

**TERMOREGOLATORE PROGRAMMABILE  
PROGRAMMABLE TEMP. CONTROLLER****TM9x****MOD-BUS SERIAL COMMUNICATION****INTRODUCTION**

This technical note is intended to describe how serial communication of TM9x controllers works. We will describe the adopted protocol and the characters sequences to be used when the instrument shall be contacted. Within this note we will call "terminal" the intelligent apparatus connected to the controller and this could be for example a personal computer, a PLC or any other machine able to handle serial communications.

There are 2 types of available serial interfaces: V.24 (RS232C) and RS485 standards. Even if RS232C standard allows full-duplex transmission, communication needs a reply from the controller after having interpreted a request coming from the terminal; it is consequently clear that practically communication will be half-duplex.

RS485 interface allows to connect many controllers to the terminal, through a simple 2 wires connection. But, for this kind of interface the terminal shall correctly handle the switching transmission-reception because this RS 485 is in any case a half duplex channel. The delay between end of request and start of reply is about 15 ms.

**CONFIGURATION PARAMETERS**

Serial communication parameters on the controller are:

- Prt**    *OFF*: Serial interface disabled  
          *Mdb*: Serial interface enabled with ModBus RTU communication protocol

When the *Prt* parameter is set to *Mdb*, the following parameters will be also shown

- Ind**    *1 - 99*: Identification number of the controller
- bdr**    Transmission speed: *3, 6, 12, 24, 48, 96* respectively meaning 300, 600, 1200, 2400, 4800, 9600 baud
- AnS**    *LOC*: The keyboard of the controller is enabled while the terminal can only read data and parameters without modifying them (with the exception of parameters **AnS** and **kEy**)  
          *rEM*: The keyboard of the controller allows to see parameters without modifying them (with the exception of parameters **AnS** and **kEy**) while the terminal can read and modify all the parameters.

**HARDWARE COMMUNICATION PROTOCOL**

Transmission is a serial asynchronous half duplex transmission (the controller doesn't transmit while receiving and when it is transmitting, it doesn't receive data) with 1 start bit, 8 bits per character, no parity bit and 1 stop bit; communication speed can be set by changing a parameter of the controller and it can be 300, 600, 1200, 2400, 4800 or 9600 baud. The communication channel is RS232 standard or RS485 standard depending on the PCB installed in the controller.

## SOFTWARE COMMUNICATION PROTOCOL

MOD-BUS protocol Modicon standard has been chosen as a reference. Recognized MOD-BUS functions are nos. 3, 4 and 6 (Note: nos. 3 and 4 are interpreted as if they were the same command, i.e. *words reading*; no. 6 is the *single word writing* function).

There are two differences between the protocol implemented on TM9x controllers and the standard one:

- 1) there is no *broadcasting* function, that is it has been not previewed that a command sent to address 0 will be recognized by all controllers on the network, independently from their address, set with the **Ind** parameter;
- 2) if the terminal sends a function code different from the recognized ones (3, 4 and 6), the controller will send an *error reply* with error code 1.

All data are transmitted in binary mode.

START condition is recognized when the delay between two subsequent characters is longer than 1 T.U. (Time Unit = time requested to send 1 character).

A request sent by the terminal has always the following format:

- 1<sup>st</sup> byte: controller address (to select the required instrument among many others)
- 2<sup>nd</sup> byte: function code (3, 4 or 6)
- 3<sup>rd</sup> byte: more significant byte of the word to read or write
- 4<sup>th</sup> byte: less significant byte of the word to read or write
- 5<sup>th</sup> byte: reading: higher part of the number of words to be read; writing: higher part of the value to be written
- 6<sup>th</sup> byte: reading: lower part of the number of words to be read; writing: lower part of the value to be written
- 7<sup>th</sup> byte: lower part of CRC-16
- 8<sup>th</sup> byte: higher part of CRC-16.

CRC-16 bit calculation is made in compliance with MOD-BUS specifications.

Parameters value will be always expressed as 16 bits binary with sign: this means than we will have values between -32768 e +32767. We must be very careful with the high byte transmitted because if its value is greater than 127 (7Fhex), the final value of the parameter will be negative, equal to  $((\text{high byte} * 256) + \text{low byte}) - 65536$ .

