



## CTT TEMPERATURE MONITOR DEVICE COMMUNICATION PROTOCOL

### CTT MODBUS-RTU COMMUNICATION PROTOCOL

#### MODBUS PROTOCOL

Modbus is a master-slave communication protocol able to support up to 247 slaves organized as a bus or as a star network;

The physical link layer can be RS232 for a point to point connection or RS485 for a network.

The communication is half-duplex.

The network messages can be Query-Response or Broadcast type.

The Query-Response command is transmitted from the Master to an established Slave and generally it is followed by an answering message.

The Broadcast command is transmitted from the Master to all Slaves and is never followed by an answer.

#### **MODBUS use two modes for transmission.**

**A) ASCII Mode:** uses a limited character set as a whole for the communication.

**B) RTU Mode:** binary, with time frame synchronization, faster than the ASCII Mode, uses half so long data block than the ASCII Mode.

**CTT temperature monitor device employ RTU mode.**

#### **GENERIC MESSAGE STRUCTURE:**

| START OF FRAME | ADDRESS FIELD | FUNCTION CODE | DATA FIELD | ERROR CHECK | END OF FRAME |
|----------------|---------------|---------------|------------|-------------|--------------|
|----------------|---------------|---------------|------------|-------------|--------------|

START OF FRAME = Starting message marker

ADDRESS FIELD = Includes device address in which you need to communicate in Query-Response mode. In case the message is a Broadcast type it includes 00.

FUNCTION CODE = Includes the operation code that you need to perform.

DATA FIELD = Includes the data field.

ERROR CHECK = Field for the error correction code.

END OF FRAME = End message marker.

#### **Mode RTU communication frame structure:**

START OF FRAME = silence on line for time  $\geq 4$  characters

ADDRESS FIELD = 1 character

FUNCTION CODE = 1 character

DATA FIELD = N characters

ERROR CHECK = 16 bit CRC

END OF FRAME = silence on line for time  $\geq 4$  characters

#### **Wait time for response :**

- typical : 15 mS

- worst case : 20 mS.



### **READING OF THE REGISTERS (Function Code \$ 03)**

Reads the binary contents of holding registers ( 2X references) in the slave.

Broadcast is not supported.

The Query message specified the starting register and quantity of register to be read.

#### **QUERY:**

| START OF FRAME | ADDRESS FIELD | FUNCTION CODE | START ADDRESS | No. OF REGISTERS | ERROR CHECK | END OF FRAME |
|----------------|---------------|---------------|---------------|------------------|-------------|--------------|
|----------------|---------------|---------------|---------------|------------------|-------------|--------------|

START OF FRAME = Starting message marker.  
ADDRESS FIELD = CTT device address (01...F7 HEX) (1 byte).  
FUNCTION CODE = Operation code ( 03 HEX) (1 byte).  
START ADDRESS = First register address to be read (2 byte).  
No. OF REGISTERS = Number of registers ( max 32) to be read (4 byte for 1 measure value).  
ERROR CHECK = Check sum.  
END OF FRAME = End message marker.

#### **WARNING:**

It is possible to read more than one variable at the same time (max 16) only if their addresses are consecutive and the variables on the same line cannot be divided.

The register data in the response message are packet 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.

#### **RESPONSE:**

| START OF FRAME | ADDRESS FIELD | FUNCTION CODE | No. OF BYTES | D0, D1, ..., Dn | ERROR CHECK | END OF FRAME |
|----------------|---------------|---------------|--------------|-----------------|-------------|--------------|
|----------------|---------------|---------------|--------------|-----------------|-------------|--------------|

START OF FRAME = Starting message marker.  
ADDRESS FIELD = CTT device address (01...F7 HEX) (1byte).  
FUNCTION CODE = Operation code ( 03 HEX) (1 Byte).  
No. OF SEND BYTES= Number of data bytes ( 00...?? HEX) (1 byte). 1 register requires 2 data bytes.  
D0, D1, .., Dn = data bytes ( 00...?? HEX) (Nr. of register x 2 = n. byte).  
ERROR CHECK = Check sum.  
END OF FRAME = End message marker.

See the TABLE OF CTT REGISTERS.

## **SETUP OF THE CTT PARAMETERS (Function Code \$ 10)**

Write values into a sequence of holding registers (2X references).

**WARNING:** It is possible to write more than one variable at the same time only if their addresses are consecutive and the variables on the same line cannot be divided (max of 4 consecutive register on the same message).

