RZ/N2L Group

RZN2L Industrial Network SOM Kit Application Note: Modbus TCP Slave Software

Introduction

This document explains Sample Program setup procedures for Modbus TCP functionalities with the adapted Modbus protocol stack code for Renesas RZ/N2L Industrial Network SOM Kit.

Target Device

RZ/N2L

Contents

1. Overview

This document describes the procedure for testing the Modbus TCP slave function using Modbus protocol stack code compatible with the Renesas RZ/N2L Industrial Network SOM Kit.

1.1 Abbreviations / Definitions

Table 1.1 Abbreviations/Definitions

Index	Abbreviations /Definitions	Description
1	IP	Internet Protocol
2	TCP	Transmission Control Protocol
3	USB	Universal Serial Bus
4	PC	Personal Computer
5	SW	Switch
6	EWARM	Embedded Workbench® for ARM
7	LED	Light Emitting Diode

1.2 Reference

Technical information about RZ/N2L are available via Renesas.

Table 1.2 Technical Inputs for RZ/N2L

Index	Technical Inputs
1	r01uh0955ejxxxx-rzn2l.pdf (RZ/N2L User's Manual: Hardware)
2	r01an6434ejxxxx-rzt2-rzn2-fsp-getting-started.pdf (Getting started with Flexible Software Package)
3	r12ut0020edxxxx-rzn2l-som-kit-hw.pdf (RZ/N2L Industrial Network SOM Kit Use's Manual)

1.3 Limitation / Known Issue

None

2. Features

The Modbus protocol stack forRZ/N2L allows for quick and easy development of the Modbus TCP server. The following nine codes can be implemented in this stack.

1. (0x01) - Read coils
2. (0x02) - Read discrete input
3. (0x03) - Read holding registers
4. (0x04) - Read input registers
5. (0x05) - Write single coil
6. (0x06) - Write single register
7. (0x0F) - Write multiple coils
8. (0x10) - Write multiple registers
9. (0x17) - Read/Write multiple registers

3. Project Setup

3.1 Requirements

This Modbus protocol stack project has been developed and tested on these environments using the following boards and tools.

Table 3.1 RZ/N2L Requirements

Item	Vender	Description
Board	Renesas Electronics	RZ/N2L Industrial Network SOM Kit
IDE	IAR Systems	Embedded Workbench® for ARM Version 9.30.1 Please apply patch (EWARM_Patch_for_RZN2L_rev1.0.zip) which is available in http://www.renesas.com/rzn2l. Regarding how to apply the patch, please read the readme file in patch file.
	Renesas Electronics	e² studio 2023-04 FSP Smart Configurator 2023-04 RZ/N2L Flexible Software Package (FSP) v1.2.0 Please download from the link below. https://github.com/renesas/rzn-fsp/releases/tag/v1.2.0
Emulator	IAR Systems	I-jet
	SEGGER	J-Link
Master demo tool	Renesas Electronics	ModbusDemoApplication.exe (Included in this package)

3.1 Hardware

This document describes the major hardware. Refer to RZ/N2L Industrial Network SOM Kit user's manual and schematic for more board details.

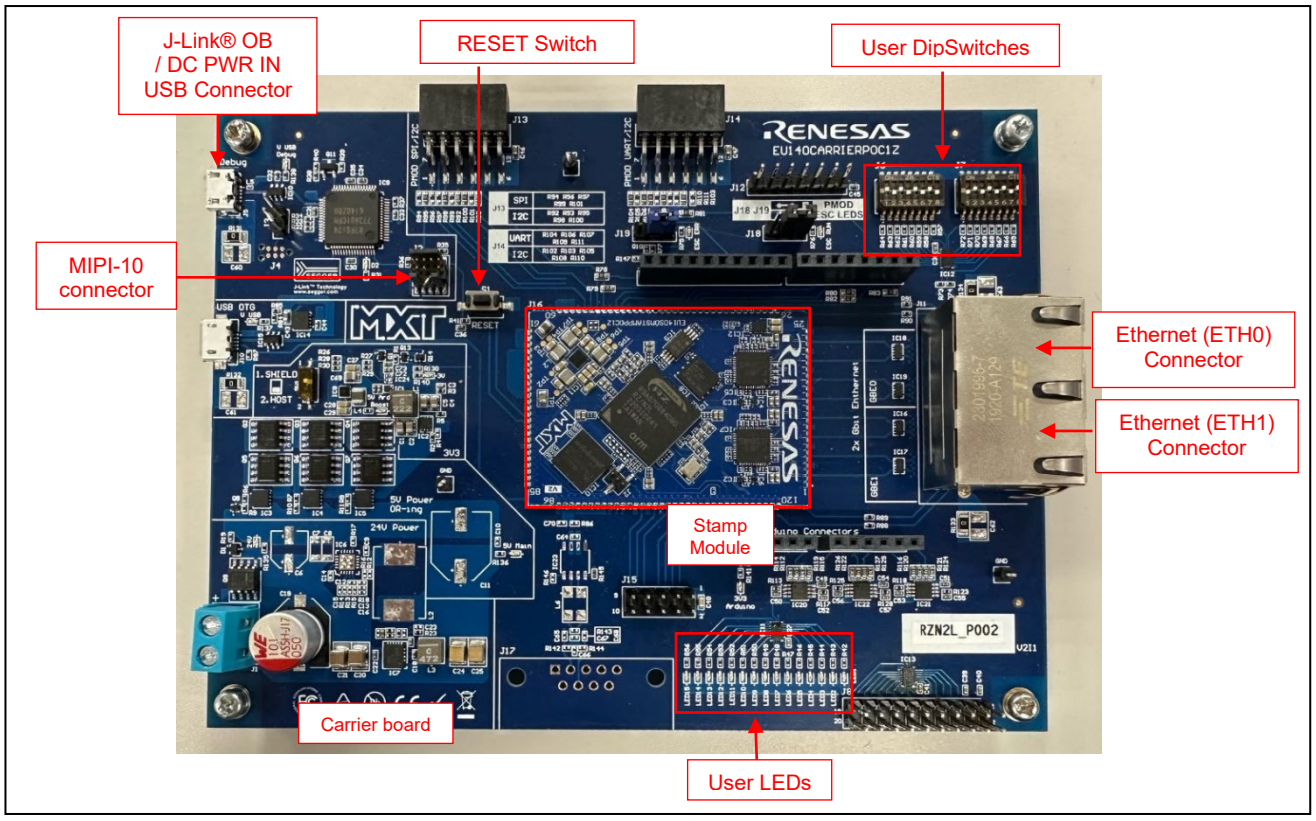

Figure 3.2 RZ/N2L Industrial Network SOM Kit

3.2 Setting the Board

Setting the board for running sample program is shown below.

