fourth bytes (H08 H02) define the high and low sections of the word of the source register in which the command has to be written. In this case the register has a length of 2 bytes (word).
The fifth and sixth bytes indicate the number of consecutive registers in which the command has to be written. In this specific case, the writing operation is carried out in 3 registers (H00 H03).
The seventh byte (H06) indicates the number of bytes that follow and that will be written. This number is always twice the number that precedes it in the frame, because each register has a length of 2 bytes.

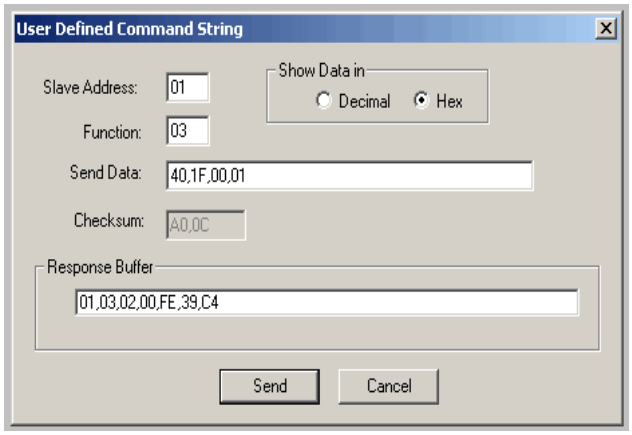
Calculating the logical area
Some manuals do not provide the logical addresses of the registers, but list the offset and logical address of the resource for each area.
On Ech200B controllers, for example, digital inputs are in the RAM area with index 8, which has a physical offset of 0x4F. The declared physical address of the digital inputs, which are grouped into a single word, is 0x63.
To obtain the logical address to read, it is necessary to follow the procedure outlined below.

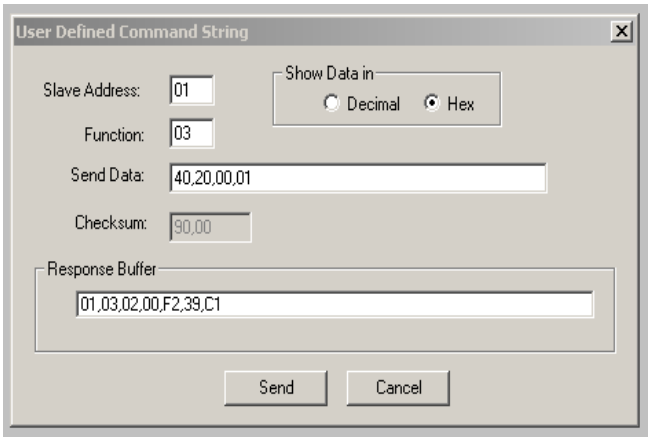
Considering that:
1. The register address is expressed by means of a word, that is 16 bits
2. The area index in the word is defined by using the 5 most significant bits
3. The logical address of the register is expressed using the remaining 11 bits

The area index in our case is 8 with is equivalent to binary number 01000 (5 bits).
The logical address is h63 – h4F, that is h14. The logical address in binary format is 00000010100.
Its complete format is 01000 00000010100, which is equivalent to hex value 0x 4014.
This means that the logical address of our register is h4014.

Reading a signed variable
Negative integers can be represented using the so-called two's complement format.
In this example we shall assume we want to represent an unsigned number with n bits. We know that n bits enable us to represent a decimal number ranging from 0 to 2^n -1.
Thus, 3 bits would enable us to represent a decimal number ranging from 0 to 7. To represent a negative number, we will therefore have to use the most significant bit to specify the sign. Consequently positive values can be expressed using the range from 0 to

[2 $^{n-1}$], while negative ones can be expressed in the range from [-2^{n-1}] to [-1]. Thus, the full range is from [2^{n-1}] to [2^{n-1} -1].
If we have 3 bits and the most significant one is reserved for the sign, it shall be possible to represent only decimal values ranging from -4 to 3.
When using the two's complement format, the most significant bit is generally referred to as sign bit.
Let's now look at a signed value like the one measured for probe St1. We know that its value is defined by a word, that is 16 bits. This means that integers can be represented in the range from -32768 to 32767.





