ModbusScope

Jens Geudens

Apr 28, 2026

CONTENTS:

CHAPTER

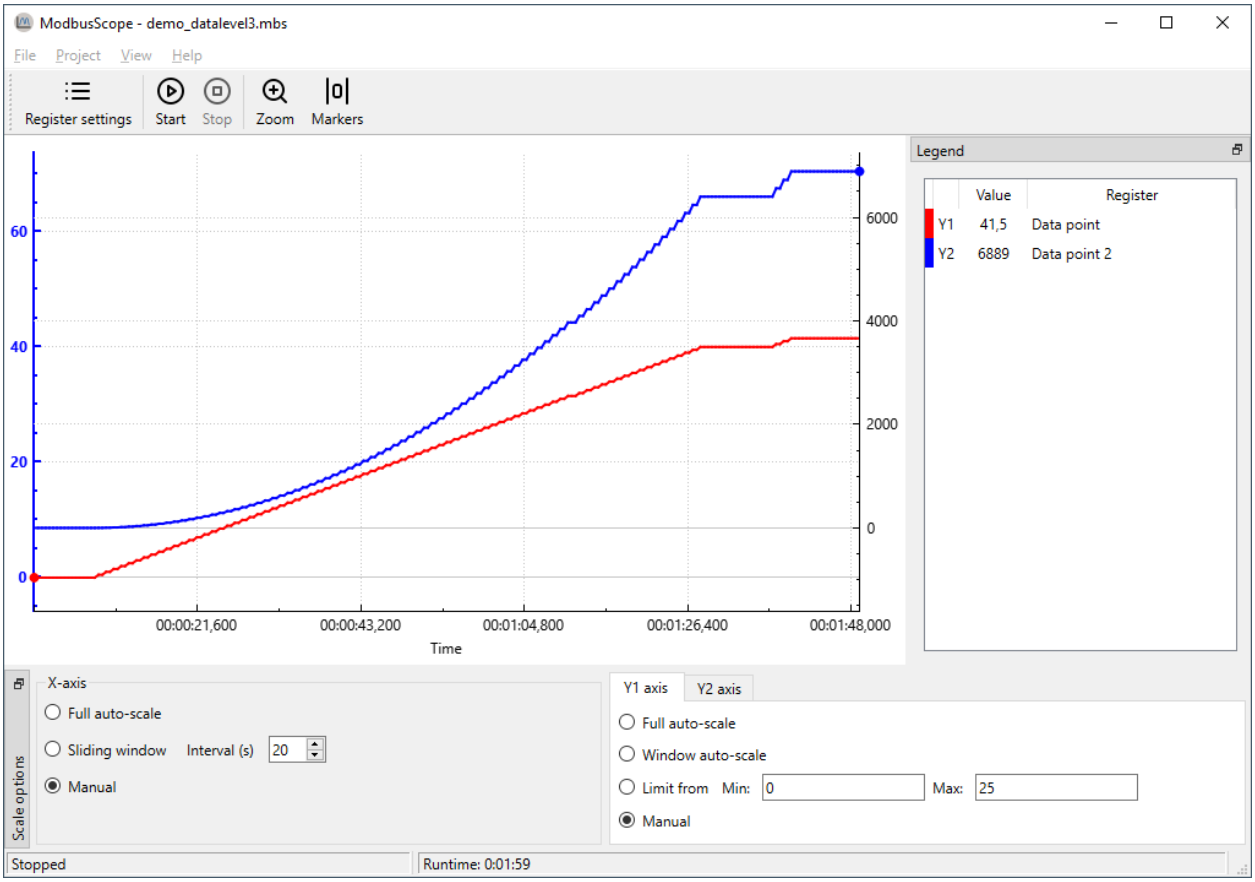
ONE

INTRODUCTION

ModbusScope is a graphical user interface tool designed for logging and visualizing data using the Modbus protocol. It supports real-time data visualization, enabling users to see graphical representations of data while logging. The tool also allows users to export logged data to CSV files for further analysis. It's highly configurable to suit various data logging needs and supports multiple Modbus devices simultaneously, making it ideal for monitoring devices in industrial automation, energy management, and environmental monitoring.

1.1 Features

- **Real-time Data Visualization**: Provides real-time graphical representations of Modbus data.
  - **Zooming**: Interactively zoom in and out of the graphs to focus on specific time periods.
  - **Markers**: Analyze data (min/max/average/...) within a specific time period.
- **Data Logging**: Continuously logs data from Modbus devices for future analysis.
- **Multiple Device Support**: Capable of handling multiple Modbus devices simultaneously.
- **CSV Export**: Allows export of logged data to CSV files for easy analysis and sharing.
- **Data manipulation**: Perform calculations and combine different multiple registers into one value.


ModbusScope - demo_datalevel3.mbs
File
Project
View
Help
Register settings
Start
Stop
Zoom
Markers
Legend
Value
Register
Y1
41,5
Data point
Y2
6889
Data point 2
60
40
20
0
6000
4000
2000
0
00:00:21,600
00:00:43,200
00:01:04,800
00:01:26,400
00:01:48,000
Time
Scale options
X-axis
Full auto-scale
Sliding window
Interval (s)
20
Manual
Y1 axis
Y2 axis
Full auto-scale
Window auto-scale
Limit from
Min:
0
Max:
25
Manual
Stopped
Runtime: 0:01:59

CHAPTER

TWO

DOWNLOAD

The latest version of *ModbusScope* can always be downloaded from the release page on Github. This page provides both the installer version and the standalone version of the software. The Windows installer is also available as Chocolatey package.

CHAPTER

THREE

OVERVIEW

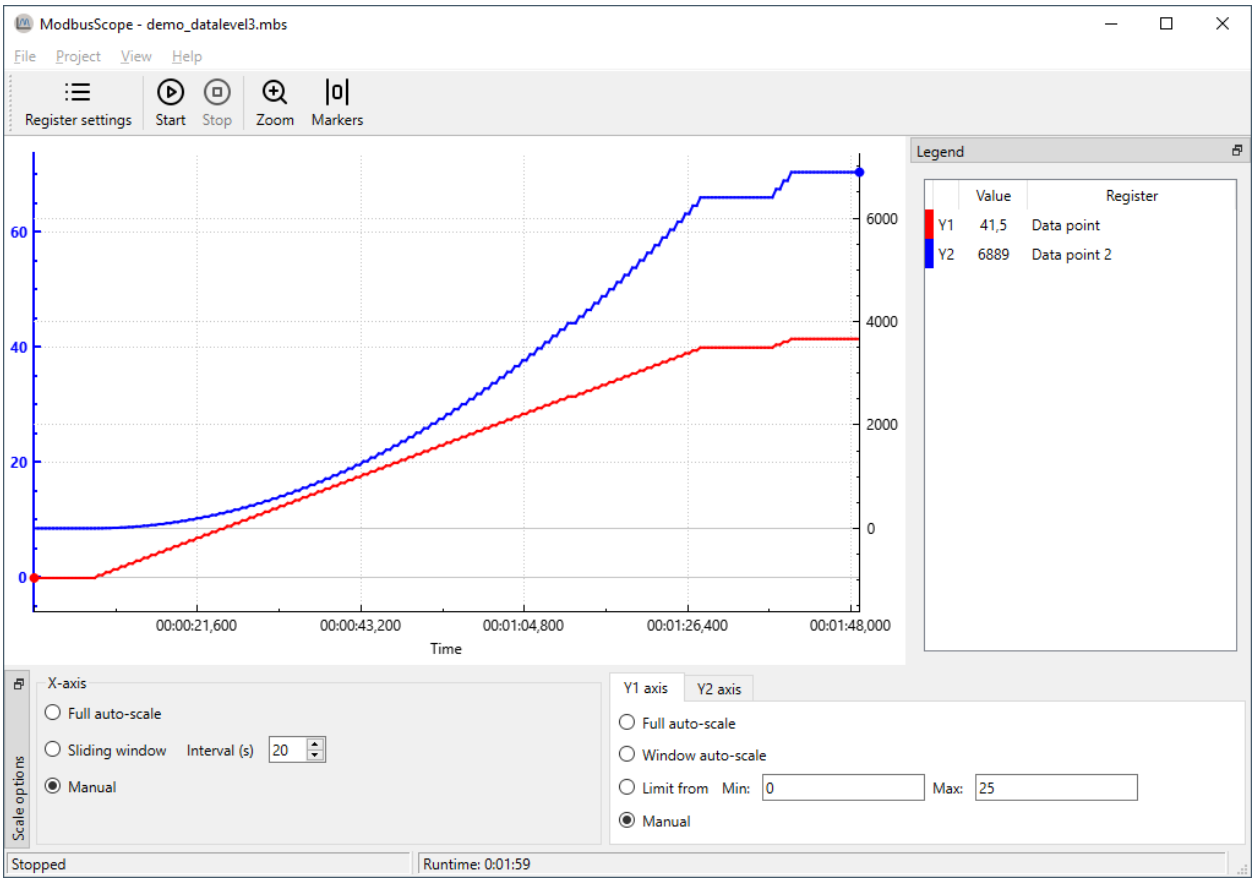
3.1 Installing

The *ModbusScope* installer or standalone version can always be downloaded from the release page.

