CTT MODBUS-RTU COMMUNICATION PROTOCOL

TEMPERATURE MONITOR DEVICE

Firmware version: v3.0 or higher

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 >=4 characters
ADDRES FIELD = 1 character
FUNCTION CODE = 1 character
DATA FIELD = N characters
ERROR CHECK = 16 bit CRC
END OF FRAME = silence on line for time >=4 characters

Wait time for response:
- typical : 15 mS
- worst case : 20 mS.

CRC GENERATION

Example of the CRC-16 generation with "C" language:

```
static unsigned char auchCRCHi [ ] = {
0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81,
0x40, 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0,
0x80, 0x41, 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40, 0x00, 0xC1, 0x81, 0x40, 0x01,
0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x01, 0xC0, 0x80, 0x41,
0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40, 0x00, 0xC1, 0x81,
0x40, 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x01, 0xC0,
0x80, 0x41, 0x00, 0xC1, 0x81, 0x40, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x01,
0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40,
0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81,
0x40, 0x01, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0,
0x80, 0x41, 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40, 0x00, 0xC1, 0x81, 0x40, 0x01,
0xC0, 0x80, 0x41, 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41,
0x00, 0xC1, 0x81, 0x40, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81,
0x40, 0x01, 0xC0, 0x80, 0x41, 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1 0x81, 0x40, 0x01, 0xC0,
0x80, 0x41, 0x00, 0xC1, 0x81, 0x40, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x01,
0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41,
0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81,
0x40
} ;

static unsigned char auchCRCLo [ ] = {
0x00, 0xC0, 0xC1, 0x01, 0xC3, 0x03, 0x02, 0xC2, 0xC6, 0x06, 0x07, 0xC7, 0x05, 0xC5, 0xC4,
0x04, 0xCC, 0x0C, 0x0D, 0xCD, 0x0F, 0xCF, 0xCE, 0x0E, 0x0A, 0xCA, 0xCB, 0x0B, 0xC9, 0x09,
0x08, 0xC8, 0xD8, 0x18, 0x19, 0xD9, 0x1B, 0xDB, 0xDA, 0x1A, 0x1E, 0xDE, 0xDF, 0x1F, 0xDD,
0x1D, 0x1C, 0xDC, 0x14, 0xD4, 0xD5, 0x15, 0xD7, 0x17, 0x16, 0xD6, 0xD2, 0x12, 0x13, 0xD3,
0x11, 0xD1, 0xD0, 0x10, 0xF0, 0x30, 0x31, 0xF1, 0x33, 0xF3, 0xF2, 0x32, 0x36, 0xF6, 0xF7,
0x37, 0xF5, 0x35, 0x34, 0xF4, 0x3C, 0xFC, 0xFD, 0x3D, 0xFF, 0x3F, 0x3E, 0xFE, 0xFA, 0x3A,
0x3B, 0xFB, 0x39, 0xF9, 0xF8, 0x38, 0x28, 0xE8, 0xE9, 0x29, 0xEB, 0x2B, 0x2A 0xEA, 0xEE,
0x2E, 0x2F, 0xEF, 0x2D, 0xED, 0xEC, 0X2C, 0xE4, 0x24, 0x25, 0xE5, 0x27, 0xE7, 0xE6, 0x26,
0x22, 0xE2, 0xE3, 0x23, 0xE1, 0x21, 0x20, 0xE0, 0xA0, 0x60, 0x61, 0xA1, 0x63, 0xA3, 0xA2,
0x62, 0x66, 0xA6, 0xA7, 0x67, 0xA5, 0x65, 0x64, 0xA4, 0x6C, 0xAC, 0xAD, 0x6D, 0xAF, 0x6F,
0x6E, 0xAE, 0xAA, 0x6A, 0x6B, 0xAB, 0x69, 0xA9, 0xA8, 0x68, 0x78, 0xB8, 0xB9, 0x79, 0xBB,
0x7B, 0x7A, 0xBA, 0xBE, 0x7E, 0x7F, 0xBF, 0x7D, 0xBD, 0xBC, 0x7C, 0xB4, 0x74, 0x75, 0xB5,
0x77, 0xB7, 0xB6, 0x76, 0x72, 0xB2, 0xB3, 0x73, 0xB1, 0x71, 0x70, 0xB0, 0x50, 0x90, 0x91,
0x51, 0x93, 0x53, 0x52, 0x92, 0x96, 0x56, 0x57, 0x97, 0x55, 0x95, 0x94, 0x54, 0x9C, 0x5C,
0x5D, 0x9D, 0x5F, 0x9F, 0x9E, 0x5E, 0x5A, 0x9A, 0x9B, 0x5B, 0x99, 0x59, 0x58, 0x98, 0x88,
0x48, 0x49, 0x89, 0x4B, 0x8B, 0x8A, 0x4A, 0x4E, 0x8E, 0x8F, 0x4F, 0x8D, 0x4D, 0x4C, 0x8C,
0x44, 0x84, 0x85, 0x45, 0x87, 0x47, 0x46, 0x86, 0x82, 0x42, 0x43, 0x83, 0x41, 0x81, 0x80,
0x40
}
unsigned short CRC16 (ptMsg, usDataLen)
unsigned char *ptMsg;                    / * message to calculate CRC upon * /
unsigned short usDataLen;                  / * number of bytes in message * /
{
    unsigned char uchCRCHi = 0xFF;          / * CRC high byte * /
    unsigned char uchCRCLo =0xFF;           / * CRC low byte * /
    unsigned uIndex;

    while (usDataLen--)                     / * pass through message buffer * /
    {
       uIndex = uchCRCHi ^ *ptMsg++;        /.* calculate the CRC * /
       uchCRCHi = uchCRCLo ^ auchCRCHi [ uIndex ] ;
       uchCRCLo = auchCRCLo [ uIndex ]
    }
    return (uchCRCHi « 8 | uchCRCLo ) ;
}
```

