The MODBUS RTU/ASCII, MODBUS/TCP plugin
PRINTED MANUAL

MODBUS RTU/ASCII, MODBUS/TCP plugin

Printed: 10/24/2022

Publisher

AGG Software

Production

© 1999-2022 AGG Software
http://www.aggsoft.com

Table of Contents

1 Introduction

MODBUS is a serial communications protocol for use with programmable logic controllers (PLCs). Simple and robust, it has since become a de facto standard communication protocol, and it is now a commonly available means of connecting industrial electronic devices.

Our MODBUS RTU/ASCII and MODBUS TCP plugin can work in different modes:

1. MODBUS master - the plugin sends MODBUS requests and processes the responses. In this mode, the program operates in the Master mode. You should know all request parameters and describe all expected response items.
2. MODBUS [Slave] - the program acts as a MODBUS slave device and sends responses from another MODBUS master PLC or device. This mode is helpful if you need to send data from a PLC to a host computer at any time using the "Write" request.
3. MODBUS [Passive] - the program processes all requests and responses of other devices communicating over an RS485 bus and then exports decoded data.

This module has the following features:

- Can send valid data requests to any MODBUS-compatible device.
- Can send requests with all major MODBUS functions.
- Calculates and verifies checksum for all data packets.
- Can request bytes, word, double words, and single registers, bits, interpret a sequence of bytes as strings.
- Can poll several MODBUS devices by a custom interval.
- Can flexibly parse all received data packets and extract register's values.

2 System requirements

The following requirements must be met for "MODBUS RTU/ASCII, MODBUS/TCP" to be installed:

Operating system: Windows 2000 SP4 and above, including both x86 and x64 workstations and servers. The latest service pack for the corresponding OS is required.
Free disk space: Not less than 5 MB of free disk space is recommended.
Special access requirements: You should log on as a user with Administrator rights in order to install this module.

The main application (core) must be installed, for example, Advanced Serial Data Logger.

Notes for Microsoft Vista and above:

Since our software saves data to the registry and installs to the Program Files folder, the following requirements must be met:

1. You need Administrator rights to run and install our software
2. The shortcut icon of our software will be located on the desktop;
3. Windows Vista will ask for your confirmation to continue the installation.

NOTE: You can configure the user account only once in order not to see the above dialog box any more. Search Google for the solution to this problem.

3 Installing MODBUS RTU/ASCII, MODBUS/TCP

1. Close the main application (for example, Advanced Serial Data Logger) if it is running;
2. Copy the program to your hard drive;
3. Run the module installation file with a double click on the file name in Windows Explorer;
4. Follow the instructions of the installation software. Usually, it is enough just to click the "Next" button several times;
5. Start the main application. The name of the module will appear on the "Modules" tab of the "Settings" window if it is successfully installed.

If the module is compatible with the program, its name and version will be displayed in the module list. You can see examples of installed modules on fig.1-2. Some types of modules require additional configuration. To do it, just select a module from the list and click the "Setup" button next to the list. The configuration of the module is described below.

You can see some types of modules on the "Log file" tab. To configure such a module, you should select it from the "File type" list and click the "Advanced" button.