1. Connect the I-jet to J2 or the USB cable to J5 for J-link OB on Carrier board.

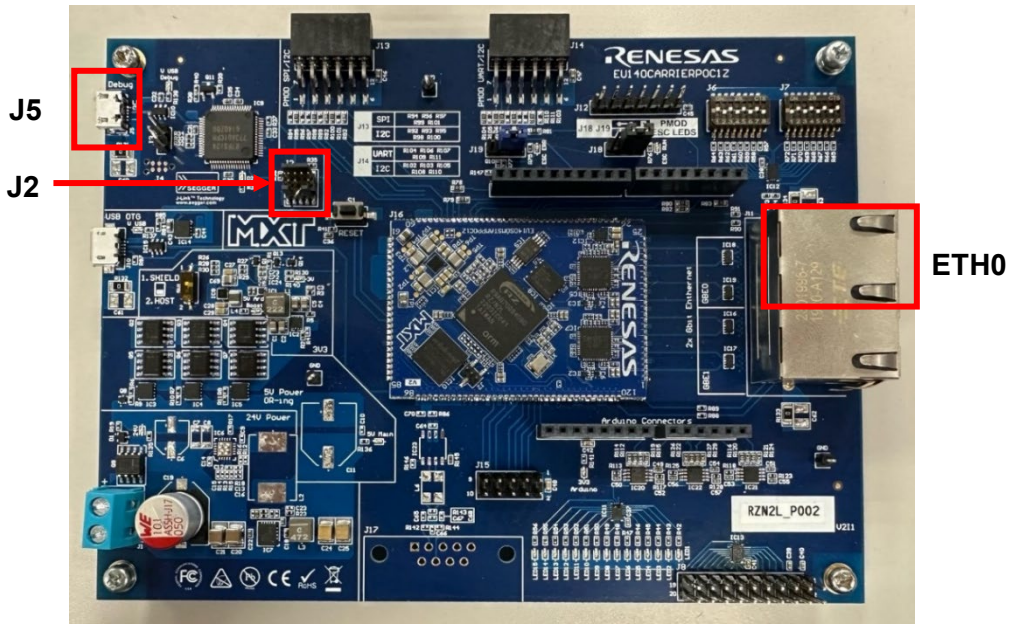

Figure 3.3 Setting the SOM Kit

2. Power is supplied by connecting USB Micro-B cable to the USB connector “J5) of the Carrier board.
3. Connect Ethernet Cable to the Ethernet Connector “ETH0”.

4. Setup a master tool

1. Open ModbusDemoApplication.exe which is included in this package.

2. Set the "Remote Modbus Server" IP Address (e.g. "192.168.1.100") and Port (e.g."502").

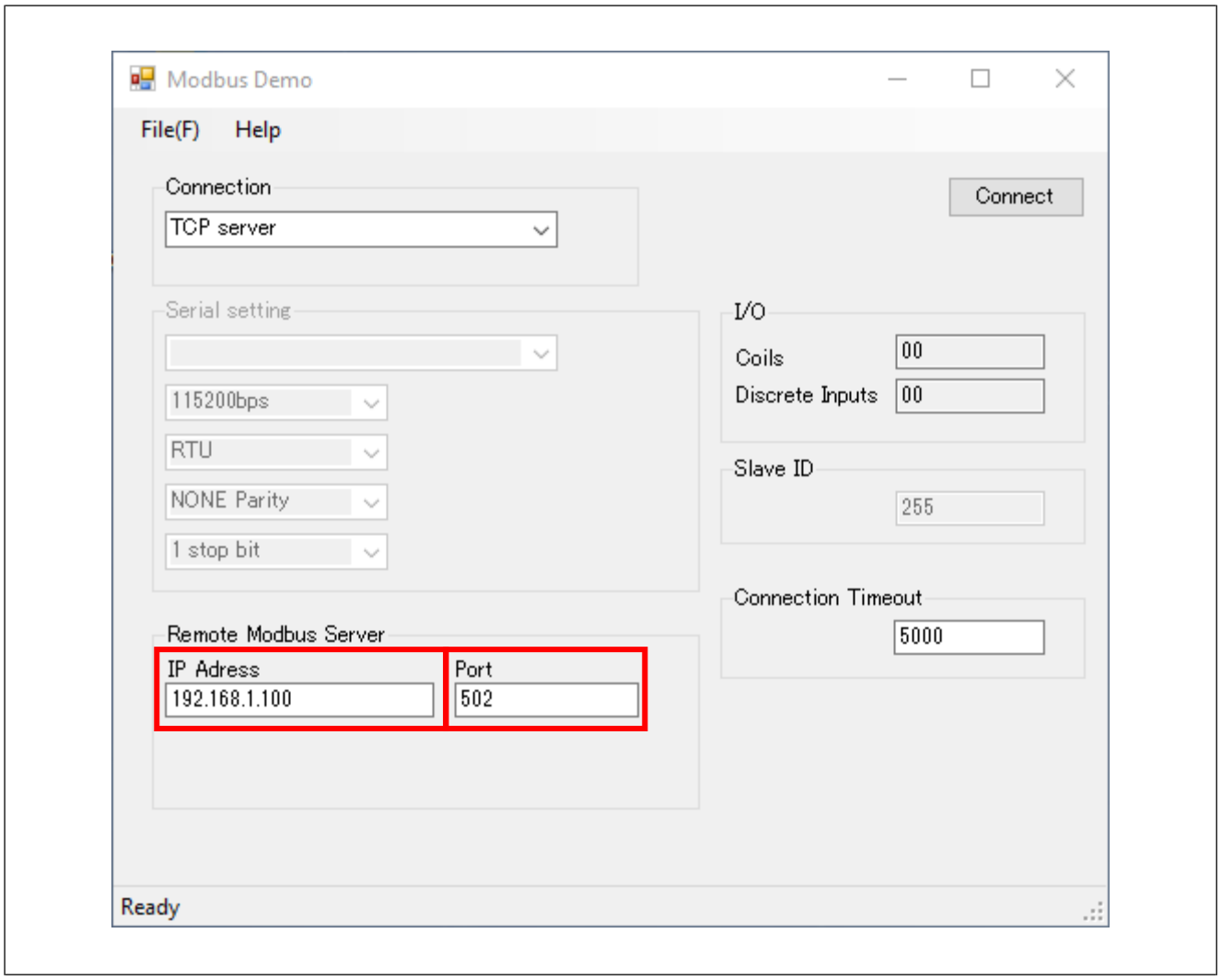

Figure 4.1 ModbusDemoApplication remote server setting

5. Running the sample application

Refer to Section 3.2 Setting the Board for board settings.

The setup differs depending on the IDE.

