13.2 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, the master device requests a cluster of slave devices for data. If any device understands that this message is addressed to it, it will reply 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

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

Modbus Master parameters for the Devices tab

Parameter	Type	Description	Required		Default Value (when not specified)	Range	
			TCP	RTU/ASCII		Min	Max
name	string	User-friendly name for a device	Yes	Yes			
description	string	Description of a device	No	No			
device_alias	string	Alphanumeric string to identify a device	Yes	Yes			
enable	boolean	Enabling/disabling of a device	No	No	1	0	1
protocol	string	Protocol to be used	Yes	Yes		Modbus RTU, Modbus TCP	
ip	string	The IP address of the TCP slave device	Yes	-			
port	integer	TCP communication port	Yes	-	502		
bind_address	string	The IP address of the network adapter used to connect to the slave device (Default: ”0.0.0.0”)	No	No	0.0.0.0		
id	integer	Modbus Slave ID	Yes	Yes			
mode	string	Choosing between RTU (”rtu”), ASCII (”ascii”) and TCP(”tcp”) modes. ASCII is the same as RTU, but with ASCII symbols.	No	No	TCP (for TCP) RTU (for Serial)	rtu, ascii, tcp	
timeout_ms	integer	Response timeout in milliseconds	No	No	10000		
device	string	Communication port (”PORT1”/”PORT2”)	-	Yes		PORT1	PORT2
baudrate	integer	Communication speed, baud/s	-	No	9600	300, 600, 1200, 2400, 4800, 9600, 19200, 38400, 57600, 115200	

databits	integer	Data bit count for communication	-	No	8	6	9
stopbits	integer	Stop bit count for communication	-	No	1	1	2
parity	string	Communication parity option	-	No	none	none, even, odd	
flowcontrol	string	Number of requests, before the link is considered lost (device status signals are changed) and reconnect attempt will be issued	-	No	none	none	
scan_rate_ms	integer	If provided and positive - all jobs will have similar scan rates - all reads and writes will be executed within this timeframe (parameter scan_rate_ms in the Signals tab will be ignored)	No	No	300		
retry_count	integer	Number of requests, before the link is considered lost (device status signals are changed) and reconnect attempt will be issued	No	No	3		
serial_delay	integer	RS485 delay between read and write operations in milliseconds	-	No	50		
keep_alive_timeout	integer	Time interval for sending a keep-alive packet (in seconds)	No	-	60		
modbus_multi_write	boolean	Use 15/16 functions to write 1 register/coil (Default: 0)	No	No	0	0	1
comm_restart_delay	integer	Time delay between disconnecting from the slave device and restarting the connection (in milliseconds) (Default: 500)	No	-	500		
update	boolean	Enable to keep updating the tags even if they have the same value.	No	No	0	0	1

Modbus Master parameters for the Signals tab

Parameter	**Type**	**Description**	**Required**		**Default Value (when not specified)**	**Range**	
			TCP	**RTU/ASCII**		**Min**	**Max**
signal_name	string	User-friendly signal name	Yes	Yes			
device_alias	string	Alphanumeric string to identify a device	Yes	Yes			

signal_alias	string	Unique alphanumeric name of the signal to be Yes used	Yes	Yes			
enable	boolean	Enabling/disabling of an individual signal	No	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 a single request	Yes	Yes			
tag_job_todo	string	Similar format to the job_todo field. Address and length must be a subset of the job field. Defines the individual tag's register(s) or coil(s). Can be described in HEX or DEC formats	Yes	Yes			
number_type	string	Type of a number. Each allowed type can be prefixed by a single or a combination of swap prefixes (SW8, SW16, SW32), e.g. SW16.UNSIGNED32, SW32.SW16.SW8.DOUBLE	Yes	Yes		DIGITAL, SIGNED8, UNSIGNED8, SIGNED16, UNSIGNED16, SIGNED32, UNSIGNED32, FLOAT, DOUBLE, SIGNED64, UNSIGNED64	
log	integer	If the log parameter is 1, the signal will be visible in the events log tab, if 0, signals will not be logged.	No	No	0		
pulse_short_time_ms	integer	The time interval for short output pulse to stay active	No	No	0		
pulse_long_time_ms	integer	The time interval for a long output pulse to stay active	No	No	0		
periodic_update_ms	integer	Signal value will be published periodically according to the value set.	No	No			

Note: If the number type is more than 16 bits (this means FLOAT, SIGNED32, DOUBLE, etc.), Modbus multiwrite should be enabled (1).

Note: SIGNED64 and UNSIGNED64 number types are only accepted from firmware version 1.11.1. It is also not recommended to use this number type if high precision is required.

Device status signals

Modbus Master has an additional signal which can be configured to show communication status. It is used to indicate if the slave device has disconnected from the master (WCC Lite). To configure such signal for Modbus protocol, job_todo field with string value is required. For Modbus master required parameters for the status signal will be: **signal_name, device_alias, signal_alias,** and **job_todo**. Job_todo must be: **communication_status.** If the signal returns the value of 1 – the device or protocol connection is on and working correctly, if 2 – the device is off or the protocol is disconnected.

Different device vendors can have different implementations of a Modbus protocol stack. A register table can be 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 two different formats - the user can select a more convenient way for him:

- hexadecimal format with every single byte separated by a | 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.

Debugging a Modbus Master application

If the configuration for Modbus Master is set up, a handler for the protocol will start automatically. If the configuration is missing or contains errors, the 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
-e [ –redis ] Show redis debug information
```

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 the command line interface and find out why the link is not functioning properly. To launch a debugging session, a user should stop the Modbus-master process and run the Modbus-master command with respective flags as shown above.

To run a debug session:

- **Step 1**: Service must be stopped by entering the following command into the wcclite:
  ```
  /etc/init.d/modbus-master stop
  ```
- **Step 2**: After the service is stopped it must be started with the preferred configuration file (JSON files found in the /etc/ folder) and a debug level 7:
  ```
  modbus-master -c /etc/modbus-master/modbus-master.json -d7
  ```
  Additional output forming options described in the table above.
- **Step 3**: Once the problem is diagnosed normal operations can be resumed with the following command:
  ```
  /etc/init.d/modbus-master start
  ```

Revision #6
Created 27 February 2025 08:00:40 by Gabriele
Updated 28 May 2025 12:22:56 by Gabriele