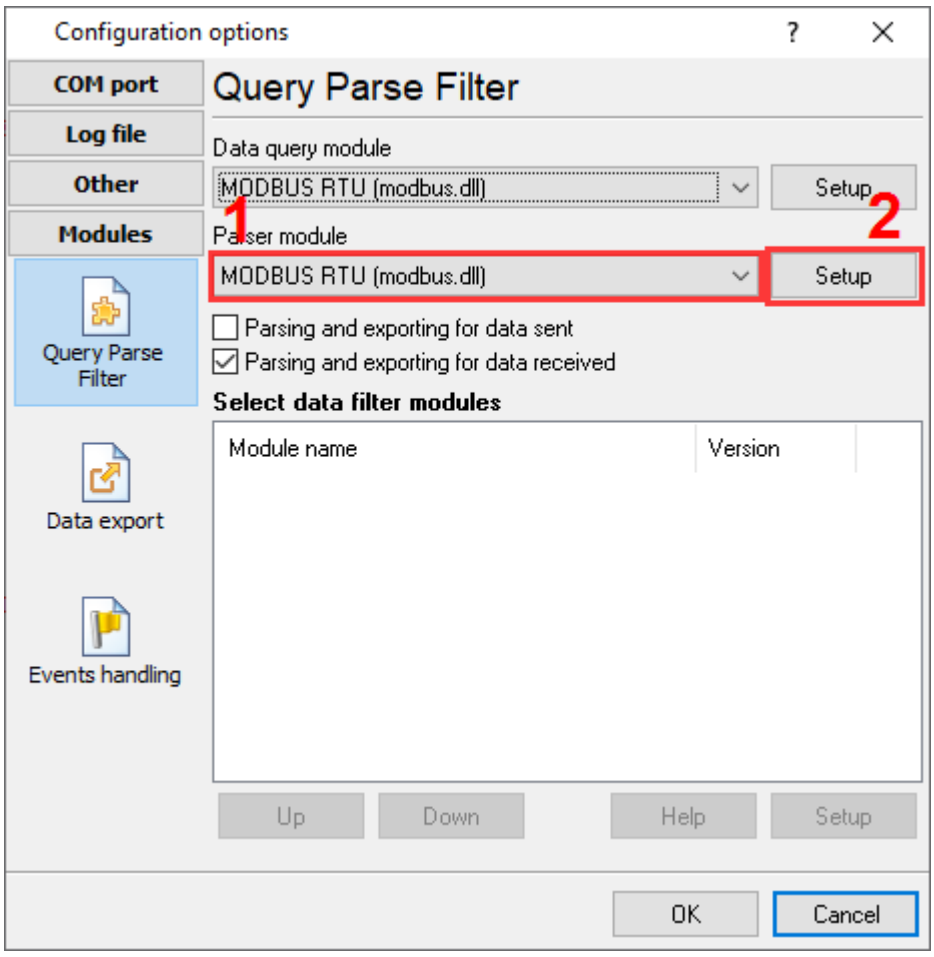

Fig. 1. Example of installed module

4 Glossary

Main program - it is the main executable of the application, for example, Advanced Serial Data Logger and asdlog.exe. It allows you to create several configurations with different settings and use different plugins.

Plugin - it is the additional plugin module for the main program. The plugin module extends the functionality of the main program.

Parser - it is the plugin module that processes the data flow, singling out data packets from it, and then variables from data packets. These variables are used in data export modules after that.

Core - see "Main program."

5 User Manual

5.1 Data query

Click "Actions - Add new request" to add a new item. The dialog window will be shown (fig. 2). Enter a request description containing any characters and click the "OK" button.

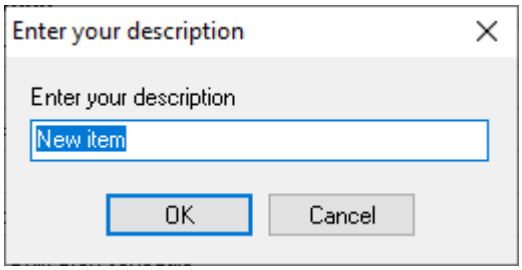

Fig. 2. Name dialog

A new MODBUS request will appear in the requests tree (fig. 3). Each MODBUS request has a few important options:

- **Device address** - it is the address of your MODBUS device in the RS232 or RS485 network. By the MODBUS protocol specification, this address can be from 0 to 255. If you specify 0 as a device address, then all devices in the network should answer this request. If you poll several identical devices on the same bus, you can specify all addresses in one request, like 1,2,3,4.
- **Function** - the MODBUS protocol function number. Usually, this value is 3 for reading holding registers or 4 for reading input registers.
- **Data address** - is the address of the first read register in the device memory.
  - **Address** - selecting this option, you should specify the full memory address, including the function code like 40001 for an input register.
  - **First register** - it's a register offset. This value is zero-based. If you want to read a register with the address 40100, you should specify 99 in this field.

- **Registers to read** - specify the number of consecutive registers in the device memory.
- **Request timeout** - is the time interval for which the program sends a request to a MODBUS device. After reaching the timeout limit, the program will automatically cancel the current request and execute the next request in the queue. The timeout value depends on the network on which master (program) and slave (device) is running. If the network is slow, then the timeout value should be larger, and if the network is fast, the timeout value can be smaller.
- **Export data for all requests at once** - you should activate this option if you read multiple values from a device in several requests and want to export it in one row to Excel or a database. In other words, this option groups data from requests with similar polling intervals and exports it simultaneously.