- When using e² studio, refer to section 5.1 and 5.3
- When using EWARM, refer to section 5.2 and 5.3

*** Replace the project name in the figure with the project name of this sample project.**

5.1 Setup sample project for e² studio

5.1.1 Startup e² studio

1. Open the e² studio and select a directory as workspace.
2. Click “Open Projects from File System…” in File tab.

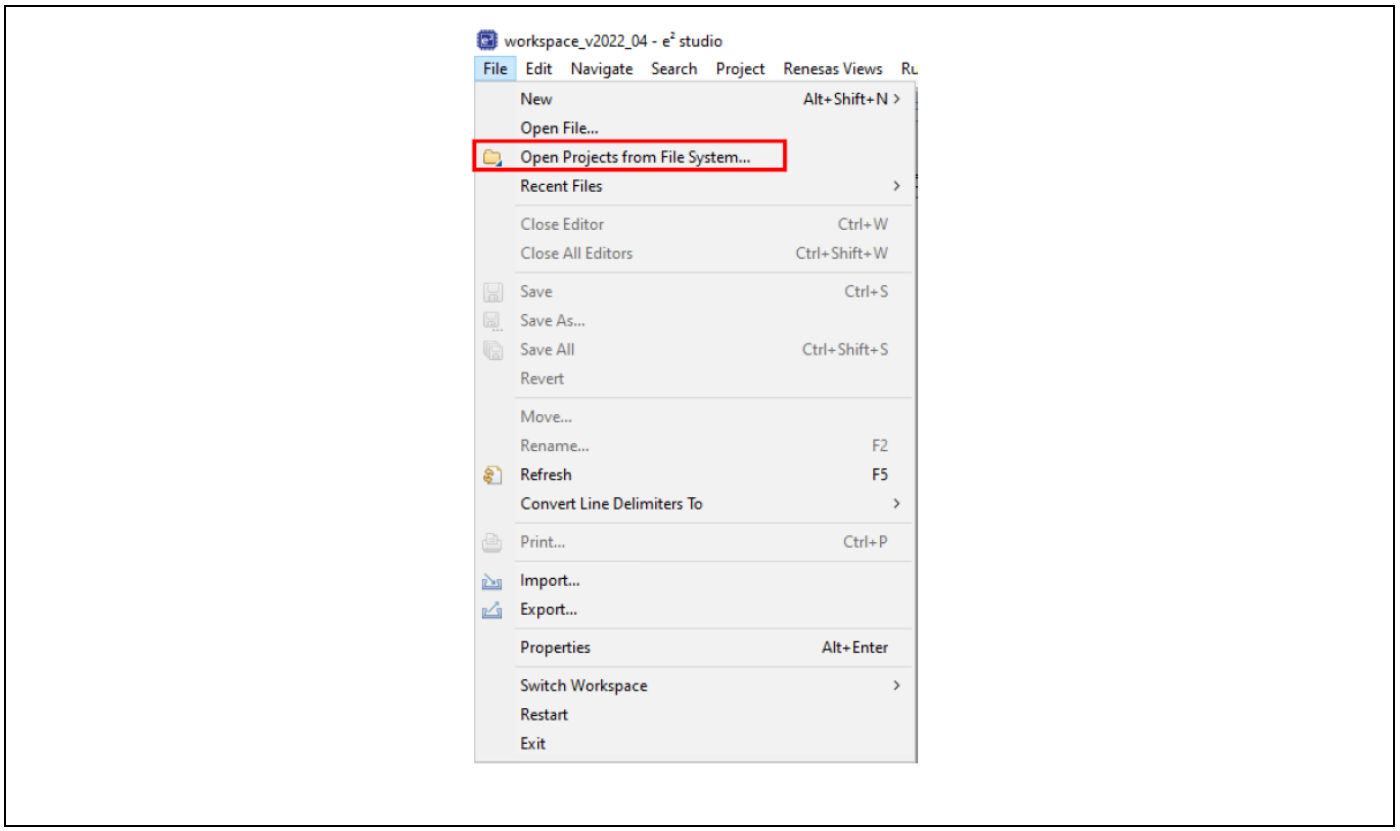

Figure 5.1 e² studio File tab

3. Import project folder.
   “\project\rzn2l_soml\modbus\single\e2studio”.

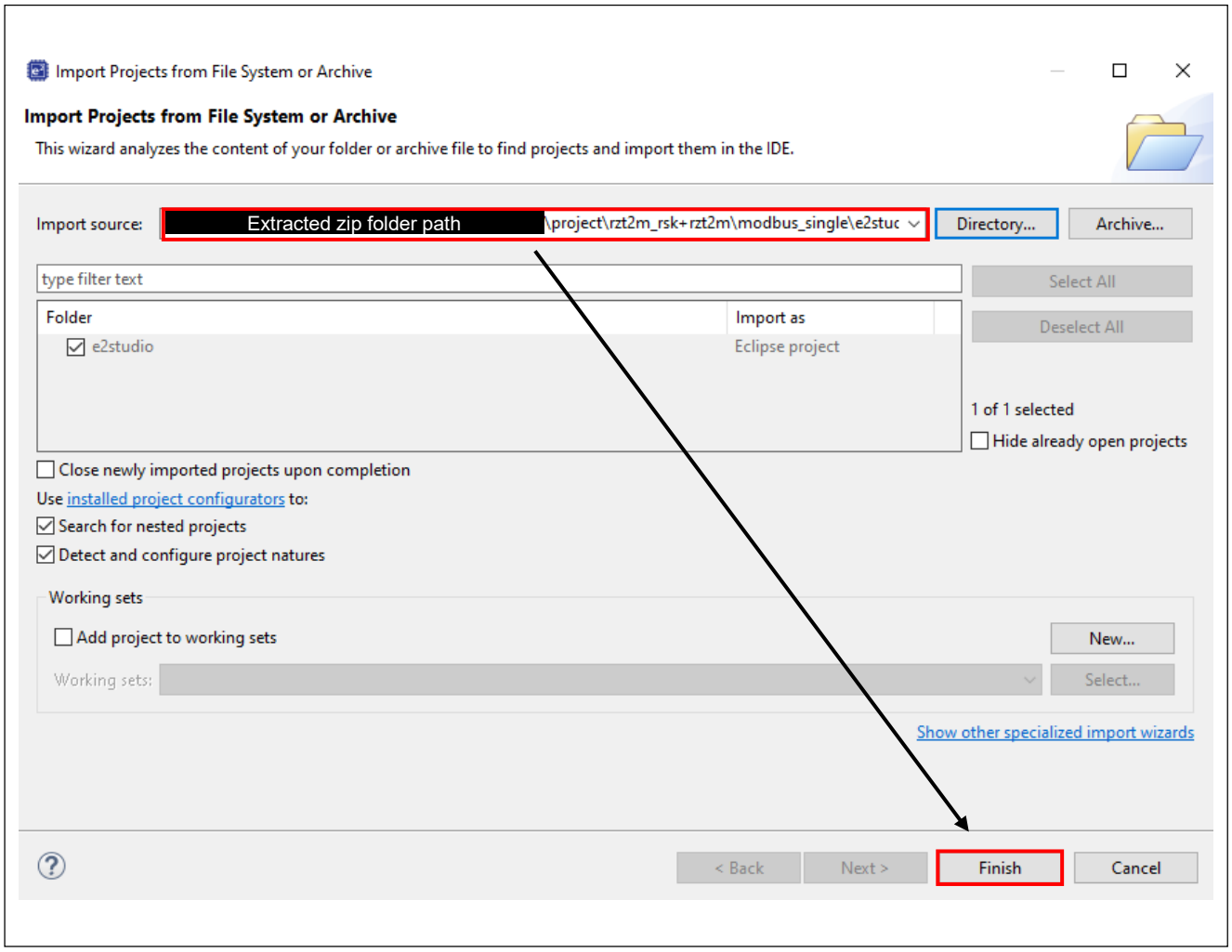

Figure 5.2 Import project on e² studio

5.1.2 Board IP address setting

Set the IP address in the following procedure. This is so that server and client are in the same domain.