### **QUERY:**

| START OF FRAME | ADDRESS FIELD | FUNCTION CODE | START ADDRESS | No. OF REGISTERS | No. OF BYTES | D0, D1, ..., Dn | ERROR CHECK | END OF FRAME |
|----------------|---------------|---------------|---------------|------------------|--------------|-----------------|-------------|--------------|
|----------------|---------------|---------------|---------------|------------------|--------------|-----------------|-------------|--------------|

|                 |   |   |  |  |  |  |  |  |
|-----------------|---|---|--|--|--|--|--|--|
| START OF FRAME  | = | Starting message marker.                        |  |  |  |  |  |  |
| ADDRESS FIELD   | = | CTT device address (01...F7 HEX)                |  |  | (1 byte).                                  |  |  |  |
| FUNCTION CODE   | = | Operation code ( 10 HEX)                        |  |  | (1 byte).                                  |  |  |  |
| START ADDRESS   | = | First register address to be written            |  |  | (2 byte).                                  |  |  |  |
| No. OF REGISTER | = | Number of registers to be written ( 1 to 4,...) |  |  | (2 byte).                                  |  |  |  |
| No. OF BYTES    | = | Number of data bytes (HEX)                      |  |  | (1 byte): 1register requires 2 data bytes. |  |  |  |
| D0,D1,...,Dn    | = | Data bytes ( 00...? HEX)                        |  |  | (1 byte) (Nr. of register x 2 = n. byte).  |  |  |  |
| ERROR CHECK     | = | Check sum.                                      |  |  |  |  |  |  |
| END OF FRAME    | = | End message marker.                             |  |  |  |  |  |  |

The normal response returns the slave address, function code, starting address and quantity of register preset.

### **RESPONSE:**

| START OF FRAME | ADDRESS FIELD | FUNCTION CODE | START ADDRESS | No. OF REGISTERS | ERROR CHECK | END OF FRAME |
|----------------|---------------|---------------|---------------|------------------|-------------|--------------|
|----------------|---------------|---------------|---------------|------------------|-------------|--------------|

|                 |   |                                      |  |  |           |  |
|-----------------|---|--------------------------------------|--|--|-----------|--|
| START OF FRAME  | = | Starting message marker.             |  |  |           |  |
| ADDRESS FIELD   | = | CTT device address (01...F7 HEX)     |  |  | (1 byte). |  |
| FUNCTION CODE   | = | Operation code ( 10 HEX)             |  |  | (1 byte). |  |
| START ADDRESS   | = | First register address to be written |  |  | (2 byte). |  |
| No. OF REGISTER | = | Number of registers to be written    |  |  | (2 byte). |  |
| ERROR CHECK     | = | Check sum.                           |  |  |           |  |
| END OF FRAME    | = | End message marker.                  |  |  |           |  |

See the TABLE OF CTT REGISTERS.

## **DIAGNOSTIC (Function Code \$ 08)**

This function provides a test for checking the communication system.

Broadcast is not supported.

The instrument's protocol has only the sub-function 0 of the diagnostics sub-functions set of the standard modbus protocol.

The Query and the Response messages are the following:

### **QUERY:**

| START OF FRAME | ADDRESS FIELD | FUNCTION CODE | SUB FUNCTION | DATA | ERROR CHECK | END OF FRAME |
|----------------|---------------|---------------|--------------|------|-------------|--------------|
|----------------|---------------|---------------|--------------|------|-------------|--------------|

START OF FRAME = Starting message marker.  
ADDRESS FIELD = CTT device address (01...F7 HEX) (1 byte).  
FUNCTION CODE = Operation code ( 08 HEX) (1 byte).  
SUB FUNCTION = Sub-function 0 (00 00 hex) (2 byte).  
DATA = Max 10 data bytes.  
ERROR CHECK = Check sum.  
END OF FRAME = End message marker.

### **RESPONSE:**

The response must be the loopback of the same data.

| START OF FRAME | ADDRESS FIELD | FUNCTION CODE | SUB FUNCTION | DATA | ERROR CHECK | END OF FRAME |
|----------------|---------------|---------------|--------------|------|-------------|--------------|
|----------------|---------------|---------------|--------------|------|-------------|--------------|

START OF FRAME = Starting message marker.  
ADDRESS FIELD = CTT device address (01...F7 HEX) (1 byte).  
FUNCTION CODE = Operation code ( 08 HEX) (1 byte).  
SUB FUNCTION = Sub-function 0 (00 00 hex) (2 byte).  
DATA = Data bytes.  
ERROR CHECK = Check sum.  
END OF FRAME = End message marker.