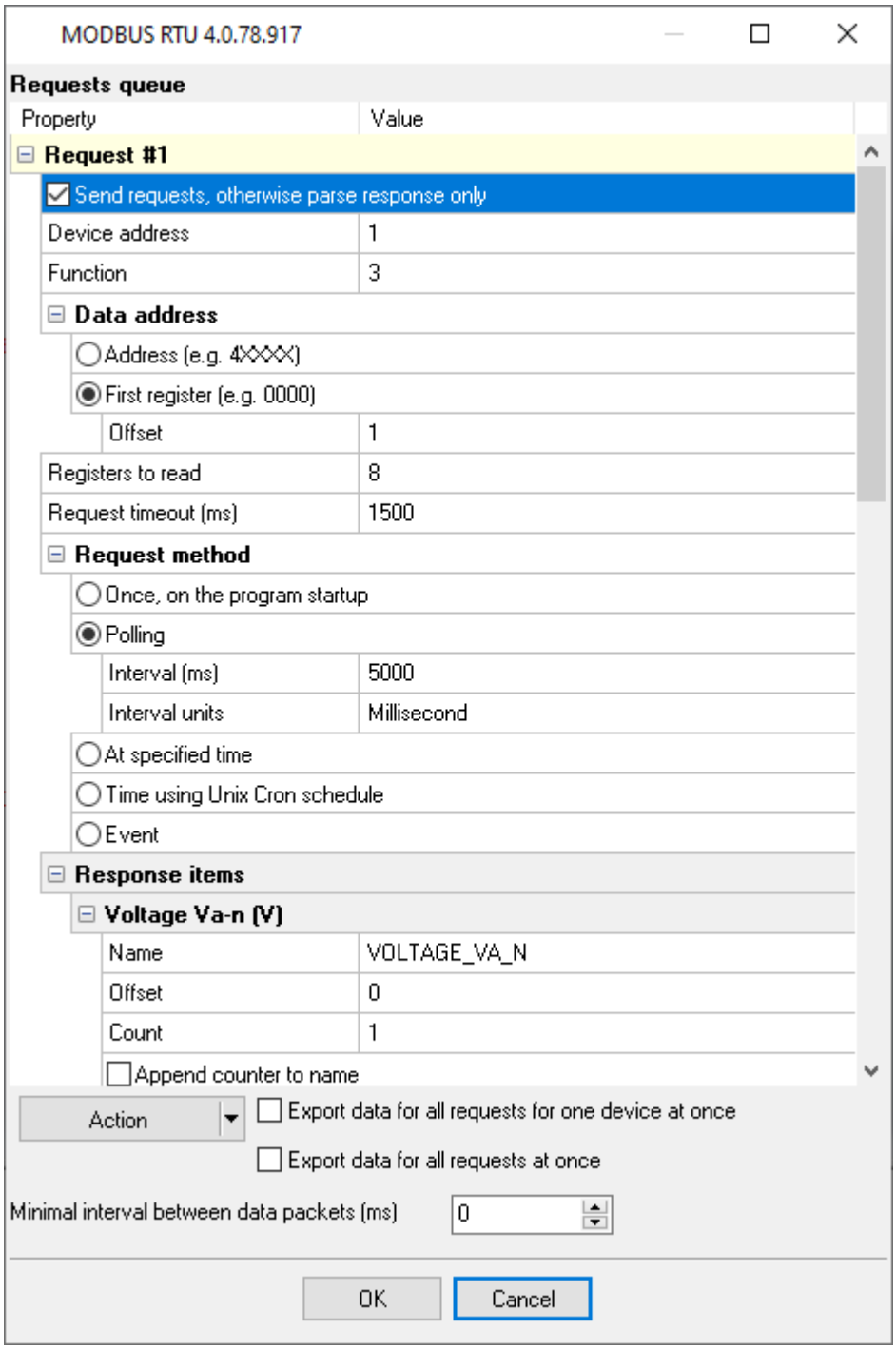

Fig. 3. MODBUS request

5.2 Request method

The plugin can send requests in the following mode:

Once, on program startup - the program will send a request once when the program starts.

Polling - the program will send a request periodically based on an interval specified. The interval between requests depends on the network on which master (program) and slave (device) is running. If the network is slow, then the time for each request will be larger and vice versa. Because the program executes all requests in the queue one by one, the time between requests depends on the number of requests in the queue.

At the specified time - the time of the day using the 24hr format (e.g., 18:00:00). You may specify several time points separated by a semicolon (e.g. 11:00:00;11:20:00;11:40:00).

Time, using Unix Cron schedule - a flexible schedule format that allows sending requests periodically or at the specified time. You can find detailed information about this format and see examples in the "Cron time format" section. The default is 0 0 12 * * *, which means "every week, every day at 12:00:00".