1. Set desired server network address in main_thread_entry.c for src folder.

   In this example will be used:

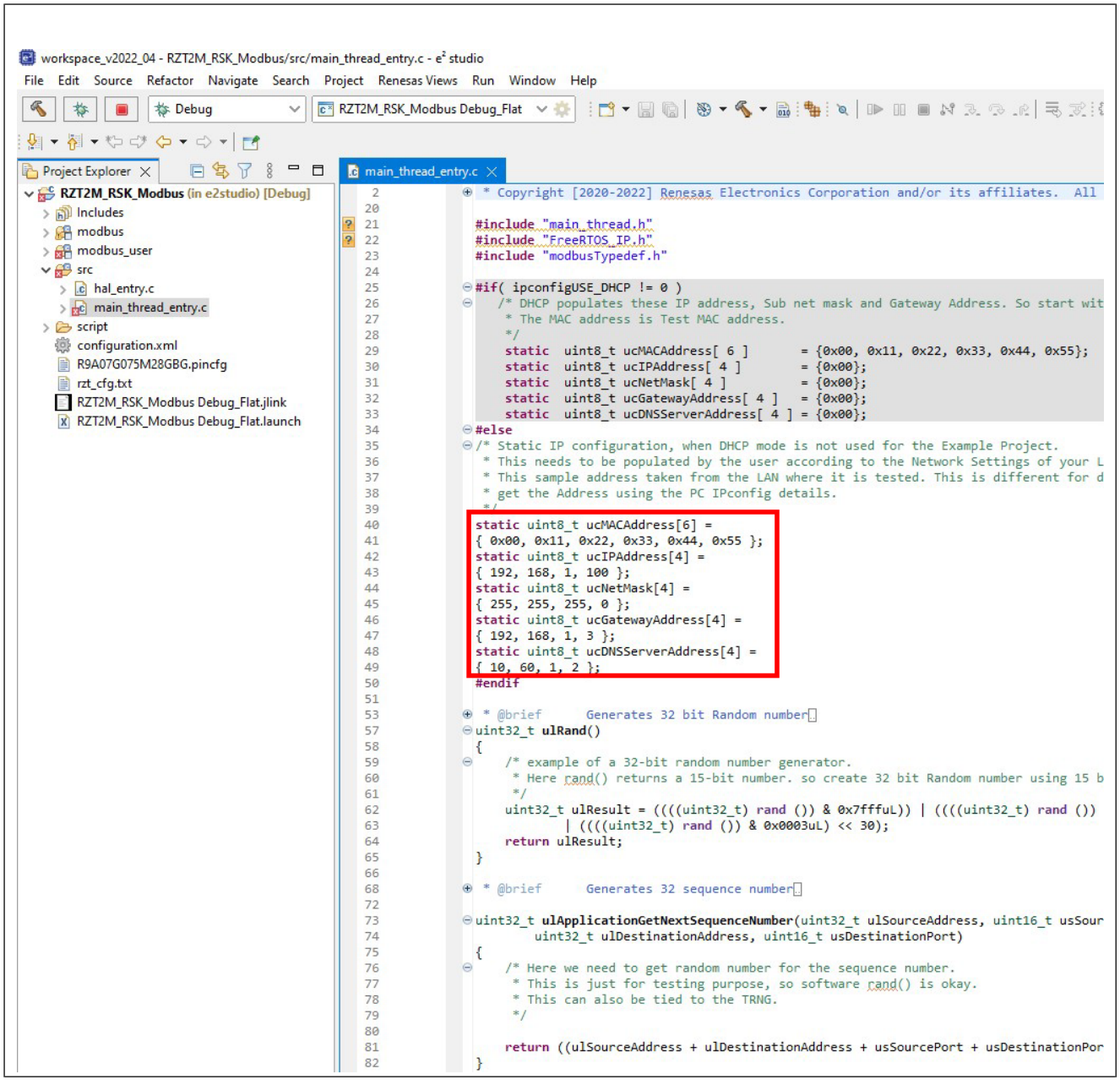

Figure 5.3 Static IP address

2. Set the IP address of the PC used must be in the same domain as the board.

   In this example will be used:

   - IP address 192.168.1.101
   - Subnet mask 255.255.255.0

5.1.3 How to generate source code and how to build

1. Click the Configuration.xml.

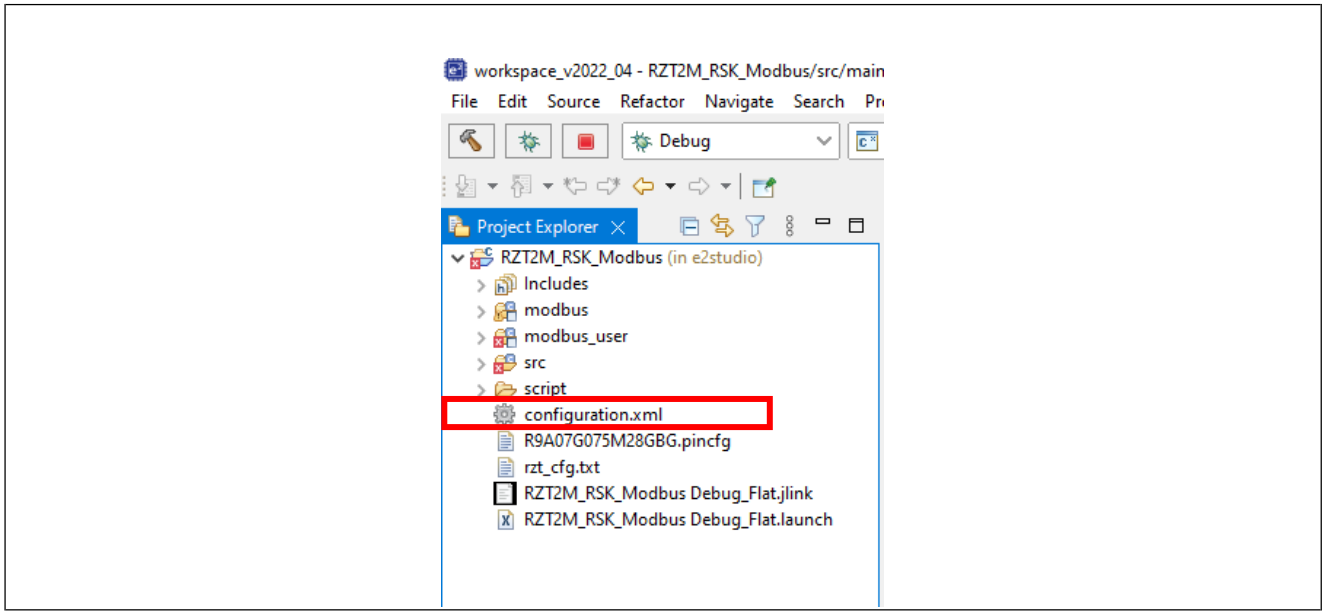

Figure 5.4 Configuration.xml

2. Click ‘Generate Project Content’ button then generate rzt, rzt_gen, rzt_cfg folder.

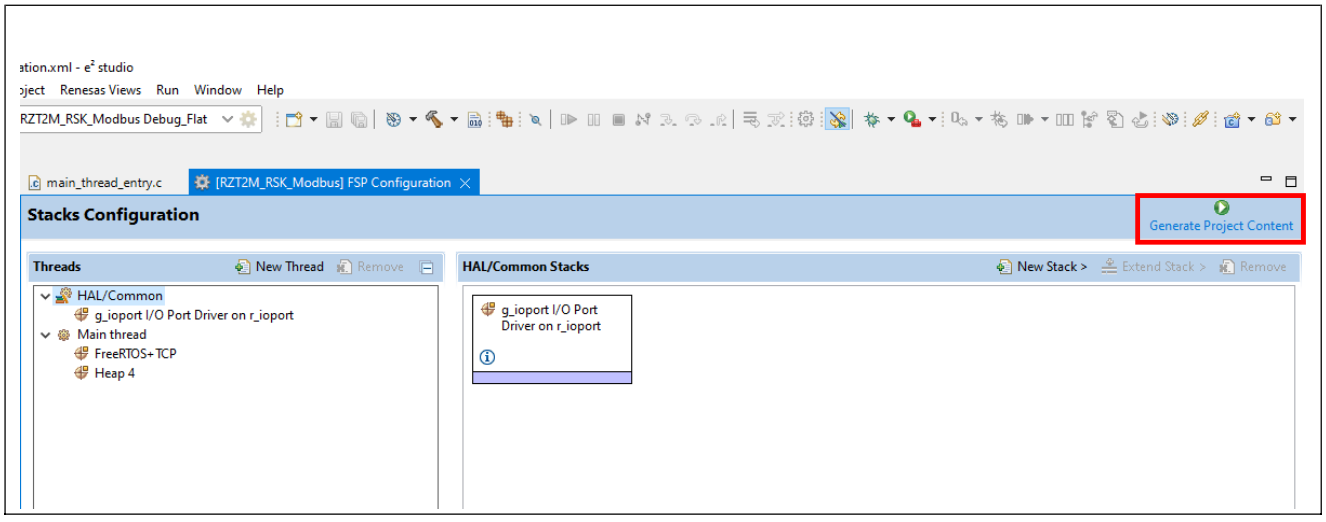

Figure 5.5 Generate Project Content

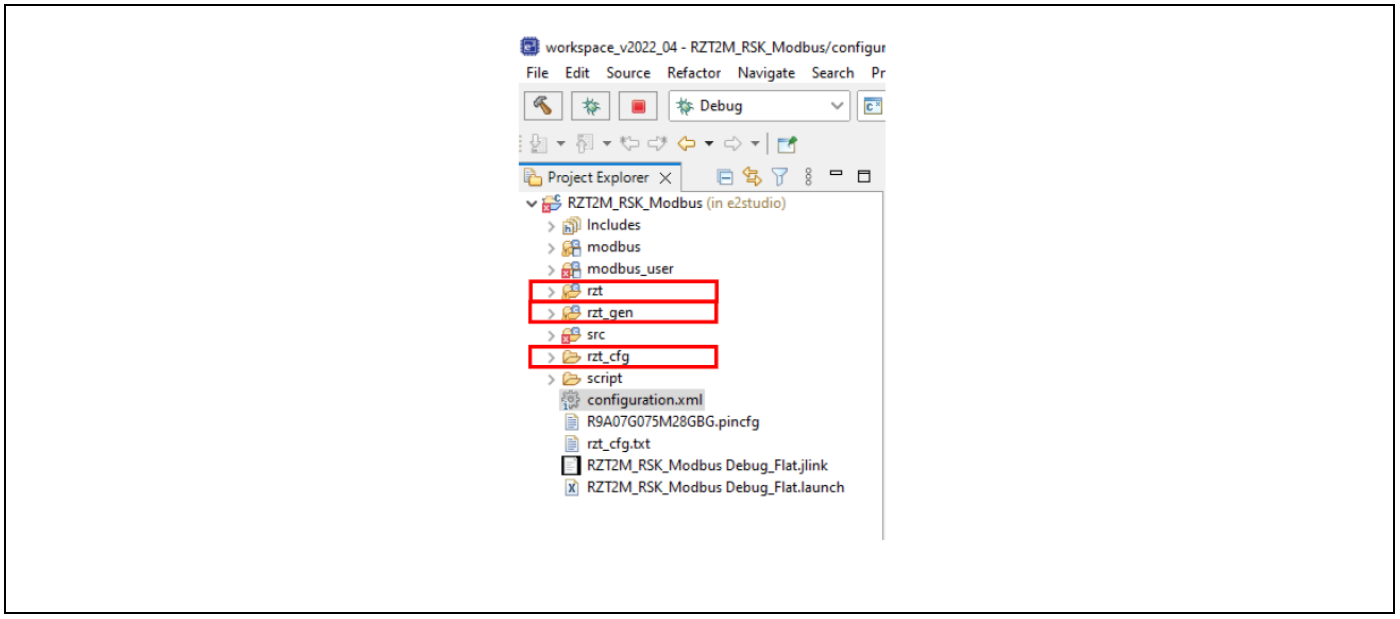

Figure 5.6 Generate project folder

3. Click the Build button in tool bar to build the project and confirm that there is no error message in build message log.

