28 Modbus

Introduction

Modbus is a serial communications protocol for use with its programmable logic controllers (PLCs). Modbus has become a de facto standard communication protocol and is now a commonly available means of connecting industrial electronic devices. It was developed for industrial applications, is relatively easy to deploy and maintain compared to other standards, and places few restrictions other than size on the format of the data to be transmitted.

Modbus enables communication among many devices connected to the same network, for example, a system that measures temperature and humidity and communicates the results to a computer. Modbus is often used to connect a supervisory computer with a remote terminal unit (RTU) in supervisory control and data acquisition (SCADA) systems. Many of the data types are named from industry usage of Ladder logic and its use in driving relays: a single-bit physical output is called a coil, and a single-bit physical input is called a discrete input or a contact.

WCC Lite supports both Modbus Master and Slave protocols. One can select between transmission over TCP/IP or serial connection (RS-485/RS232). Bytes to transmit can either be encoded according to both RTU and ASCII parts of standard.

Modbus Master

Modbus communication contains a single Master and may include more than 1, but not more than 247 devices. To gather data from peripheral devices, master device request a cluster of slave devices for data. If any device understand that this message is addressed for it, replies with data. As no timestamp is sent along with data, having recent data requires frequent polling. WCC Lite can be configured to acquire data periodically in custom-defined intervals.

Configuring datapoints

To use Modbus Master in WCC Lite, it has to be configured via an Excel configuration. This configuration contains two Excel sheets where parameters have to be filled in - Devices and Signals

Modbus Master parameters for Devices tab

Parameter	**Type**	**Description**	**Required**	**Default Value** (when not specified)	**Range**	
					Min	Max
name	string	User-friendly name for a device	Yes			
description	string	Description of a device	No			
device_alias	string	Alphanumeric string to identify a device	Yes			
enable	boolean	Enabling/disabling of a device	No	1	0	1
protocol	string	Protocol to be used	Yes		Modbus RTU, Modbus TCP	
ip	string	IP address of TCP slave device	Yes (for TCP).			
port	integer	TCP communication port	No (for TCP)	502		
bind_address	string	IP address of network adapter used to connect to slave device (Default: ”0.0.0.0”)	No (for TCP)	0.0.0.0		
id	integer	Modbus Slave ID	Yes			

mode	string	Choosing between RTU (”rtu”), ASCII (”ascii”) and TCP(”tcp”) modes. ASCII is the same as RTU, but with ASCII symbols.	No	TCP (for TCP) RTU (for Serial)	rtu, ascii, tcp	
timeout_ms	integer	Response timeout in milliseconds	No	10000		
device	string	Communication port (”PORT1”/”PORT2”)	Yes (for RTU/ASCII)		PORT1	PORT2
baudrate	integer	Communication speed, baud/s	No (for RTU/ASCII)	9600	300, 600, 1200, 2400, 4800, 9600, 19200, 38400, 57600, 115200	
databits	integer	Data bit count for communication	No (for RTU/ASCII)	8	6	9
stopbits	integer	Stop bit count for communication	No (for RTU/ASCII)	1	1	2
parity	string	Communication parity option	No (for RTU/ASCII)	none	none, even, odd	
flowcontrol	string	Number of requests, before link is considered lost (device status signals are changed) and reconnect attempt will be issued	No (for RTU/ASCII)	none	none	
scan_rate_ms	integer	If provided and positive - all jobs will have similar scan rate - all reads and writes will be executed within this timeframe (parameter scan_rate_ms in Signals tab will be ignored)	No	300		
retry_count	integer	Number of requests, before link is considered lost (device status signals are changed) and reconnect attempt will be issued	No	3		
serial_delay	integer	RS485 delay between read and write operations in milliseconds	No (for RTU/ASCII)	50		
keep_alive_timeout	integer	Time interval for sending a keep alive packet (in milliseconds)	No (for TCP)	60		
modbus_multi_write	boolean	Use 15/16 functions to write 1 register/coil (Default: 0)	No	0	0	1
comm_restart_delay	integer	Time delay between disconnecting from slave device and restarting connection (in milliseconds) (Default: 500)	No (for TCP)	500		