Event - the program executes the corresponding request when the plugin receives an external event. These events can be generated by our other plugins, like "Event generator," "Script execute," "Expressions,"

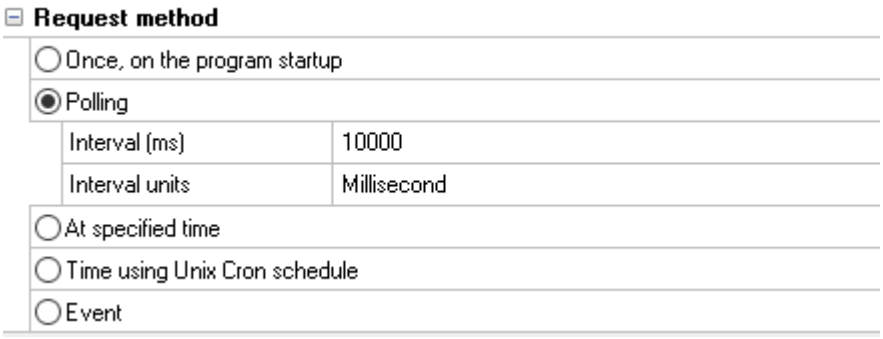

Fig. 4. Request methods

If you have added several requests to the queue, you can move them up or down. To do it, select a request, click the "Action" button, and select an action ("Move up" or "Move down").

You can also click this button to change a request's description or remove a request from the queue.

You can also perform the same actions by using the context menu that pops up when you right-click items in the request tree.

5.3 Cron time format

The CRON format is a simple yet powerful way to describe time and operation periodicity. The traditional (inherited from the Unix world) CRON format consists of five fields separated with spaces:

<Second> <Minutes> <Hours> <Month days> <Months> <Weekdays>

Any of the five fields can contain the * (asterisk) character as its value. It stands for the entire range of possible values. For example, every minute, every hour and so on. In the first four fields, you can also use the proprietary "?" (w/o quotes) character. See its description below.

Any field can contain a list of comma-separated values (for example, 1,3,7) or an interval (subrange) of values defined by a hyphen (for example, 1-5).

You can use the / character after the asterisk (*) or after an interval to specify the value increment. For example, you can use 0-23/2 in the "Hours" field to specify that the operation should be carried out every two hours (old version analog: 0,2,4,6,8,10,12,14,16,18,20,22). The value */4 in the "Minutes" field means that the operations must be carried out every four minutes. 1-30/3 is the same as 1,4,7,10,13,16,19,22,25,28.

You can use three-word abbreviations in the "Months" (Jan, Feb, ..., Dec) and "Weekdays" (Mon, Tue, ..., Sun) fields instead of numbers.

Examples

Note: the <Second> field equal 0 in all examples

Format	Description
* * * * *	every minute
59 23 31 12 5	one minute before the end of the year if the last day in the year is Friday
59 23 31 Dec Fri	one minute before the end of the year if the last day in the year is Friday (one more variant)
45 17 7 6 *	every year on the 7th of June at 17:45
0,15,30,45 0,6,12,18 1,15,31 * 1-5 *	00:00, 00:15, 00:30, 00:45, 06:00, 06:15, 06:30, 06:45, 12:00, 12:15, 12:30, 12:45, 18:00, 18:15, 18:30, 18:45, if it is the 1st, 15th or 31st of any month and only on workdays
*/15 */6 1,15,31 * 1-5	00:00, 00:15, 00:30, 00:45, 06:00, 06:15, 06:30, 06:45, 12:00, 12:15, 12:30, 12:45, 18:00, 18:15, 18:30, 18:45, if it is the 1st, 15th or 31st of any month and only on workdays (one more variant)
0 12 * * 1-5 (0 12 * * Mon-Fri)	at noon on workdays
* * * 1,3,5,7,9,11 *	every minute in January, March, May, July, September, and November
1,2,3,5,20-25,30-35,59 23 31 12 *	on the last day in the year at 23:01, 23:02, 23:03, 23:05, 23:20, 23:21, 23:22, 23:23, 23:24, 23:25, 23:30, 23:31, 23:32, 23:33, 23:34, 23:35, 23:59

0 9 1-7 * 1	on the first Monday of every month at 9 in the morning
0 0 1 * *	at midnight on the 1st of every month
* 0-11 * *	every minute till noon
* * * 1,2,3 *	every minute in January, February, and March
* * * Jan,Feb,Mar *	every minute in January, February, and March
0 0 * * *	every day at midnight
0 0 * * 3	every Wednesday at midnight

You can use the proprietary "?" character in the first four fields of the CRON format. It stands for the start time, i.e., the question mark will be replaced with the start time during the field processing: minute for the minute field, hour for the "Hours" field, month day for the month day field, and month for the month field.

