MODBUS Server

MODBUS Server communication protocol

Supported device types and versions

The protocol implements a server (slave) communication with arbitrary devices that support the MODBUS RTU standard either in a serial communication version or a MODBUS over TCP/IP variant.

Communication line configuration

- Line category: Serial (serial communication), SerialOverUDP Device Redundant (serial communication).
- Line category TCP/IP-TCP (MODBUS over TCP/IP). Use a symbolic address *ALL* or *, in order for the KOM process to listen on a selected TCP port on all existing network interfaces. TCP port 502 is commonly used, but any of the ports can be used.
  Line number - set the value for example 1.
  Note: KOM process works as a multitasking TCP server and that is why it is able to handle multiple clients at the same time.

Line protocol parameters

Configuration dialog box - table **Parameters**.
They influence some optional parameters of the protocol. Following line protocol parameters can be set:

Table 1

Parameter	Meaning	Unit	Default value
Silent Interval	A delay before the start of transmission of each data packet.	ms	50
No Request Timeout	If timeout passes and no valid request comes, all stations on the line will go to a communication error. However, the values of output I/O tags will not be invalidated (as this is a server protocol).	mi:ss	1:00
Single Server	If the value of the parameter is set to YES, the KOM process replies by SLAVE_DEVICE_FAILURE error to each request which is sent to a non-existent station (a station with an unknown address). If the value is NO, the KOM process ignores this request and does not send a reply to it.	YES /NO	YES
Moxa Timeout	Switching time of Moxa redundant devices in case of communication error or some problems. As this is a server protocol that waits for requests from external devices, a failure to receive a communication request for a longer time than the value of this parameter is considered to be an error. It is effective only for SerialOverUDP Device Redundant line.	sec	10 sec

Communication station configuration

- Communication protocol "**Modbus Server**".
- The station address is a decimal figure in the range of 1 up to 247. Address 0 is reserved for broadcast.

It is possible to configure more stations with different addresses on one line, the KOM process will reply on behalf of every configured station. See also the line protocol parameters Single Server.

Station protocol parameters

Configuration dialog box - tab **Parameter**.
They influence some optional parameters of the protocol. Following station protocol parameters can be set:

Table 2

Parameter	Meaning	Unit	Default value

Addressing model	Sets an address model of MODBUS protocol: **MODBUS PDU** data are addressed from *0* up to *65535*. **MODBUS data Model** data are addressed from *1* up to *65536*. **Note:** *MODBUS PDU* is a default value. If the *MODBUS data Model* is set, the object with the address X is addressed as X-1 in *MODBUS PDU*. After a change of this parameter, the KOM process must be restarted.	MODBUS PDU MODBUS data Model	MODBUS PDU

I/O tag configuration

Possible value types: **Ai**, **Ao**, **Di**, **Do**, **Ci**, **Co**.

I/O tag address:

In the MODBUS protocol, the basic address space is divided into registers of Coils type (reading/writing), Discrete Inputs (reading), Holding Registers (reading/writing), and Input Registers (reading).

Every address space is independent, providing 2-byte addressing, i.e. addresses from 0 up to 65535.

The I/O tag with an address starting with *%IGNORE* will be ignored.

Address format of I/O tag:
Address format is *[I|U|L|LI|S|SI|f|F|C|D]Fn.Address[.BitNr]* in which:

- The first optional character defines the type of I/O tag:
  - **I** - Integer 16 bit
  - **U** - Unsigned 16 bit (default)
  - **L** - Unsigned long (4 bytes = 2 registers) - two registers with *Address* and *Address+1* are read, unsigned, and transmitted as big-endian (see Note)
  - **LI** - Unsigned long (4 bytes = 2 registers) - two registers with *Address* and *Address+1* are read and transmitted as little-endian, unsigned (see Note)
  - **S** - Signed long (4 bytes = 2 registers) - two registers with *Address* and *Address+1* are read, signed, and transmitted as big-endian (see Note)
  - **SI** - Signed long (4 bytes = 2 registers) - two registers with *Address* and *Address+1* are read and transmitted as little-endian, signed (see Note)
  - **f** - Float 32 bit (two registers) in *big-endian* format (bytes B4, B3, B2, B1 will be sent, B4 is highest byte and B1 is the lowest byte of float)
  - **F** - Float 32 bit (two registers) in *little-endian* format (bytes B2, B1, B4, B3 will be sent, B4 is highest byte and B1 is the lowest byte of float)
  - **C** - Request counter up (16 bit unsigned, which is incremented by every read request). Works only for Fn=3 or Fn=4
  - **D** - Request counter down (16 bit unsigned, which is decremented by every read request). Works only for Fn=3 or Fn=4
    Note: Request counter up/down can be used to configure a "watchdog" I/O tag to monitor the status and speed of Modbus communication.
- Parameter *Fn* is a function of Modbus protocol for data reading, which inserts I/O tag into proper address space:
  - **1** - Coils: binary statuses
  - **2** - Discrete Inputs: binary inputs
  - **3** - Holding Registers: status registers
  - **4** - Input Registers: input registers
- Parameter *Address* is a 2-bytes address of a register in the range of 0 up to 65535.
- Parameter *BitNr* optionally specifies a bit of register in the range of 0 up to 15.
  Note: coexistence of an I/O tag without *BitNr* parameter and multiple I/O tags with *BitNr* parameter having the same *Address* is possible.

Implementation of protocol supports the following functions (commands of MODBUS Client for a D2000 KOM process):

- **1** - Read Coils: reading of binary status - KOM process sends values of I/O tags of Do type.
- **2** - Read Discrete Inputs: reading of binary inputs - KOM process sends values of I/O tags of Do type.
- **3** - Read Holding Registers: reading of status registers - KOM process sends values of I/O tags of Co, Ao type (signed/unsigned).
- **4** - Read Input Registers: reading of input registers - KOM process sends values of I/O tags of Co, Ao type (signed/unsigned).
- **5** - Write Single Coil: writing of binary statuses - KOM process writes a received binary value in I/O tag of Di, Do type into system.
- **15** - Write Multiple Coils - KOM process writes all received binary values of I/O tag of Di, Do type into system.
- **6** - Write Single Register: writing of status registers - KOM process writes the received value in I/O tag of Ai, Ao, Ci, Co type into system.
- **16** - Write Multiple registers: writing of multiple registers - KOM process writes all received values in I/O tags of Ai, Ao, Ci, Co type into system.

Note: This is a server type of protocol that is primarily intended for sending the values out of the D2000 system. Therefore the I/O tags should be configured as an output (Ao, Co, Do) because of the manipulation of their values directly or by control objects. If the I/O tag is configured as input (Ai, Ci, Di), the KOM process is unable to send a valid value in a reply to reading by functions 1 - 4 until the value is written by function 5, 15, 6, or 16 from outside.

If the KOM process does not have the valid value of I/O tag or request to read a nonexistent I/O tag is received, an implicit value False or 0 is sent as a response to a read request (MODBUS protocol does not support the transfer of value quality).

Literature

- MODBUS APPLICATION PROTOCOL SPECIFICATION V1.1b, December 28, 2006. http://www.modbus.org

Blog

You can read blogs about the Modbus protocol:

- Communication – Modbus protocol
- Communication - Modbus in practice
- D2000 and UniPi Neuron
- What load can Raspberry Pi handle?

Changes and modifications

-

Document revisions

- Ver. 1.0 - April 24th, 2009 - document creating
- Ver. 1.1 - November 21st, 2010 - document update.
- Ver. 1.2 - November 11th, 2011 - document update.
- Ver. 1.3 - July 22th, 2019 - Implementation of signed/unsigned long values (L, Ll, S, Sl)

Related pages:

Communication protocols