RZ/T2N Group

OpenPLC Application Note

Introduction

This document provides the procedure to install OpenPLC and OpenPLC Runtime on the Windows PC and the evaluation board.

This guide provides the following information:

- Building procedure
- Preparation for the test environment for OpenPLC

Any GitHub repositories, libraries, or code referenced in this document may be subject to licenses and terms defined by their respective authors or copyright holders. Users are solely responsible for reviewing, understanding, and complying with all applicable licenses and terms associated with such content. Our company assumes no responsibility or liability for compliance with these third-party licenses.

The RZ/T2N OpenPLC Package is based on RZ Linux BSP Plus for Kernel 6.12. Additional information on building the BSP and booting the evaluation board is available at the following website.

RZ Linux BSP Plus for Kernel 6.12: https://renesas-rz.github.io/rz_linux_bsp_plus/

Target Reference Board

- RZ/T2N Evaluation Board

Target Software

- RZ Linux BSP Plus for Kernel 6.12
  URL: https://renesas-rz.github.io/rz_linux_bsp_plus/
- OpenPLC

Contents

1. Environmental Requirement 3
2. Build Instructions 4
3. Install the OpenPLC Runtime to the evaluation board 6
4. Appendix 12
4.1 Modbus TCP communication with Windows PC 12
4.2 OpenPLC Editor Setup and Communication Procedure 25
5. Revision History 40
Website and Support 41
Notice 42
General Precautions in the Handling of Microprocessing Unit and Microcontroller Unit Products 43

1. Environmental Requirement

The environment for preparing OpenPLC environment is listed in **Table 1-1.**

Figure 1-1 shows the recommended environment for this package.

A Windows PC can be used as a serial terminal interface with software such as TeraTerm.

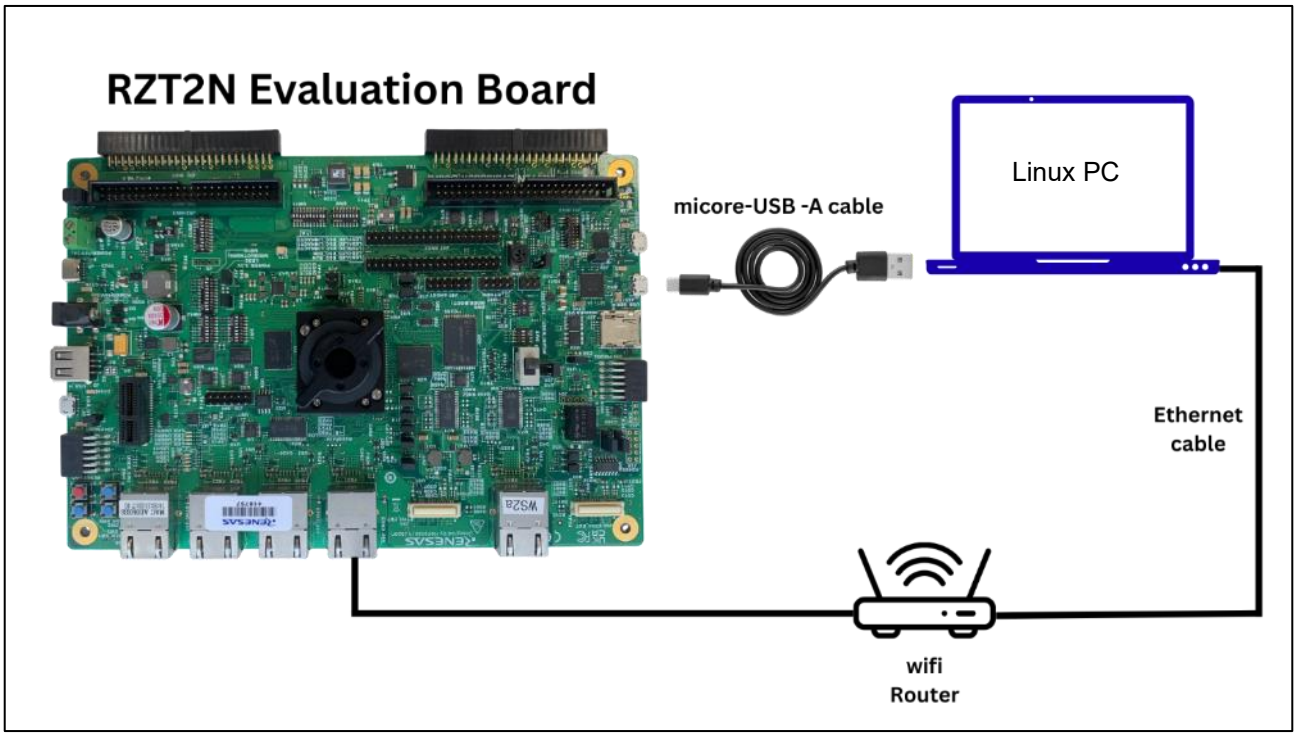


Figure 1-1. Recommend environment.

Table 1-1. Equipment and Software for Developing Environments of Linux Platform

Equipment		Description
Linux Host PC		Used as build/debug environment. 100GB free space on HDD or SSD is necessary.
	OS	Ubuntu 22.04 LTS 64 bit OS must be used. 22.04 inside a docker container also OK.
Windows Host PC		Used as debug environment, controlling with terminal software.
	OS	Windows 11
	Terminal software	Used for controling serial console of the target board. Tera Term (latest version) is recommended. Available at https://ttssh2.osdn.jp/index.html.en
	VCP Driver	Virtual COM Port driver which enables to communicate Windows Host PC and the target board via USB which is virtually used as serial port. Available at: http://www.ftdichip.com/Drivers/VCP.htm
USB serial to MiniUSB Cable		Serial communication (UART) between the Evaluation Board Kit and Windows PC. The type of USB serial connector on the Evaluation Board Kit is Mini USB type B.
microSD Card		Used to boot the system, and store applications.

2. Build Instructions

This chapter provides the procedure to build the Board Support Package.

(1) Build RZ Linux BSP Plus for Kernel 6.12

Read the following site and build RZ Linux BSP Plus for Kernel 6.12. After that, proceed to the next step.

https://renesas-rz.github.io/rz_linux_bsp_plus/RZTN/getting_started/

(2) Build Initialize

Initialize a build using the 'oe-init-build-env' script in Poky and point TEMPLATECONF to platform conf path.

```
$ cd ${WORK}
$ TEMPLATECONF=$PWD/meta-renesas/meta-rz-distro/conf/templates/bsp-plus-conf/ source \
poky/oe-init-build-env build
```

(3) Edit local.conf

Enable the following packages to install the OpenPLC Runtime. Add the following lines to “${WORK}/build/conf/local.conf”.

```
# Build openplc
EXTRA_IMAGE_FEATURES ?= "tools-sdk dev-pkgs "
HOSTTOOLS += " fallocate pkg-config"
IMAGE_INSTALL:append = " pkgconfig util-linux libtool python3 python3-pymodbus \
python3-flask python3-flask-login python3-pyserial python3-sqlite3 sqlite3 \
libmodbus cmake make gcc-dev autoconf-dev binutils-dev libbfd cpp cpp-symlinks gcc \
gcc-symlinks binutils binutils-symlinks autoconf g++ g++-symlinks automake git \
bison bison-dev flex "
INCOMPATIBLE_LICENSE_EXCEPTIONS:append = " libtool:GPL-3.0-only gcc-dev:GPL-3.0-only \
autoconf-dev:GPL-3.0-only libbfd:GPL-3.0-only cpp:GPL-3.0-only \
cpp-symlinks:GPL-3.0-only gcc:GPL-3.0-only gcc-symlinks:GPL-3.0-only \
autoconf:GPL-3.0-only g++:GPL-3.0-only g++-symlinks:GPL-3.0-only \
automake:GPL-3.0-only "
PREFERRED_VERSION_bison = "3.8.2"

# Set rootfs over 2.1G
IMAGE_ROOTFS_SIZE = "2202009"
```

(4) Start a build

Run the commands below to start a build. Building an image can take up to a few hours depending on the user’s host system performance.

Build the target file system image using bitbake.

```
$ MACHINE=rzt2n-dev bitbake core-image-minimal
```

After the building is successfully completed, a similar output will be seen, and the command prompt will return. The following log is an example.

```
NOTE: Tasks Summary: Attempted 3512 tasks of which 8 didn't need to be rerun and all succeeded.
```

All necessary files listed in the following Table 2-1. Image filestable will be generated by the bitbake command and will be in the “${WORK}/build/tmp/deploy/images“ directory.

Table 2-1. Image files

RZ/T2N	**Linux kernel**	Image-rzt2n-dev.bin
	Device tree file	r9a07g076m48-dev.dtb
	root filesystem	core-image-minimal-rzt2n-dev.rootfs.tar.bz2
	Boot loader	bl2_bp_xspi0-rzt2n-dev.srec fip-rzt2n-dev.srec
	SD image	core-image-minimal-rzt2n-dev.rootfs.wic.gz core-image-minimal-rzt2n-dev.rootfs.wic.bmap
	Flash Programmer	HDR NM Flash_Programmer_SCIF_CR52_RZT2N_EVK.mot

If you want to know the components installed to the root filesystem, check the manifest file. The manifest file is created to the following path after building the images:

```
${WORK}/build/tmp/deploy/images/rzt2n-dev/core-image-minimal-rzt2n-dev.rootfs.manifest
```

(5) Boot the evaluation board

Refer to the following website for instructions on booting the evaluation board.
https://renesas-rz.github.io/rz_linux_bsp_plus/RZTN/getting_started/

The build procedure on the website can be skipped because it has already been completed in Chapter 2 of this application note.

After booting the evaluation board, continue with Chapter 3 of this application note.

3. Install the OpenPLC Runtime to the evaluation board

This chapter provides the procedure to install the OpenPLC Runtime to the evaluation board from the OpenPLC application on the Windows PC.

(1) Install OpenPLC Runtime to the evaluation board

Run the commands below to install OpenPLC Runtime.

```
root@rzt2n-dev:~# git clone https://github.com/thiagoralves/OpenPLC_v3.git
root@rzt2n-dev:~# cd OpenPLC_v3/
root@rzt2n-dev:~/OpenPLC_v3# git checkout -b tmp a283d0dbac813116babdf81c53144c4d455edf49
root@rzt2n-dev:~/OpenPLC_v3# export PKG_CONFIG_PATH=/usr/local/lib/pkgconfig
root@rzt2n-dev:~/OpenPLC_v3# export LD_LIBRARY_PATH=/usr/local/lib:$LD_LIBRARY_PATH
root@rzt2n-dev:~/OpenPLC_v3# ldconfig
root@rzt2n-dev:~/OpenPLC_v3# ./install.sh custom
```

Confirm the logs below to verify that OpenPLC Runtime was installed successfully.

```
Installing
OpenPLC on Custom Platform
[MATIEC COMPILER]
…
checking for gcc ... gcc
checking whether the C compiler works... Yes
…
[ 90%] Built target asiopal
[100%] Built target asiodnp3 Install the project...
--
Install configuration: "Release"
…
Moving Files...
Compiling for Linux
Generating object files... Generating glueVars
Compiling main program...
Compilation finished successfully!
```

Notice: Sometimes this installation could fail in the “libmodbus” stage, in this case, you can also run step 0 to run the OpenPLC Runtime.

(2) Run the webserver

Moidfy the file "~/OpenPLC_v3/webserver/monitoring.py" on the evaluation board as below:

```
@@ -1,6 +1,6 @@
 import time, threading
 from struct import *
-from pymodbus.client.sync import ModbusTcpClient
+from pymodbus.client import ModbusTcpClient

 class debug_var():
     name = ''
```

Run the commands below to start the webserver.

```
root@rzt2n-dev:~/OpenPLC_v3# cd webserver/
root@rzt2n-dev:~/OpenPLC_v3/webserver# python3 webserver.py
```

Be sure to change the directory into "webserver".

The below error occurs when you execute commands not under directory "webserver".

```
* Serving Flask app "webserver" (lazy loading)
* Environment: production
WARNING: This is a development server. Do not use it in a production deployment.
Use a production WSGI server instead.
* Debug mode: off
* Running on http://0.0.0.0:8080/ (Press CTRL+C to quit)
```

(3) Check the OpenPLC Runtime on the browser application in Windows PC

The main runtime webserver is available at your target IP address on port 8080. For example, if you installed

OpenPLC Runtime on a RZT2N, and your RZT2N IP address is 10.166.29.238, then you can access OpenPLC.

Runtime by opening your web browser and pointing it to http://10.166.29.238:8080.



Figure 3-1. OpenPLC Runtime Web Interface Access URL

Once you access OpenPLC webserver, you should see in your browser window a login page like this:

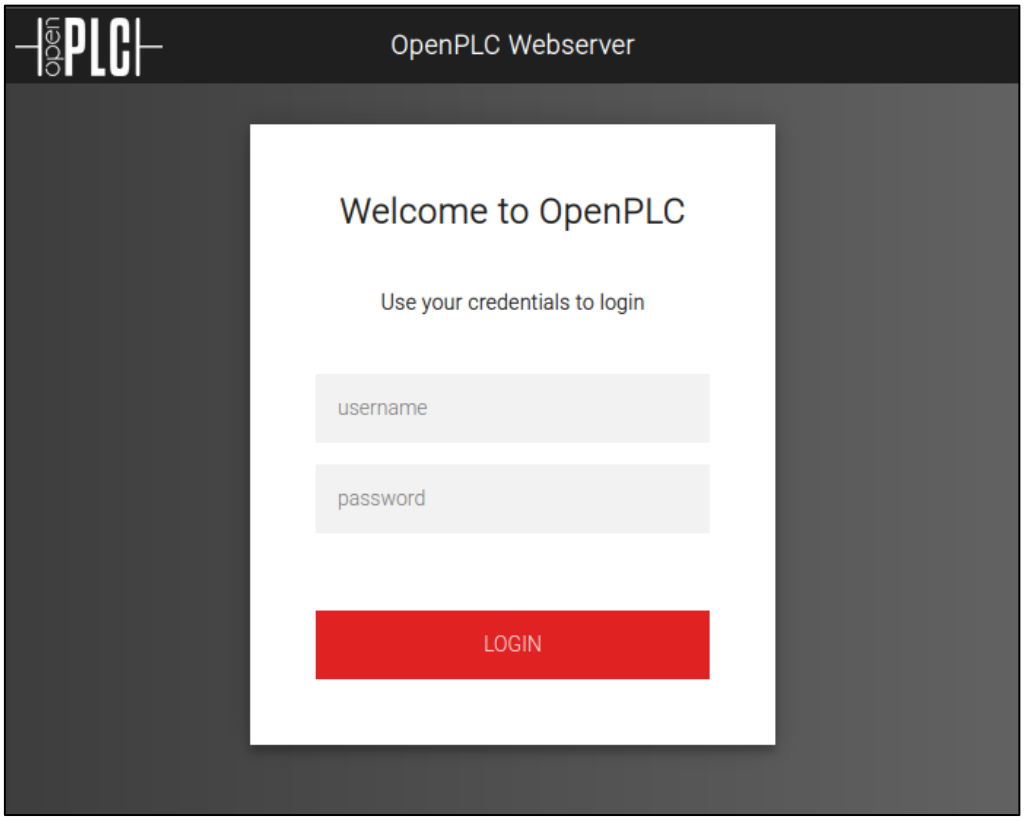


Figure 3-2. OpenPLC Web Server Login Page

The default username and password are openplc (Username) and openplc (Password).

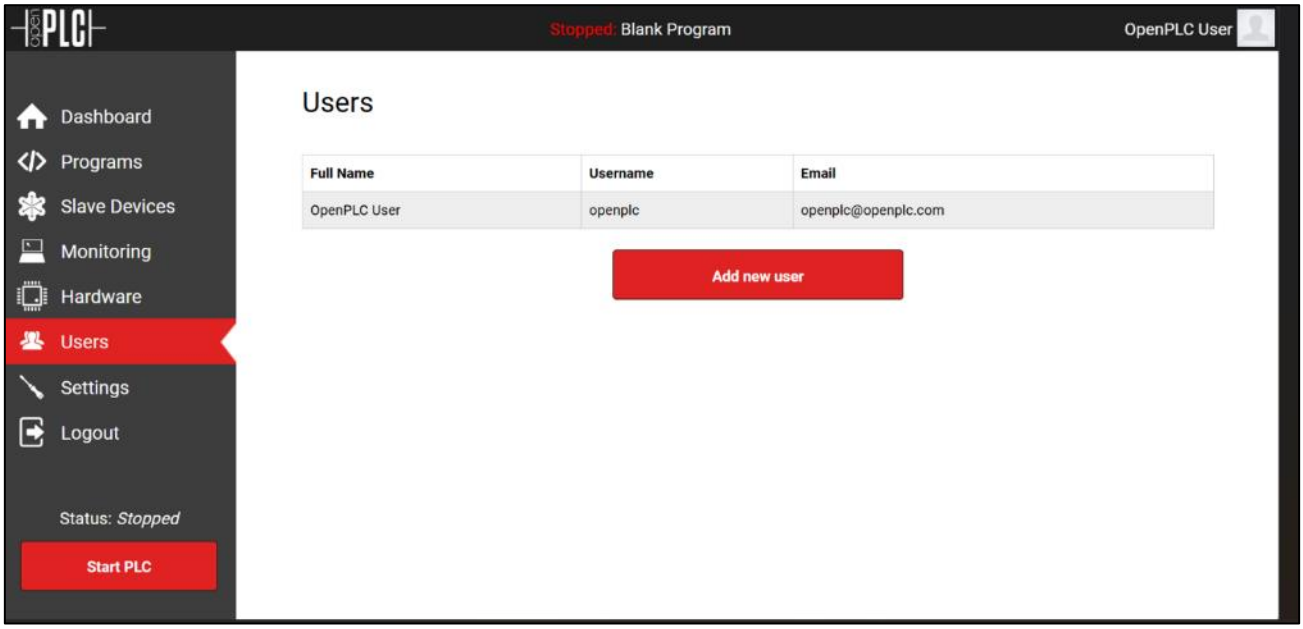


Figure 3-3. OpenPLC User Management Page

Confirm the logs below to veify that login OK:

```
10.231.33.40 - - [09/Jun/2026 10:33:26] "GET /dashboard HTTP/1.1" 302 -
10.231.33.40 - - [09/Jun/2026 10:33:26] "GET /login HTTP/1.1" 200 -
10.231.33.40 - - [09/Jun/2026 10:33:26] "GET /static/openplc_logo.gif HTTP/1.1" 304 -
10.231.33.40 - - [09/Jun/2026 10:33:31] "POST /login HTTP/1.1" 302 -
10.231.33.40 - - [09/Jun/2026 10:33:31] "GET /dashboard HTTP/1.1" 200 -
10.231.33.40 - - [09/Jun/2026 10:33:31] "GET /static/openplc_logo.gif HTTP/1.1" 304 -
10.231.33.40 - - [09/Jun/2026 10:33:31] "GET /static/arrow.png HTTP/1.1" 304 -
10.231.33.40 - - [09/Jun/2026 10:33:31] "GET /static/programs-icon-64x64.png HTTP/1.1" 304 -
10.231.33.40 - - [09/Jun/2026 10:33:31] "GET /static/users-icon-64x64.png HTTP/1.1" 304 -
10.231.33.40 - - [09/Jun/2026 10:33:31] "GET /static/logout-icon-64x64.png HTTP/1.1" 304 -
10.231.33.40 - - [09/Jun/2026 10:33:31] "GET /static/modbus-icon-512x512.png HTTP/1.1" 304 -
10.231.33.40 - - [09/Jun/2026 10:33:31] "GET /static/settings-icon-64x64.png HTTP/1.1" 304 -
10.231.33.40 - - [09/Jun/2026 10:33:31] "GET /static/monitoring-icon-64x64.png HTTP/1.1" 304 -
10.231.33.40 - - [09/Jun/2026 10:33:31] "GET /static/hardware-icon-980x974.png HTTP/1.1" 304 -
10.231.33.40 - - [09/Jun/2026 10:33:31] "GET /static/default-user.png HTTP/1.1" 304 -
10.231.33.40 - - [09/Jun/2026 10:33:31] "GET /static/home-icon-64x64.png HTTP/1.1" 304 -
10.231.33.40 - - [09/Jun/2026 10:33:31] "GET /runtime_logs HTTP/1.1" 200 -
10.231.33.40 - - [09/Jun/2026 10:33:32] "GET /runtime_logs HTTP/1.1" 200 -
10.231.33.40 - - [09/Jun/2026 10:33:33] "GET /runtime_logs HTTP/1.1" 200 -
10.231.33.40 - - [09/Jun/2026 10:33:34] "GET /runtime_logs HTTP/1.1" 200 -
10.231.33.40 - - [09/Jun/2026 10:33:35] "GET /runtime_logs HTTP/1.1" 200 -
```