Modbus Master parameters for Signals tab

Parameter	**Type**	**Description**	**Required**	**Default Value** (when not specified)	**Range**	
					Min	Max
signal_name	string	User-friendly signal name	Yes			

device_alias	string	Alphanumeric string to identify a device	Yes			
signal_alias	string	Unique alphanumeric name of the signal to be Yes used	Yes			
enable	boolean	Enabling/disabling of an individual signal	No	1	0	1
job_todo	string	Request to send according to modbus specification without device address and checksum. This field can be identical on several tags to fetch them in single request	Yes			
tag_job_todo	string	Similar format to job_todo field. Address and length must be a subset of job field. Defines the individual tag's resgister(s) or coil(s). Can be described in HEX or DEC formats	Yes			
number_type	string	Type of a number (FLOAT, DOUBLE, DIGITAL, etc.)	Yes			
log	integer	Size of this signal's log in Event log.	No	0		
pulse_short_time_ms	integer	Time interval for short output pulse to stay active	No			
pulse_long_time_ms	integer	Time interval for long output pulse to stay active	No			

Different device vendors can have different implementations of a Modbus protocol stack. A register table can be a one of the primary differences. WCC Lite Modbus Master transmits the most significant word (byte) first, however, devices from some vendors might require transmitting the least significant word (byte) first. If that is the case, make sure to switch bytes as needed. To find out more about setting a correct number format, one should consult a section `number_type`.

Modbus job or tag (as a task to be completed) can be built in a two different formats - user can select a more convenient way for him:

- hexadecimal format with every single byte separated by | symbol. Device address, bytes containing output information and CRC (LRC) bytes should be excluded from the message;
- decimal format containing function number, first address and address count, separated by ; symbol. All other information should be excluded from the message;

`job_todo` can group several `tag_job_todo`'s. That way one Modbus message can be used to extract several tags. Grouping is accomplished dynamically meaning that if several identical jobs are found, their tags are grouped automatically.

Modbus Master and Slave both have an additional signal which can be configured to show communication status. It is used to indicate if the slave device has disconnected from master (WCC Lite). To configure such signal, two columns should be filled with particular values. To a newly created additional signal one should make `job_todo` equal to `device_status` and `tag_job_todo` equal to communication_status. Communication error status is set when a predefined count of messages (three by default, defined in `poll_retry_count` column) fail to be received or are considered invalid.

Debugging a Modbus Master application

If configuration for Modbus Master is set up, handler for protocol will start automatically. If configuration is missing or contains errors, protocol will not start. It is done intentionally to decrease unnecessary memory usage.

Modbus Master command line debugging options

`modbus-master`

```
-h [ –help ] Display help information
```

```
-V [ –version ] Show version
-d<debug level> Set debugging level
-c [ –config ] Config path
-r [ –raw ] Show raw telegram data
-f [ –frame ] Show frame data
-s [ –serial ] Show serial port data
–tcp Show tcp packets
–ascii Show ASCII messages
–rtu Show RTU messages
-e [ –redis ] Show redis debug information
-R [ –readyfile ] Ready notification file
```

If Modbus Master does not work properly (e.g. no communication between devices, data is corrupted, etc.), a user can launch a debug session from command line interface and find out why link is not functioning properly. To launch a debugging session, a user should stop modbus-master process and run modbus-master command with respective flags as shown above.

Modbus Slave

WCC Lite can act as one (or several) of slave devices in a communication line. This can be used to transmit data to SCADA systems or other RTU devices. It can reply to a messages from Modbus Master with matching device and register addresses.

