CZAKI THERMO-PRODUCT
ul. 19 Kwietnia 58
05-090 Raszyn-Rybie
tel. (22) 7202302
fax. (22) 7202305
www.czaki.pl
handlowy@czaki.pl

PROGRAMMABLE TEMPERATURE TRANSMITER

TMD-11

USER'S GUIDE

Table of Contents

1. Short description

TMD-11 is a microprocessor-based temperature converter, designed for Pt100 and Ni100 sensors.

The converter has an RS-485 interface which enables the connection of multiple transducers to a common bus. Configuration of the TMD-11 and reading of measurement results is done via the RS-485 interface.

In order to ensure a collision-free, error-resistant data transmission - the ModBus-RTU communication protocol was used.

Basic information about the Modbus protocol is provided in point 4 of the Manual.

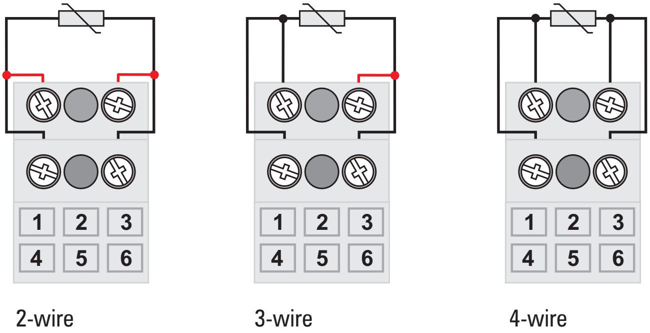
The signal from the temperature sensor is measured with an analog-to-digital converter. The measured signal is filtered, scaled and linearized.

If necessary - additional compensation of the lead wire resistance is performed.

The 4-wire connection of the sensor ensures the highest measurement accuracy and allows to eliminate the influence of the lead wire resistance.

The design of the TMD-11 converter enables the creation of dispersed measurement systems, ensuring high accuracy of the obtained results.

2. Connection diagrams



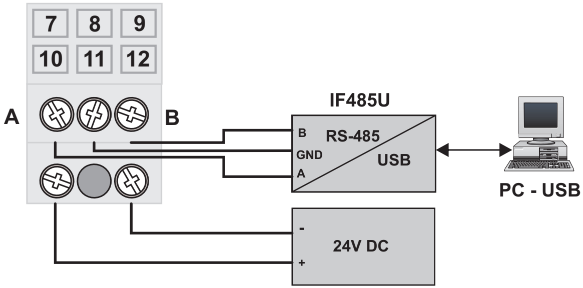
2-wire

3-wire

4-wire

Img. 2.1 Sensor wiring configurations

Attention. The sensor lead resistance can be corrected by entering its value into the Rp register (page 13, register address $888)



Img. 2.2 Power and RS-485 connections

3. Configuration

Configuration of the TMD-11 is performed over RS-485 bus by writing settings to converter internal registers.
The free of charge configuration program tmdkonfig.exe is available at www.czaki.pl.

3.1 Service mode

In order to facilitate the configuration of the converter - e.g. (when its settings are not known), the TMD-11 can be set into the service mode after pressing the [SRV] button.
In the service mode, the converter works with the fixed settings:

bus address = 247
baud rate = 19200
parity = even

The service settings do not change the converter configuration settings (registers), which can be set according to the user's needs.
Operation in the service mode is indicated by the blinking ST diode.
The sensor error signaling is disabled in the service mode.

To exit the service mode, press the [SRV] button again or send the 'reset' command to the converter.

3.2 Converter signaling led's

TMD-11 has three signaling LEDs:

RX - signaling of message receipt
TX - signaling that a response has been sent
ST - status signaling:
    blinking - operation in the service mode
    continuous - sensor error (outside the service mode)

4. RS-485 interface

RS-485 is a multipoint communications standard in automation environment. Industrial standard EIA 485 defines bidirectional, half-duplex data transmission bus.
The maximum length of the bus depends on the choosen baudrate.
As a rule of thumb, the speed in bit/s multiplied by the length in metres should not exceed 10^8.

Recommended cable: shielded twisted pair 24AWG with $Z > 100\ \Omega$.
The screen must be earthed on one side.
Note - Ethernet cables of STP category 5 (cables defined by EIA 568 standard) can be used.

