Modbus Wizard - Read Me

Introduction

This document provides an overview of the Modbus Wizard program, its functionalities, and instructions on how to make it executable. The program integrates three main functionalities into a single tool for working with Modbus devices: Modbus TCP Register Scan, Brute Force Connection for Modbus RTU Devices, and Modbus RTU Scanner.

Libraries and Their Versions

Library	Version	Purpose
Tkinter	Built-in Python library	Provides the GUI framework for creating windows, dialogs, buttons, and other interface elements.
Pymodbus	2.5.3	A Python library for Modbus protocol support, enabling communication with Modbus devices. The specific version is required for the RTU handshake.
MinimalModbus	2.1.1	A simple and easy-to-use Modbus RTU/ASCII implementation for Python, used for the auto-detection wizard to brute force connection parameters.
PySerial	3.5	Provides serial port communication capabilities, used for the RTU connection.
Logging	Built-in Python library	Provides logging capabilities to track and debug the execution of the program.
Threading	Built-in Python library	Used to handle concurrent execution, allowing the program to perform scans in the background.
Struct	Built-in Python library	Provides functions to convert between Python

		values and C structs represented as Python bytes objects.
Time	Built-in Python library	Used to handle timing functions such as delays.
FileDialog	Built-in Python library	Provides file dialog interfaces to save and load files.

Explanation of the Program

Modbus TCP Register Scan

Objective: Implement a comprehensive scan of all available registers on a Modbus TCP device. This is crucial because the registers provided by the manufacturer may sometimes be inaccurate or incomplete.

Implementation:
- Scan all available registers from the start to the end address.
- Include any register that is defined on the device.
- If a register is not defined, mark it as 'not defined'.
- The results are displayed in the GUI, showing defined and undefined registers.

Brute Force Connection for Modbus RTU Devices

Objective: Establish a connection to Modbus RTU devices when communication parameters (baud rate, parity, stop bits) are unknown.

Implementation:
- Use a brute force approach to try all possible communication settings (baud rate, parity, stop bits) until a successful connection is established.
- Display the connection settings used upon successful connection.
- The auto-detection wizard attempts various combinations of settings and scans the registers to verify connectivity.

Modbus RTU Scanner

Objective: Scan Modbus RTU devices using known communication parameters.

Implementation:
- Allow the user to specify the communication parameters (port, baud rate, parity, data bits, stop bits, timeout).
- Connect to the device and scan the specified range of registers.
- Display the defined and undefined registers in the GUI.
- Support for multiple formats (decimal, binary, hex, float, etc.) for displaying register values.

Drive to Make the Program

The motivation behind creating this program is to provide a comprehensive and user-friendly tool for working with Modbus devices. Many existing tools rely on manufacturer-provided registers and known communication parameters, which may not always be accurate or available. This program addresses these limitations by:

1. Ensuring Comprehensive Scans:
   - Scanning all available registers ensures that users can identify all active registers on their Modbus devices, even if the manufacturer's documentation is incomplete or inaccurate.
2. Brute Force Connection:
   - The brute force approach for Modbus RTU devices ensures that users can establish a connection even when the communication parameters are unknown, which is a common challenge in fieldwork.
3. Ease of Use:
   - The integrated GUI makes it easy for users to interact with their Modbus devices without needing to write code or use complex command-line tools. The tool is designed to be accessible to both technical and non-technical users.

By combining these functionalities into a single program, users are provided with a robust and versatile tool for their Modbus communication needs.

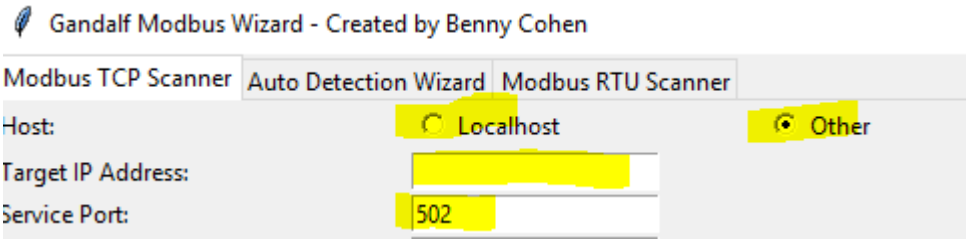
Modbus Wizard - Modbus TCP Scanner User Guide

Overview

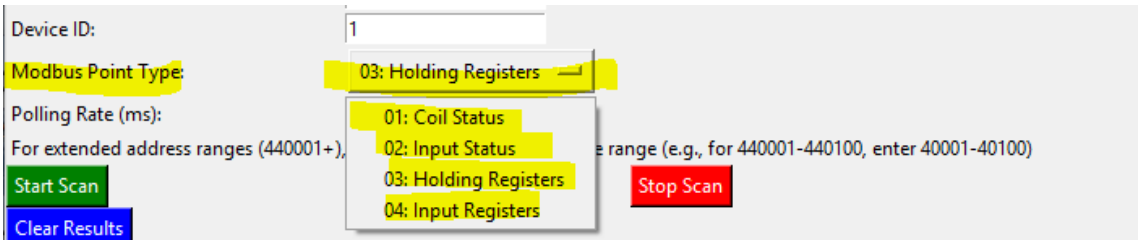
The Modbus TCP Scanner tab allows you to scan and read Modbus TCP registers easily. Here's a quick guide to using it.