### **DIAGNOSTIC EXAMPLE**

#### **QUERY**

| Field Name        | Example ( Hex) |
|-------------------|----------------|
| Slave Address     | 01             |
| Function Code     | 08             |
| Sub-function Hi   | 00             |
| Sub-function Lo   | 00             |
| Data Hi           | F1             |
| Data Lo           | A7             |
| Error Check (CRC) | ??<br>??       |

#### **RESPONSE**

| Field Name        | Example ( Hex) |
|-------------------|----------------|
| Slave Address     | 01             |
| Function Code     | 08             |
| Sub-function Hi   | 00             |
| Sub-function Lo   | 00             |
| Data Hi           | F1             |
| Data Lo           | A7             |
| Error Check (CRC) | ??<br>??       |

## **REPORT SLAVE ID (Function Code \$ 11)**

This function returns the type of the instrument and the current status of the slave run indicator. Broadcast is not supported. The Query and the Response messages are the following:

### **QUERY:**

| START OF FRAME | ADDRESS FIELD | FUNCTION CODE | ERROR CHECK | END OF FRAME |
|----------------|---------------|---------------|-------------|--------------|
|----------------|---------------|---------------|-------------|--------------|

START OF FRAME = Starting message marker.  
ADDRESS FIELD = CTT device address (01...F7 HEX) (1 byte).  
FUNCTION CODE = Operation code ( 11 HEX) (1 byte).  
ERROR CHECK = Check sum.  
END OF FRAME = End message marker.

### **RESPONSE:**

| START OF FRAME | ADDRESS FIELD | FUNCTION CODE | BYTE COUNT | SLAVE ID | RUN INDICATOR STATUS | ERROR CHECK | END OF FRAME |
|----------------|---------------|---------------|------------|----------|----------------------|-------------|--------------|
|----------------|---------------|---------------|------------|----------|----------------------|-------------|--------------|

START OF FRAME = Starting message marker.  
ADDRESS FIELD = CTT device address (01...F7 HEX) (1 byte).  
FUNCTION CODE = Operation code ( 11 HEX) (1 byte).  
BYTE COUNT = Number of data bytes (18 HEX) (1 byte).  
SLAVE ID = Slave ID identifier (54 HEX) (1 byte).  
RUN INDICATOR STATUS = Run indicator status (FF HEX) (1 byte).  
DATA = Data bytes.  
ERROR CHECK = Check sum.  
END OF FRAME = End message marker.

The normal response has the slave ID identifier (54 HEX) and the run indicator status (FF HEX) plus 10 data bytes (byte count is 12, 0C Hex). Last two data bytes carry firmware version (bytes 11 ,12 ).

## **ERROR MESSAGE FROM SLAVE TO MASTER**

When a slave device receives a not valid query, it does transmit an error message.

### **RESPONSE:**

| START OF FRAME | ADDRESS FIELD | FUNCTION CODE | ERROR CODE | ERROR CHECK | END OF FRAME |
|----------------|---------------|---------------|------------|-------------|--------------|
|----------------|---------------|---------------|------------|-------------|--------------|

START OF FRAME = Starting message marker.  
ADDRESS FIELD = CTT device address (01...F7 HEX) (1 byte ).  
FUNCTION CODE = Operation code with bit 7 high (1 byte).  
ERROR CODE = Message containing communication failure (1 byte).  
ERROR CHECK = Check sum.  
END OF FRAME = End message marker.

### **ERROR EXAMPLE QUERY**

| Field Name          | Example ( Hex) |
|---------------------|----------------|
| Slave Address       | 01             |
| Function Code       | 03             |
| Starting Address Hi | 00             |
| Starting Address Lo | 00             |
| Number Of Word Hi   | 00             |
| Number Of Word Lo   | 05             |
| Error Check (CRC)   | ??             |
|                     | ??             |

### **RESPONSE**

| Field Name        | Example ( Hex) |
|-------------------|----------------|
| Slave Address     | 01             |
| Function Code     | 83 (1)         |
| Error Code        | 02 (2)         |
| Error Check (CRC) | ??             |
|                   | ??             |

(1):Function Code transmitted by master with bit 7 high.  
(2): Error type:  
01= Illegal Function  
02= Illegal data address  
03= Illegal data value