In the example shown in the figure above, the value of probe St1 is 0x FEF2, which is equivalent to 1111111011110010.
The most significant bit corresponds to the upper value (1), which means that the value is negative.
If we convert the binary number with the one's complement method (NOT logical function) and add 1:

One's complement : 0000000100001101
+ 1

0000000100001110

The resulting absolute value in decimal format is 270. In this case we know that the value should be interpreted as a negative value. Therefore, the value of probe St1 is -27.0 °C.

The same result can be obtained by subtracting from value FFFF, that is 1111111111111111, FEF2 (see above) and adding 1.

1111111111111111
1111111011110010
+ 1

0000000100001110

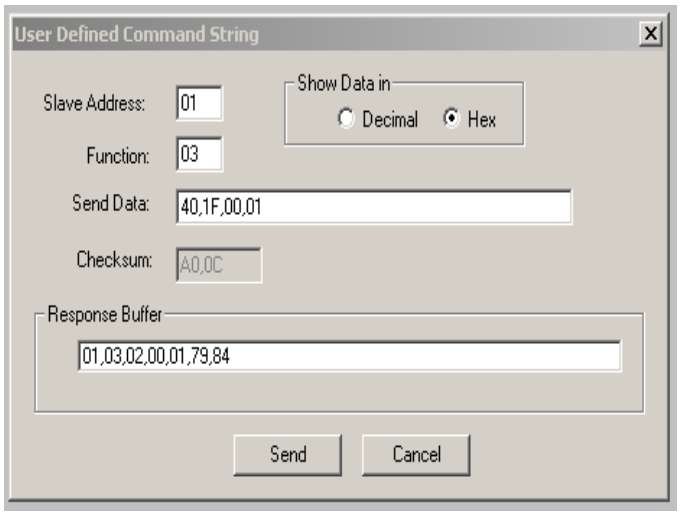
Thus, it is generally easier to use the following formula:

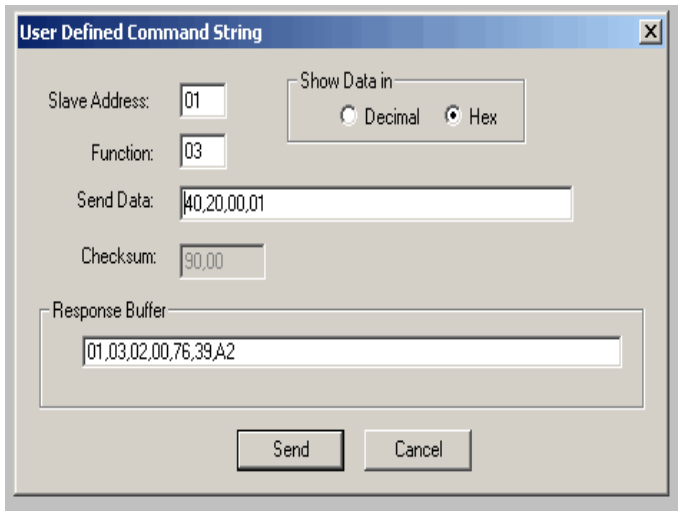
(FFFF- Value) +1

Examples of commands supported by Ech200B controllers

1. Reading a probe

The first example explains how to read regulation probe St1. On this controller the value of probes is divided into 2 words. We have deliberately chosen model Ech200B because it is the only Eliwell device that offers the possibility of reading probes with 2 words. The physical addresses of probe St1 are 6E and 6F: 6E contains the most significant byte while 6F contains the least significant one.