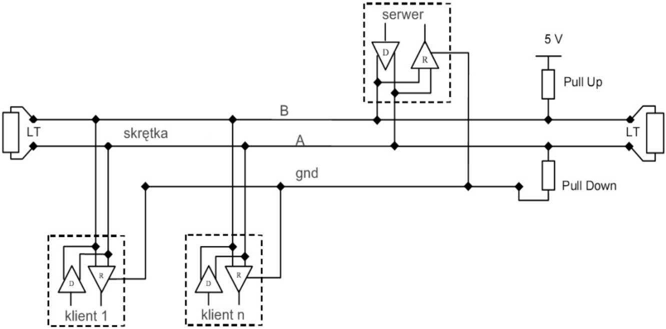


Fig. 2.3 Recommended connection of the RS-485 Modbus-RTU interface

5. ModBus - overview

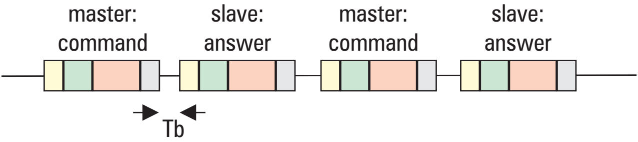
The ModBus standard defines a communication protocol that enables collision-free communication over common serial line (eg. RS-485).

Data exchange in the ModBus network consists in sending messages between the master and individual slaves, while the slave sends the message only in response to the server message.
Each slave must be assigned a unique number (1 ÷ 247), which is his address in the network. The master is communicating with the selected slave, providing his address in the message.
In addition to the slave's address, the message includes: command code, data and the message checksum.

address	function	data	CRC
1 byte	1 byte	0÷252 bytes	2 bytes

Fig. 3.1 ModBus-RTU message frame

The command code informs the slave about the action requested by the master.
After reading the message content, the slave executes the command (e.g. sends the measurement result) or returns information about an error.



Tb - interruption in access to the bus: min 3.5 * transmission time of a single byte (11 bits)

Fig. 3.2 Sending messages via the serial bus

Each slave device "handles" a certain list of commands - depending on the type and functionality of the device. Commands are usually requests to write or read registers or device I / O lines.
The list of commands supported by a given device and the list of available registers is given in the device's documentation.

Frame checksum

Each ModBus message ends with a 16-bit -CRC checksum.
In the TMD-11 converter, the initial value of the polynomial used to calculate the CRC is 0xA001 (hexadecimal).

Data format

In the ModBus protocol, registers are the basic "portion" of data.
The register is two bytes in size.
ModBus messages are transmitted byte by byte, from the most significant to the least significant bit. Each data byte is preceded by a START bit and ended with a parity check (PAR) bit and a STOP bit:

START	b0	b1	b2	b3	b4	b5	b6	b7	PAR	STOP

Parity control.

The PAR bit is set depending on the selected parity check mode:

Parity	even	odd
EVEN	PAR = 0	PAR = 1
ODD	PAR = 1	PAR = 0
NO	PAR = 1	PAR = 1

number of ,1' in byte

eg. for byte 0110 1010 (bin)
PAR = 0 (EVEN) or PAR=1(ODD).

5.1 Data format

In TMD-11, data is encoded in the Big-Endian format, i.e.
most significant byte of data is sent first.

Sending logs.

When sending a register value, the high byte (MSB) is sent first,
and then the low byte of the register (LSB)
e.g. 0x1234 is sent in the form: 0x12, 0x34

Float (32bit):

the Modbus standard does not define how to encode float numbers, but the most common encoding is the so-called Big-Endian.
Bytes are sent in order A B C D, where
A - the most significant byte,
D - least significant byte

Float numbers are written in the form of 4 byte values:

x = (-1)*s * 2 ^ (e-127)*(1.f)

where s = b31, e = (b2...b9)2, i f = b10b11...b32.

Strings:

A single character is 1 byte in size. The register consists of 2 characters.
Sending a string of characters is performed like sending a string of registers: first the high byte is sent, then the low one.
Example: the text 'TMD-11' is sent in the following order:
,T'_'M'_'-'_'D'_'1'_'1'

Example	*data*	*bytes*
0x1234	register (2 bytes)	0x12 34
0x4320F7CF	float 160.968 (4 bytes)	0x43 20 F7 CF
'abcdef'	string (2 chars per register)	a b c d e f

6. List of supported ModBus functions

03 (0x03) Reading internal registers (holding registers)
04 (0x04) Readout of input registers
06 (0x06) Write a single register
08 (0x08) Diagnostic functions:
10 (0x10) Write internal registers

6.1 Function 0x03 (Read registers)

This function is used to read internal device registers (e.g. measurement values)

Request:

function code	1B	0x03
register address	2B	0x0000 ÷ 0xFFFF
quantity of registers	2B	1÷125

Response:

function code	1B	0x03
number of bytes	1B	2xN
register value		2xN

Error:

error code	1B	0x83
exception code:	1B	
address outside the range		0x02
data outside the range		0x03

6.2 Function 0x04 (Readout of input registers)

This function is used to read device input registers (e.g. measurement values).

Request:

function code	1B	0x04
register address	2B	0x0000÷0xFFFF
number of registers	2B	N=1÷125

Response:

function code	1B	0x04
number of bytes	1B	2xN
register value	2B	2xN

Error:

error code	1B	0x84
exception code	1B	
address outside the range		0x02
data outside the range		0x03

6.3 Function 0x06 (Write single register)

The function enables saving a single device register (e.g. network address of the transducer).