The PLC can be started successfully with the default Blank Program. After the Runtime is started, the Dashboard shows **Status: Running** and **Program: Blank Program** (see **Error! Reference source not found.**), and the “Stop PLC” button becomes available, which confirms that the Runtime is executing the program. The Runtime Logs panel on the Dashboard may display “There was a problem retrieving the logs”; this only affects the web display and does not affect PLC operation. The full startup sequence is shown in the log below, which is output on the board console when the PLC starts:

- The Runtime starts and loads the default Blank Program.
- The Interactive Server listens on port 43628 (used by the web interface to communicate with the Runtime).
- The Modbus server is enabled and listens on port 502 (standard Modbus TCP port).
- “Server: waiting for new client...” indicates the Runtime is ready to accept connections.

The warnings “Slave Devices (mbconfig.cfg file not found)” and “Persistent Storage file not found” are expected for a fresh setup with no slave devices or persistent variables configured and can be ignored.

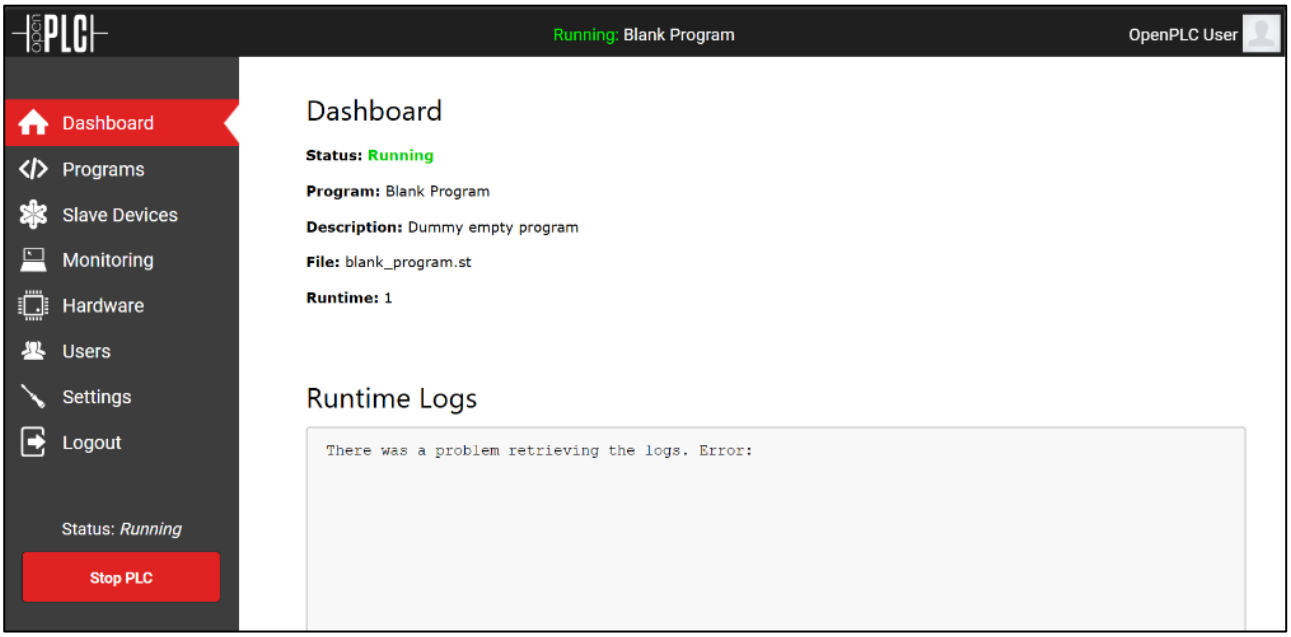


Figure 3-4. OpenPLC Runtime Running with Default Blank Program

The following logs are displayed on the board when the PLC starts.

```
OpenPLC Runtime starting...
Skipping configuration of Slave Devices (mbconfig.cfg file not found)
Interactive Server: Listening on port 43628
Interactive Server: waiting for new client...
Warning: Persistent Storage file not found
Setting main thread priority to RT
Locking main thread memory
Getting current time
Interactive Server: Client accepted! Creating thread for the new client ID: 4...
Interactive Server: waiting for new client...
Interactive Server: Thread created for client ID: 4
Issued runtime_logs() command
Interactive Server: client ID: 4 has closed the connection
Terminating interactive server connections
10.231.33.40 - - [09/Jun/2026 10:33:39] "GET /runtime_logs HTTP/1.1" 200 -
Openning database
Enabling Modbus on port 502
Interactive Server: Client accepted! Creating thread for the new client ID: 4...
Interactive Server: waiting for new client...
Interactive Server: Thread created for client ID: 4
Issued start_modbus() command to start on port: 502
Server: Listening on port 502
Server: waiting for new client...
```

4. Appendix

This chapter provides the procedure to utilize the OpenPLC Runtime on the evaluation board. The explanation in this chapter is based on RZ/T2N as an example.

OpenPLC Runtime is an open-source PLC (Programmable Logic Controller) runtime that runs on standard Linux-based hardware. OpenPLC supports IEC 61131-3 standard programming languages including Ladder Diagram and Structured Text, and provides industrial communication protocols such as Modbus TCP, DNP3, and EtherNet/IP. This makes it suitable for industrial automation prototyping and evaluation on embedded platforms.

Follow the following description to know how to prepare the test environment and test it on the evaluation board.

4.1 Modbus TCP communication with Windows PC

(1) Download Simply Modbus TCP Client Tool

This test tool turns your Windows PC into a Modbus TCP Client or Modbus RTU-over-TCP Client. Connect to Modbus TCP or server (slave) devices to test, troubleshoot, and monitor Modbus TCP communication.

Notes: This is a demo version. It allows a limited number of requests per session. Restart the application to reset the counter if the demo popup appears.

- Access the download page: https://www.simplymodbus.ca/TCPclient.htm
- Scroll to Free Download and choose one Compiler:
  Ex: Compiler 14 - SimplyModbusTCPclient8.1.2Install.zip
- Download the installer and install the application:

A) Extract the downloaded ZIP file, then run setup.exe and follow the installation wizard (Next → Next → Finish). Navigate to the extracted folder STM812 (note: the folder name may vary depending on the version downloaded).



Figure 4-1. Extracted SimlyModbus TCP Client Package

B) Once the installation is complete, open "Simply Modbus TCP Client 8.1.2". The following window will appear:

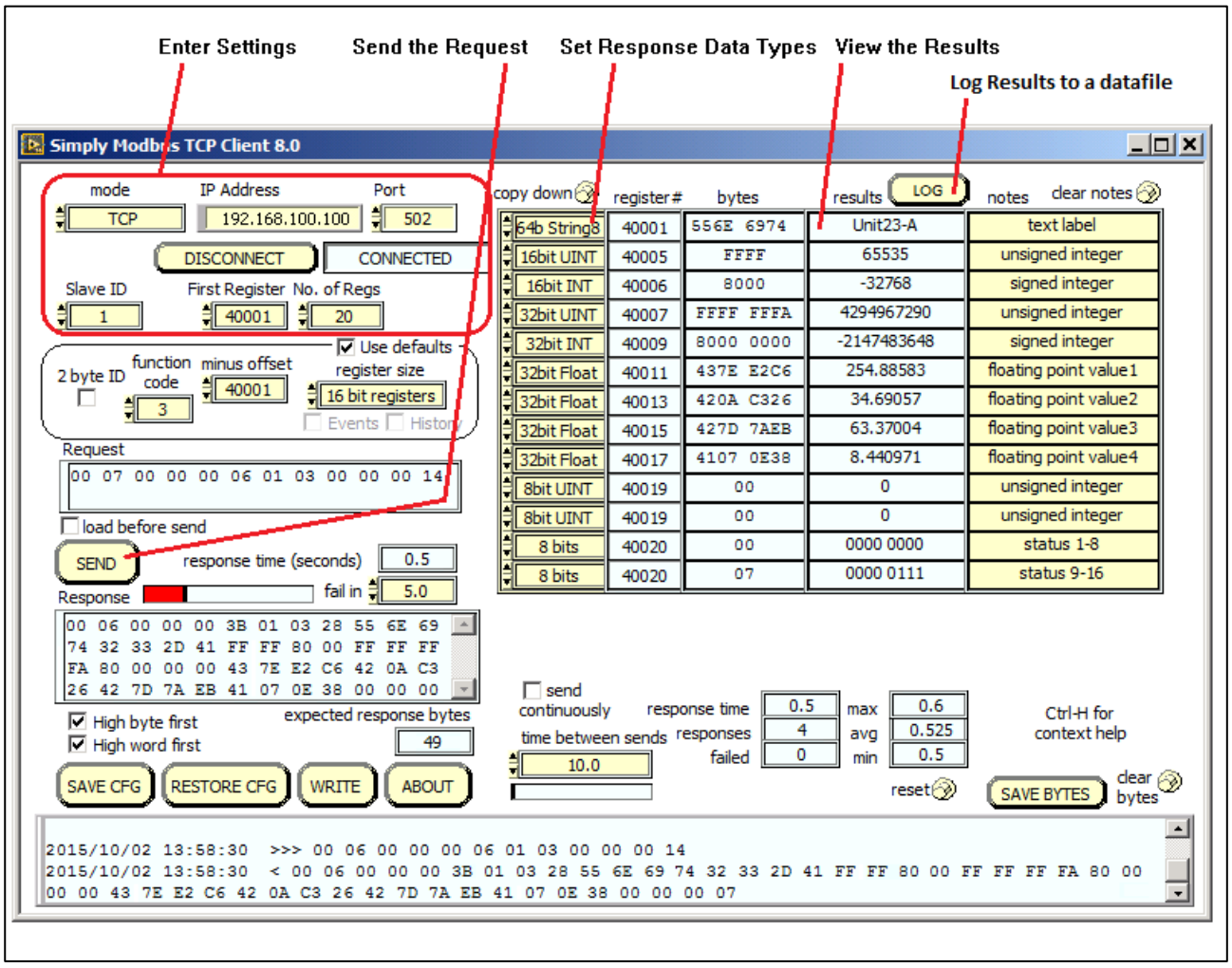


Figure 4-2. SimlyModbus TCP Client Main Window

(2) Set the IP addresses

Set the IP address of the evaluation board and the opposite device. The below settings are the examples. Note that the evaluation board and the opposite device are in the same network, or they are connected directly:

A) Evaluation Board:

```
$ ifconfig eth1 192.168.5.127 netmask 255.255.255.0 up
$ route add default gw 192.168.5.1 eth1
```

B) Windows PC : The requested operation requires elevation (Run as administrator).

```
$ netsh interface ip set address "Ethernet interface" static 192.168.5.20 255.255.255.0
```

(3) Check connection

Check connection between board and Windows PC by ping command:

A) Evaluation Board:

```
$ ping -c 10 192.168.5.20
```

B) Windows PC :

```
$ ping -n 10 192.168.5.127
```

(4) Start OpenPLC webserver on the board

Run the commands below to start the webserver.

```
root@rzt2n-dev:~/OpenPLC_v3# cd webserver/
root@rzt2n-dev:~/OpenPLC_v3/webserver# python3 webserver.py > /tmp/webserver.log 2>&1 &
root@rzt2n-dev:~/OpenPLC_v3/webserver# tail -f /tmp/webserver.log
Press CTRL+C to quit
```

(5) Access OpenPLC Web UI from PC browser

Open a browser on the Windows PC that guide at step “Check the OpenPLC Runtime on the browser application in Windows PC”

(6) Create the ST program file

Create and upload a Structured Text (ST) program that reads a Modbus Holding Register input, multiplies it by 2, and writes the result to another register.

- Create the ST program file on the board or somewhere in your Windows File Explorer
- Ex: vi /tmp/test_modbus.st
- Paste the following program save file:

```
PROGRAM test_modbus
  VAR
    input_val  AT %QW0 : INT;
    output_val AT %QW1 : INT;
  END_VAR

  output_val := input_val * 2;

END_PROGRAM

CONFIGURATION Config0
  RESOURCE Res0 ON PLC
    TASK task0(INTERVAL := T#100ms, PRIORITY := 0);
    PROGRAM instance0 WITH task0 : test_modbus;
  END_RESOURCE
END_CONFIGURATION
```

(7) Upload program via Web UI

- In the browser, navigate to Programs → click Choose File “test_modbus.st” has done in (6) → click Upload Program:

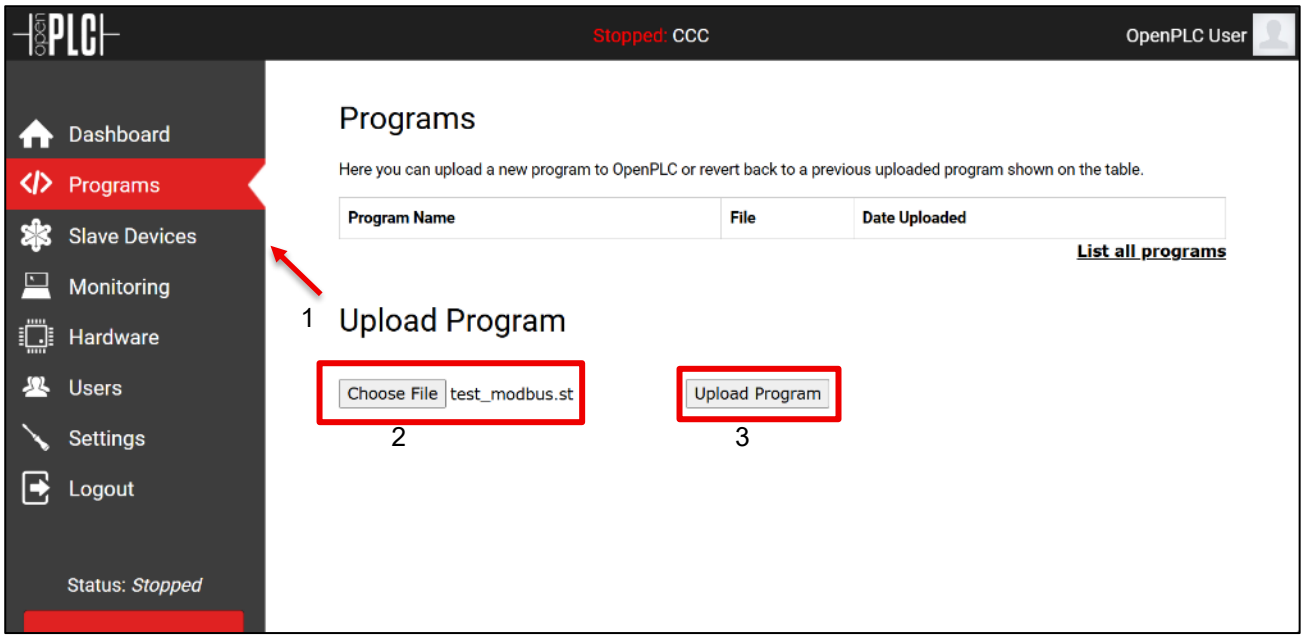


Figure 4-3. Uploading the Modbus Test Program to OpenPLC

- Edit “Name” and “Description” → click “Upload program”.
  Ex: the name is “Test Modbus”, description is “Read QW0 input, multiply by 2, write to QW1”

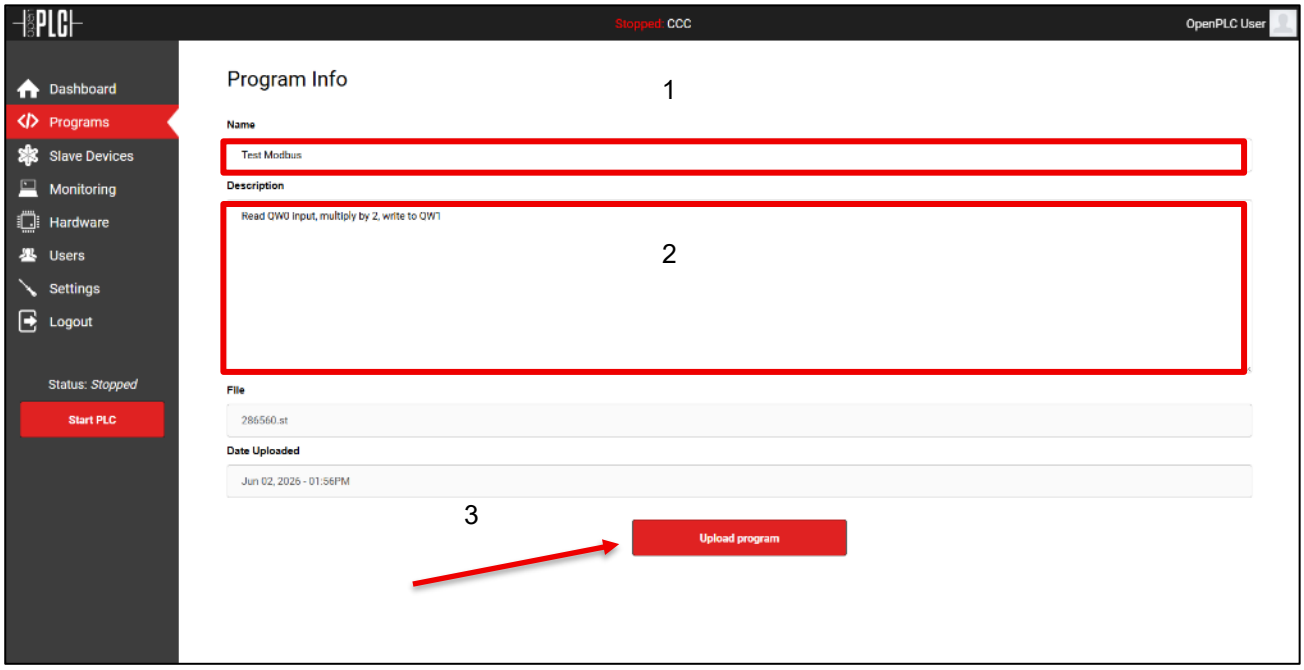


Figure 4-4. OpenPLC Program Information Configuration

(8) Compile and launch program

After successful compilation, click **Launch program**. Once the "Go to Dashboard" button appears, click it to proceed to the dashboard.