Note: The ”Error Check (CRC)” field must be computed referring to the characters from the first of ADDR to the last of DATA inclusive.

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)	??
	??

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)	??
	??

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	ADDITIONAL 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 (11 HEX) (1 byte).
BYTE COUNT = Number of data bytes (0A HEX) (1 byte).
(here start DATA field)
SLAVE ID = Slave ID identifier (54 HEX) (1 byte).
RUN INDICATOR STATUS = Run indicator status (FF HEX) (1 byte).
ADDITIONAL DATA = 8 Data bytes to compose the follow Hex string.
[0x24] [0x43] [0x74] [0x74] [0x36] [0x73] [SW rev HI][SW rev LO]

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 8 data bytes (byte count is 10, 0A Hex). Last two data bytes carry firmware version (bytes 11 and 12 of the frame).

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
high.	
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

(2): Error type:
01= Illegal Function
02= Illegal data address
03= Illegal data value

TABLE OF CTT REGISTERS

This table is referred to the CTT4 device with 4 input channels and CTT8 device with 8 input channels.

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

Max 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

Instantaneous absolute temperature register (read only)
- CTT8: from 0x0280 (CH1) to 0x0287 (CH8), 8 register 2-byte wide, integer value of Celsius degree
- CTT4: from 0x0280 (CH1) to 0x0283 (CH4), 4 register 2-byte wide, integer value of Celsius degree

NOTE: These registers return binary values from -30 (0xFFE2) to +200(0x00C8)

Max absolute temperature setting register (read only)
- CTT8: from 0x0288 (CH1) to 0x028F (CH8), 8 register 2-byte wide, integer value of Celsius degree
- CTT4: from 0x0288 (CH1) to 0x028B (CH8), 4 register 2-byte wide, integer value of Celsius degree

NOTE: These registers return binary values from -30 (0xFFE2) to +200(0x00C8)

Alarm temperature setting register (read / write)
- CTT8: from 0x0300 (CH1) to 0x0307 (CH8), 8 register 2-byte wide, integer value of Celsius degree
- CTT4: from 0x0300 (CH1) to 0x0303 (CH4), 4 register 2-byte wide, integer value of Celsius degree

Trip temperature setting register (read / write)
- CTT8: from 0x0310 (CH1) to 0x0317 (CH8), 8 register 2-byte wide, integer value of Celsius degree
- CTT4: from 0x0310 (CH1) to 0x0313 (CH4), 4 register 2-byte wide, integer value of Celsius degree

State (diagnostic) of the input channels (read only)
- CTT8: from 0x0290 (CH1) to 0x0297 (CH8), 8 register 2-byte wide, integer value of the state of channel
- CTT4: from 0x0290 (CH1) to 0x0293 (CH4), 4 register 2-byte wide, integer value of the state of channel

- Value 0x0000 Input is right connect
- Value 0x0001 Input is shorted, display of that channel mean SHR
- Value 0x0002 Input is open; display of that channel mean oPE
- Value 0x0003 Input is failure; display of that channel mean Fdc

Fan control setting register (read / write)
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

Reset Max temperature register (write only)
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

Alarm and Trip status register (read only)
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 (read only)

MSByte of register 0x271 (read only)

- bit 0 FAULT relay (active LOW 0 = relay energized)
- bit 1 FAN relay (active HIGH 1 = relay energized)
- 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 (read / write)

MSByte of register 0x274:

Number of ACTIVE channel

-value 0x00 : 3 channel, display mean c h 3

-value 0x01 : 4 channel, display mean c h 4

LSByte of register 0x274:

FAN mode

- value 0x00 : active on CH1÷3, display mean f a N o n
- value 0x01 : disabled, display mean f a N o f f
- value 0x02 : only on CH 4, display mean f a N 4

TROUBLESHOOTING

If response from CTT doesn't happen:

- check connection from CTT and RS232/RS485 converter or other interface if used;
- check if data outgoing from the RS232 serial port of the PC or PLC 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 terminal emulator);
- if the RS232/485 converter is not our model EMI, be sure the automatic turnaround-time is set in range 1 to 2 mS

contrel elettronica srl
I-26900 Lodi - ITALY - Via S. Fereolo, 9
Tel. +39 0371 30207 / 30761 Fax +39 0371 32819
http://www.contrel.it - E-mail: contrel@contrel.it