For example, if you specify:

? ? * * *

The task will be run at the moment of startup and will continue being run simultaneously (if the user does not restart the program again, of course) – the question marks are replaced with the time the program was started at. For example, if you start the program at 8:25, the questions marks will be replaced like this:

25 8 * * * *

Here are some more examples:

- ? ? ? ? * - run _only_ at startup;
- ? * * * * - run at startup (for example, at 10:15) and continue being run in exactly one hour: at 11:15, 12:15, 13:15 and so on;
- * ? * * * - run every minute during the startup hour;
- */5 ? * * * - run on the next day (if CRON is not restarted) at the same hour every minute and so on every day, once in five minutes, during the startup hour.

5.4 Data parser

All data export modules use parser variables containing parsed (decoded) values. Our MODBUS RTU/ASCII, MODBUS/TCP picks out significant data blocks (data packets) from the common data flow, analyzes the extracted data packet, and checks its integrity using CRC (cyclical redundancy check).

All parser items are assigned with a corresponding request in the queue. You can define one or more parser items (variables) in one request.

You can add a new parser item (variable) to the request by clicking "Actions - Add response item." Before, you should select a caption of the corresponding request. A new parser item (variable) will appear in the "Response items" group (fig. 5).

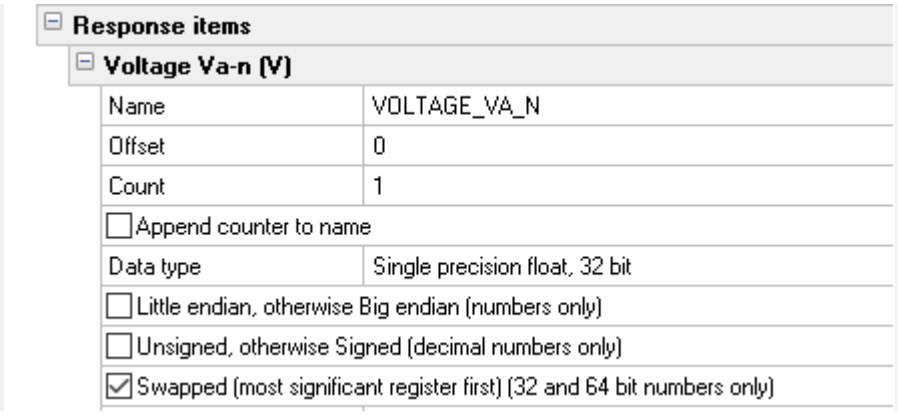

Fig. 5. Data parser items.

Each response item has a few important options:

- **Name** - the of the parser variable. This name you'll bind with fields in data publication modules.
- **Offset** - the device can respond few data bytes, but you need only some of them. The "Offset" field contains a byte offset of the data from the beginning of the data block. This value is zero-based. If the first byte of your value is located at the beginning of the data block, then this value should be 0. You can specify -1 here. Then the program will automatically calculate the value offset. Please, note, the data block does not include a data packet header (address and data size).
- **Count** - is the number of values (nor bytes) with the same parameters (data type and default value), located one after another since the offset. If you specify more than one here, then a value index (1, 2, 3, etc.) will be added to the parser item name. Please note that it is a count of values, not bytes or registers (one value can allocate one or more bytes or registers).
- **Data type** - is the data type of the value. Each value can utilize one (for the "Byte" data type) or more bytes.
- **Default value** - this value will be used if the parser can't parser data block for this parser item. For example, if the data block has a small size or offset is too large.

5.5 Custom formulas for response items

You can execute basic math calculations on read values or bytes. The most expressions were copied from the Expressions plugin, where you can find the full list of supported functions.

Special variables

{VALUE} - it is a special placeholder of the current response item value. This value is already decoded to the configured data type.

If you select "Custom formula" in the "Data type" field for a response item, the plugin also uses the following variable:

BYTExxSyy - these variables contain the selected number of bytes from a response data. The response data does not include MODBUS packet header and checksum (device address, function code, etc.).

xx - a hexadecimal value of a zero-based offset of the first byte.

yy - a hexadecimal value of a number of bytes.

Examples

BcdToStr(BYTE00S04) - converts data bytes (0-3) from BCD coding to a string.

BcdToInt(BYTE00S02) * IntPower(10, SmallintToStrBE(BYTE02S02)) - converts data bytes (0-1) from BCD coding to
decimal and to raises to the power of values from bytes 2 and 3.

{VALUE} * (VTRFwM)/10 - multiplies the current response item's value to another value extracted from the same response before.

{VALUE} * K_RMS_FACTOR - another similar example.