ModbusScope can be easily installed by double-clicking on the provided installer file and following the on-screen instructions. The installer will handle the installation of all necessary files on your computer. At the end of the installation process, you have the option to set ModbusScope as the default application for opening `.mbs` files. This allows you to quickly open your ModbusScope project files by simply double-clicking on them in your file explorer.

3.2 Getting started

3.2.1 Open ModbusScope



3.2.2 Configuring Modbus connections and devices

To set up *ModbusScope*, follow these steps in order:

1. **Configure your connection** — Open *Settings* > *Connection* and enter the details of how to communicate with your Modbus device (e.g., TCP address/port or serial settings).
2. **Add and configure devices** — Open *Settings* > *Device* and add a device for each Modbus slave you need to communicate with. Configure device-specific settings like the slave ID and timeout. Link the device to the connection you created in step 1.

3.2.3 Configure Modbus registers

Once you have set up your connections and devices, you can add Modbus registers using the register settings window. For each register, you can:

- Change the name, color, and data type
- Specify which device the register belongs to
- Optionally perform calculations on the register data

3.2.4 Log data

- Click on the "Start Logging" button.
- View the data as it is logged in real time using the provided tools.
  - Adjust the time or value axis scale.

3.2.5 Export data

- After logging data, click on the "Export" button.
- Choose the file location and export the data as a CSV file for further analysis.

3.2.6 Save configuration

Save your project configuration by going to *File* > *Save Project As...* to easily reload your setup in future sessions.

CHAPTER

FOUR

GRAPHVIEW

4.1 Start/stop log

Once you have added the desired Modbus registers, you can begin logging data by pressing the *Start Logging* button. ModbusScope will communicate with each Modbus slave specified in your registers. Each device is connected through a TCP or RTU (serial) connection as configured. Once the *Start Logging* button is pressed, ModbusScope will start logging the data and will automatically display the values on the graph.

> **NOTE**: When you press the *Start Logging* button, it will clear any data that is already present in the graph and start logging new data.

When you have finished testing and collecting data, you can stop logging by pressing the *Stop Logging* button. Once logging is stopped, you can further examine the collected data by using the various tools provided by the application.

4.2 Adjust scale settings

While ModbusScope is logging data, you can view the already logged values in the graph view. The various scale settings allow you to examine the data in different ways and zoom in or out as needed. You can adjust the scale settings to suit your needs and easily examine the data while new values are being added to the log. The software provides several scale settings that you can use to customize the view of the data.

4.2.1 X-axis

The x-axis of the graph can be scaled in three ways: *full auto-scale*, *sliding window*, and *manual* mode.

- *Full auto-scale* will automatically adjust the maximum of the x-axis to show all data of the graph.
- *Sliding window* allows you to view only the values of the last time period, which is configurable.
- *Manual* scale setting means that the scaling is fixed and will not change automatically while logging, even when new values are logged, the current time period stays the same.

4.2.2 Y1- and Y2-axis

Compared to the x-axis, the y-axis has two similar modes: *full auto-scale* and *manual*. These options work the same way as they do for the x-axis. In addition to these, the y-axis also has two other modes: *window auto-scale* and the *limit from* setting.

- *Window auto-scale* automatically adjusts the range of the y-axis based on the values currently visible in the graph.
- *Limit from* setting allows the user to set the minimum and maximum values for the y-axis.

There are two Y-axes available and the user can select which Y-axis to use per graph. This can be done in the *register* dialog or by double-clicking the Y-axis indicator in the legend.

4.3 Zoom graph

The graph-view in *ModbusScope* supports zooming to allow for a more detailed examination of the logged data. By using the scroll wheel on your mouse, both the x- and y-axis switch to manual setting and the range of the axis will be increased or decreased based on the scroll wheel movement. The current position of the mouse cursor is used as a reference point for the zoom action, allowing you to focus on specific areas of the graph.

It is also possible to select a single axis to zoom in and out of by clicking on it. This means that when you use the mouse wheel, only the selected axis will be zoomed, while the range of the other axis will remain the same. To deselect an axis, you can click anywhere in the graph view. Double-clicking an axis will reset it to *full auto-scale* mode.

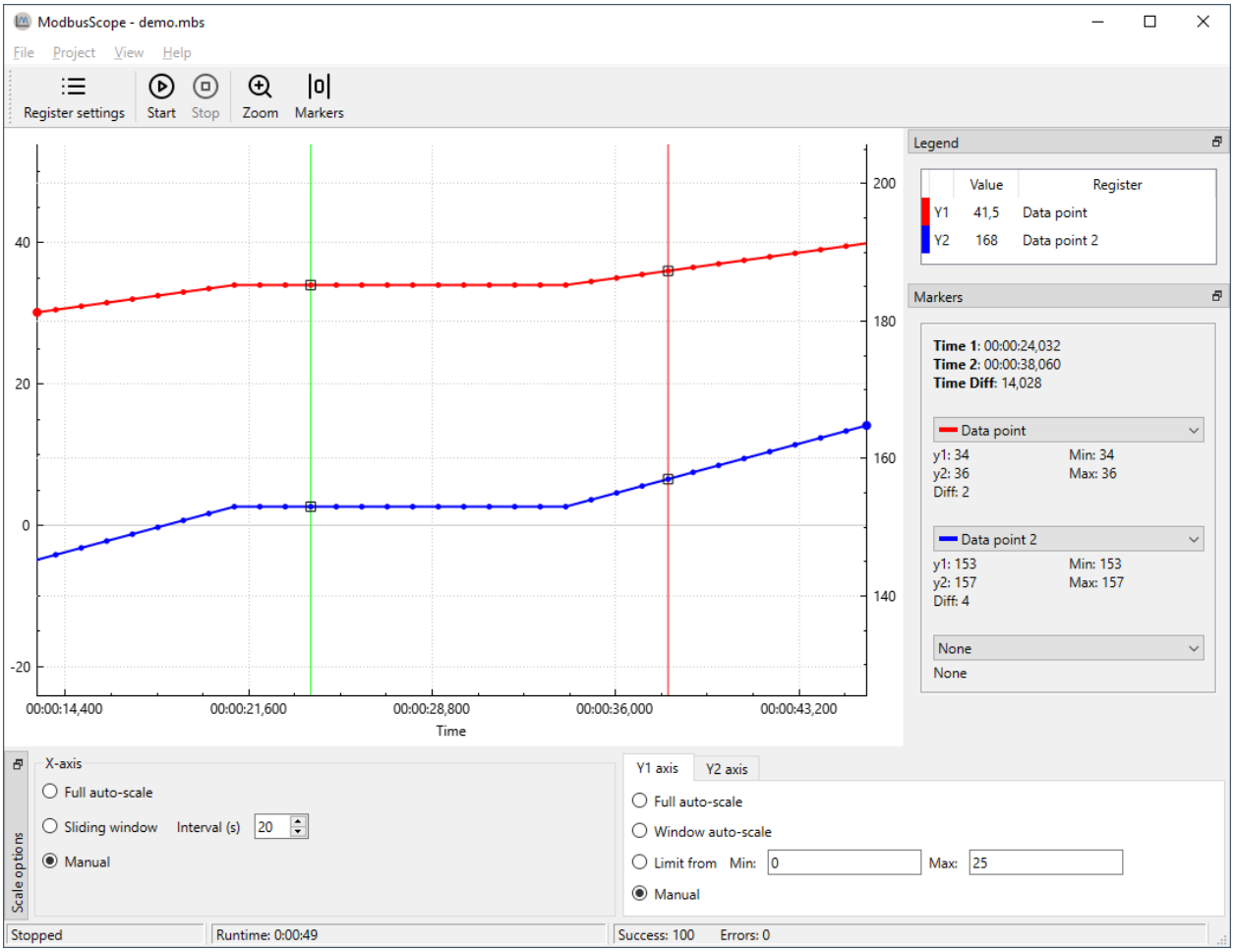
The *Zoom* button in *ModbusScope* allows the user to zoom in on a specific region of the graph by drawing a rectangle over the area of interest using the mouse.

4.4 Enable/Disable markers

ModbusScope offers a feature to investigate dynamic behavior of a system by measuring the time and value differences between two points on the graph. To use this feature, you need to add two vertical markers to the graph. Once the markers are added, you can move them to the desired positions and *ModbusScope* will calculate and display the time and value differences between them. This allows you to easily compare and analyze changes in the system over a specific period of time.

To add the left marker (green vertical line), press the `Ctrl` key and click on the location of interest with the left mouse button. The position must coincide with a sample in the graph. To add the right marker (red vertical line), press the `Ctrl` key and click on the location of interest with the right mouse button.

Once you have added the markers by using the `Ctrl` key and left or right mouse click, a panel called *Markers* will appear on the right side of the screen. This panel displays information about the markers, including the value of the registers at the left marker (Time 1) and the right marker (Time 2). It also shows the time and value difference between the two markers, which allows you to easily compare and analyze the data.