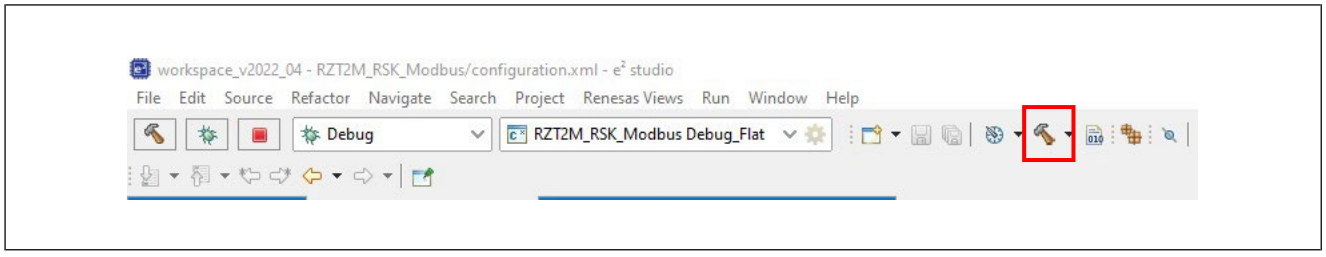

Figure 5.7 Build button

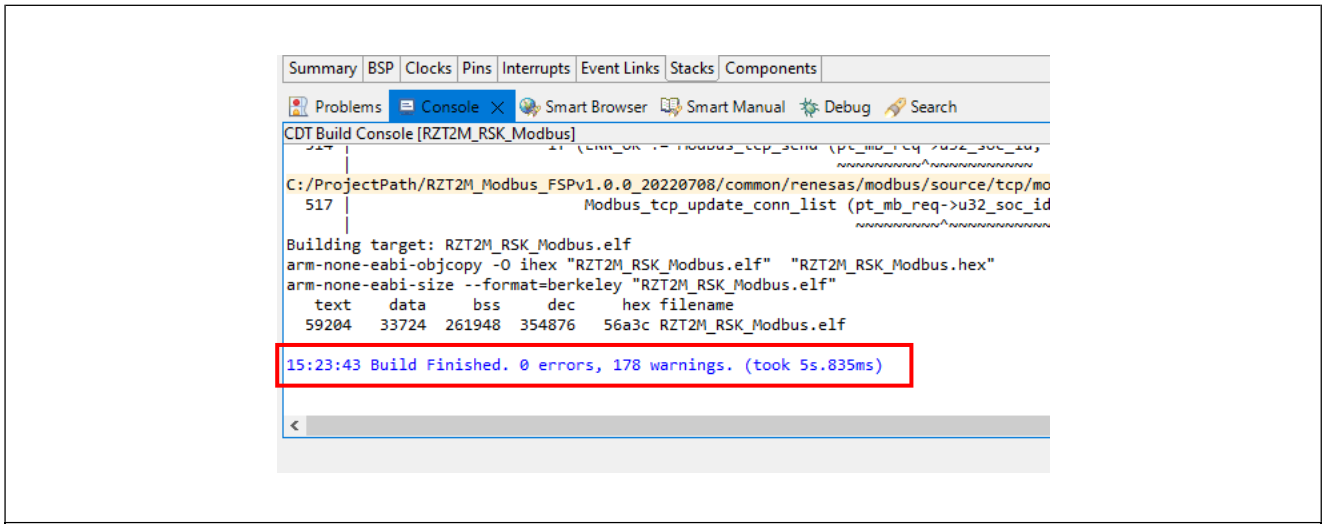

Figure 5.8 Build message

5.1.4 Download application and run debugger

1. Click the Debug button in tool bar to download the built application program and launch the debugger.

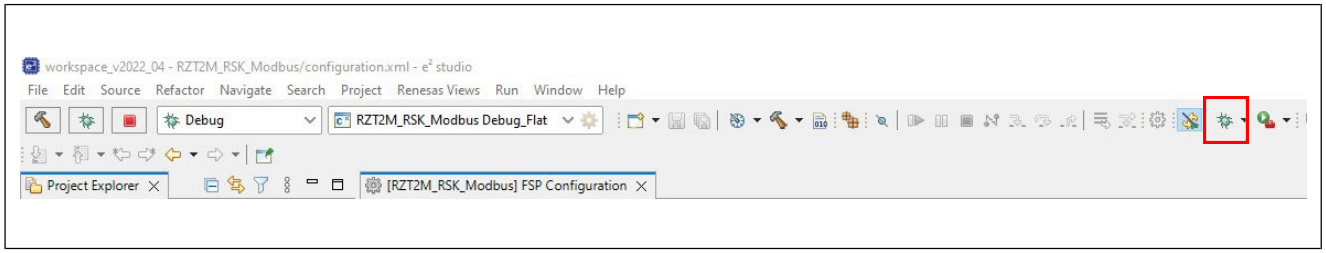

Figure 5.9 Debug button

2. Click to “Switch” button.

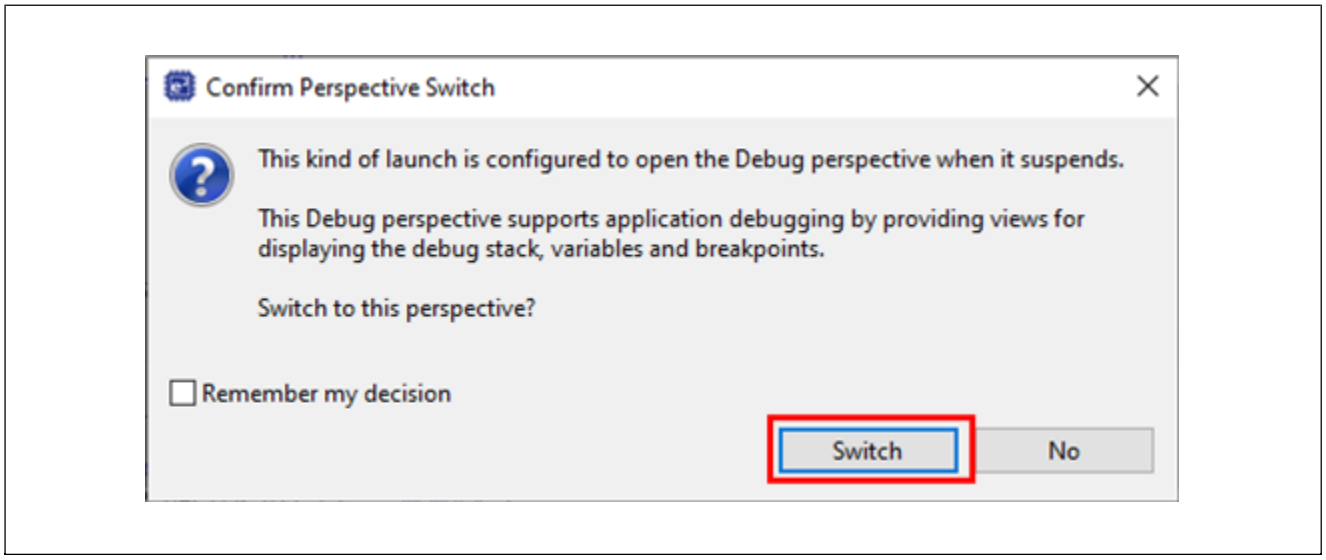

Figure 5.10 Confirm Perspective Switch

3. The program will break at “system_init” for startup.

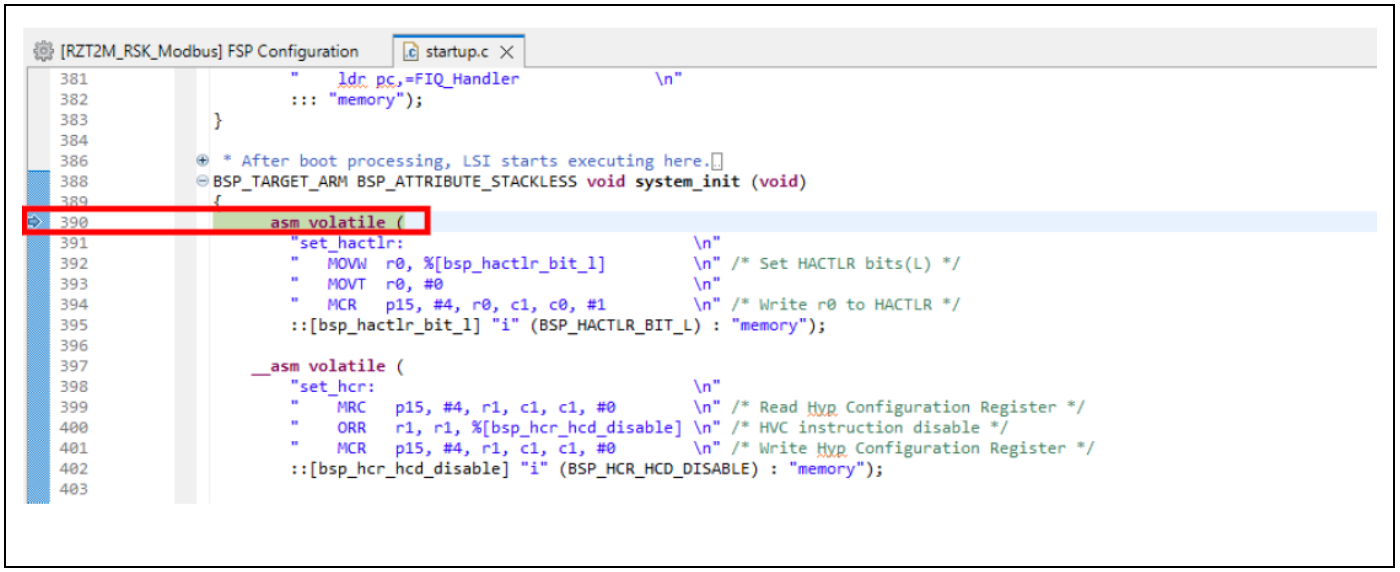

Figure 5.11 Break point 1

4. Before running the loaded program, please change the CPSR register of CR52 general registeron Registers tabs.

- Change the “T” register bit (bit 5 in CPSR register), which is Thumb execution state bit, from “1” to “0” to switch the instruction mode from “Thumb” to “Arm”.
  - For example, when the register value is “0x000001**f**a”, set it to “0x000001**d**a”.

Please note that the program halts at Default_Handler() when running if the value of “T” bit in CPSR register is not changed