5.6 MODBUS slave mode

In this mode, the program acts as a MODBUS slave device and serves requests from a MODBUS master. The plugin supports all major MODBUS function codes: 1,2,3,4,5,6,15,16. The plugin does not send data without a valid request from a MODBUS master.

This mode is useful when a PLC initiates a connection and wants to send something to a host computer using the "WRITE" command like "Write Single Coil," "Write Single Register," "Write Multiple Coils," or "Write Multiple registers,"

The configuration window (fig. 6) is slightly different in this mode. For example, you do not need to configure data polling parameters.

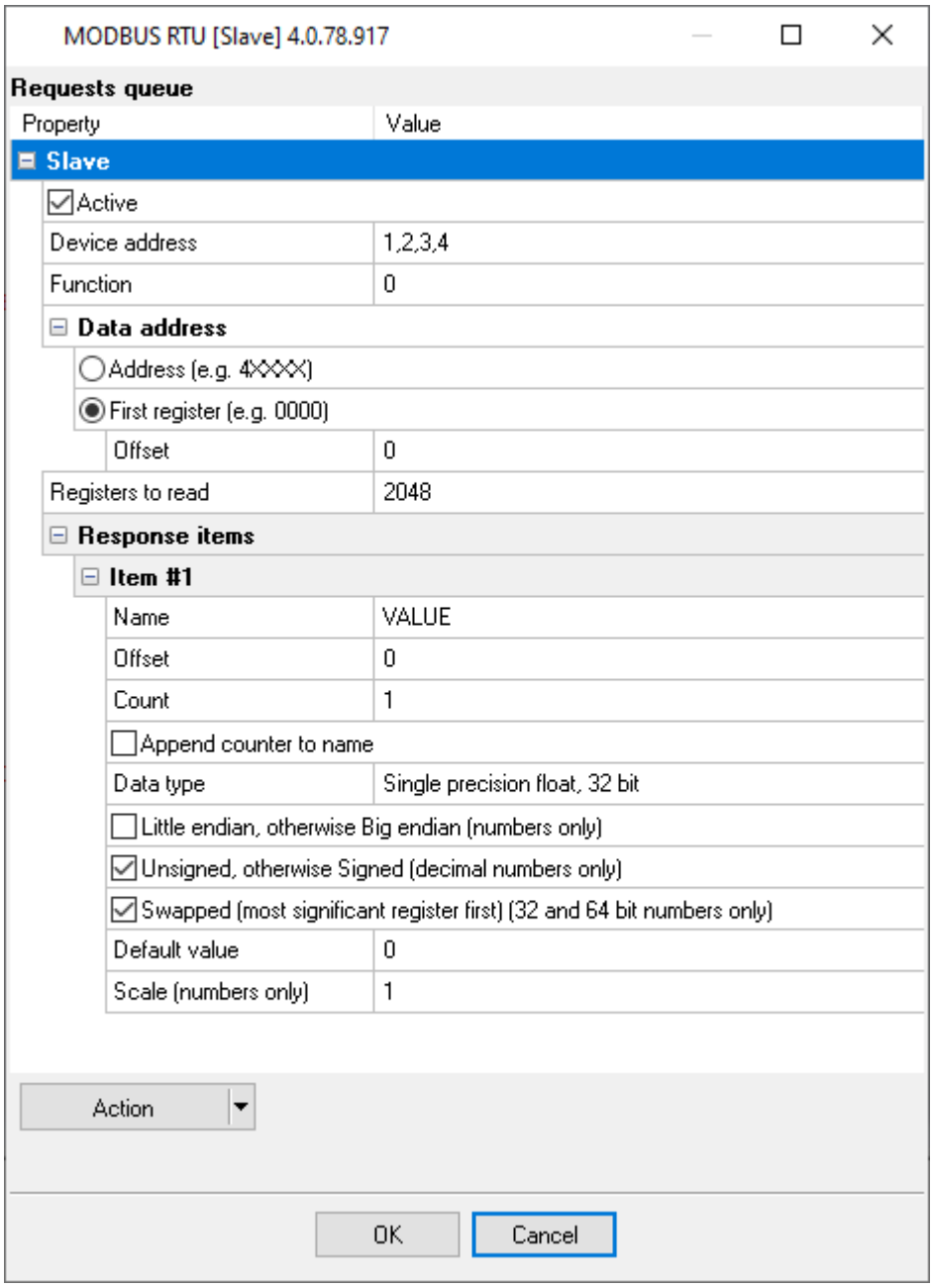

Fig. 6. MODBUS slave emulation

Active - if this option is checked, the request parameters are active, and the plugin emulates the slave device described in the parameters below.