ModbusScope - demo.mbs
File
Project
View
Help
Register settings
Start
Stop
Zoom
Markers
Legend
Value
Register
Y1
41,5
Data point
Y2
168
Data point 2
Markers
Time 1: 00:00:24,032
Time 2: 00:00:38,060
Time Diff: 14,028
Data point
y1: 34
y2: 36
Diff: 2
Min: 34
Max: 36
Data point 2
y1: 153
y2: 157
Diff: 4
Min: 153
Max: 157
None
None
200
180
160
140
40
20
0
-20
00:00:14,400
00:00:21,600
00:00:28,800
00:00:36,000
00:00:43,200
Time
Scale options
X-axis
Full auto-scale
Sliding window
Interval (s)
20
Manual
Y1 axis
Y2 axis
Full auto-scale
Window auto-scale
Limit from
Min:
0
Max:
25
Manual
Stopped
Runtime: 0:00:49
Success: 100
Errors: 0

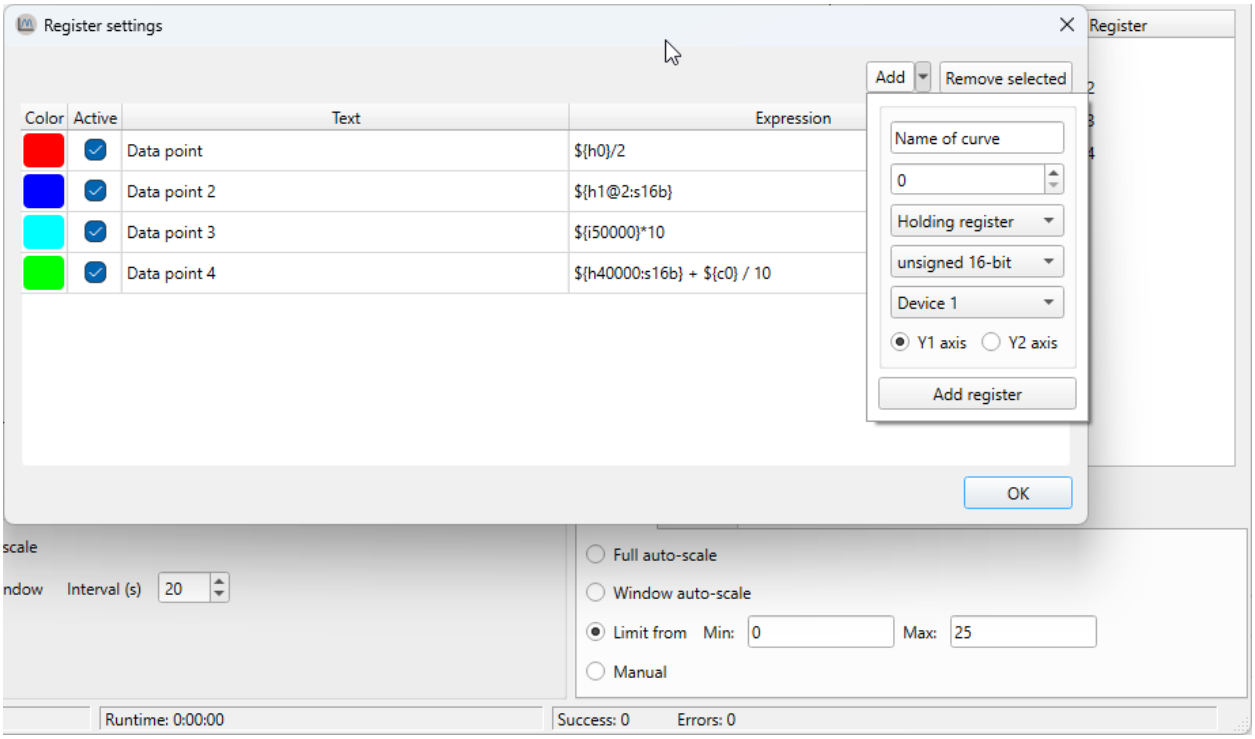
CHAPTER

FIVE

CONFIGURATION

5.1 Configure register settings

When you first open *ModbusScope*, no Modbus registers are added. To add registers, click on *Register Settings* in the toolbar on the interface. This will open a window where you can add and edit the registers that you want to monitor.



5.1.1 Add Modbus registers

In the *Register settings* dialog, you can add Modbus registers either manually or by importing them from a `.mbc` file. When you open a `.mbs` file in *ModbusScope*, all registers are added again, which allows you to continue working with the previously used registers.

Once the registers have been added, you can adjust them as needed. This includes:

- **Name and color**: Give each register a meaningful name and choose a color for its graph line
- **Expression**: Optionally define a calculation or transformation of the raw register data
- **Y-axis**: Select which Y-axis is used for this register

NOTE: The number of registers that are actively polled can greatly impact the sample rate. To achieve a higher resolution in time, you can either reduce the number of actively polled registers or make sure that the registers are in consecutive addresses so that they can be polled in one packet. This will help to increase the sample rate and improve the resolution of the data.

Object types

In *ModbusScope*, a register expression has the form `${REG[@DEVICE][:TYPE]}`:

- `REG`: the Modbus address (use either Modicon notation or prefix notation, see below).
- `@DEVICE`: optional device id to specify which configured device to read from. If not specified, the first device is used.
- `:TYPE`: optional data type/size: 16b, s16b, 32b, s32b, f32b (see list below). For coils/discrete inputs, the type is ignored.

Examples:

- `${45332}` → 16-bit unsigned, device 1
- `${45332: s16b}` → 16-bit signed, device 1
- `${45332@2: 32b}` → 32-bit unsigned, device 2
- `${45332@3}` → 16-bit unsigned, device 3

Register address

Two notations are supported:

- Modicon 5-digit (compact): encodes object type and address in one number (limited to 1–9999 per field).
- Prefix notation (full range): use a letter prefix and the full 0–65535 address, e.g. `c0-c65535`, `d0-d65535`, `i0-i65535`, `h0-h65535`.

ModbusScope will prefer the Modicon 5-digit form when the address fits; otherwise it will use the prefix form.

Modicon 5-digit notation	Prefix example	Object type	Modbus function code
1 - 9999	c0 - c65535	Coil	1
10001 - 19999	d0 - d65535	Discrete inputs	2
30001 - 39999	i0 - i65535	Input registers	4
> 40001	h0 - h65535	Holding registers	3

Supported register types

ModbusScope currently supports the following types for holding and input registers:

- `16b`: Unsigned 16-bit value
- `s16b`: Signed 16-bit
- `32b`: Unsigned 32-bit value
- `s32b`: Signed 32-bit value
- `f32b`: 32-bit float (IEEE 754)

For coils and discrete inputs, the register type is ignored. The endianness of 32-bit registers can be configured per device with the `32-bit little-endian` setting in the settings dialog.

5.1.2 Expressions

Expressions are used to define calculations or transformations of data in *ModbusScope*. They can be used to convert raw data from a Modbus register into a more meaningful value, or to perform mathematical operations on multiple registers. These expressions can be defined using several mathematical operators and functions, as well as references to specific registers. For example, you could use an expression to calculate the power usage of a system by combining and the voltage and current readings from two different registers into a single expression.

ModbusScope supports various binary operators that are commonly used in expressions such as `|` for bitwise OR, `&` for bitwise AND, `<<` for bitwise left shift, and `>>` for bitwise right shift. It also supports basic arithmetic operators like `+`, `-`, `*`, `/`, `%`, and `^` for addition, subtraction, multiplication, division, modulus and exponentiation respectively. In addition to the above operators, *ModbusScope* also supports hexadecimal numbers represented with the `0x` prefix and binary numbers represented with `0b` prefix. It also supports floating-point numbers; use either a decimal point (`.`) or a comma (`,`) as the decimal separator. The parser uses the first separator encountered in a number.