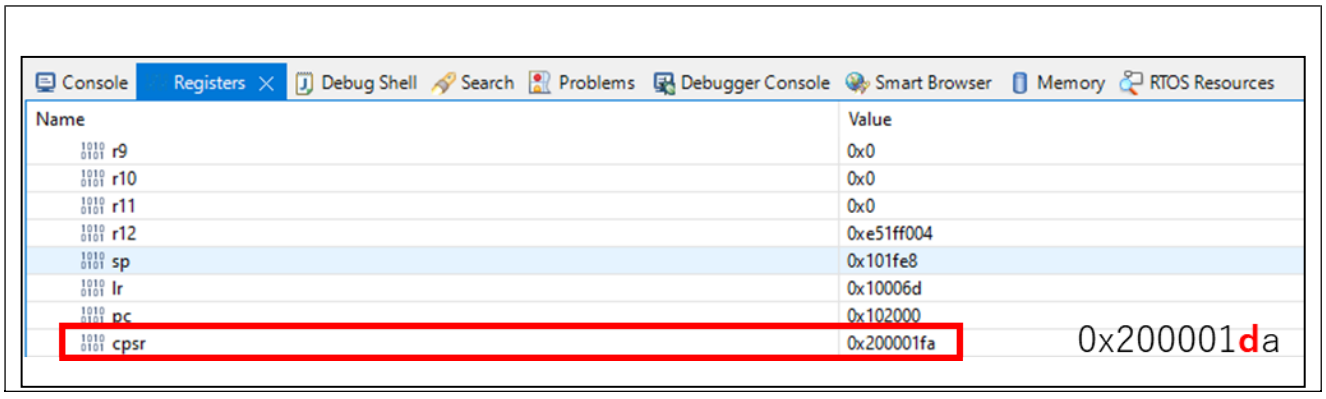

Figure 5.12 CPSR register of CR52 generic register on Registers tab

5. Click the Resume button. The program will break at the first of main function.

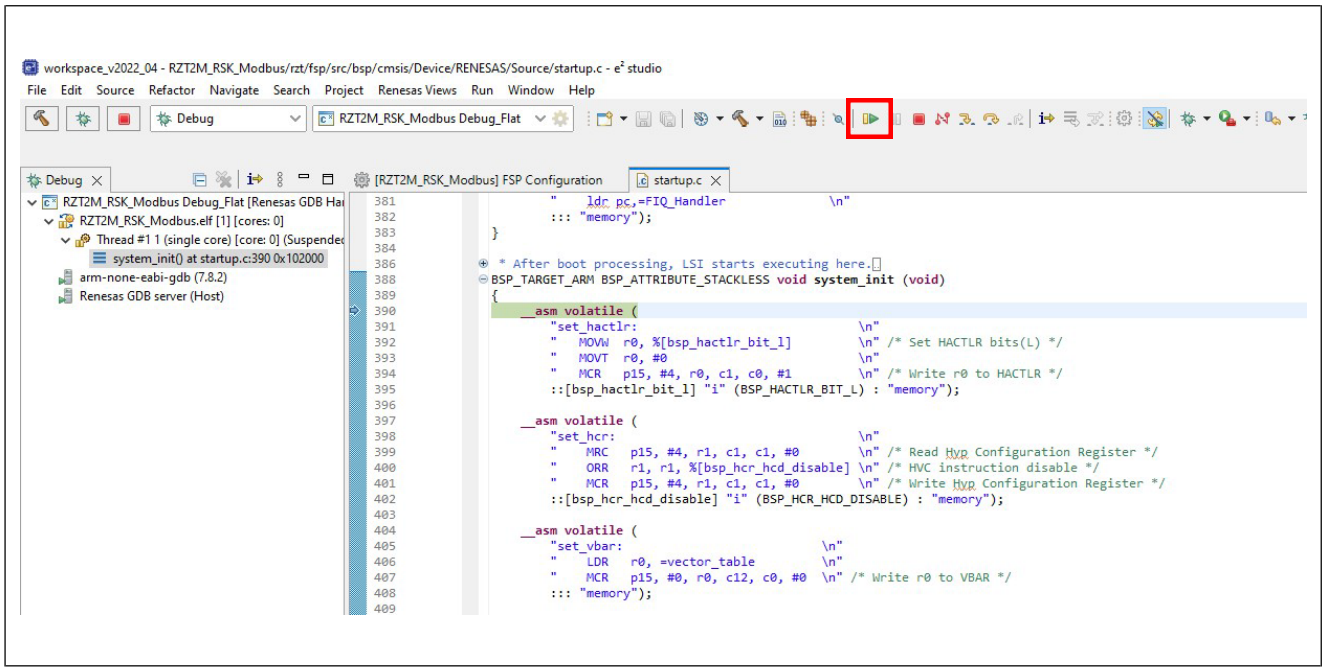

Figure 5.13 Resume button

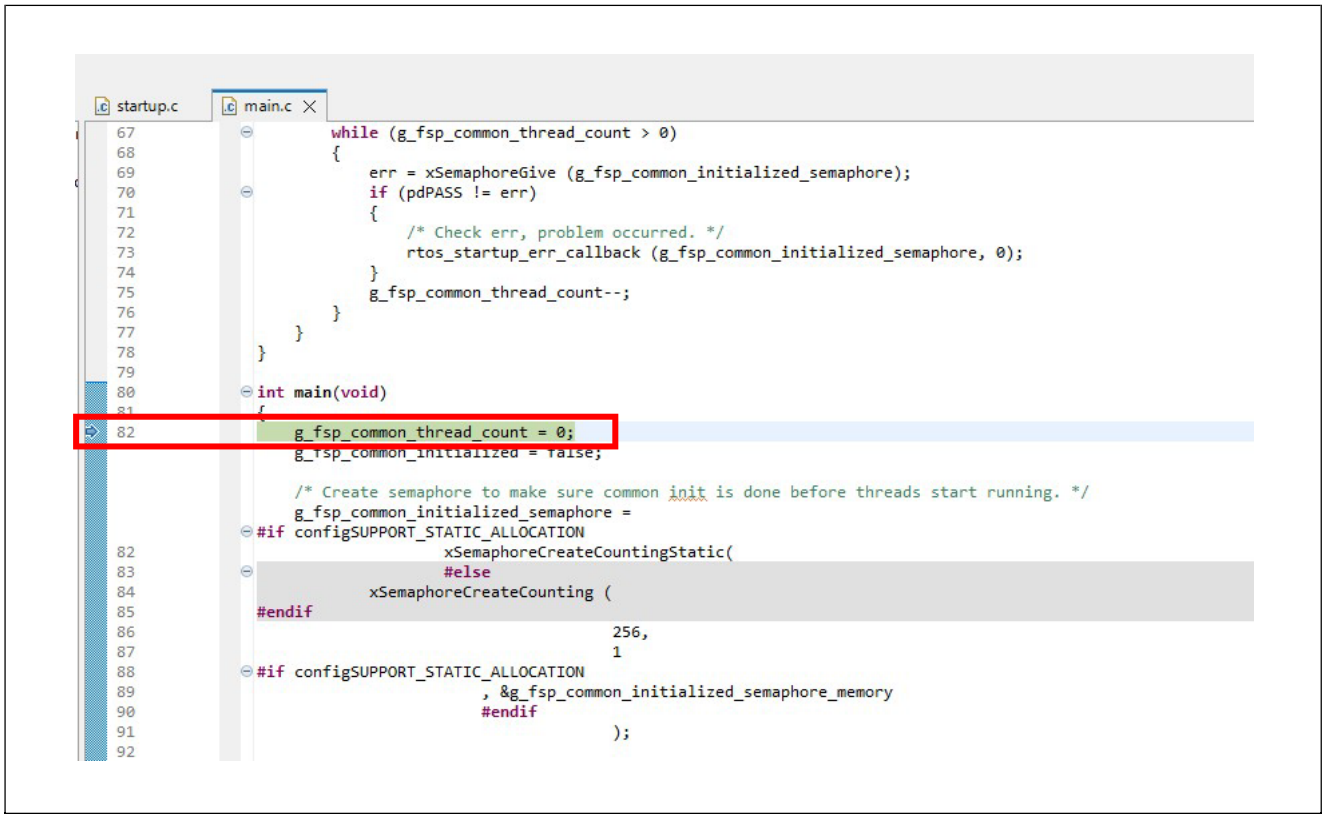

Figure 5.14 Break point2

6. Click the **Resume** button again to execute the program. If the program is working properly, it will be waiting for the TCP/IP connection request.

5.2 Setup sample project for EWARM

5.2.1 Startup EWARM