## **TABLE OF CTT REGISTERS**

Decimal value is calculated as follow

Dec. Value = (MSB Byte \*256) + LSB Byte

### **Instantaneous temperature register (read only)**

- CTT8: from 0x0258 (CH1) to 0x25F (CH8), 8 register 2-byte wide, integer value of Celsius degree

- CTT4: from 0x0258 (CH1) to 0x25B (CH4), 4 register 2-byte wide, integer value of Celsius degree

NOTE: These register return also the coded status of the probe input; a valid value of temperature is added the integer 0x0019; follow explaining example

- Value 0x0000 Input is shorted, display of that channel mean SHR

- Value 0x0001 Input is open; display of that channel mean OPE

- Value 0x0019 Input is right connect to PT100; measured temperature is 0 °C

### **Peak temperature register (read only)**

- CTT8: from 0x0260 (CH1) to 0x267 (CH8), 8 register 2-byte wide, integer value of Celsius degree

- CTT4: from 0x0260 (CH1) to 0x263 (CH4), 4 register 2-byte wide, integer value of Celsius degree

### **Alarm temperature register (read / write)**

- CTT8: from 0x0300 (CH1) to 0x307 (CH8), 8 register 2-byte wide, integer value of Celsius degree

- CTT4: from 0x0300 (CH1) to 0x303 (CH4), 4 register 2-byte wide, integer value of Celsius degree

### **Trip temperature register (read / write)**

- CTT8: from 0x0310 (CH1) to 0x317 (CH8), 8 register 2-byte wide, integer value of Celsius degree

- CTT4: from 0x0310 (CH1) to 0x313 (CH4), 4 register 2-byte wide, integer value of Celsius degree

### **CTT4 only read / write register**

Fan OFF temperature: 0x0272, 1 register 2-byte wide, integer value of Celsius degree

Fan ON temperature: 0x0273, 1 register 2-byte wide, integer value of Celsius degree

### **CTT4 / CTT8: register to reset peak temperature register 0x27F**

2 byte wide; write the 0xA55A value in these register to reset the peak temperature registers; only the value 0xA55A is accepted; other value are discarded;

### **Status register**

Trip signalling led status: MSByte of register 0x270 (read only)

- CTT8: from bit 0 (CH1) to bit 7 (CH8): 0 = led OFF; 1 = led ON

- CTT4: from bit 0 (CH1) to bit 3 (CH4): 0 = led OFF; 1 = led ON

Alarm signalling led status: LSByte of register 0x270 (read only)

- CTT8: from bit 0 (CH1) to bit 7 (CH8): 0 = led OFF; 1 = led ON

- CTT4: from bit 0 (CH1) to bit 3 (CH4): 0 = led OFF; 1 = led ON

### **Relay and HOLD status: MSByte of register 0x271 (read only)**

- bit 0 FAULT relay (active LOW 0 = relay energized)

- bit 1 FAN relay (active HIGH 1 = relay energized) NOTE: is valid ONLY on CTT4 model

- bit 2 ALARM relay (active HIGH 1 = relay energized)

- bit 3 TRIP relay (active HIGH 1 = relay energized)

- bit 4 unused

- bit 5 HOLD MODE (active LOW 0 = HOLD active)

- bit 6 unused

- bit 7 unused

LSB byte : unused

### **FAN status: register 0x274 (read / write) (only CTT4 model)**

LSByte: FAN mode

MSByte: Number of ACTIVE channel

- value 0x00: active on CH1-3 (display mean fan on)

- value 0x00: 3 channel (display mean ch 3)

- value 0x01: disabled (display mean fan off)

- value 0x01: 4 channel (display mean ch 4)

- value 0x02: only on CH 4 (display mean fan 4)

## **TROUBLESHOOTING**

If response from CTT doesn't happen:

- check connection from CTT and RS232/RS485 converter ;

- check if data outgoing from the RS232 serial port of the PC come in the RS232/485 converter

- try to increase the wait time for response ( 30 mS is good);

- check if the transmitted data stream is **EXACTLY** as in example, monitoring the data on the RS485 serial line with a terminal ( eg. Hyperterminal or other emulator);

- if the RS232/485 converter is not our model CUS, be sure the turnaround-time is set in range 1 to 2 mS



---

I-26900 Lodi - ITALY - via S. Fereolo, 9  
Tel. ++39 0371 30207/30761 Fax. ++39 0371 32819 E-mail: [control@control.it](mailto:control@control.it)  
<http://www.control.it> - <http://www.control.net>