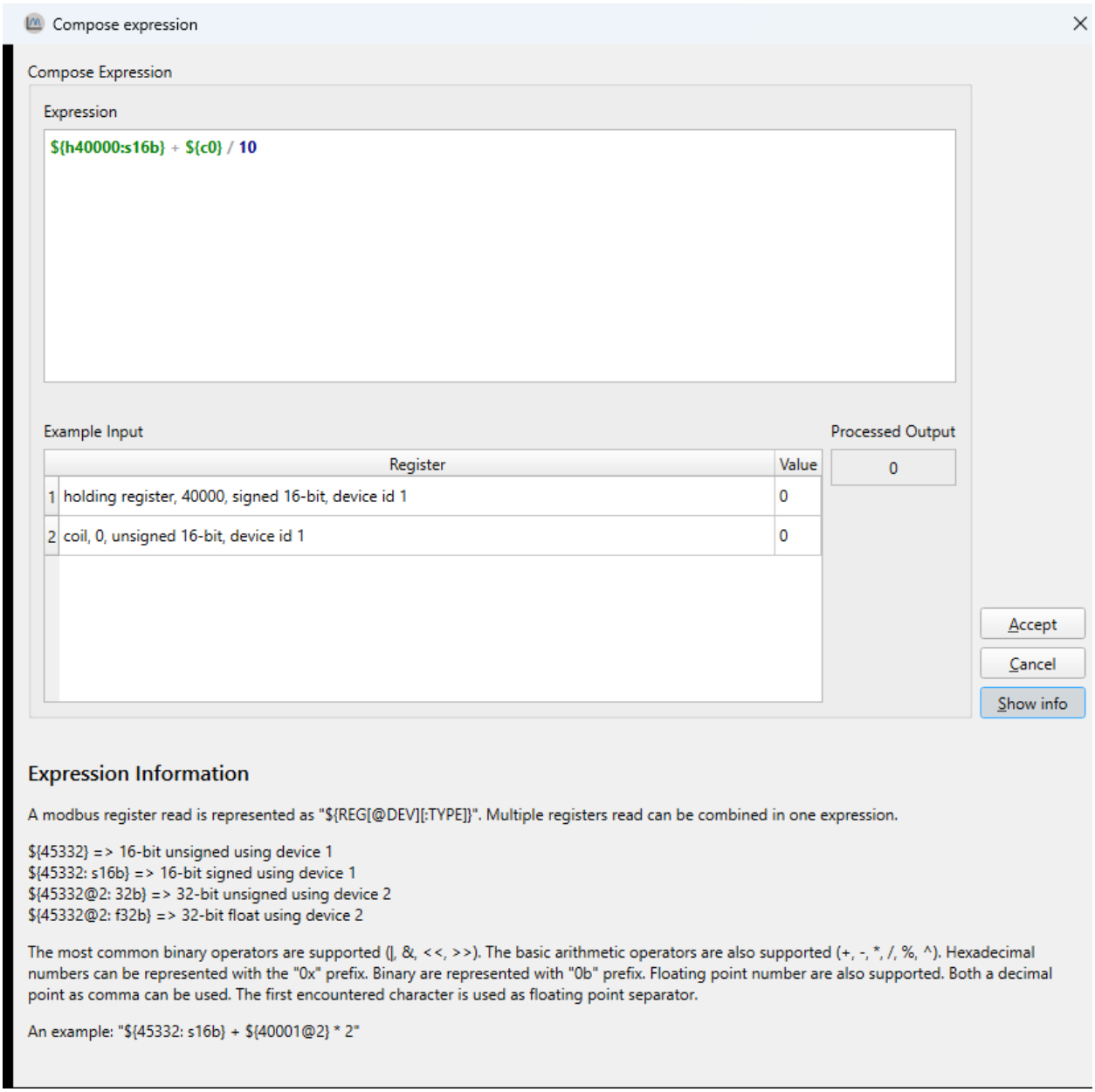
Some examples of valid expressions are

- `${40001:  s16b} + ${40002@2} * 2`
- `${40001:  s32b} * 1.1`
- `${30001} + 0x1000`
- `${40001} & 0b11111000`
- `(${30001} >> 8) & 0xFF`

5.1.3 Compose expression window

The *compose expression* window is a feature in *ModbusScope* that allows the user to create custom calculations using registers and other mathematical operations. Expressions allow for more flexibility in defining the data that is logged and displayed on the graph, and it can be used to create expressions that are specific to the user's needs. The compose expression window can be accessed from the register settings dialog, and it provides a user-friendly interface for creating and editing expressions.

This window can be opened by clicking the edit button (`...`) in the expression cell in the register settings dialog. This allows you to easily test and verify that the expression is functioning correctly before applying it to the logged data. The expression can be updated freely, and the register definition will be validated while you are entering it. When the register definition is green, it means that it is valid. The *example input* table can be used to enter values to test and verify the expression with actual values. This allows you to ensure that the expression is working as intended and to make any necessary adjustments before using it to log data. It is possible to combine multiple register reads in one expression by using mathematical operators and functions.

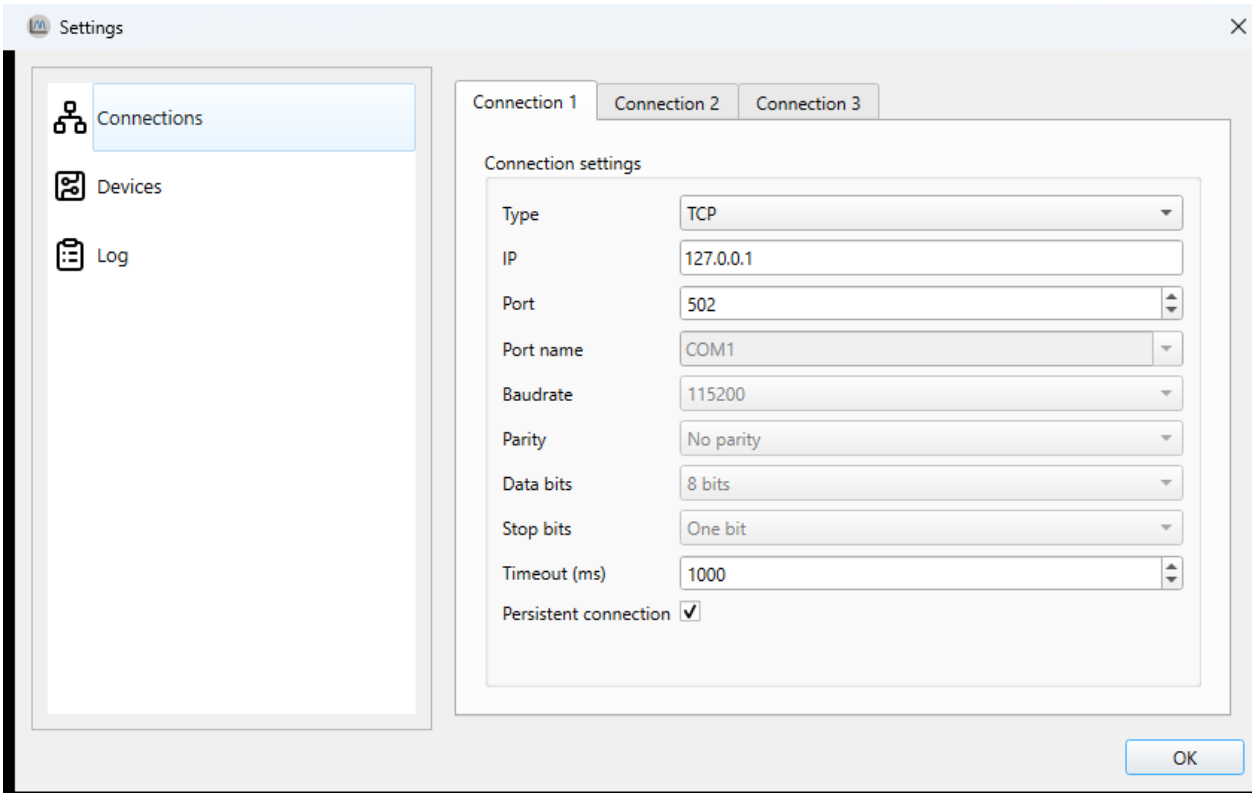


Expression error

If an expression or a test input causes an error, no output is shown in the *compose expression* window and the error message is displayed.

5.2 Configure connection settings

Open *Settings* > *Connection* to configure connection-related options.



Connections define **how** ModbusScope communicates with external devices. Each connection can be configured independently to use either TCP (Modbus TCP/IP) or RTU (serial) communication.

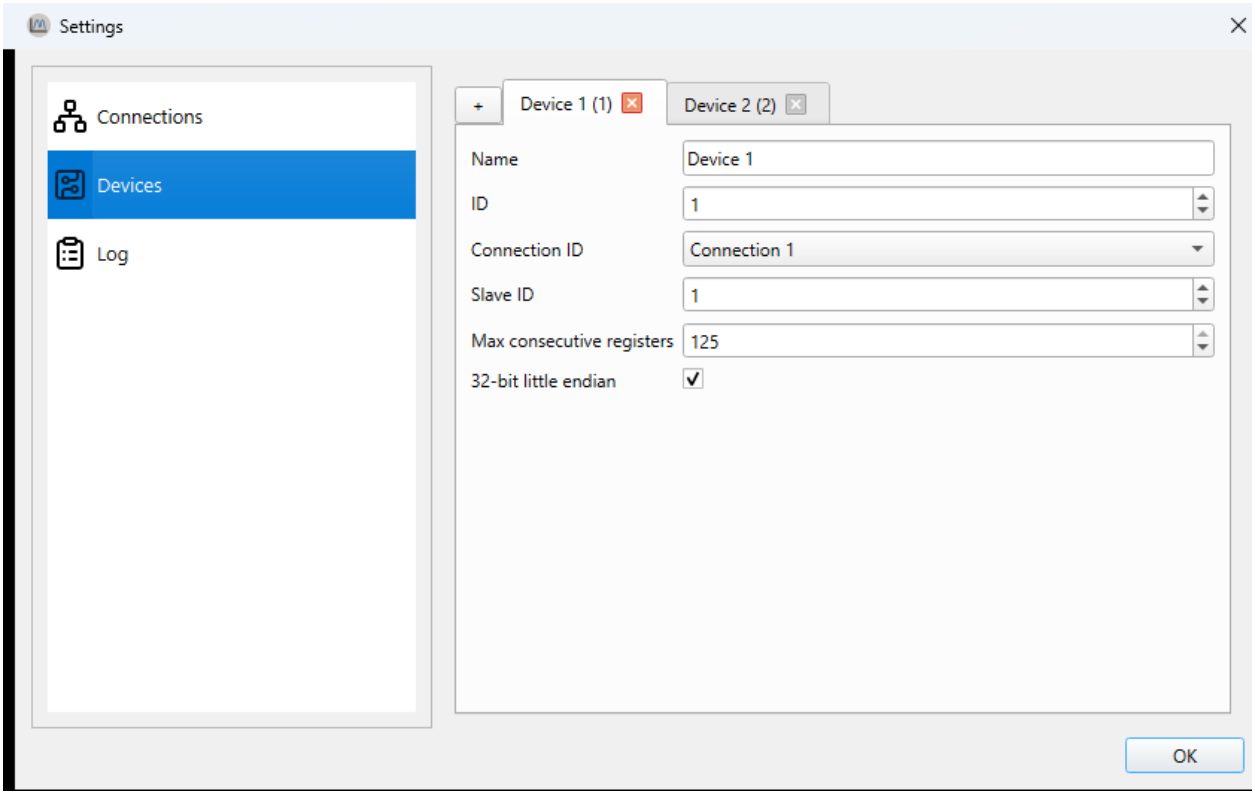
Common connection settings:

- **Enabled**: Enable or disable this connection.
  - Not possible for connection 1.
- **Protocol**: Select TCP for Modbus TCP/IP or RTU for Modbus RTU (serial). Modbus ASCII is not supported.
- **IP address (TCP only)**: The IPv4 address of the Modbus TCP slave.
- **Port (TCP only)**: TCP port used by the slave (default Modbus port is 502).
- **Serial port (RTU only)**: The local serial port name (e.g., COM1, /dev/ttyS0).
- **Baud rate (RTU only)**: Serial communication speed (e.g., 9600, 19200).
- **Parity (RTU only)**: Serial parity (None, Even, Odd) which must match the serial bus configuration.
- **Data bits (RTU only)**: Number of data bits per serial frame (commonly 7 or 8).
- **Stop bits (RTU only)**: Number of stop bits per serial frame (commonly 1 or 2).
- **Timeout**: How long the application will wait for a response from the slave before timing out.
- **Persistent connection**: When enabled, the connection stays open between poll cycles to reduce connection overhead. The connection will be reinitialized on error.

  Note: Devices are linked to connections in the Device settings. Each device specifies which connection to use for communication and configures device-specific parameters like Slave ID, Timeout, Max consecutive registers, and 32-bit endianness.

5.3 Device settings

Open *Settings* > *Device* to configure device-specific options and link devices to connections.



5.3.1 Understanding Devices

A device represents a single Modbus slave (external equipment like a heat pump, sensor, or controller) that you want to communicate with. Each device has its own configuration settings and is linked to a communication connection. This separation enables you to:

- Configure multiple devices with different Modbus settings, even if they share the same connection
- Manage Modbus protocol parameters on a per-device basis

5.3.2 Device Configuration Options

The device settings control parameters that affect Modbus protocol behavior:

- **Name**: A user-friendly identifier for the device
- **ID**: ID of device (used as reference for registers)
- **Connection ID**: Which connection to use for communicating with this device
- **Slave ID**: The Modbus unit/slave address (typically 1, but can be different for each device on the same physical connection)
- **Timeout**: How long ModbusScope will wait for a response from the slave before reporting a timeout error
- **Max consecutive registers**: Limit on how many consecutive registers to request in a single Modbus read (some devices limit this due to memory constraints)

- **32-bit little-endian**: Controls whether 32-bit values are stored in little-endian or big-endian byte order (must match your device's configuration)

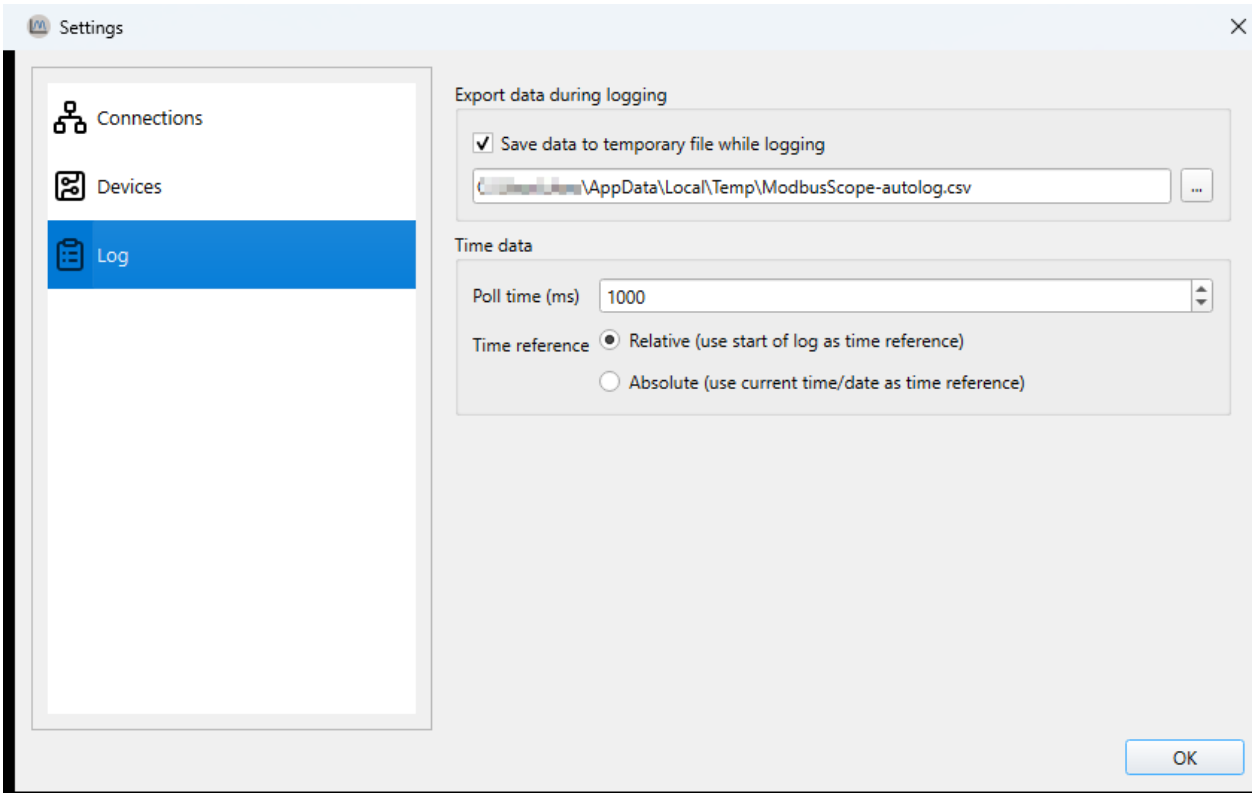
5.3.3 Working with Devices

When you add a register, you specify which device it belongs to. ModbusScope will use the device's settings (slave ID, timeout, connection, etc.) when reading that register. This allows different registers to use different devices and different protocol settings as needed.

> **NOTE**: In Register settings, you link a register to a device by using the `@DEVICE` syntax in the expression (e.g., `${40001@2}` or `${40001@3}`). You must configure at least one device before adding registers.

5.4 Configure log settings

Open *Settings* > *Log* to configure logging options (sample rate, temporary file behavior, time stamp mode).



ModbusScope creates a data file in the general temporary folder by default when a logging session is started. The data points are appended to the file during the logging session so that the data can be recovered in case of an unforeseen crash or if the user forgets to save the data before quitting the application. The temporary file is cleared every time a polling session is started so that new data can be logged. Some of this behavior can be customized in the *log settings*. The user can choose to disable the feature or change the location of the temporary data file. This allows the user to ensure that the data is saved in a location that is convenient for them.

By default, *ModbusScope* will log data points every 250 milliseconds. This is the default sample rate, and it can be adjusted in the *log settings*. The user can increase or decrease the sample rate to suit their needs. Additionally, by default, *ModbusScope* will log timestamps relative to the start of the log session. This means that the time stamp of each data point is recorded as the time elapsed since the start of the logging session. However, this behavior can be changed by enabling the *use absolute times* option in the *log settings*. When this option is enabled, absolute timestamps are logged instead, meaning that the actual date and time of each data point is recorded in the log file. This configuration

allows you to choose the time stamp format (relative or absolute) that is most appropriate for your use case and to easily compare logged data collected at different times.

5.4.1 Optimize logging interval

The minimum logging interval is determined by several factors such as the Modbus protocol and the register addresses. When the requested register addresses aren't in successive order, the Modbus protocol has an inherent slowdown and *ModbusScope* will split the read request into several packets. This will negatively impact the minimum logging interval because of the Modbus end of frame timeout. To achieve a fast logging interval, it's important to limit the number of registers and make sure that consecutive registers are polled. This can allow for a faster logging interval.