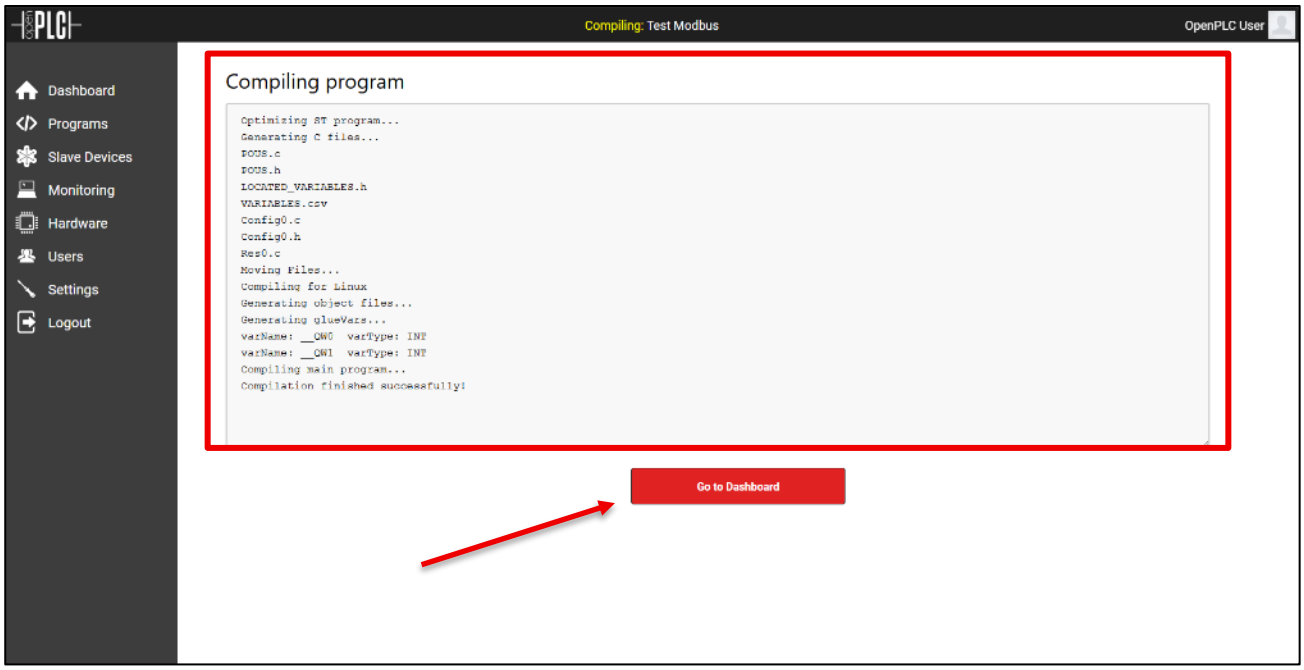


Figure 4-5. Successful Program Compilation

(9) Start PLC

- Click Start PLC.

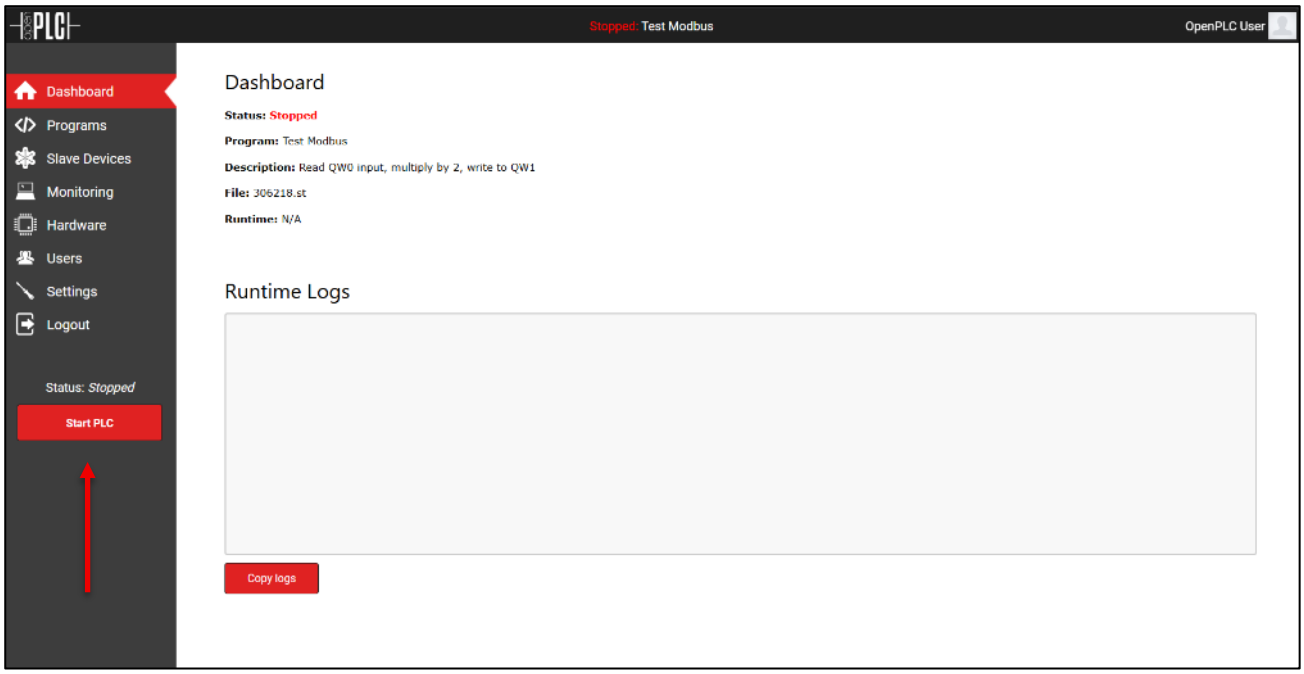


Figure 4-6. Starting the OpenPLC Runtime

- Confirm that the Dashboard shows Status: Running | Program: Test Modbus

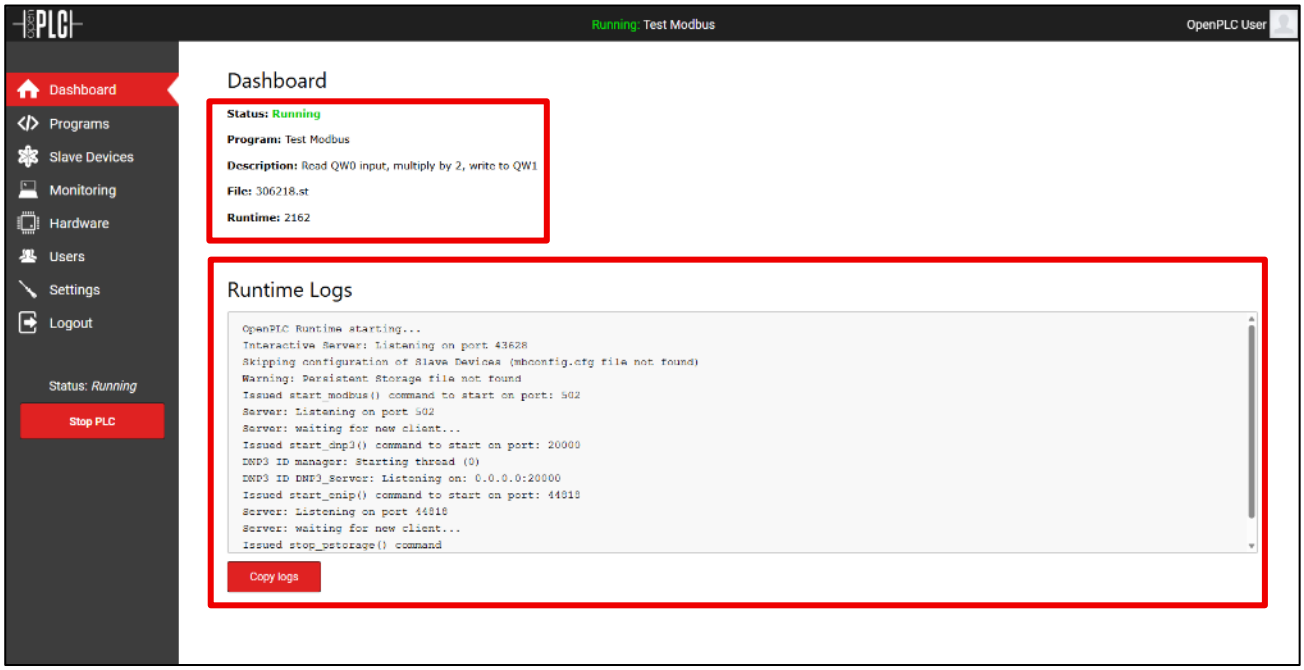


Figure 4-7. OpenPLC Runtime Running the Test Modbus Program

- Confirm logs below on RZ/T2N:

```
10.231.33.40 - - [16/Sep/2026 05:04:29] "GET /runtime_logs HTTP/1.1" 200 -
10.231.33.40 - - [16/Sep/2026 05:04:30] "GET /runtime_logs HTTP/1.1" 200 -
OpenPLC Runtime starting...
Skipping configuration of Slave Devices (mbconfig.cfg file not found)
Interactive Server: Listening on port 43628
Interactive Server: waiting for new client...
Warning: Persistent Storage file not found
Setting main thread priority to RT
Locking main thread memory
Getting current time
Interactive Server: Client accepted! Creating thread for the new client ID: 4...
Interactive Server: waiting for new client...
Interactive Server: Thread created for client ID: 4
Issued runtime_logs() command
Interactive Server: client ID: 4 has closed the connection
Terminating interactive server connections
192.168.5.20 - - [02/Jun/2026 14:21:12] "GET /runtime_logs HTTP/1.1" 200 -
Openning database
Enabling Modbus on port 502
Interactive Server: Client accepted! Creating thread for the new client ID: 4...
Interactive Server: waiting for new client...
Interactive Server: Thread created for client ID: 4
Issued start_modbus() command to start on port: 502
Server: Listening on port 502
Server: waiting for new client...
Interactive Server: client ID: 4 has closed the connection
Enabling DNP3 on port 20000
Terminating interactive server connections
Interactive Server: Client accepted! Creating thread for the new client ID: 4...
Interactive Server: waiting for new client...
Interactive Server: Thread created for client ID: 4
```

```
Issued start_dnp3() command to start on port: 20000
Interactive Server: client ID: 4 has closed the connection
Terminating interactive server connections
Enabling EtherNet/IP on port 44818
DNP3 ID manager: Starting thread (0)
channel state change: OPENING
DNP3 ID DNP3_Server: Listening on: 0.0.0.0:20000
DNP3 Enabled
Interactive Server: Client accepted! Creating thread for the new client ID: 9...
Interactive Server: waiting for new client...
Interactive Server: Thread created for client ID: 9
Issued start_enip() command to start on port: 44818
Server: Listening on port 44818
Server: waiting for new client...
Interactive Server: client ID: 9 has closed the connection
Disabling Persistent Storage
Terminating interactive server connections
Interactive Server: Client accepted! Creating thread for the new client ID: 9...
Interactive Server: waiting for new client...
Interactive Server: Thread created for client ID: 9
Issued stop_pstorage() command
Interactive Server: client ID: 9 has closed the connection
Terminating interactive server connections
Interactive Server: Client accepted! Creating thread for the new client ID: 9...
Interactive Server: waiting for new client...
Interactive Server: Thread created for client ID: 9
Interactive Server: client ID: 9 has closed the connection
Terminating interactive server connections
Interactive Server: Client accepted! Creating thread for the new client ID: 9...
Interactive Server: waiting for new client...
Interactive Server: Thread created for client ID: 9
Interactive Server: Client accepted! Creating thread for the new client ID: 11...
Interactive Server: client ID: 9 has closed the connection
Terminating interactive server connections
Interactive Server: waiting for new client...
Interactive Server: Thread created for client ID: 11
Interactive Server: Client accepted! Creating thread for the new client ID: 9...
Issued runtime_logs() command
Interactive Server: waiting for new client...
Interactive Server: Thread created for client ID: 9
Interactive Server: client ID: 11 has closed the connection
Terminating interactive server connections
Interactive Server: client ID: 9 has closed the connection
Terminating interactive server connections
…
```

- Verfiy Modbus port is listening on the board

```
root@rzt2n-dev:/OpenPLC_v3/webserver# netstat -tln | grep 502
tcp        0      0 0.0.0.0:502             0.0.0.0:*               LISTEN

root@rzt2n-dev:/OpenPLC_v3/webserver# cat core/LOCATED_VARIABLES.h

__LOCATED_VAR(INT,__QW0,Q,W,0)

__LOCATED_VAR(INT,__QW1,Q,W,1)
```

(10) Connect Simply Modbus to Board

- Open Simply Modbus TCP Client (Launch the application from the Start menu or desktop shortcut.)

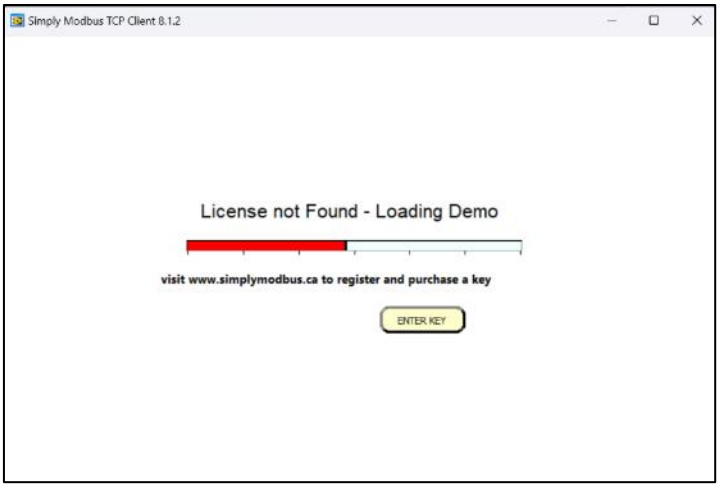


Figure 4-8. SimlyModbus TCP Client Launch Screen

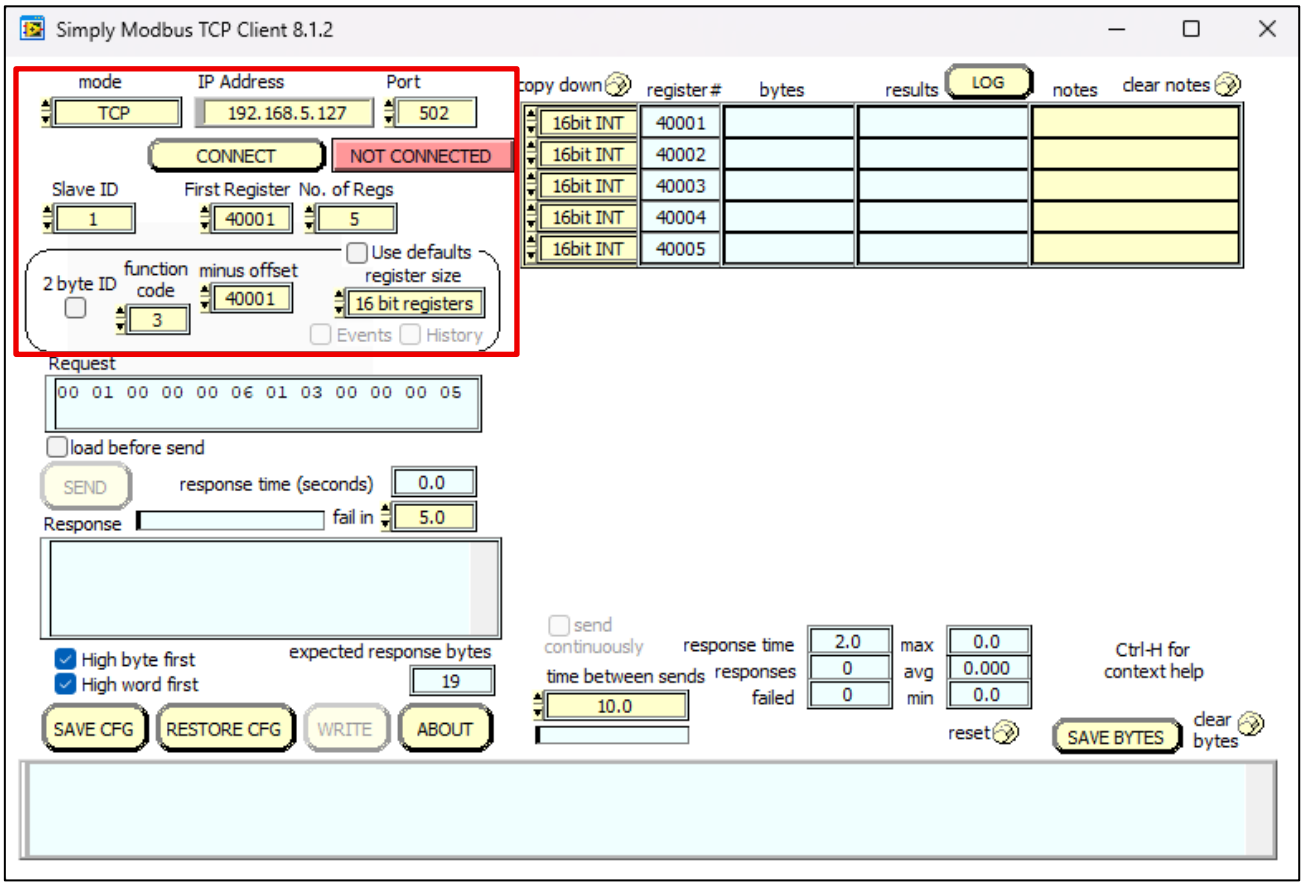


Figure 4-9. SimlyModbus TCP Client Configuration

Table 4-1. SimlyModbus TCP Client Configuration Parameters

Field	Value	Description
mode	TCP	Modbus TCP protocol
IP Address	192.168.5.127	Board ETH3 (interface eth0) address
Port	502	Modbus TCP standard port
Slave ID	1	OpenPLC default slave ID
function code	3	Read Holding Registers
First Register	40001	Start from register 1
No. of Regs	5	Read 5 registers

- Click the CONNECT button. The status indicator should change from NOT CONNECTED to CONNECTED. If connection fails, verify PLC is Running on dashboard and port 502 is listening on the board.