Device address - it is the address of an emulated slave device. You can specify several addresses in one request, like 1,2,3,4. The plugin does not respond to requests addressed to other addresses.

Function - the function code allows you to define the supported memory areas in the emulated slave device.

0 - Emulate all memory areas.

1 - Coils.
2 - Discrete inputs.
3 - Holding registers.
4 - Input registers.

You can also combine function code like 1,3 or 3,4

Data address and **Registers to read** options allow you to define a range of an emulated memory area in the slave device. You can use the following values to emulate the entire memory space of 65535 registers.

Offset: 0
Register to read: 65535

Please note, a larger number of registers require more computer memory. It becomes significant if you emulate many slave devices.

If a master device tries to access data from unallocated memory space, the slave returns an exception code to the master.

Response items - this list allows you to define a custom name and a data type for some values in the slave's memory. The **Offset** parameter defines an absolute offset in the address space for the named value. Other parameters have the same meaning as described before.

If this list is empty or omits some values, the plugin exports all unnamed values as unsigned 16-bit decimals with an autogenerated name like "**V**NNNNN" or "**B**NNNNN."

V - it is a prefix for registers
B - it is a prefix for coils and discrete inputs.

NNNNN - is an absolute offset of a value in the address space in the decimal format.

5.7 MODBUS passive mode

In this mode, the program processes all MODBUS read-write requests and responses and exports data values. If you use the "Spy" mode in our data logger software, you must enable parsing for sent and received data (fig. 7). The plugin supports all major MODBUS function codes: 1,2,3,4,5,6,15,16. The plugin does not send any data.

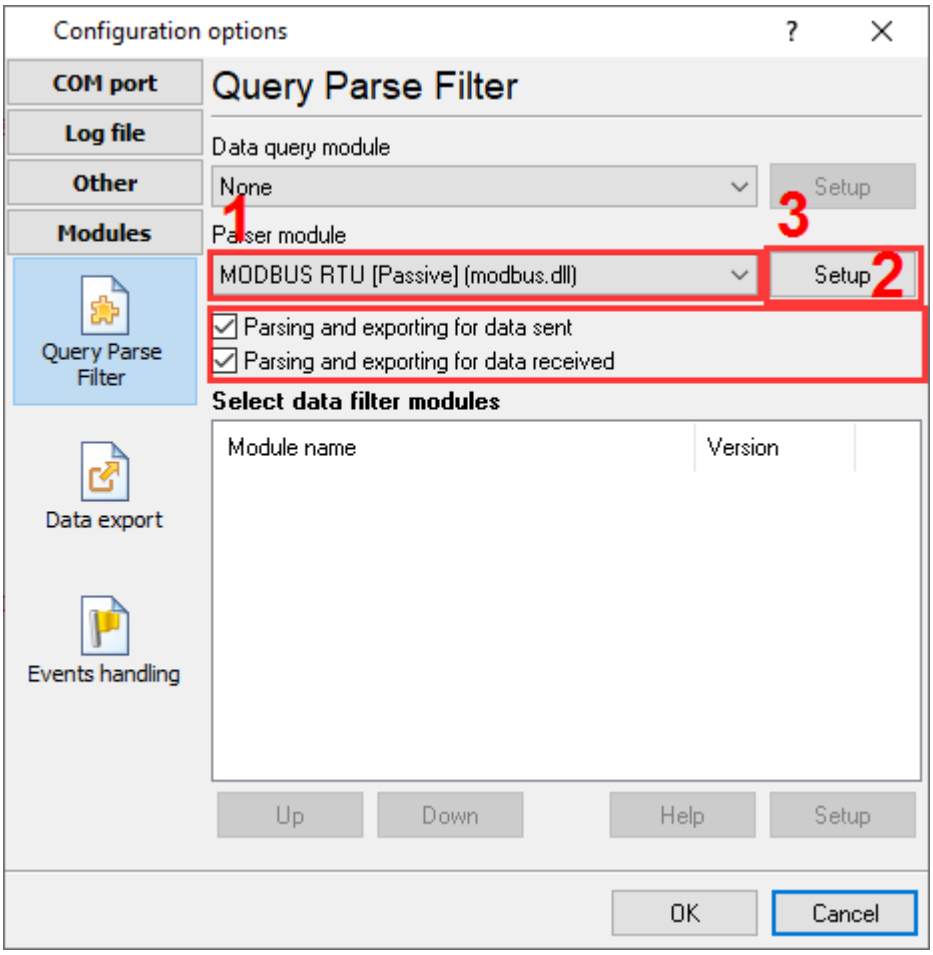

Fig. 7. Data parsing options

This mode helps monitor all data exchange over an RS485 bus and get all data in parallel with another application or two communicating external devices.