CHAPTER

SIX

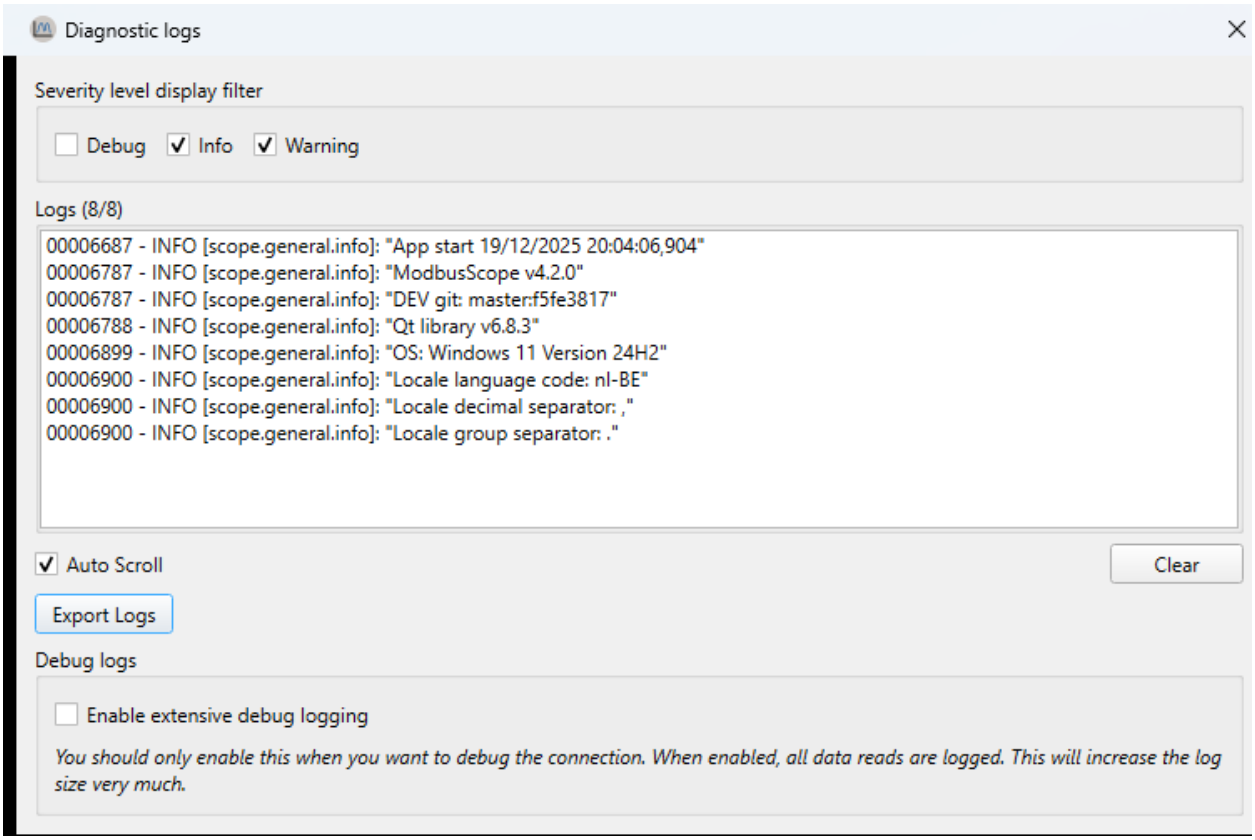
DIAGNOSTICS

6.1 Diagnostic logging

During a logging session, various errors may occur. To facilitate quick examination and resolution of these errors, *ModbusScope* has built-in logging capabilities. For example, if a slave responds too slowly, a timeout may be generated. All register reads are recorded in the diagnostic logs along with the results, allowing communication or connection errors to be easily examined. Other common errors include Modbus exceptions, such as when a register configuration is incorrect. The log will also include basic information about *ModbusScope* and the system it is running on, such as the operating system and its specific version.

6.2 Logs window

Opening the *diagnostic logs* window can be done via *Help* > *Diagnostic logs...*.



The logs can be viewed in the logs window. The log list will dynamically update as new logs are added. By using the filters, specific categories of logs can be hidden or shown. Specific logs can be selected and copied to the clipboard by right-clicking on them. Alternatively, all logs can be exported using the *Export Logs* button. Additionally, extensive debug logging can be enabled to log extra (internal) information about the status of every read. However, this option will dramatically increase the number of logs generated.

CHAPTER

SEVEN

IMPORTING AND EXPORTING

7.1 Saving and opening configuration

The configuration of registers, as described in previous sections, can be saved and loaded in a project file. These settings can be saved in a `.mbs` file by going to *File* > *Save Project As...* and loaded by going to *File* > *Open Project...* or by dragging a `.mbs` file into the application.

7.1.1 Deprecation notice

Starting from *ModbusScope* v4, the format of the `.mbs` project file has changed in an incompatible manner. This was required to support the full 16-bit address range of all object types. Project files saved with *ModbusScope* v3.x.x will be automatically converted on opening.

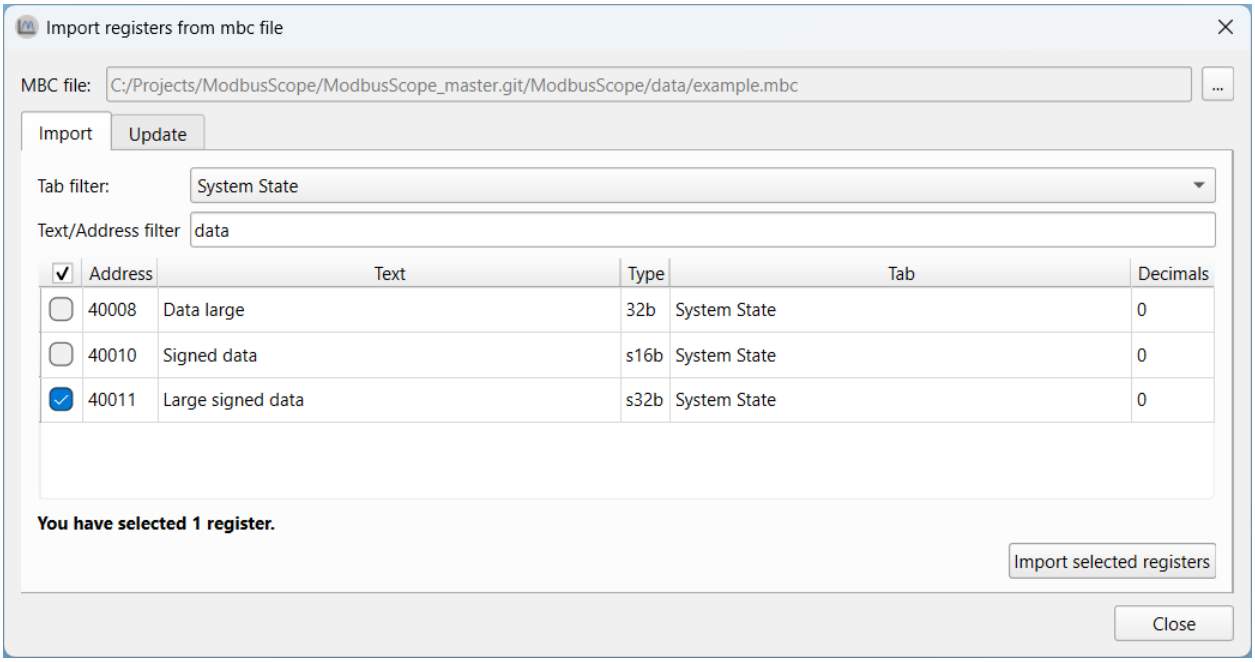
Support for older project files (pre v3.x.x) has been dropped. It is no longer possible to use these project files in the latest version of *ModbusScope*. A workaround would be to open the project file in ModbusScope v3.x.x and save the project file. The resulting project file will be updated and is compatible with the v4.x.x version.

7.2 Exporting data/image

Current log results can be exported as an image or as data (`.csv`) file. You can select either *File* > *Save Data File As...* or *File* > *Export Image As...* to do so. It is important to note that saving a project/data file or exporting an image can only be done when logging is not active.

7.3 Import register definitions from *mbc* file

ModbusControl is a proprietary application not available to the public. It can be used to read and write data from Modbus slaves. It is possible to import the register definitions from a *ModbusControl* project file (`.mbc`) into *ModbusScope* by clicking on *Import from .mbc file* in the register dialog or by dragging and dropping the `.mbc` file into the main screen of *ModbusScope*. This makes it easy to add register definitions to *ModbusScope* from a *ModbusControl* project file.