Configuring datapoints

To use Modbus Slave in WCC Lite, it has to be configured via an Excel configuration. This configuration contains two Excel sheets where parameters have to be filled in - Devices and Signals

> ⚠ If TCP/IP is used as a trasmission medium, only devices with IPs predefined in host column are allowed to connect. All other connections are rejected

Modbus Slave parameters for Devices tab

Parameter	**Type**	**Description**	**Required**	**Default Value** (when not specified)	**Range**	
					Min	Max
name	string	User-friendly name for a device	Yes			
description	string	Description of a device	No			
device_alias	string	Alphanumeric string to identify a device	Yes			
enable	boolean	Enabling/disabling of a device	No	1	0	1
protocol	string	Protocol to be used	Yes		Modbus serial Slave, Modbus TCP Slave	
host	string	Space separated host IP addresses of master device	Yes (for TCP).			
port	integer	TCP port to listen for incoming connections	Yes (for TCP)			
bind_address	string	IP address of network adapter used to connect to slave device (Default: ”0.0.0.0”)	No (for TCP)	0.0.0.0		
keep_alive_timeout	integer	Minimum time a connection can be idle without being closed in miliseconds	No (for TCP)	60		

mode	string	Choosing between RTU (”rtu”), ASCII (”ascii”) and TCP(”tcp”) modes. ASCII is the same as RTU, but with ASCII symbols.	No	TCP (for TCP) RTU (for Serial)	rtu, ascii, tcp	
device	string	Communication port (”PORT1”/”PORT2”)	Yes (for serial)		PORT1	PORT2
baudrate	integer	Communication speed, baud/s	No (for serial)	9600	300, 600, 1200, 2400, 4800, 9600, 19200, 38400, 57600, 115200	
databits	integer	Data bit count for communication	No (for serial)	8	6	9
stopbits	integer	Stop bit count for communication	No for serial)	1	1	2
parity	string	Communication parity option	No (for serial)	none	none, even, odd	
flowcontrol	string	Communication device’s flow control option.	No (for serial)	none	none	

Modbus Slave parameters for Signals tab

Parameter	**Type**	**Description**	**Required**	**Default Value** (when not specified)	**Range**	
					Min	Max
signal_name	string	User-friendly signal name	Yes			
device_alias	string	Alphanumeric string to identify a device	Yes			
signal_alias	string	Unique alphanumeric name of the signal to be Yes used	Yes			
enable	boolean	Enabling/disabling of an individual signal	No	1	0	1
number_type	string	Type of a number (FLOAT, DOUBLE, DIGITAL, etc.)	Yes			
log	integer	Size of this signal’s log in Event log.	No	0		
common_address	integer	Address of a slave device	Yes			
function	integer	Function number	Yes			
info_address	integer	Register address	Yes			
size	integer	Register/Coil size	Yes			

Mapping values to registers

Internally stored values aren’t organised in a register-like order, therefore mapping should be done by the user. This mapping includes setting an address of the device WCC Lite is simulating as well as function number, register number and how much 16-bit registers are used to store a value. These values should be set in `common_address`, `function`, `info_address` and `size` columns respectively in the Excel configuration.

To find out how many register should be used for storing a values, how values can have their values swapped, a user should consult a section `number_type` (18.2.4).

⚠ If a Modbus master device requests a data from a register that is mapped but doesn’t yet have initial value, ILLEGAL DATA ADDRESS error code will be returned. The same error code is returned if a requested size of value is bigger that defined or if register is not configured at all.

Debugging a Modbus Slave application

If configuration for Modbus Slave is set up, handler for protocol will start automatically. If configuration is missing or contains errors, protocol will not start. It is done intentionally to decrease unnecessary memory usage.

Modbus Slave command line debugging options

`modbus-slave`

```
-h [ –help ] Display help information
-V [ –version ] Show version
-d<debug level> Set debugging level
-c [ –config ] Config path
-r [ –raw ] Show raw telegram data
-f [ –frame ] Show frame data
-s [ –serial ] Show serial port data
–tcp Show tcp packets
–ascii Show ASCII messages
–rtu Show RTU messages
-e [ –redis ] Show redis debug information
-R [ –readyfile ] Ready notification file
```

⚠ If Modbus Slave does not work properly (e.g. no communication between devices, data is corrupted, etc.), a user can launch a debug session from command line interface and find out why link is not functioning properly.

ℹ To launch a debugging session, a user should stop `modbus-slave` process and run `modbus-slave` command with respective flags as shown above.

Revision #9
★Created 11 October 2021 08:30:22 by Tautvilis
Updated 4 July 2022 08:31:05