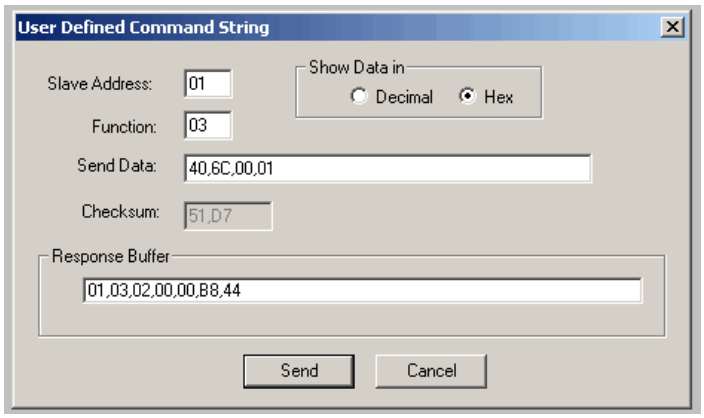


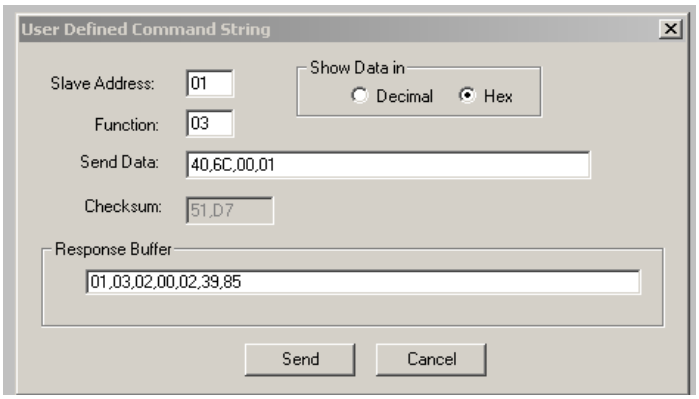


The controller replies with a word for logical address 0x4020. So, we need to take the low section and use it as least significant byte for the probe. For address 0x401F we instead use the low section as most significant byte. The result is a word with values 0x01 and 0x76, which means that the hex value of the probe is 0176 and that its decimal value is 374. Thus, the corresponding value in °C is 37.4 °C..

2. Reading an alarm

We shall now see how to read an alarm, for example the one related to a high digital pressure, through the tripping of the pressure switch.
The probe alarms are in the RAM, thus in area 8 .
Alarms, which can be active and inactive, are grouped into words and described by the status of a bit situated in the location referred to in the manual. The physical address of the high pressure alarm, for example, is 0xBB and its status is described by the status of the index bit 1 (bit_1).
The first operation we need to perform consists in finding the logical address, which in this case is 0x406C.

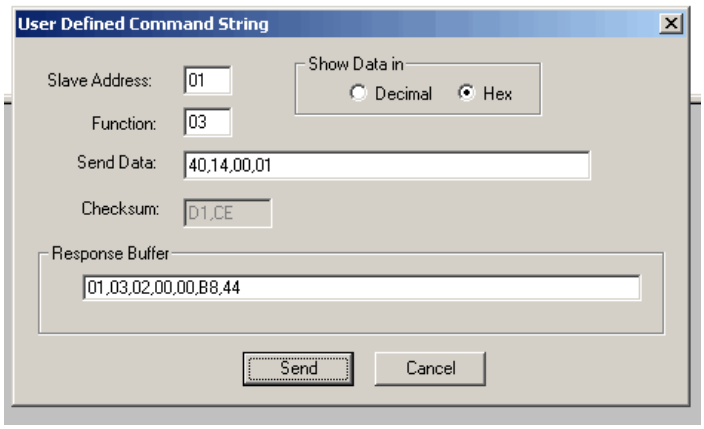


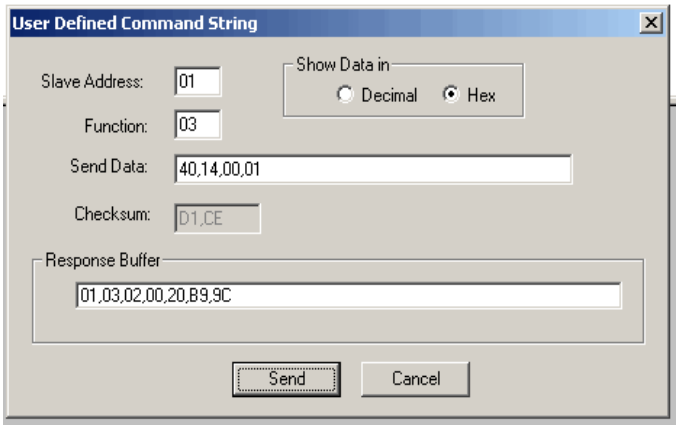


The image on the left shows that the address contains no alarm. The controller replies in fact with a word with value 0x 0000. In the image on the right, the regulator replies with a word with value 0x 0002.
In binary format, this word is equivalent to 0000 0000 0000 0010.
The bit with index 1 (bit_1) corresponds to a high logical value, which means that the high digital pressure alarm is active.

3. Reading the status of a digital input

The status of digital inputs is stored in the RAM (area 8). This area contains the status of the 5 available inputs along with the status of probes, if these have been configured as digital.
Now, let's look at digital input 5 with logical address 0x4014.