Interface Elements & Usage

1. **Host Selection**:
    - **Localhost**: For testing with simulated values from Modsim.
    - **Other**: For real devices, select this and input the device's IP address.



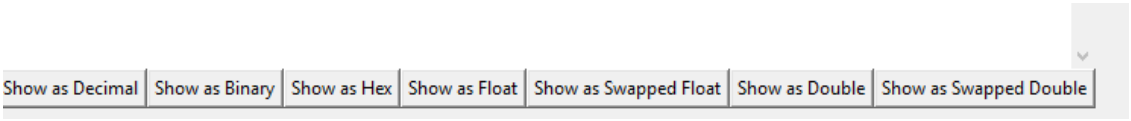
2. **Target IP Address**:
    - Enter the IP address when "Other" is selected.
3. **Service Port**:
    - Default is `502`, commonly used for Modbus. Adjust if your device uses a different port.
4. **Start and End Address**:
    - Define the range of registers to scan. You can scan hundreds of registers at once, more flexible than some tools.
5. **Device ID**:
    - Default is `1`. Change if needed to match your device.
6. **Modbus Point Type**:
    -
    - Select from Coil Status, Input Status, Holding Registers, or Input Registers using the dropdown menu.



7. **Polling Rate**:
    - Set the scanning rate in milliseconds. Default is `50ms`. Adjust if necessary.
8. **Buttons**:
    - **Start Scan**: Begins the scanning process.
    - **Stop Scan**: Halts the scanning.
    - **Clear Results**: Clears the displayed results.
    - **Download Log**: Saves the log file for reviewing scan results and errors.
9. **Result Display**:

- Shows the results of the scan. Undefined registers or errors are displayed in red for easy identification.

10. **Format Buttons**:
    -
    - Toggle between different formats (Decimal, Binary, Hex, Float, Swapped Float, Double, Swapped Double) to view the register values as per your requirement.



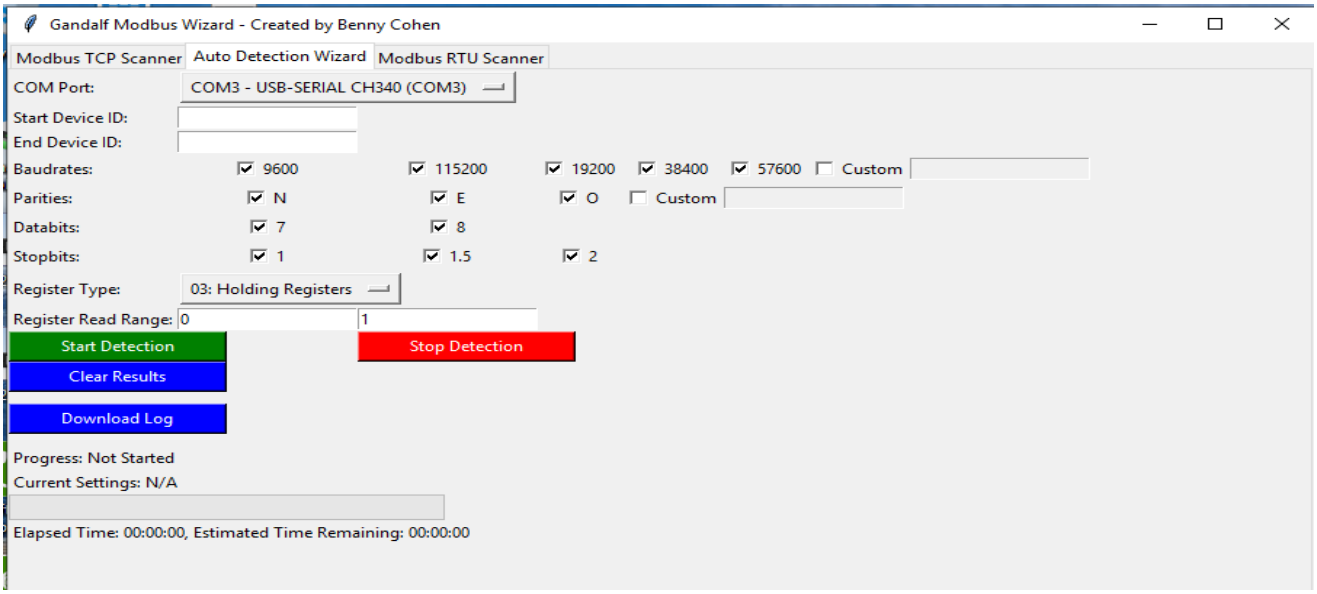
Modbus Wizard - Auto Detection Wizard User Guide

Overview

The Auto Detection Wizard helps you automatically detect Modbus RTU device settings and registers. This guide provides a quick overview of the interface elements and their usage.