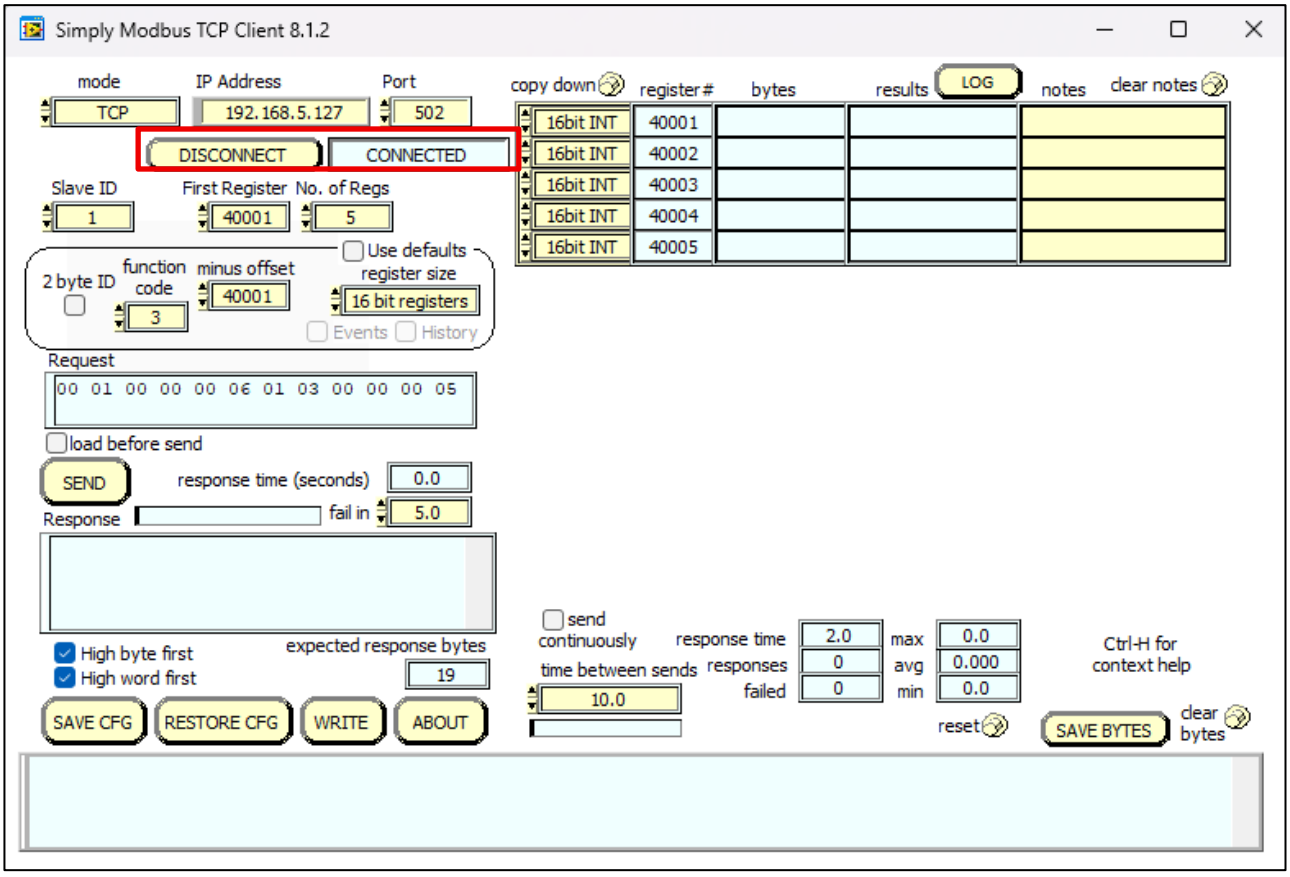


Figure 4-10. SimlyModbus TCP Client Connected to OpenPLC

(11) Write Register Value

Write a value to Holding Register %QW0 (mapped to Modbus register 40001). The PLC program will read this as input_val and compute output_val = input_val × 2.

- Open the **Write window**: In the main Simply Modbus window, click the **WRITE** button. A new window will appear.

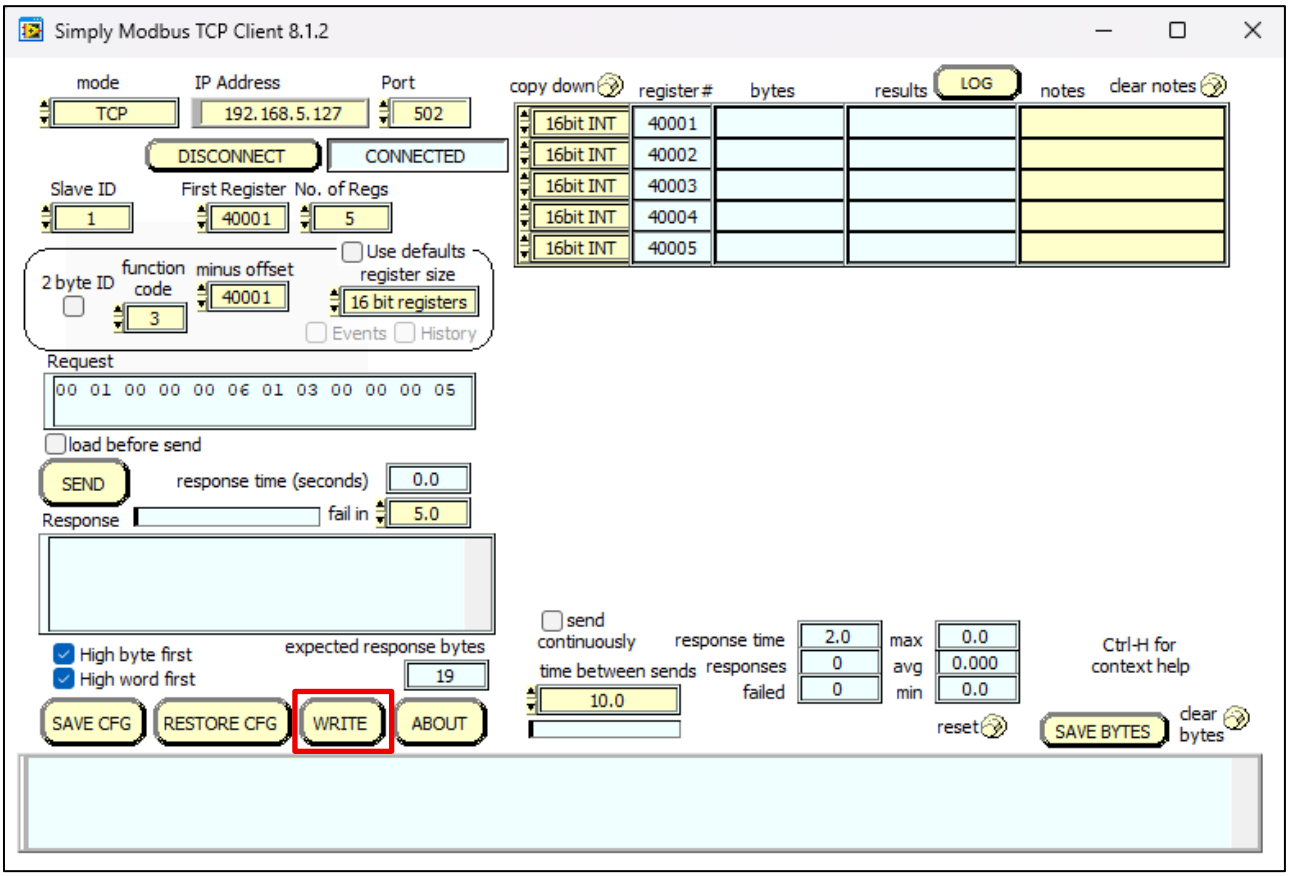


Figure 4-11. Launching the Write Register Function

- Configure the following parameters and click SEND in the write window. Check the Response field:

Table 4-2. Modbus Write Register Parameters

Field	Value	Description
function code	6	Write Single Register (FC06)
First Register	40001	Maps to %QW0 (input_val)
Values to Write	1	Write 1 register
Values to Write	100	Test value to write

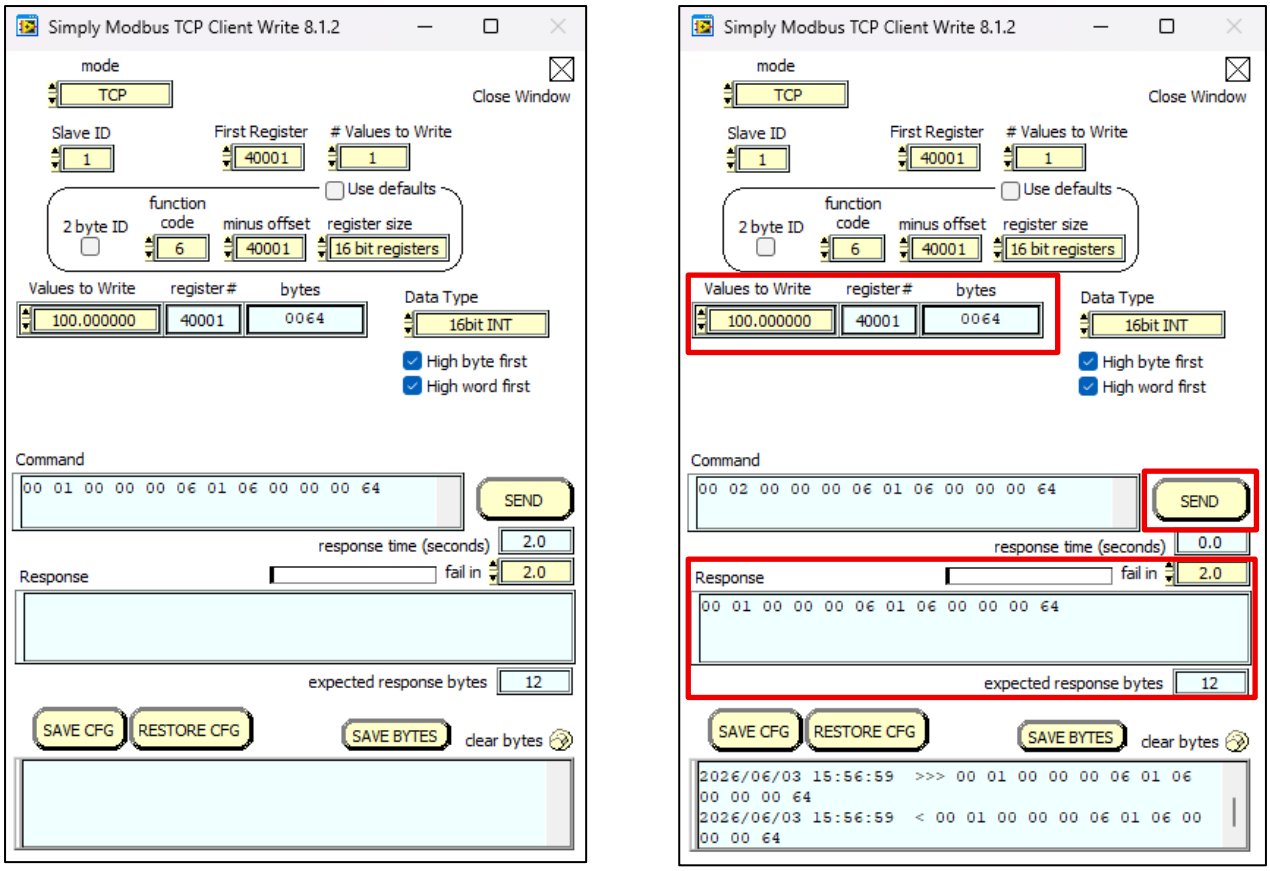


Figure 4-12. Successful Write Operation to Holding Register 40001

- Expected response (hex): 0x64 = 100 decimal. Response containing 00 64 (hex for 100) confirms the write was successful.

(12) Read Register Value & Verify

- Return to the main Simply Modbus window.
- Configure reading parameters:

Table 4-3. Modbus Read Holding Register Parameters

Field	Value	Description
function code	3	Read Holding Registers (FC03)
First Register	40001	Start from register 1
No. of Regs	5	Read 5 registers

- Click **SEND** and verify results

Table 4-4. Expected Modbus Register Values

Register	Variable	Expected Value
40001	%QW0 (input_val)	100
40002	%QW1 (output_val)	200
40003-40005	-	0

- Register 40002 showing **200** confirms the PLC program is executing correctly - reading input from Modbus, computing the result, and writing back to the Modbus register.

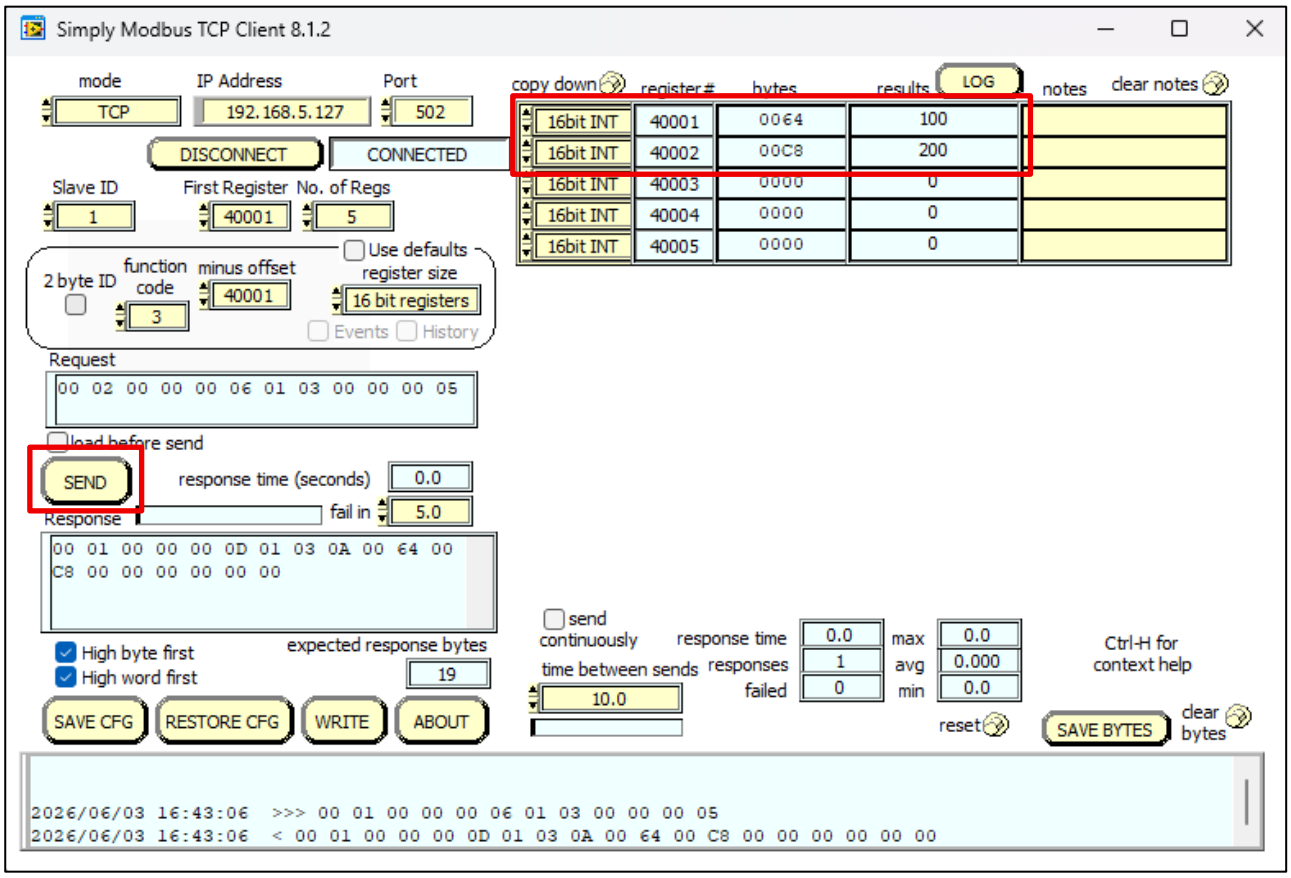


Figure 4-13. Verification of Modbus Register Values

(13) Monitor Variables on OpenPLC Web UI

- Navigate to Monitoring page: In the browser, click **Monitoring** in the left sidebar:

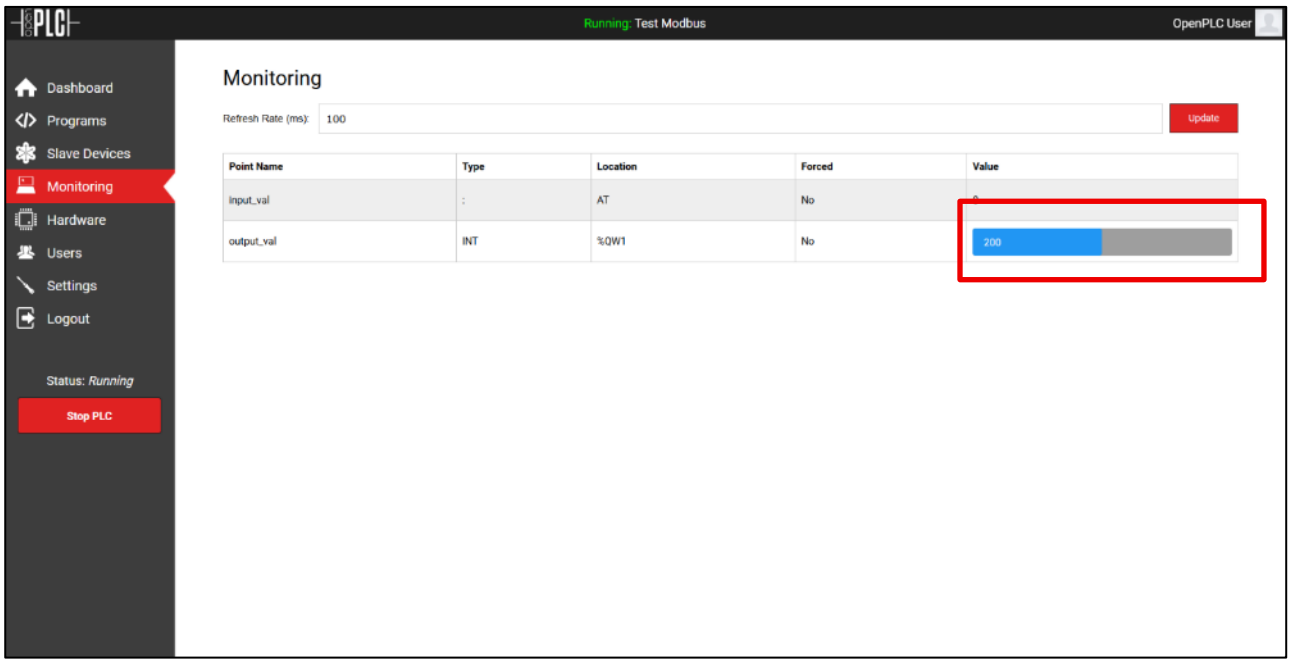


Figure 4-14. Monitoring PLC Variables in the OpenPLC Web Interface

- Tests with different values:

Table 4-5. Expected PLC Output Values

Write to 40001	Expected 40002
50	100
100	200
500	1000
1000	2000

Note**:** In OpenPLC v3, variables declared with AT addressing (e.g. input_val AT %QW0) may display "AT" in the Location column on the Monitoring page instead of the actual address. This is a known UI display issue in OpenPLC v3 and does not affect the actual program execution. The variable is correctly mapped to its Modbus register and operates as expected.