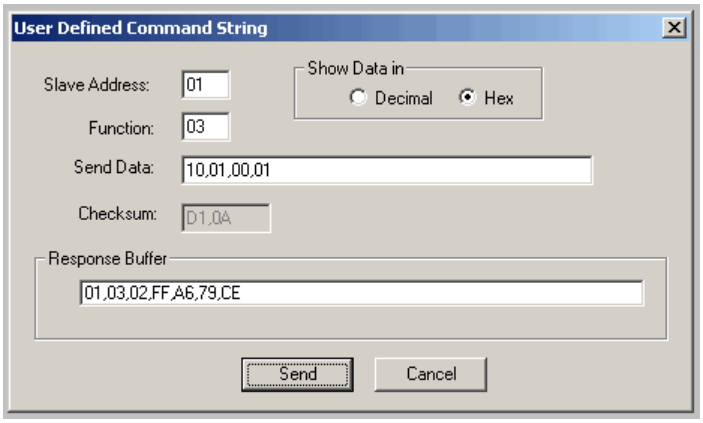
From the left image we see that the controller replies to the command with a word that has a value of 0x 0000, which means that the corresponding binary value is 0000 0000 0000 0000.
The status of digital input 5 can be found in index 5 (bit_5), as explained in the manual. In this case, the logical value of bit 5 is low, which means that the input is open.

In the example on the left, the device replies with a word that has a value of 0x 0020 that is equivalent to the binary value 0000 0000 0010 0000.

Examples of commands supported by Ert400B controllers

1. **Reading the probe**

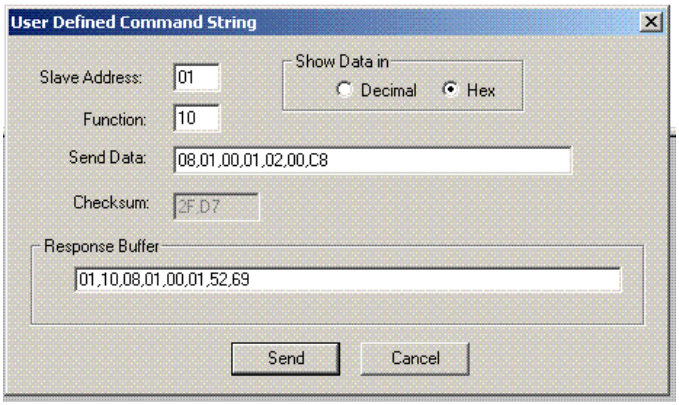
Let's now suppose we need to read the regulation probe and that, for the purposes of this example, its value is negative.
The ERT 400 manual provides the logical addresses of resources and in this case the area offset is irrelevant.
In the example above, the logical address of the regulation probe St1 is 4097, that is 0x10 01. The whole probe value set on this controller has the length of a word, but its bytes are contiguous.
This means that it is not necessary to read two words and group the least significant bytes.



The controller replies with value 0xFF A6 that is equivalent to decimal value 65446.
As explained above, to obtain the probe value, it is necessary to use the two's complement method, that is (FF FF – FF A6) +1 = 5A
The absolute value of the probe is 90 °C\10. Thus, the regulation temperature is -9.0°C.

2. **Writing a variable**

The example below explains how to write a variable using the set point. Let's suppose we wish to change the cooling set point and set it to 20.0°C.



The logical address of the Set cooling variable is 2049, that is 0x 08 01.
Thus, we need to use the write command 0x10.
The structure of the frame shall be the following:

01: client network address
10: write command
0801: logical area where the cooling set point is located
0001: number of variables that have to be written
02: number of bytes to write
00C8: hex value to write, that is 200 (20.0°C)

DISCLAIMER

This document is the exclusive property of Eliwell and may not be reproduced or circulated unless expressly authorized by Eliwell. Although Eliwell has done everything possible to guarantee the accuracy of this document, it declines any responsibility for damage arising from its use. The same applies to any person or company involved in preparing and writing this document.
Eliwell reserves the right to make changes or improvements at any time without notice.

Eliwell Controls s.r.l.
Via dell'Industria, 15 • Zona Industriale Paludi • 32010 Pieve d'Alpago (BL) ITALY
Telephone +39 0437 986 111 • Facsimile +39 0437 989 066
Sales +39 0437 986 100 (Italy) • +39 0437 986 200 (other countries)
Email: saleseliwell@invensyscontrols.com
Technical helpline +39 0437 986 300 • E-mail techsuppeliwell@invensyscontrols.com
www.eliwell.it