Interface Elements & Usage

1. **COM Port**:
   - Select from the dropdown list. Ensure your device is connected and drivers are installed if it is not listed.
2. **Start Device ID / End Device ID**:
   - Define the range of slave IDs to scan. Modbus RTU devices often have IDs different from 1.
3. **Baudrates**:
   - Select common baud rates or enter a custom one. Multiple selections increase scan time.
4. **Parities**:
   - Choose from None (N), Even (E), Odd (O), or enter a custom parity.
5. **Databits**:
   - Typically set to 8, but ensure it matches your device settings.
6. **Stopbits**:
   - Choose 1, 1.5, or 2 as per your device configuration.
7. **Register Type**:
   - Select either Holding Registers or Input Registers based on what you need to scan.
8. **Register Read Range**:
   - Define the range of registers to scan. A minimum range of 2 registers is required to avoid handshake issues.
9. **Buttons**:
   - **Start Detection**: Begins the detection process.
   - **Stop Detection**: Stops the detection process.
   - **Clear Results**: Clears the displayed results.
   - **Download Log**: Saves the log file for reviewing detection results and errors.
10. **Progress & Current Settings**:
    - Displays the progress, current settings, and estimated time remaining during the scan.



Modbus Wizard - Modbus RTU Scanner User Guide

Overview

The Modbus RTU Scanner helps you scan and read Modbus RTU devices by establishing a serial connection. This guide provides a brief overview of the interface elements and their usage.

Interface Elements & Usage

1. **COM Port**:
    - Select the COM port from the dropdown list. Ensure your device is connected and drivers are installed if it is not listed.
    - If you used the Auto Detection Wizard first, the serial parameters will already be selected, and you should have an established serial connection.
2. **Baud Rate**:
    - Choose the baud rate from the predefined options or enter a custom one if necessary.
3. **Parity**:
    - Choose the parity bit: None (N), Even (E), Odd (O), or custom.
4. **Data Bits**:
    - Select the number of data bits, typically 8.
5. **Stop Bits**:
    - Choose the number of stop bits, typically 1.
6. **Timeout (s)**:
    - Enter the timeout duration in seconds for communication.
7. **Start Address / End Address**:
    - Define the range of registers to scan.
8. **Device ID**:
    - Enter the slave ID of the device.
9. **Modbus Point Type**:
    - Select the type of Modbus point to scan, such as Holding Registers or Input Registers.
10. **Polling Rate (ms)**:
    - Set the polling rate in milliseconds.
11. **Batch Size**:
    - Define the number of registers to read in one batch.
12. **Buttons**:
    - **Connect**: Establishes the serial connection.
    - **Disconnect**: Terminates the serial connection.
    - **Start Scan**: Begins the scanning process.
    - **Stop Scan**: Stops the scanning process.
    - **Clear Results**: Clears the displayed results.
    - **Download Log**: Saves the log file for reviewing scan results and errors.
13. **Results Display**:
    - Shows the scan results in the specified format (Decimal, Binary, Hex, Float, Swapped Float, Double, Swapped Double).

Gandalf Modbus Wizard - Created by Benny Cohen

Modbus TCP Scanner | Auto Detection Wizard | Modbus RTU Scanner

COM Port: COM3

Baud Rate: 19200

Parity: E

Data Bits: 8

Stop Bits: 1.0

Timeout (s): 1

Start Address: 1

End Address: 55

Device ID: 1

Modbus Point Type: 03: Holding Registers

Polling Rate (ms): 50

Already Connected | Disconnect

Start Scan | Stop Scan

Batch Size: 20

Clear Results

Download Log

```
Address: 31, Value: 715
Address: 32, Value: 421
Address: 33, Value: 61
Address: 34, Value: 451
Address: 35, Value: 369
Address: 36, Value: 742
Address: 37, Value: 848
Address: 38, Value: 197
Address: 39, Value: 788
Address: 40, Value: 502

Undefined Registers:
Address: 41
Address: 42
Address: 43
Address: 44
Address: 45
Address: 46
Address: 47
Address: 48
```

Show as Decimal | Show as Binary | Show as Hex | Show as Float | Show as Swapped Float | Show as Double | Show as Swapped Double