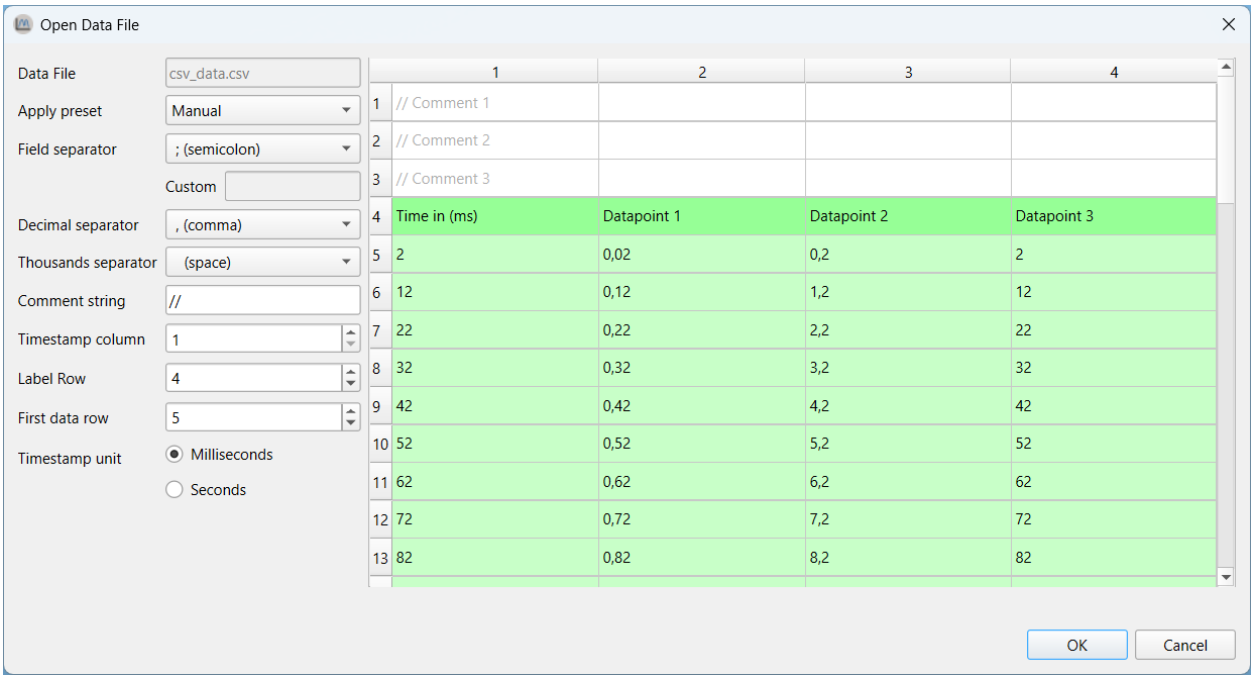

Import registers from mbc file
MBC file:
C:/Projects/ModbusScope/ModbusScope_master.git/ModbusScope/data/example.mbc
...
Import
Update
Tab filter:
System State
Text/Address filter
data
Address
Text
Type
Tab
Decimals
40008
Data large
32b
System State
0
40010
Signed data
s16b
System State
0
40011
Large signed data
s32b
System State
0
You have selected 1 register.
Import selected registers
Close

CHAPTER

EIGHT

OPEN DATA FILE

When a data file that was created by *ModbusScope* is opened, the data will be loaded automatically. This can be done by selecting *File* > *Open Data File…* or by dragging the file into the application. If a non-*ModbusScope* csv export file is detected, the program will automatically determine the necessary settings for parsing the file. This includes the field and decimal separators. These settings can be adjusted manually if needed. Once the settings are configured, the rest of the file will be loaded and the data can be viewed in the graph as with a normal *ModbusScope* logging session.



8.1 Parse settings

The format of a `.csv` file can vary, so correct settings must be in place for parsing the file. These settings can be adjusted and the results of the parsing will be reflected in the grid display.

8.1.1 Locale related

Depending on the configured region, the format of the data will be different. To provide maximum flexibility when opening a data file, *ModbusScope* allows you to select these settings freely.

- **Field separator**: Symbol used to separate data fields from each other.
- **Decimal separator**: Symbol used to separate the integer part from the fractional part of a number written in decimal form (e.g., "." in 12.45).

- **Thousand separator**: Symbol used to separate groups of digits (e.g., "." in 1.000).

8.1.2 File structure related

ModbusScope can read data from a `.csv` file if it's formatted similarly to the files exported by *ModbusScope*. This includes having a column for timestamps and one or more columns for data.

Supported data file format

```
Time (ms);Datapoint 1;Datapoint 2;Datapoint 3
2;0,02;0,2;2
12;0,12;1,2;12
22;0,22;2,2;22
32;0,32;3,2;32
```

Settings

Since there is no standard for the contents of a `.csv` file, some settings need to be filled in to correctly parse time data.

- **Comment string**: Symbol(s) at the beginning of a line that indicate(s) that a line should be ignored when parsing. Shouldn't be longer than 2 characters.
- **Time stamp column**: Sometimes the time data isn't in the first column. This setting can be used to select the correct column. All columns to the right of the time data column will be parsed for data
- **Label row**: This setting indicates the row with the labels (names) for the graphs. It can be used to skip header lines in the file. The label row should contain the same number of fields as the data rows.
- **Data row**: Similar to the label row, but for the data

8.1.3 Functionality

ModbusScope adds some extra functionality when opening a data file. The *time-stamp unit* can be selected between milliseconds and seconds. When seconds is selected, the time-stamp will be converted to milliseconds during the load process.

8.2 Presets

When analyzing several data files of which the settings can't be auto-detected, it's useful to save the settings as a preset. *ModbusScope* allows you to create a configuration file with custom presets. This configuration file loads when opening a datafile, and you can select the correct preset. You can also configure a keyword per preset; if a data file name contains the keyword, the preset will be automatically selected.

8.2.1 Locations

ModbusScope searches for the `presets.xml` configuration file in 2 specific locations. The first location is the documents folder of the current Windows user: `C:/Users/<USER>/Documents/`. The preset configuration file should be in a subfolder named *ModbusScope*. If the preset configuration file isn't found in either location, *ModbusScope* will try to find the file in the same location as the main executable. If unsuccessful, *ModbusScope* will use the built-in presets. For valid preset configuration files found, the built-in presets will be replaced with the one specified in the file.

8.2.2 Keyword

As mentioned before, a preset can be automatically selected based on the presence of a keyword in the name of the data file.

8.2.3 Example preset configuration file

```
<modbusscope>
 <parsepreset>
  <name>Default (be)</name>
  <keyword>-be</keyword>
  <fieldseparator><![CDATA[;]]></fieldseparator>
  <decimalseparator><![CDATA[,]]></decimalseparator>
  <thousandseparator><![CDATA[ ]]></thousandseparator>
  <commentSequence><![CDATA[//]]></commentSequence>
  <column>1</column>
  <labelrow>1</labelrow>
  <datarow>2</datarow>
 </parsepreset>

 <parsepreset>
  <name>be-seconds</name>
  <fieldseparator><![CDATA[;]]></fieldseparator>
  <decimalseparator><![CDATA[,]]></decimalseparator>
  <thousandseparator><![CDATA[ ]]></thousandseparator>
  <commentSequence><![CDATA[//]]></commentSequence>
  <column>1</column>
  <labelrow>1</labelrow>
  <datarow>2</datarow>
  <timeinmilliseconds>false</timeinmilliseconds>
 </parsepreset>

</modbusscope>
```

CHAPTER

NINE

RELEASE NOTES

The latest *ModbusScope* installer or standalone version can always be downloaded from the release page.

9.1 v4.2.2 (28/04/2026)

9.1.1 Changed

- Update URL for update check

9.2 v4.2.1 (31/03/2026)

9.2.1 Fixed

- Fix incorrect data displayed when polling multiple devices on the same connection (e.g. multiple slave IDs on a serial bus)

9.2.2 Changed

- Rework diagnostic logging to generate fewer logs when extensive logging is disabled
- Improve copying diagnostic logs to the clipboard
- Improve enabling debug logs

9.3 v4.2.0 (09/01/2026)

9.3.1 Added

- Added concept of devices to allow polling multiple devices on a single connection

9.3.2 Changed

- Major rework of settings window
  - Combined several settings dialogs into one window
- Major refactor of documentation and user manual
- Improve note handling (more padding, round values to less decimals)
- Update default colours of graphs

9.3.3 Removed

- Replaced workaround to force light theme with standard function

9.4 v4.1.1 (11/09/2025)

9.4.1 Added

- Warn when registers are selected, but not imported yet when closing import MBC dialog

9.4.2 Fixed

- Always use light mode even when system is configured as dark mode (Github #372)

9.5 v4.1.0 (03/07/2025)

9.5.1 Added

- Add option to highlight graph
- Add option to compare/import changes from mbc with existing mbs (Github #342)

9.5.2 Changed

- Select consistent theme across platforms
- Various UI changes to improve consistency (scale dock, about window, . . .) (Github #349)
- Make sure color of register remains visible when register is selected in register window
- Various improvements to importing MBC registers (Github #344)

9.6 v4.0.1 (23/12/2024)

9.6.1 Added

- Diagnostic dialog is opened when stats section in status bar is clicked
- Add most recent project file menu (Github #222)
- Add median poll time of last 50 samples to status bar
- Support other files with “open with” (Github #329)

9.6.2 Changed

- Scaling settings of secondary value axis is now saved in project file

9.7 v4.0.0 (18/06/2024)

9.7.1 Added

- Add extended register syntax to handle full address range (e.g `${h65535}`)
- Highlight expression syntax error in register dialog
- Implement easier editing of expression in register dialog

9.7.2 Fixed

- Fixed warning when loading mbs file with unsupported datalevel
- Don't deselect axis when Ctrl key is pressed (Github #304)

9.7.3 Changed

- Improve formatting of large and small values (Github #287)

9.7.4 Removed

- Removed correction of corrupt values (STMStudio)
- Removed support for opening very old mbs files (ModbusScope v2.x.x or earlier)
  - Old mbs files can be updated by opening and saving using ModbusScope v3.x.x.