- Kill webserver v3

```
$ pkill -f "webserver.py"
$ pkill -f "openplc"
```

4.2 OpenPLC Editor Setup and Communication Procedure

This section describes how to use OpenPLC Editor to create and upload a PLC program to the RZ/T2N evaluation board and verify the communication between the Windows PC and the board via Modbus TCP protocol.

Note: OpenPLC Editor v4 requires OpenPLC Runtime v4 to be installed on the evaluation board. OpenPLC Runtime v3 (installed in the previous section in chapter 3 and chapter 4.1) is not compatible with OpenPLC Editor v4 due to a protocol mismatch — Runtime v3 uses plain HTTP without authentication, while Editor v4 requires HTTPS/TLS with JWT-based authentication on port 8443. This incompatibility exists regardless of which port is configured. Therefore, OpenPLC Runtime v4 must be installed separately on the board to use OpenPLC Editor v4.

(1) Download OpenPLC

The steps may be changed by the vendor, so follow the instructions of the official site. Visit the site below and download the installer:

OPENPLC Development System V4 | OPENPLC Store International

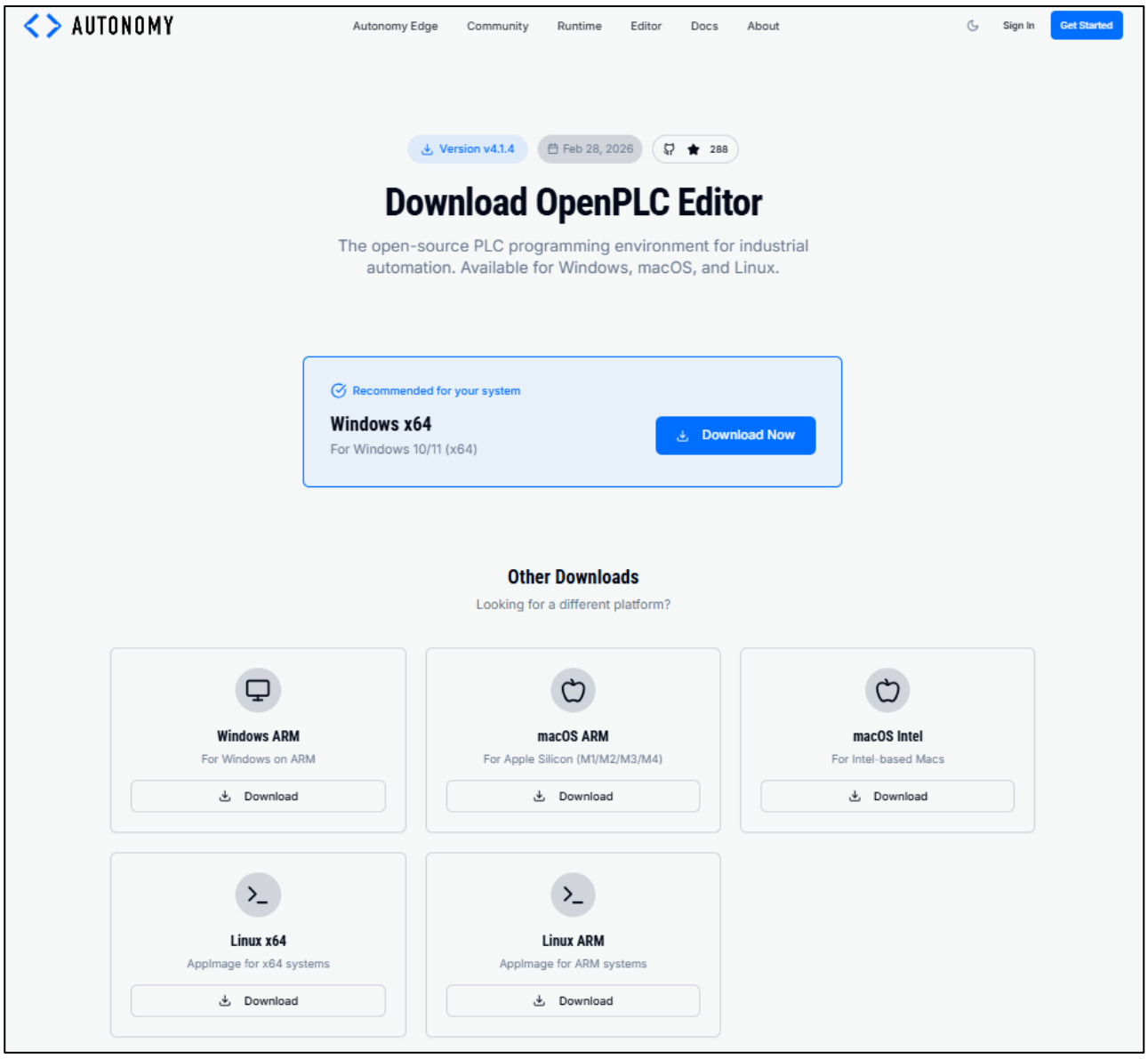


Figure 4-15. OpenPLC Editor Download Page

(2) Run the installer and install the OpenPLC

Follow the installer, there is no problem with the installation if you press "Next>" to proceed. Once the installation is complete, press "Finish". After that, an icon will appear on your desktop. When you start it up, "OpenPLC" will start up.

- Double click to install exe file and waiting to finish:

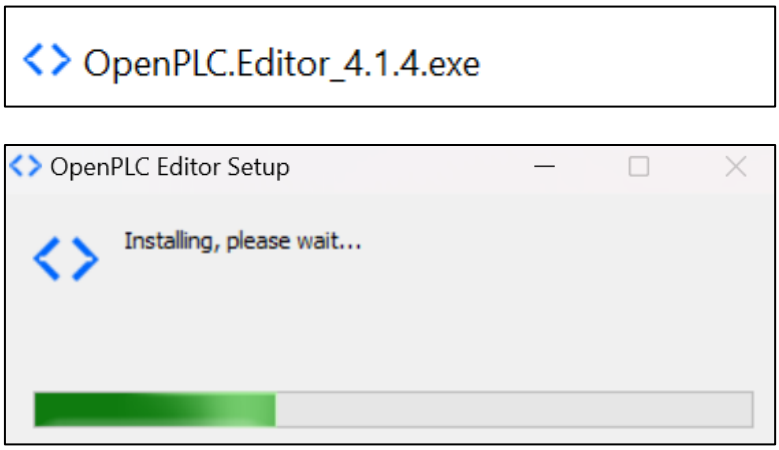


Figure 4-16. Installing OpenPLC Editor

(3) Create new project

- Click New Project.

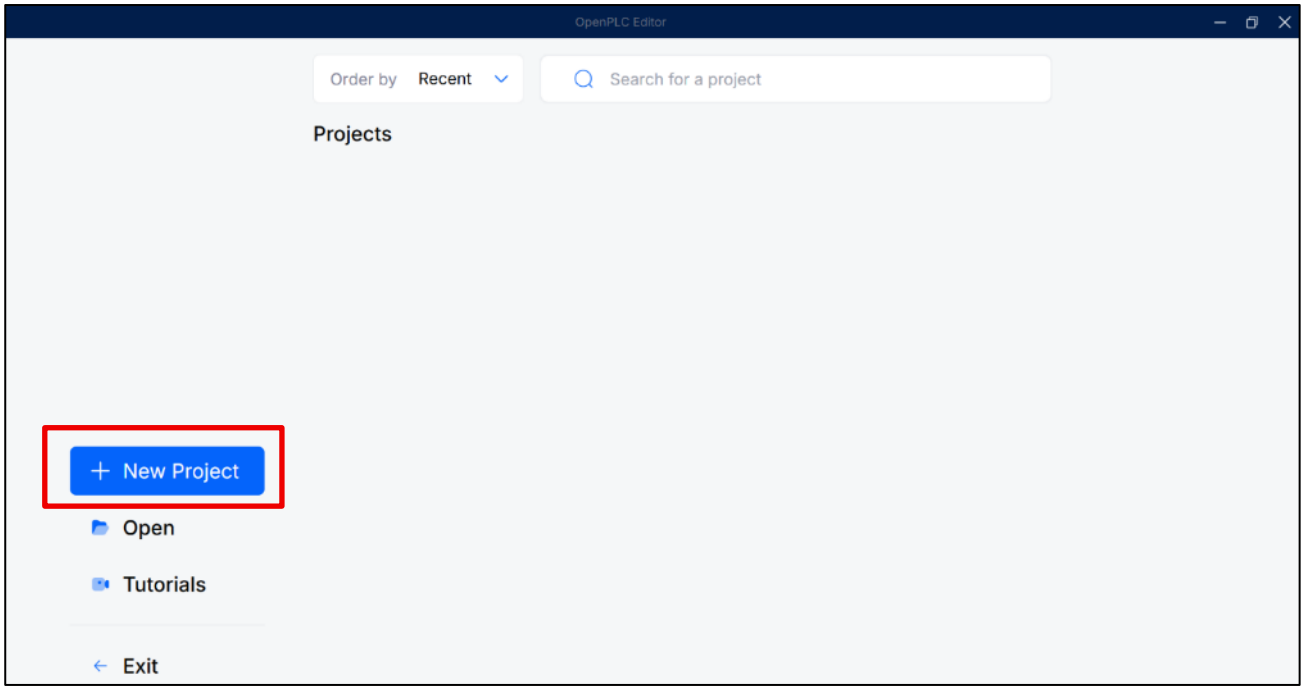


Figure 4-17. OpenPLC Editor Home Screen

- Click PLC Project → Next

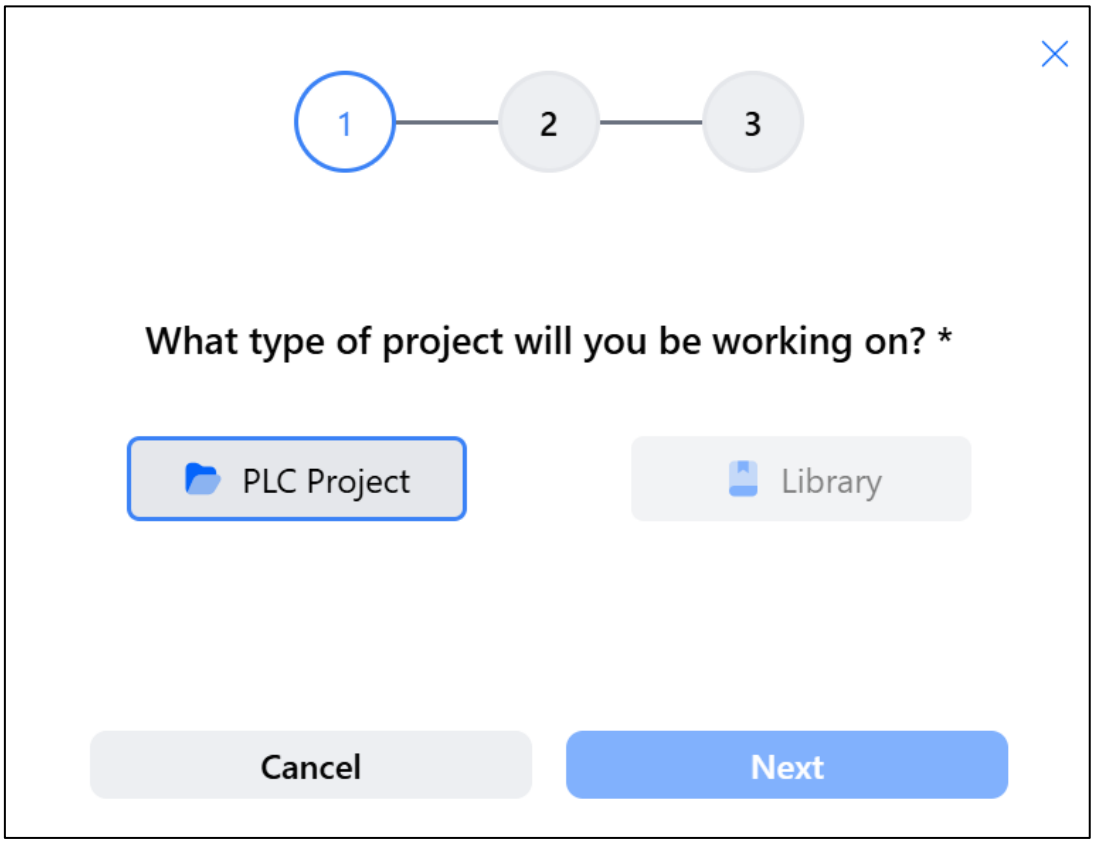


Figure 4-18. Creating a New PLC Project

- Give a name and choose empty for project

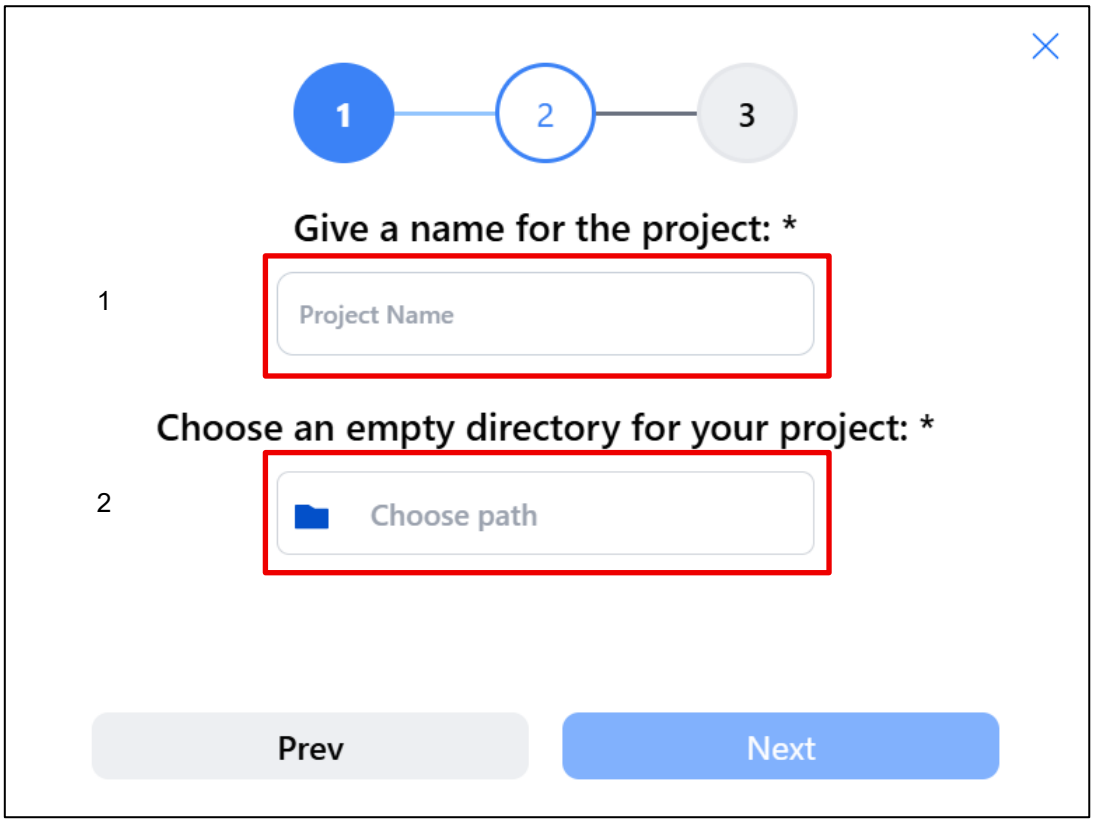


Figure 4-19. Configuring New PLC Project Settings

- Select Language and Cyclic Task:

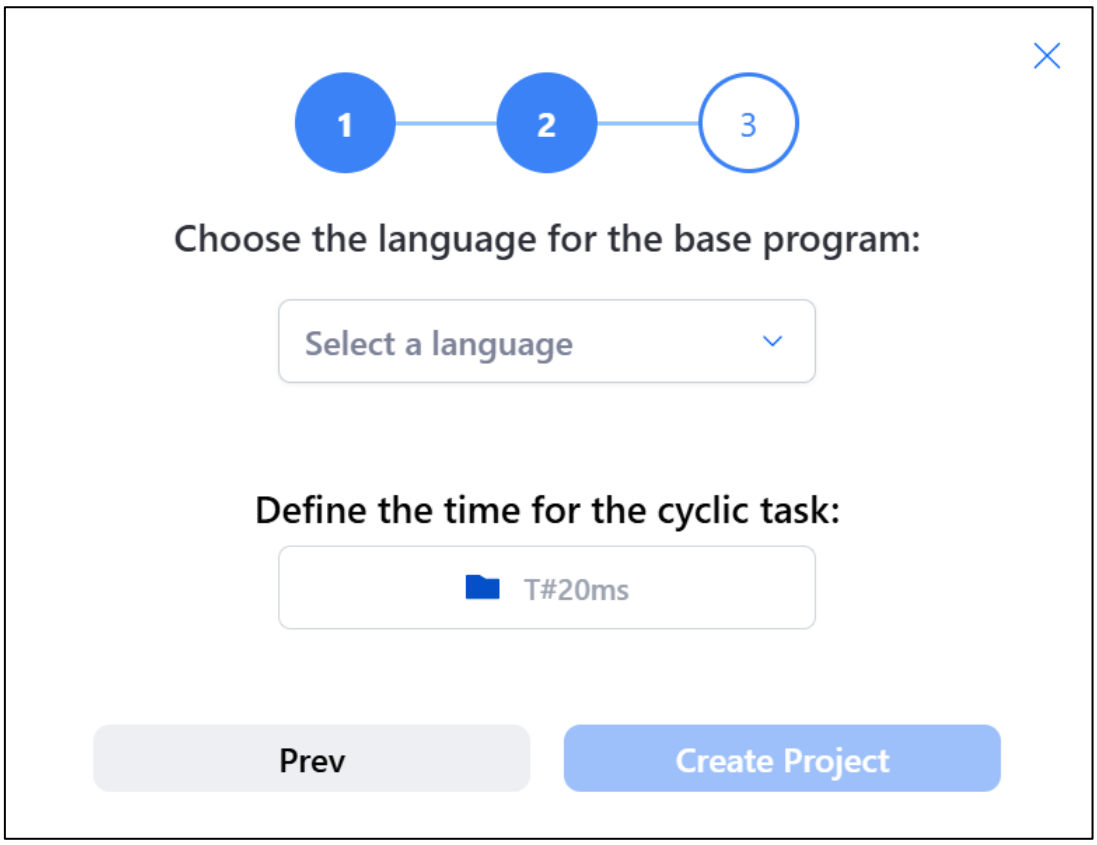


Figure 4-20. Selecting the Programming Language and Cyclic Task

- Select language: **Ladder Diagram**:

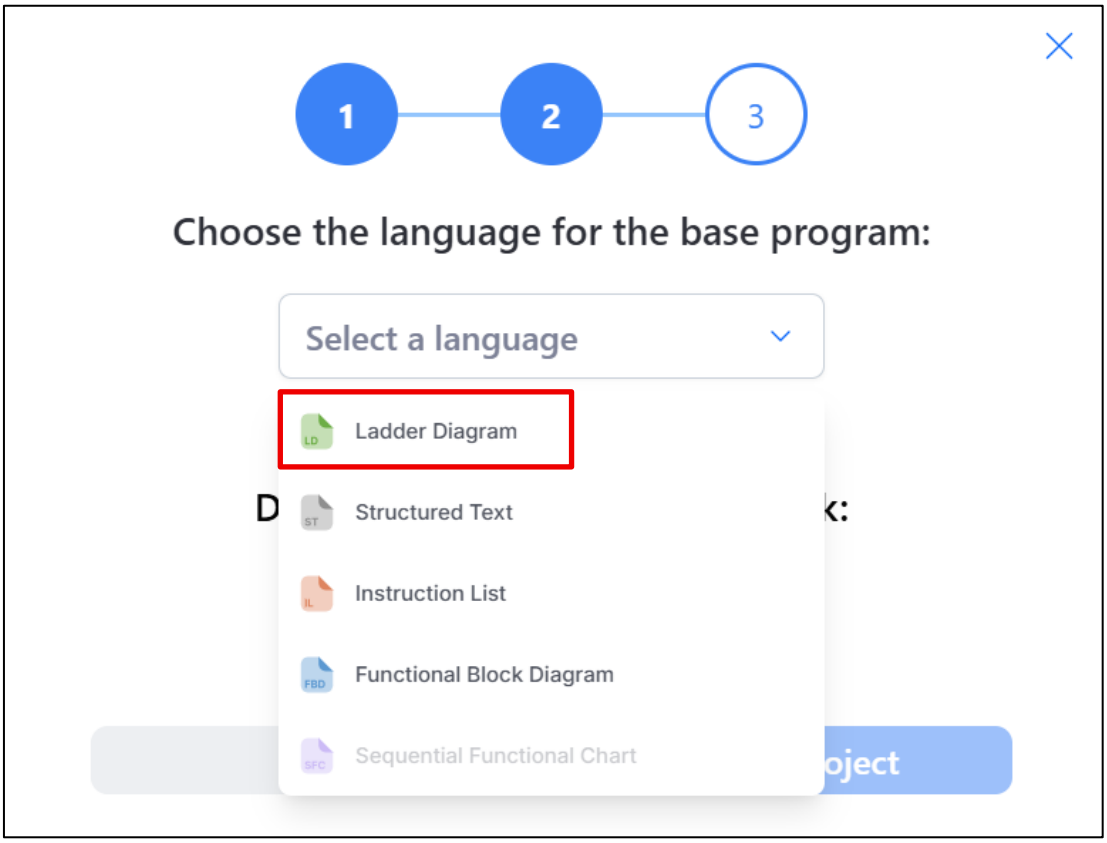


Figure 4-21. Ladder Diagram Language Selection

- Define Cyclic task time → OK → Create Project

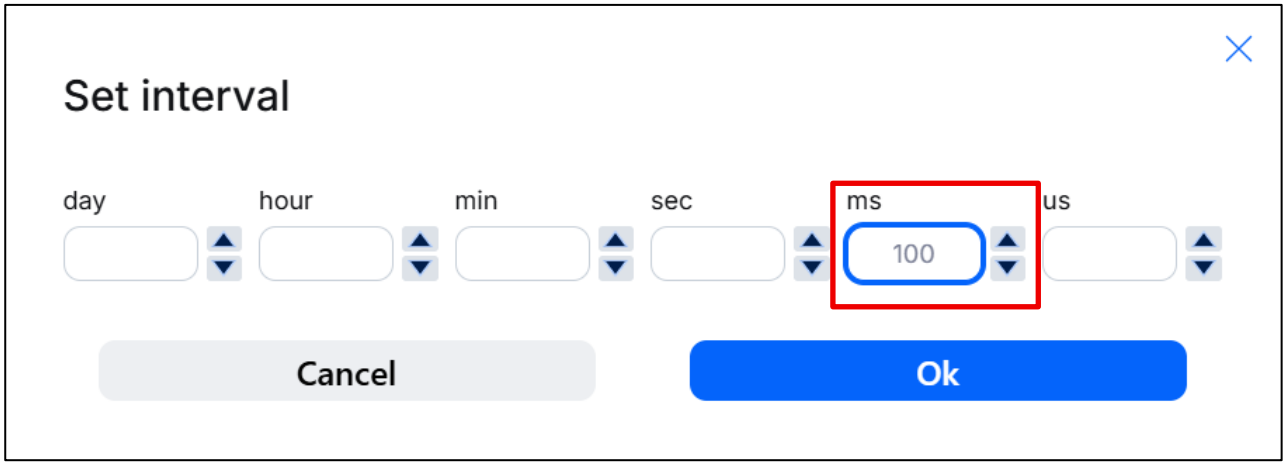


Figure 4-22. Configuring the Cyclic Task Interval

(4) Define Variables

- Click “Plus Icon” → Add **input_val** and **output_val**

Table 4-6. PLC Variable Definitions

#	Name	Class	Type	Location
0	input_val	Local	INT	%QW0
1	output_val	Local	INT	%QW1

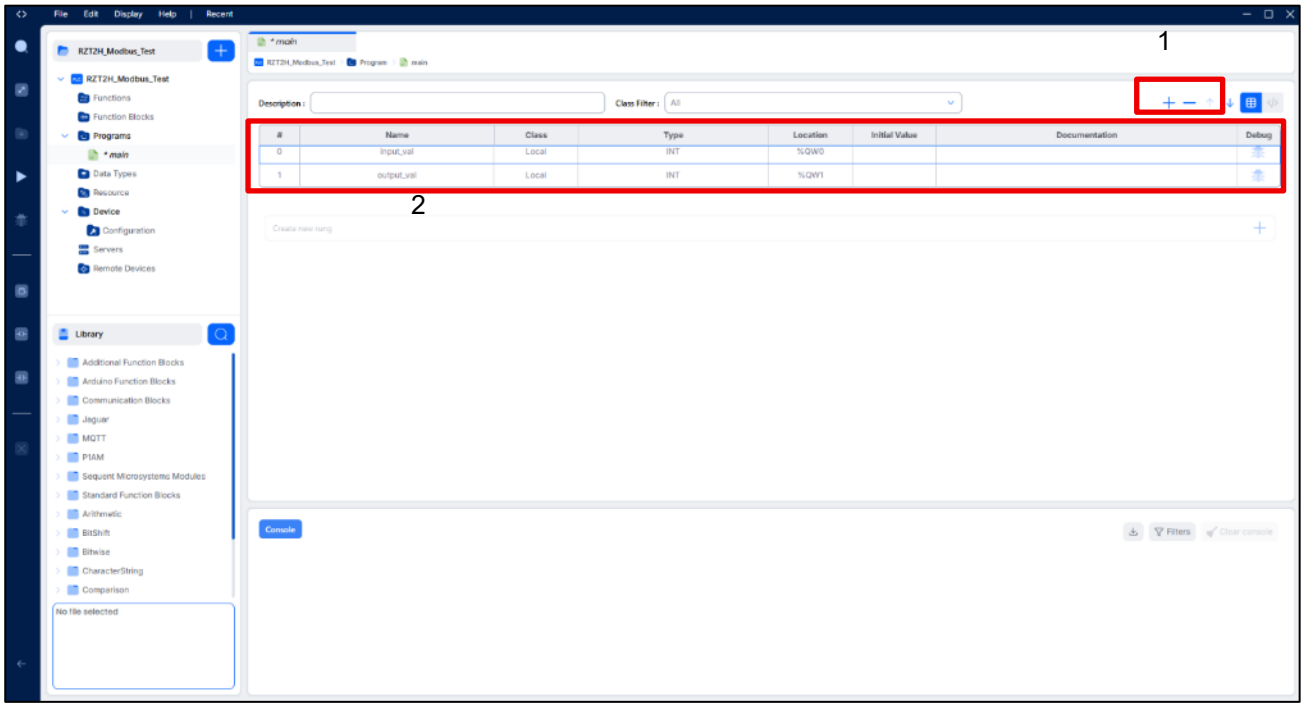
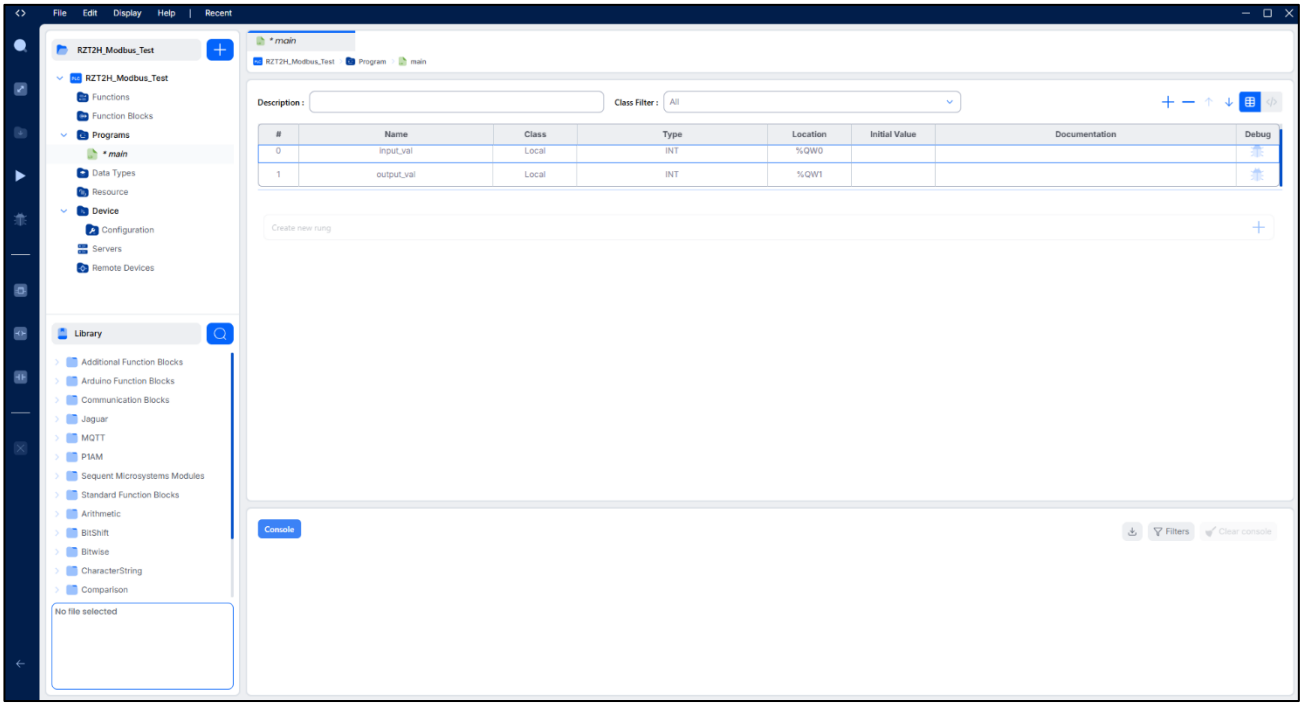


Figure 4-23. Adding Input and Output Variables

- Create new rung: In the Ladder Diagram editor, click the **"+"** button on the right side of the **"Create new rung"** text to add a new rung. A new rung will appear with a left rail and right rail as shown below.
Figure 4-24. **Creating a New Ladder Logic Rung**



- Add Function Block **MUL: Library in left side bar** → **Arithmetic** → **MUL** → drag and drop it into the center of the rung.

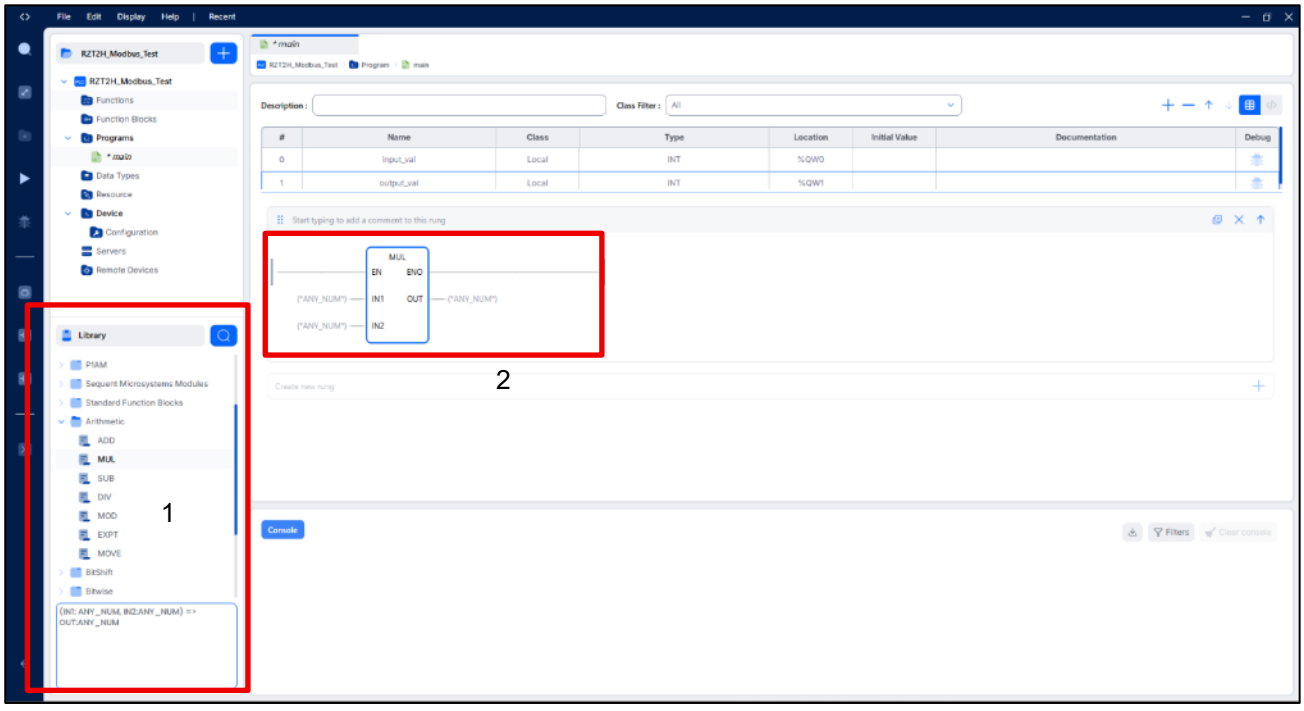


Figure 4-25. Adding the MUL Function Block to the Ladder Diagram

- Assign Variables to MUL Block:

Assign IN1 → input_val:
1. Click on (*ANY_NUM*) on the left side of IN1
2. Type input_val → Press Enter

Assign IN2 → value :
1. Click on (*ANY_NUM*) on the left side of IN2
2. Type 5 → Press Enter

Assign OUT → output_val:
1. Click on (*ANY_NUM*) on the right side of OUT
2. Type output_val → Press Enter

Press Ctrl + S to save the project.