### **'3' o '4' Functions : words reading**

The reply of the controller to the reading function will be a frame of  $(5 + 2N)$  bytes, where N is the number of requested words and whose meaning is:

- 1<sup>st</sup> byte: controller address
- 2<sup>nd</sup> byte: function code (3, 4 or 6)
- 3<sup>rd</sup> byte: number of bytes of transmitted data (equal to  $2N$ , where N is the number of requested words)
- 4<sup>th</sup>, 5<sup>th</sup>: value of the first requested word (high and low bytes respectively)
- 6<sup>th</sup>, 7<sup>th</sup>: valore della seconda word richiesta
- ...
- $(2+2N)$ <sup>th</sup>,  $(3+2N)$ <sup>th</sup>: value of the N<sup>th</sup> requested word
- $(4+2N)$ <sup>th</sup>,  $(5+2N)$ <sup>th</sup>: CRC-16 (low and high byte respectively)

**'6' Function: single word writing**

The reply of the controller to the writing function will be a frame of 8 bytes:

- 1<sup>st</sup> byte: controller address
- 2<sup>nd</sup> byte: function code (3, 4 or 6)
- 3<sup>rd</sup> , 4<sup>th</sup> byte: word address to be written (high and low bytes respectively)
- 5<sup>th</sup> , 6<sup>th</sup> byte: written value (high and low bytes respectively)
- 7<sup>th</sup> , 8<sup>th</sup> byte: CRC-16 (low and high byte respectively)

**Error reply**

If the command cannot be completed successfully a frame of 5 bytes is sent back to the terminal stating the type of error that happened:

- 1<sup>st</sup> byte: controller address
- 2<sup>nd</sup> byte: function code with the more significant bit set (function code + 80hex)
- 3<sup>rd</sup> byte: error code
- 4<sup>th</sup> , 5<sup>th</sup> byte: CRC-16 (low and high byte respectively)

**ERROR CODES**

Value Error

- 1 Not recognized function code
- 2 Illegal address
- 3 Illegal value
- 9 Illegal number of requested data
- 10 Write-protected data

**READING / WRITING LOCATIONS**

| Address |   | Reading | Writing |
|---------|---|---------|---------|
| 0x0001  | Offset *  | Yes     | Yes     |
| 0x0002  | Key lock (0=OFF , 1=Lo , 2=Hi)                  | Yes     | Yes     |
| 0x0100  | Heating proportional band *                     | Yes     | Yes     |
| 0x0101  | Heating derivative time (or upper hysteresis) * | Yes     | Yes     |
| 0x0102  | Heating integral time (or lower hysteresis) *   | Yes     | Yes     |
| 0x0103  | Heating cycle time *                            | Yes     | Yes     |
| 0x0300  | MAIN Set point *                                | Yes     | Yes     |
| 0x1000  | Process value *                                 | Yes     | No      |

\* = as for limits please refer to the programming manual

Please remember that writing is possible only in accordance with "kEy" keylock and "AnS" serial mode parameters. Particularly if "AnS" is set to "rEM" then parameters can be set through serial communication but not through keyboard. On the contrary, when in "LoC" mode, parameters can be adjusted through keyboard but not through serial communication.

### Reading example

Let's see how to read the 4 parameters of the heating control:

- proportional band [100%],
- derivative time [10'],
- integral time [4'],
- cycle time [10'']

of the controller with address 04.

The starting address of the readings is 0x0100, words to be read are 4, bytes sequence to be sent to the controller is:

```
id    com  < address > < words > < CRC  >
0x04 0x03 0x01 0x00 0x00 0x04 0x45 0xA0
```

The reply of the controller will be:

```
id    com  bytes < P.B.  > < der t > < int t > < cycle t > < CRC  >
0x04 0x03 0x08 0x00 0x64 0x00 0x0A 0x00 0x04 0x00 0x0A 0xF8 0x1A
```

So, for this example, proportional band is set to 100 (0x0064), derivative time is 10 (0x000A), integral time is 4 (0x0004) and cycle time is 10 (0x000A).

### Writing example

The only writing command is "0x06", that is a writing command for a single word. It is not possible to write more than one parameter with one only command.

Let's see how to write the Main set point with a value of 150 on the same controller of the previous example (id = 4). The bytes sequence to be sent to the controller is:

```
id    com  < address > < data  > < CRC  >
0x04 0x06 0x03 0x00 0x00 0x96 0x09 0xB5
```

The reply of the controller will be:

```
id    com  < address > < data  > < CRC  >
0x04 0x06 0x03 0x00 0x00 0x96 0x09 0xB5
```



**THERMOSYSTEMS s.r.l.**

phone: (+39) 0363 350159 fax: (+39) 0363 350362

Via delle Industrie, 8 - 24040 Fornovo San Giovanni (BG) - ITALY

web: [www.thermosystems.it](http://www.thermosystems.it)

e-mail: [info@thermosystems.it](mailto:info@thermosystems.it)