Request:

function code	1B	0x06
register address	2B	0x0000 ÷ 0xFFFF
register value	2B	0x0000 ÷ 0xFFFF

Response:

function code	1B	0x06
register	2B	address 0x0000 ÷ 0xFFFF
register value	2B	0x0000 ÷ 0xFFFF

Error:

error code	1B	0x86
exception code	1B	
address outside the range		0x02
data outside the range		0x03

6.4 Function 0x10 (Write to registers)

The function enables saving several device registers (e.g. saving a real number or object description).

Request:

function code	1B	0x10
address of the 1st register	2B	0x0000÷0xFFFF
number of registers (N)	2B	0x0001÷0x0078
number of bytes	1B	Nx2
register values	Nx2	0x0001÷0x0078

Response:

function code	1B	0x10
address of the 1st register	2B	0x0000÷0xFFFF
number of bytes	2B	0x0001÷0x0078

Error:

error code	1B	0x90
exception code	1B	
address outside the range		0x02
data outside the range		0x03

6.5 Diagnostic function 0x08

This function is used to perform the transmitter diagnostics.
When called, the sub-function code is given as a parameter.

Request:

function code	1B	0x08
sub-function code	2B	0x0000,0x0001,0x0004
data	2BxN	

Response * (see description of subfunction 0x0004):

function code	1B	0x03
sub-function code	2B	0x0001
data	2B	0x0000

Diagnostic sub-functions.

Echo

sub-function code	2B	0x0000

* N data 2B

The transducer repeats the received data - the message sent by the server will be identical to the message sent in the response.

Restart Communications Option

sub-function code	2B	0x0001
data	2B	0x0000

The transmitter serial port is reset and reinitialized. If the transmitter was set to the listening mode - no sends responses and exits listening mode. If the transducer was not in the listening mode - it sends a response to the command - as in the echo subfunction.

NOTE:

Changing the settings of communication parameters (baud rate, parity check) and changing the address in the network - will be performed after the reset of the communication interface!

7. TMD-11 Modbus registers

Address *	Size	Description	Data type
		Output registers (modbus function 0x04)	
004 (30005)	2R	measured temperature [°C]	float
006 (30007)	2R	sensor resistivity [Ohm]	float
100 (30101)	1R	calibration register 1	hex
101 (30102)	1R	calibration register 2	hex
102 (30103)	1R	serial number	hex
103 (30104)	1R	device type id	hex
104 (30105)	4R	PCB version	string (8 chars)
105 (30106)	4R	firmware version	string (8 chars)
Holding registers (reading function 0x03, writing 0x06 or 0x10)			
200 (40201)	1R	protection	integer
201 (40202)	1R	slave address	1÷247
202 (40203)	1R	baudrate	= 1 for 9600 bod
			= 2 for 19200
			= 3 for 38400
			= 4 for 57600
			= 5 for 115200
203 (40204)	1R	parity control	= 1 : even parity
			= 2 : odd parity
			= 3 : no parity (2 stop bits)
204 (40205)	1R	sensor type	= 0 : Pt-100
			= 1 : Ni-100
205 (40206)	1R	sensor wiring	= 2 : 2 wire
			= 3 : 3 wire
			= 4 : 4 wire
206 (40207)	2R	connections resistance [Ohm]	float (32 bit)
208 (40209)	8R	description	string (16 char)

** Registers addresses are given in decimal code.*
Addresses in the style of PLCs are given in parentheses.
The data size is given as a multiple of the size of a single register (1R = 2 bytes)

8.Technical Data

Sensor type	Pt100, Ni100 (EN 60751)
Connection	2, 3 or 4 wires
Measuring range	
Pt100	-200 ... 850°C
Ni100	-60 ... 180°C
Sensor current	<0.2 mA
Accuracy	±0.05°C +0.05% of FS
Temperate coefficient	< 0.01% /°C
Conversions speed	10 per second
Power supply	12 ... 36 V DC / 0.2 W
Operating temperature	0 ... +60°C
Humidity range	< 90% without condensation
Dimensions (height x width x depth)	98 x 17.5 x 56.4mm
Weigth	~50g

Output - Modbus

Physical layer	RS-485
Communication protocol	MODBUS-RTU
Slave address range	1 ... 247
Supported baudrate	9600, 19200, 38400, 57600, 115200
Supported parity	Odd/Even/None
Data byte format	8E1, 8O1, 8N1
Response time	< 0.01s

9. Default settings

Slave address	247
Parity control	8E1
Baudrate	19200
Sensor type	Pt100
Sensor connection	3-wire

10. Contents of the package

Transmiter
Printed user guide.