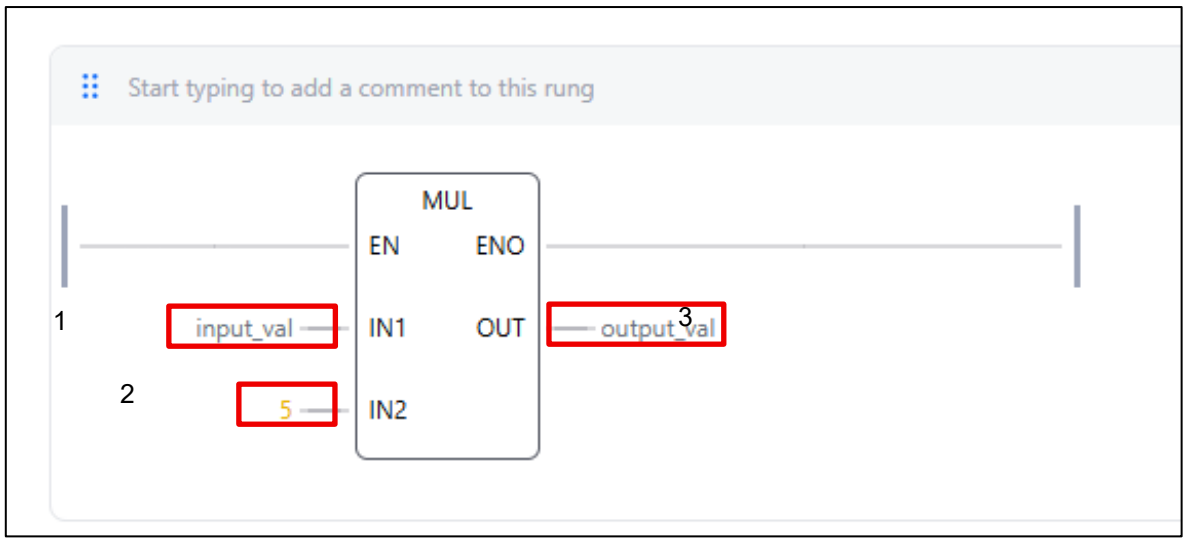


Figure 4-26. Configuring the MUL Function Block

(5) Install OpenPLC Runtime v4 on the Board

Note: OpenPLC Runtime v4 must be checked out at tag v4.0.9 to ensure compatibility with OpenPLC Editor v4. Using a newer version (e.g. 4.1.0-RC3) will result in a "Runtime Version Mismatch" error when connecting from the Editor.

Run the following commands on the board terminal to install OpenPLC Runtime v4.

- Clone the repository

```
$ git clone https://github.com/Autonomy-Logic/openplc-runtime.git
$ cd openplc-runtime
$ git checkout v4.0.9
```

- Create Python virtual environment and install main dependencies

```
$ python3 -m venv .venv
$ source .venv/bin/activate
$ pip install -r requirements.txt
$ deactivate
```

- Install plugin virtual environments

```
# modbus_master plugin
$ python3 -m venv venvs/modbus_master
$ source venvs/modbus_master/bin/activate
$ pip install pymodbus==3.11.2
$ deactivate

# modbus_slave plugin
$ python3 -m venv venvs/modbus_slave
$ source venvs/modbus_slave/bin/activate
$ pip install pymodbus==3.11.2
$ deactivate

# opcua plugin
$ python3 -m venv venvs/opcua
$ source venvs/opcua/bin/activate
$ pip install asyncua
$ deactivate

# runtime venv
python3 -m venv venvs/runtime
source venvs/runtime/bin/activate
pip install -r requirements.txt
deactivate
```

- Build the C core

```
$ mkdir -p build && cd build
$ cmake ..
$ make
$ cd ..
```

- Bypass installation check

```
$ touch /openplc-runtime/.installed
```

(6) Configure Modbus TCP Plugin on the Board

- Create the Modbus slave configuration file:

```
$ cat > /openplc-runtime/core/src/drivers/plugins/python/modbus_slave/modbus_slave_config.json << 'EO
F'
{
  "network_configuration": {
    "host": "0.0.0.0",
    "port": 502
  }
}
EOF

$ cp /openplc-runtime/plugins_default.conf /openplc-runtime/plugins.conf
```

- Enable the Modbus slave plugin:

```
$ sed -i 's/^modbus_slave,\(.*\),0,0,/modbus_slave,\1,1,0,/' /openplc-runtime/plugins.conf
```

(7) Start OpenPLC Runtime v4

```
$ source .venv/bin/activate
$ ./start_openplc.sh > /tmp/webserver.log 2>&1 &
$ sleep 8
$ tail -f /tmp/webserver.log
Press CTRL+C to quit
```

- Create admin account (first time only)

```
$ curl -k -X POST https://127.0.0.1:8443/api/create-user \
  -H "Content-Type: application/json" \
  -d '{"username":"openplc","password":"openplc","role":"admin"}'
```

Note**:** The create-user step is required only on the first run. If the account already exists, the response will be {"msg":"User already created!"} - this can be safely ignored.

- Verify Runtime v4 is listening on port 8443:

```
$ netstat -tln | grep 8443
```

Expected output:

```
 tcp 0 0 0.0.0.0:8443 0.0.0.0:* LISTEN
```

(8) Connect OpenPLC Editor to the Board

- In OpenPLC Editor, navigate to Device > Configuration and configure as shown in Table 4-7:

Table 4-7. OpenPLC Editor Device Configuration Settings

Field	Value
Device	OpenPLC Runtime v4
IP Address	Your IP Address

- Click Connect. A login dialog will appear. Enter the following credentials in Table 4-8:

Table 4-8. OpenPLC Runtime Default Login Credentials

Field	Value
Username	openplc
Password	openplc

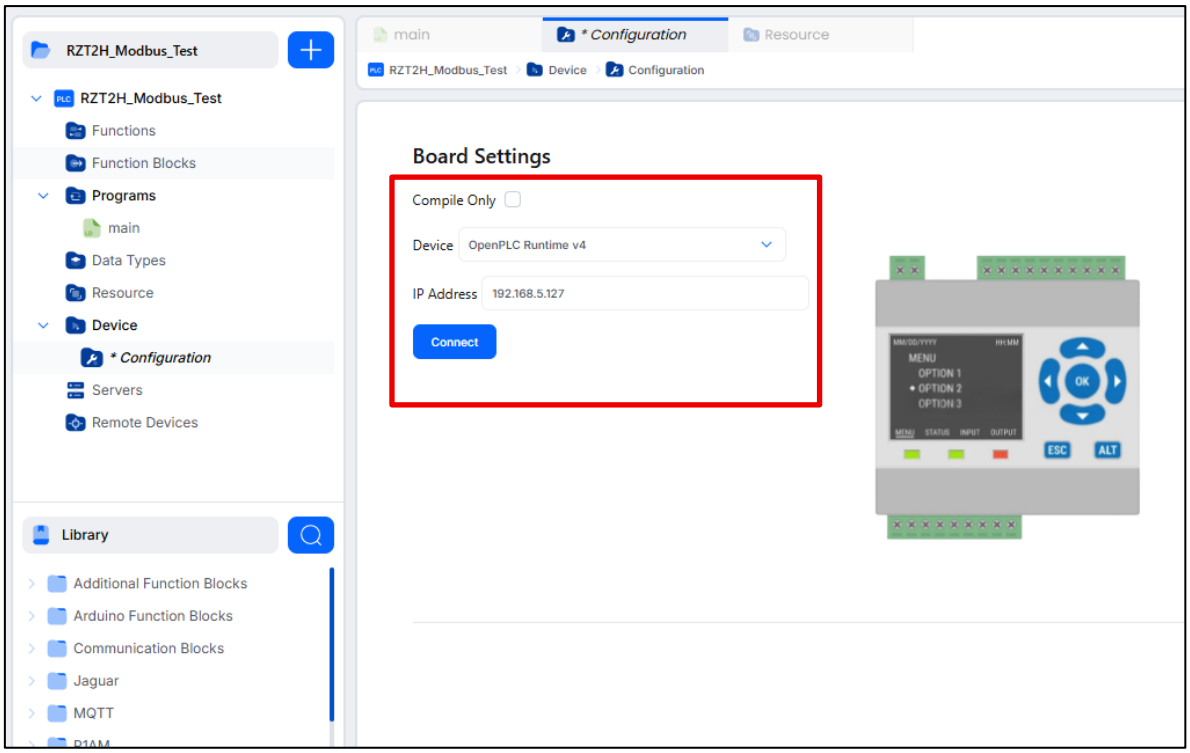


Figure 4-27. Configuring the OpenPLC Runtime Connection

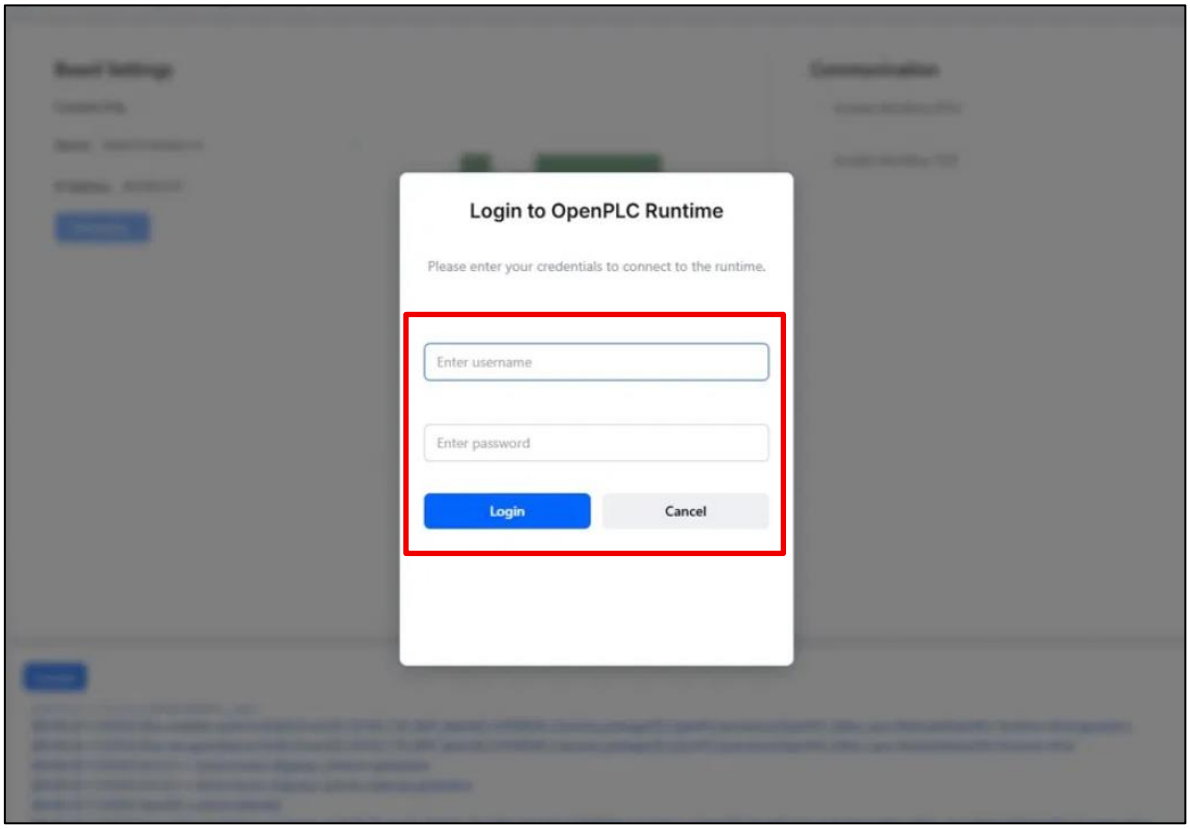


Figure 4-28. Logging in to OpenPLC Runtime

(9) Upload and Run the Program

- Now connected OK but PLC will show **EMPTY**
- Click the Run **Compile** button in the left sidebar. The Editor will compile the Ladder Diagram, upload it to the board, and automatically start the PLC. After the process completes, the status should show **Connected | PLC: RUNNING** with Scan Cycle Statistics displayed.

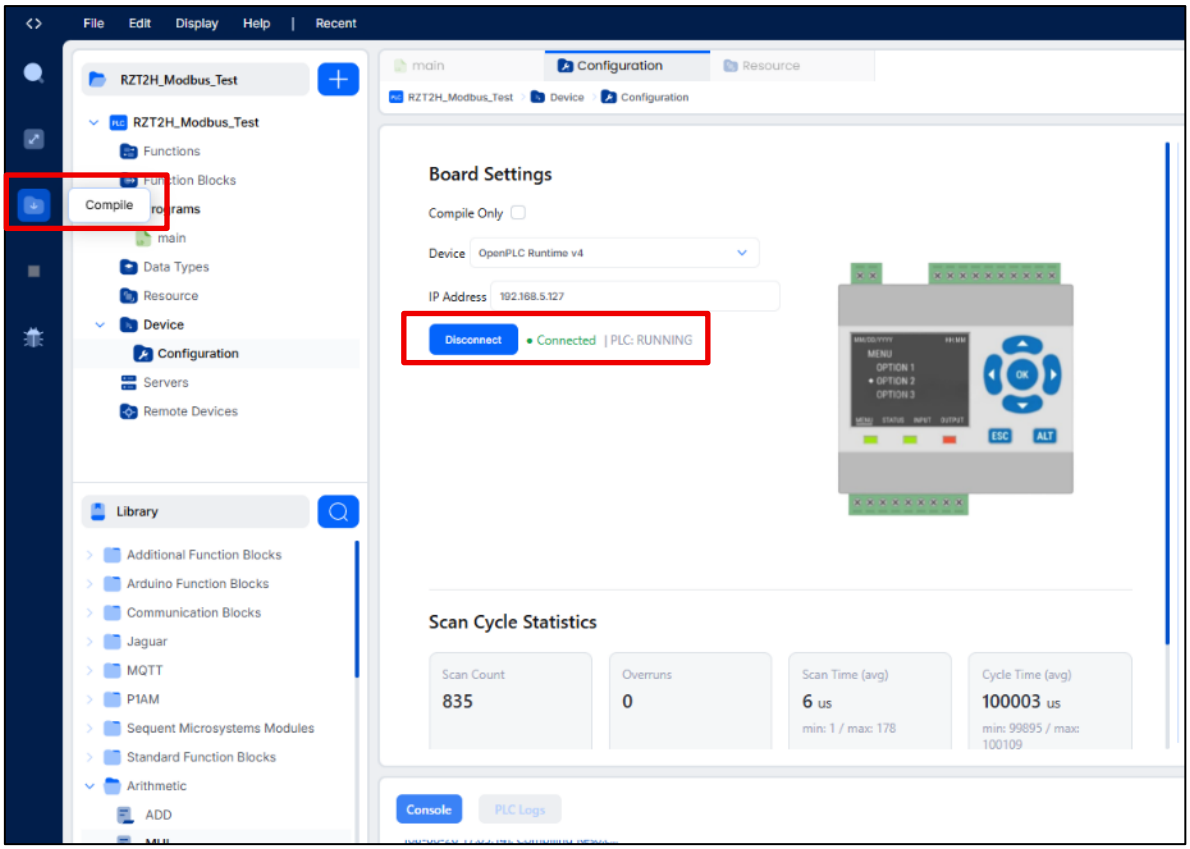


Figure 4-29. Successful Deployment of the PLC Program

(10) Re-enable Modbus Plugin After Each Upload:

- Each time a new program is uploaded from the Editor, the Modbus slave plugin must be re-enabled on the board. Run the following commands:

```
$ chmod 644 /openplc-runtime/plugins.conf
$ sed -i 's/^modbus_slave,\(.*\),0,0,/modbus_slave,\1,1,0,/' /openplc-runtime/plugins.conf
```

- Then two time click Run (triangle play) button in the left sidebar again in the Editor (below compile button) to stop and start the PLC again. Verify port 502 is listening:

```
$ netstat -tln | grep 502
```

Expected output:

```
tcp  0  0  0.0.0.0:502  0.0.0.0:*  LISTEN
```

(11) Verify Communication with Simply Modbus TCP Client

- Open Simply Modbus TCP Client 8.1.2 on the Windows PC and configure as follows:

Table 4-9. Modbus Read Holding Register Parameters

Field	Value
IP Address	192.168.5.127
Port	502
Function code	3 (Read Holding Registers)
First Register	40001
No. of Regs	5

- Click “CONNECT”, status will change from “NOT CONNECTED” to “CONNECTED”

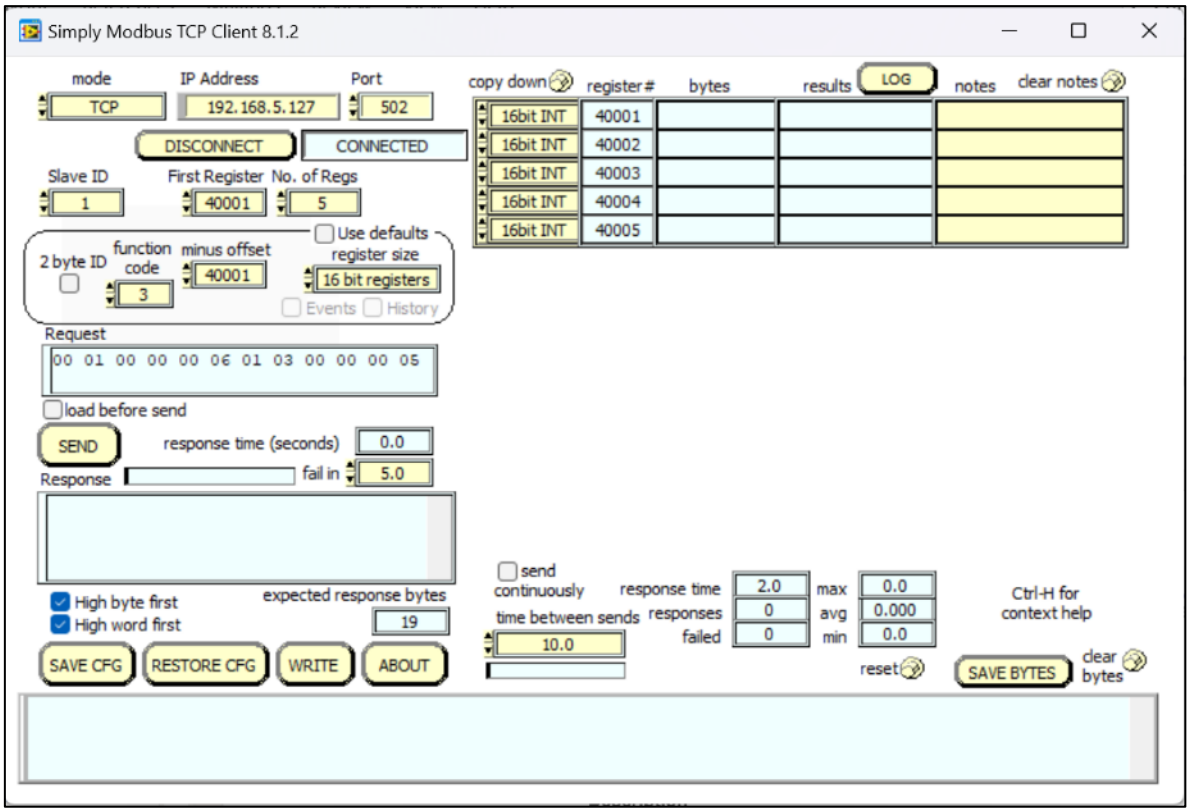


Figure 4-30. SimlyModbus TCP Client Connected to OpenPLC Runtime

- Then click WRITE to write value 100 to register 40001:

Table 4-10. Modbus Write Register Parameters

Field	Value
Function code	6 (Write Single Register)
First Register	40001
Values to Write	100

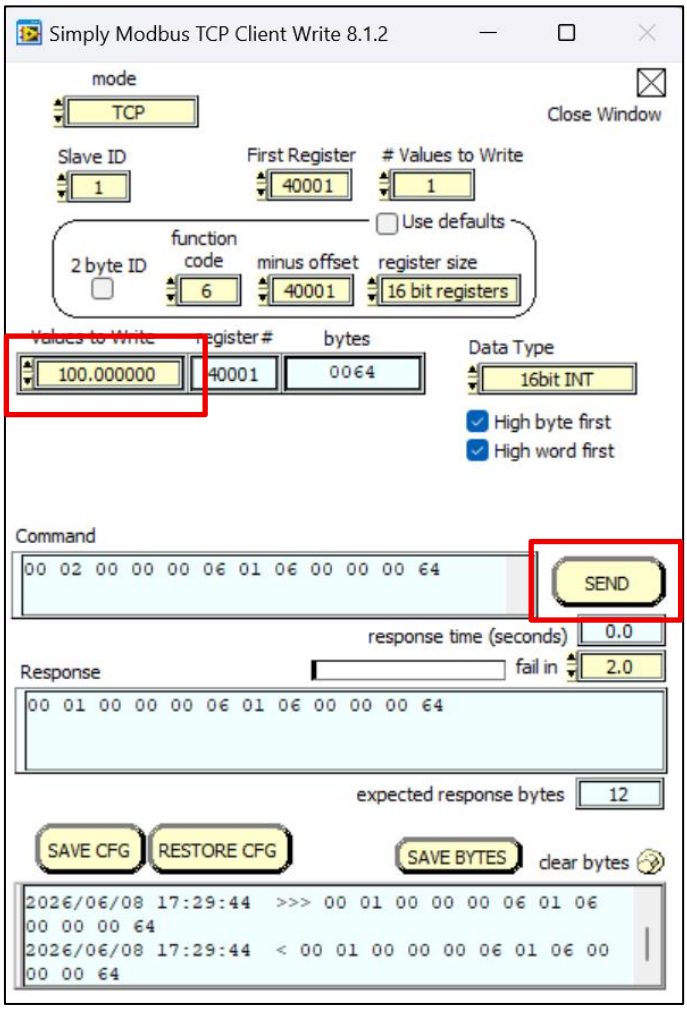


Figure 4-31. Writing Value 100 to Holding Register 40001

- Click SEND. Then switch back to FC3 and read the registers. Expected results:

Table 4-11. Expected Register Values After Program Execution

Register	Variable	Expected Value
40001	%QW0 (input_val)	100
40002	%QW1 (output_val)	500

Register 40002 showing 500 (= 100 x 5) confirms the Ladder Diagram program uploaded from OpenPLC Editor is executing correctly on the RZ/T2N board and communicating via Modbus TCP.

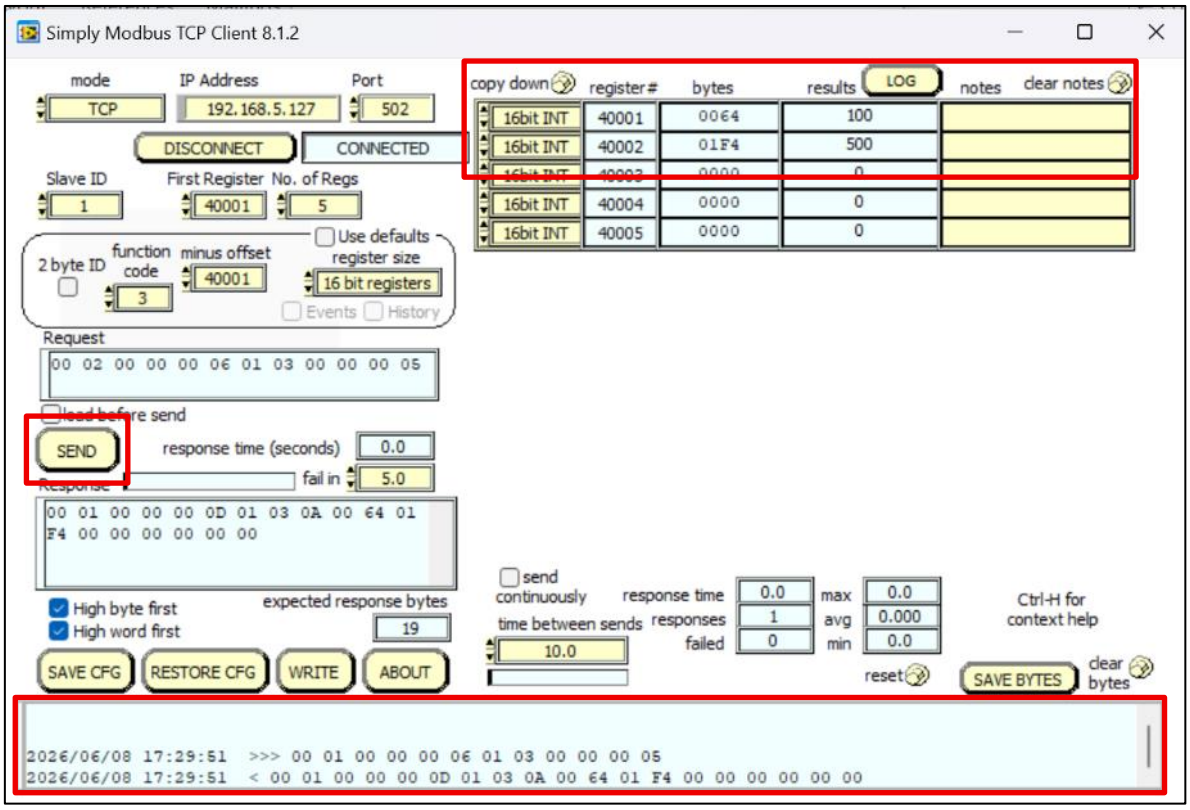


Figure 4-32. Verifying PLC Output Through Modbus Registers

Note: After each program upload from OpenPLC Editor, the Modbus slave plugin is disabled automatically. Always run the re-enable commands in step 4.2 (10) before testing with Simply Modbus TCP Client.

(12) Summary

Table 4-12. Summary of Software Components and Communication Protocols

Component	Role	Protocol	Port
OpenPLC Editor v4	Write Ladder Diagram, compile and upload PLC program, start/stop PLC	HTTPS	8443
OpenPLC Runtime v4	Execute PLC program on board, manage plugins, expose REST API	HTTPS	8443
Modbus Slave Plugin	Map PLC variables (%QW0, %QW1) to Modbus holding registers (40001, 40002)	Modbus TCP	502
Simply Modbus TCP Client	Write input value to register 40001, read output value from register 40002 to verify	Modbus TCP	502

- Complete the following flow:

1. Editor upload program → Runtime compile → PLC RUNNING
2. Simply Modbus write 100 → register 40001 → %QW0 = 100
3. PLC caculate: output = input × 5 → %QW1 = 500
4. Simply Modbus read → register 40002 → %QW1 = 500

5. Revision History

Rev.	Date	Description	
		Page	Summary
1.00	Oct. 29, 2026	-	First edition issued.

Website and Support

Renesas Electronics Website
http://www.renesas.com/

Contact Us
http://www.renesas.com/contact/

All trademarks and registered trademarks are the property of their respective owners.

Notice

1. Descriptions of circuits, software and other related information in this document are provided only to illustrate the operation of semiconductor products and application examples. You are fully responsible for the incorporation or any other use of the circuits, software, and information in the design of your product or system. Renesas Electronics disclaims any and all liability for any losses and damages incurred by you or third parties arising from the use of these circuits, software, or information.
2. Renesas Electronics hereby expressly disclaims any warranties against and liability for infringement or any other claims involving patents, copyrights, or other intellectual property rights of third parties, by or arising from the use of Renesas Electronics products or technical information described in this document, including but not limited to, the product data, drawings, charts, programs, algorithms, and application examples.
3. No license, express, implied or otherwise, is granted hereby under any patents, copyrights or other intellectual property rights of Renesas Electronics or others.
4. You shall be responsible for determining what licenses are required from any third parties, and obtaining such licenses for the lawful import, export, manufacture, sales, utilization, distribution or other disposal of any products incorporating Renesas Electronics products, if required.
5. You shall not alter, modify, copy, or reverse engineer any Renesas Electronics product, whether in whole or in part. Renesas Electronics disclaims any and all liability for any losses or damages incurred by you or third parties arising from such alteration, modification, copying or reverse engineering.
6. Renesas Electronics products are classified according to the following two quality grades: “Standard” and “High Quality”. The intended applications for each Renesas Electronics product depends on the product’s quality grade, as indicated below.
   "Standard": Computers; office equipment; communications equipment; test and measurement equipment; audio and visual equipment; home electronic appliances; machine tools; personal electronic equipment; industrial robots; etc.
   "High Quality": Transportation equipment (automobiles, trains, ships, etc.); traffic control (traffic lights); large-scale communication equipment; key financial terminal systems; safety control equipment; etc.
   Unless expressly designated as a high reliability product or a product for harsh environments in a Renesas Electronics data sheet or other Renesas Electronics document, Renesas Electronics products are not intended or authorized for use in products or systems that may pose a direct threat to human life or bodily injury (artificial life support devices or systems; surgical implantations; etc.), or may cause serious property damage (space system; undersea repeaters; nuclear power control systems; aircraft control systems; key plant systems; military equipment; etc.). Renesas Electronics disclaims any and all liability for any damages or losses incurred by you or any third parties arising from the use of any Renesas Electronics product that is inconsistent with any Renesas Electronics data sheet, user’s manual or other Renesas Electronics document.
7. No semiconductor product is absolutely secure. Notwithstanding any security measures or features that may be implemented in Renesas Electronics hardware or software products, Renesas Electronics shall have absolutely no liability arising out of any vulnerability or security breach, including but not limited to any unauthorized access to or use of a Renesas Electronics product or a system that uses a Renesas Electronics product. RENESAS ELECTRONICS DOES NOT WARRANT OR GUARANTEE THAT RENESAS ELECTRONICS PRODUCTS, OR ANY SYSTEMS CREATED USING RENESAS ELECTRONICS PRODUCTS WILL BE INVULNERABLE OR FREE FROM CORRUPTION, ATTACK, VIRUSES, INTERFERENCE, HACKING, DATA LOSS OR THEFT, OR OTHER SECURITY INTRUSION (“Vulnerability Issues”). RENESAS ELECTRONICS DISCLAIMS ANY AND ALL RESPONSIBILITY OR LIABILITY ARISING FROM OR RELATED TO ANY VULNERABILITY ISSUES. FURTHERMORE, TO THE EXTENT PERMITTED BY APPLICABLE LAW, RENESAS ELECTRONICS DISCLAIMS ANY AND ALL WARRANTIES, EXPRESS OR IMPLIED, WITH RESPECT TO THIS DOCUMENT AND ANY RELATED OR ACCOMPANYING SOFTWARE OR HARDWARE, INCLUDING BUT NOT LIMITED TO THE IMPLIED WARRANTIES OF MERCHANTABILITY, OR FITNESS FOR A PARTICULAR PURPOSE.
8. When using Renesas Electronics products, refer to the latest product information (data sheets, user’s manuals, application notes, “General Notes for Handling and Using Semiconductor Devices” in the reliability handbook, etc.), and ensure that usage conditions are within the ranges specified by Renesas Electronics with respect to maximum ratings, operating power supply voltage range, heat dissipation characteristics, installation, etc. Renesas Electronics disclaims any and all liability for any malfunctions, failure or accident arising out of the use of Renesas Electronics products outside of such specified ranges.
9. Although Renesas Electronics endeavors to improve the quality and reliability of Renesas Electronics products, semiconductor products have specific characteristics, such as the occurrence of failure at a certain rate and malfunctions under certain use conditions. Unless designated as a high reliability product or a product for harsh environments in a Renesas Electronics data sheet or other Renesas Electronics document, Renesas Electronics products are not subject to radiation resistance design. You are responsible for implementing safety measures to guard against the possibility of bodily injury, injury or damage caused by fire, and/or danger to the public in the event of a failure or malfunction of Renesas Electronics products, such as safety design for hardware and software, including but not limited to redundancy, fire control and malfunction prevention, appropriate treatment for aging degradation or any other appropriate measures. Because the evaluation of microcomputer software alone is very difficult and impractical, you are responsible for evaluating the safety of the final products or systems manufactured by you.
10. Please contact a Renesas Electronics sales office for details as to environmental matters such as the environmental compatibility of each Renesas Electronics product. You are responsible for carefully and sufficiently investigating applicable laws and regulations that regulate the inclusion or use of controlled substances, including without limitation, the EU RoHS Directive, and using Renesas Electronics products in compliance with all these applicable laws and regulations. Renesas Electronics disclaims any and all liability for damages or losses occurring as a result of your noncompliance with applicable laws and regulations.
11. Renesas Electronics products and technologies shall not be used for or incorporated into any products or systems whose manufacture, use, or sale is prohibited under any applicable domestic or foreign laws or regulations. You shall comply with any applicable export control laws and regulations promulgated and administered by the governments of any countries asserting jurisdiction over the parties or transactions.
12. It is the responsibility of the buyer or distributor of Renesas Electronics products, or any other party who distributes, disposes of, or otherwise sells or transfers the product to a third party, to notify such third party in advance of the contents and conditions set forth in this document.
13. This document shall not be reprinted, reproduced or duplicated in any form, in whole or in part, without prior written consent of Renesas Electronics.
14. Please contact a Renesas Electronics sales office if you have any questions regarding the information contained in this document or Renesas Electronics products.

(Note1) “Renesas Electronics” as used in this document means Renesas Electronics Corporation and also includes its directly or indirectly controlled subsidiaries.
(Note2) “Renesas Electronics product(s)” means any product developed or manufactured by or for Renesas Electronics.

(Rev.5.0-1 October 2020)

Corporate Headquarters

TOYOSU FORESIA, 3-2-24 Toyosu,
Koto-ku, Tokyo 135-0061, Japan
www.renesas.com

Trademarks

Renesas and the Renesas logo are trademarks of Renesas Electronics Corporation. All trademarks and registered trademarks are the property of their respective owners.

Contact information

For further information on a product, technology, the most up-to-date version of a document, or your nearest sales office, please visit:
www.renesas.com/contact/.

© 2026 Renesas Electronics Corporation. All rights reserved

General Precautions in the Handling of Microprocessing Unit and Microcontroller Unit Products

The following usage notes are applicable to all Microprocessing unit and Microcontroller unit products from Renesas. For detailed usage notes on the products covered by this document, refer to the relevant sections of the document as well as any technical updates that have been issued for the products.

1. Precaution against Electrostatic Discharge (ESD)

   A strong electrical field, when exposed to a CMOS device, can cause destruction of the gate oxide and ultimately degrade the device operation. Steps must be taken to stop the generation of static electricity as much as possible, and quickly dissipate it when it occurs. Environmental control must be adequate. When it is dry, a humidifier should be used. This is recommended to avoid using insulators that can easily build up static electricity. Semiconductor devices must be stored and transported in an anti-static container, static shielding bag or conductive material. All test and measurement tools including work benches and floors must be grounded. The operator must also be grounded using a wrist strap. Semiconductor devices must not be touched with bare hands. Similar precautions must be taken for printed circuit boards with mounted semiconductor devices.

2. Processing at power-on

   The state of the product is undefined at the time when power is supplied. The states of internal circuits in the LSI are indeterminate and the states of register settings and pins are undefined at the time when power is supplied. In a finished product where the reset signal is applied to the external reset pin, the states of pins are not guaranteed from the time when power is supplied until the reset process is completed. In a similar way, the states of pins in a product that is reset by an on-chip power-on reset function are not guaranteed from the time when power is supplied until the power reaches the level at which resetting is specified.

3. Input of signal during power-off state

   Do not input signals or an I/O pull-up power supply while the device is powered off. The current injection that results from input of such a signal or I/O pull-up power supply may cause malfunction and the abnormal current that passes in the device at this time may cause degradation of internal elements. Follow the guideline for input signal during power-off state as described in your product documentation.

4. Handling of unused pins

   Handle unused pins in accordance with the directions given under handling of unused pins in the manual. The input pins of CMOS products are generally in the high-impedance state. In operation with an unused pin in the open-circuit state, extra electromagnetic noise is induced in the vicinity of the LSI, an associated shoot-through current flows internally, and malfunctions occur due to the false recognition of the pin state as an input signal become possible.

5. Clock signals

   After applying a reset, only release the reset line after the operating clock signal becomes stable. When switching the clock signal during program execution, wait until the target clock signal is stabilized. When the clock signal is generated with an external resonator or from an external oscillator during a reset, ensure that the reset line is only released after full stabilization of the clock signal. Additionally, when switching to a clock signal produced with an external resonator or by an external oscillator while program execution is in progress, wait until the target clock signal is stable.

6. Voltage application waveform at input pin

   Waveform distortion due to input noise or a reflected wave may cause malfunction. If the input of the CMOS device stays in the area between V_{IL} (Max.) and V_{IH} (Min.) due to noise, for example, the device may malfunction. Take care to prevent chattering noise from entering the device when the input level is fixed, and also in the transition period when the input level passes through the area between V_{IL} (Max.) and V_{IH} (Min.).

7. Prohibition of access to reserved addresses

   Access to reserved addresses is prohibited. The reserved addresses are provided for possible future expansion of functions. Do not access these addresses as the correct operation of the LSI is not guaranteed.

8. Differences between products

   Before changing from one product to another, for example to a product with a different part number, confirm that the change will not lead to problems. The characteristics of a microprocessing unit or microcontroller unit products in the same group but having a different part number might differ in terms of internal memory capacity, layout pattern, and other factors, which can affect the ranges of electrical characteristics, such as characteristic values, operating margins, immunity to noise, and amount of radiated noise. When changing to a product with a different part number, implement a systemevaluation test for the given product.

© 2026 Renesas Electronics Corporation. All rights reserved