9.8 v3.9.0 (04/12/2023)

9.8.1 Added

- Add support for other object types (discrete output coils, discrete input contacts, input registers, holding registers)

9.9 v3.8.1 (12/08/2023)

9.9.1 Added

- Allow in place editing of expressions

9.9.2 Changed

- Allow screenshot during logging (Github #280)

9.10 v3.8.0 (02/06/2023)

9.10.1 Added

- Improve visibility of errors (highlight curve in legend)

9.10.2 Fixed

- Fixed diagnostic log cleared when "extensive logging" is disabled (Github #265)

9.10.3 Changed

- Change add register dialog (wizard) to drop-down frame
- Keep focus of selected register when filtering during mbc import (Github #169)

9.11 v3.7.0 (03/02/2023)

9.11.1 Changed

- Rework toolbar (new icons and remove some actions)

- Update notification is now only visible after 14 days since release
- Improve marker indicators (Z-order)
- Select value axis scale options tab on axis selection (Github #253)
- Update dependencies

9.11.2 Added

- Add support for 32-bit floating point type (Github #250)
- Add indicator on axis to indicate value axis configuration of curve

9.12 v3.6.3 (21/11/2022)

9.12.1 Fixed

- Fix menu on incorrect screen (Github #248)

9.13 v3.6.2 (06/11/2022)

9.13.1 Fixed

- Fix marker data calculations (Github #240)
- Fix markers in combination with value axis (Github #240)

9.14 v3.6.1 (04/10/2022)

9.14.1 Fixed

- Fix zooming of axis with mouse wheel (Github #234)

9.15 v3.6.0 (02/10/2022)

9.15.1 Changed

- When starting only set scaling to auto when it is set to manual (Github #210)
- Improve import/export of csv data (especially when modifying file in Excel) (Github #220)
- Remove unused space in scale dock

9.15.2 Added

- Second Y-axis on right-side (Github #188)
- Add option to quickly add a register

9.15.3 Fixed

- Fix blurry tick labels in Qt6 build
- Fix crash when there is no data and marker is active (Github #223)
- Fix crash when a register is added and data is present (Github #229)

9.16 v3.5.1 (11/05/2022)

- Crash on showing tooltip with empty graphs (Github #208)
- Incorrect tab stop order when adding new registers (Github #209)
- Revert Windows build back to Qt5.15.2 to fix blurry tick labels

9.17 v3.5.0 (04/05/2022)

- Convert min/max settings of Y axis to floating point (Github #183)
- Add *Save Project* menu item
- Implement changing order of registers by drag and drop (Github #78)
- Allow scrolling to negative times when there is negative time data (Github #198)
- Update to Qt6

9.18 v3.4.0 (02/03/2022)

- Allow full 16-bit addressing of holding registers (Github #181)
- Expand expressions with register definition to provide more flexibility for user (Github project #6)
  - This changes adds the ability to combine multiple registers from potential multiple connections in one curve.
- Add syntax highlighting to expressions, including invalid token (Github #142)

9.19 v3.3.1 (23/12/2021)

- Select next register row after deletion in register dialog
- Improve note positioning when sliding window (Github #168)
- Hide markers on data load (Github #171)
- Fix opening project file via menu (Github #173)
- Let text filter also work on register number during mbc import (Github #174)

9.20 v3.3.0 (08/10/2021)

- Various user experience improvements
  - Replace check boxes with radio buttons
  - Rename menu items
  - Add extra menu items to tool bar
- Reload previous mbc file when import mbc dialog is opened (Github #156)
- Improve documentation (fixed Github #161 and open data file chapter)
- Fix that marker points are visible when graph isn't (Github #157)
- Include connection id in duplicate check when opening mbs file (Github #164)
- Updated used libraries and internal improvements

9.21 v3.2.1 (15/04/2021)

- Fix crash when starting ModbusScope with project file (Github #155)

9.22 v3.2.0 (03/04/2021)

Improvements

- Add more visible update notification
- Make documentation and project page (with issue) more visible (Github #152)

Fixes

- When data file is loaded, reset value in legend after inspection with `Control` key
- Reset scaling to auto when clearing data

9.23 v3.1.0 (23/02/2021)

Improvements

- Add support for modbus RTU (serial port)
- Add options to export diagnostic log
  - Copy specific logs to clipboard
  - Export complete log to file (Github #135)
- Minor graphical tweaks to markers

9.24 v3.0.0 (14/11/2020)

Improvements

- Replace fixed operations (multiply, divide, shift, ...) with custom user-defined expression
  - Link to doc
  - Fixed operations will be automatically converted to custom expression on project load
- Rework and move the user manual to ReadTheDocs
- Expand information in logs
- Improve handling of large time periods (Github #139)
  - Don't wrap around when period is larger than one day

Backward compatibility When loading old ModbusScope files (pre v3.x.x), the existing operations (multiply, divide, shift, bitmask, ...) will be converted to a single expression that is used in the new ModbusScope. This conversion makes sure that users won't notice any differences in functionality.

When exporting the settings (project file), the new expression will be saved. Older ModbusScope versions won't be able to parse this expressions. The operations will be reset to the defaults, but other register info will be loaded correctly.

9.25 v2.1.1 (03/07/2020)

- Fix update check (add OpenSSL dll to install) (Github #136)

9.26 v2.1.0 (15/06/2020)

Defects

- Fix tooltip

Improvements

- Add support for 32 bit registers (Github #129)
- Add support for persistent connection (default on) (Github #18)
- Minimize scale dock
- Disable bit mask for signed numbers
- Rework logging to be able improve logging in the future

9.27 v2.0.0 (03/03/2020)

Internal (code changes)

- Add more tests
- Fix most issues reported by static code analysis (Coverity)
- Change linking from static to dynamic
  - Application changes from one large executable to a smaller executable with extra dll's

Features

- Added possibility to poll 2 different slaves in the same log
- Added possibility to change graph color from legend
- Added filter (error/info) in diagnostics window
- Added mbc filter based on description / register number
- Toggle markers option
- Rectangle zoom
- Integrated most used features from GraphViewer
  - More flexible configuration of parsing settings
  - Advanced auto detection of parsed settings
  - Presets of parse settings
- Improved file loading/parsing
  - Improve file loading windows (keep showing raw data on invalid parse settings) (Github #120)
  - Speed up loading large data file (Github #121)
  - Add progress bar on file load (Github #122)

Bugs

- Small bug fixes (Github #111)

9.28 v1.6.1 (06/04/2019)

Bugfixes

- Fix error when writing notes to imported data file (Github #109)
- Automatically remove field separator from register names (Github #106)

Improvements

- When importing a mbc file already selected registers are now disabled dynamically.

9.29 v1.6.0 (25/01/2019)

Most of the work in this release isn't visible for the user. A complete rework of the communication module has been done. libmodbus was dropped in favor of Qt Modbus. The communication module has also been completely reworked to be able to implement integration and unit testing.

9.30 v1.5.0 (02/10/2018)

Features

- Add support for notes (small texts in graph)
  - Editable (even after data load)
  - Saved with data in csv
- Update legend component (thanks to @Fornax)
- Add tab filter when importing mbc files (implements Github #96)

Bug fixes

- Fix slow drag issue (Github #104)
- Keep visibility state when adding/removing graphs (Github #102)
- Absolute time is off by 2 hours (Github #103)

Under the hood

- Add initial unit tests for some modules
- Add Travis build

9.31 v1.4.0 (17/02/2018)

Features

- Rework Modbus communication code
- Update libraries
  - QCustomPlot v2.0.0 (final)
  - libmodbus to v3.14
- Improve support for absolute timestamp

- Add option to show/hide all graphs (Github #99)
- Add logging (Github #71)

Bugfixes

- Fix some minor bugs (Github #95) (thanks to @pluyckx), (Github #89)

9.32 v1.3.0 (01/04/2017)

Features

- Rework tooltip (show value under cursor in legend when control key is pressed) (Github #90)
- Import registers from mbc file (drag and drop or button in register dialog) (Github #91)
- Add window auto scale on y-axis (Github #36)
- Add meta data when exporting data (keep color when importing) (Github #63)
- Added extra marker calculations (minimum, maximum, average, median, slope, ...) (Github #79)
- Use delete button to remove registers (Github #34)
- Improve communication (only split Modbus read on specific Modbus exception)
- Update QModbusPlot to v2.0.0 (beta)
- Add command line argument to enable OpenGL (–opengl)

Bugs

- Make sure legend window (when docked) is present on screenshot (Github #80)
- Small fixes (Github #82, Github #83, Github #85, Github #88)

9.33 Older releases

Older releases can be found on Github