The plugin would only process a response from a MODBUS slave device if the plugin received a valid request from a MODBUS master before. The interval between a request and a response is lower than a timeout interval defined in the "Request timeout" parameter.

The configuration window for this mode is shown in figure 8.

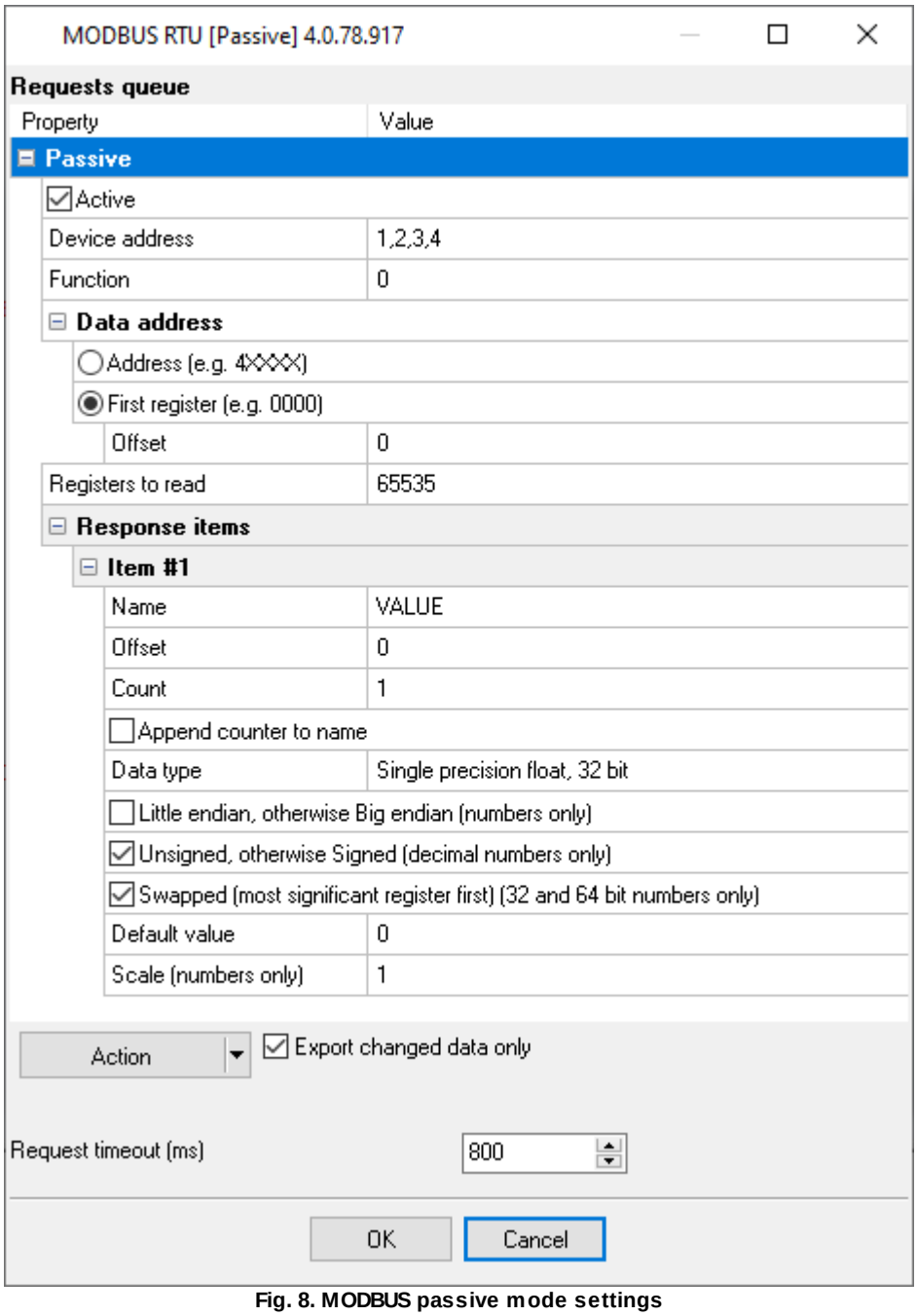

Fig. 8. MODBUS passive mode settings

Request and response settings are identical to the "MODBUS slave" mode.

The plugin can export data from reading and writing requests. If a master intensively polls a slave device but data changes rarely, you can enable the "Export changed values only" option, and the plugin will ignore repeated read or write requests.