1. Open the EWARM.
2. Click “Open Workspace…” in File tab.

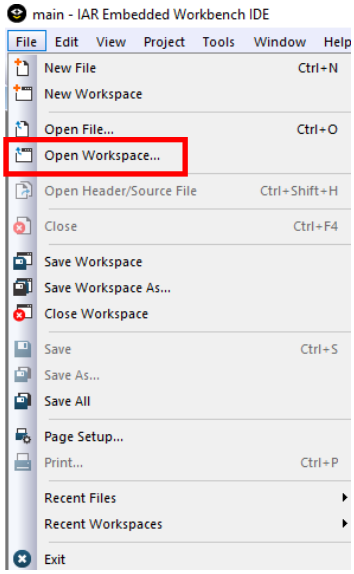

Figure 5.15 EWARM file tab

3. Select the Workspace File(.eww) and click the “Open” button.
   “\project\rzn2l_som\modbus_single\ewarm\RZN2L_SOM_Modbus.eww”.

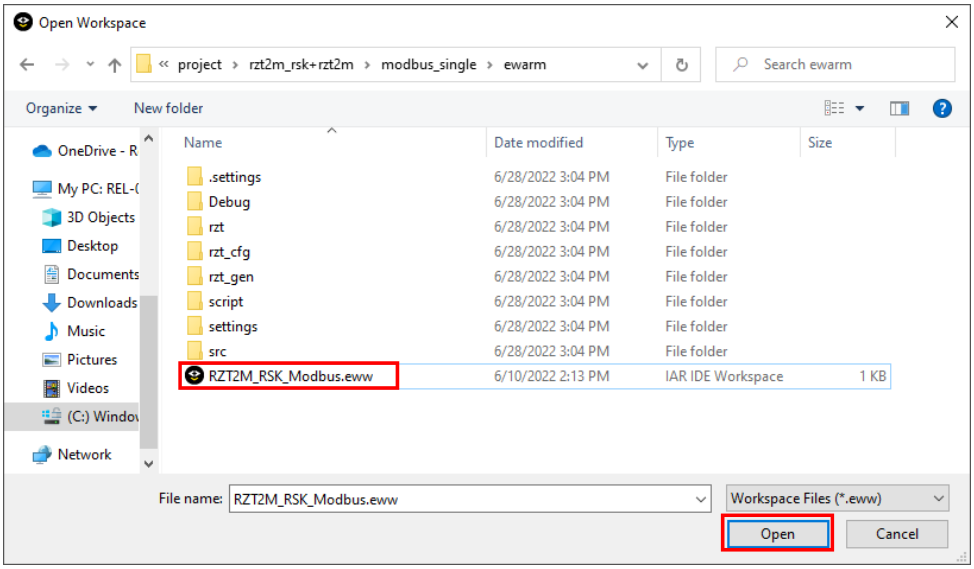

Figure 5.16 Open project file

5.2.2 Board IP address setting

Set the IP address on the following procedure. The server and client must be in the same domain.

1. Set desired server network address in main_thread_entry.c.

In this example will be used:

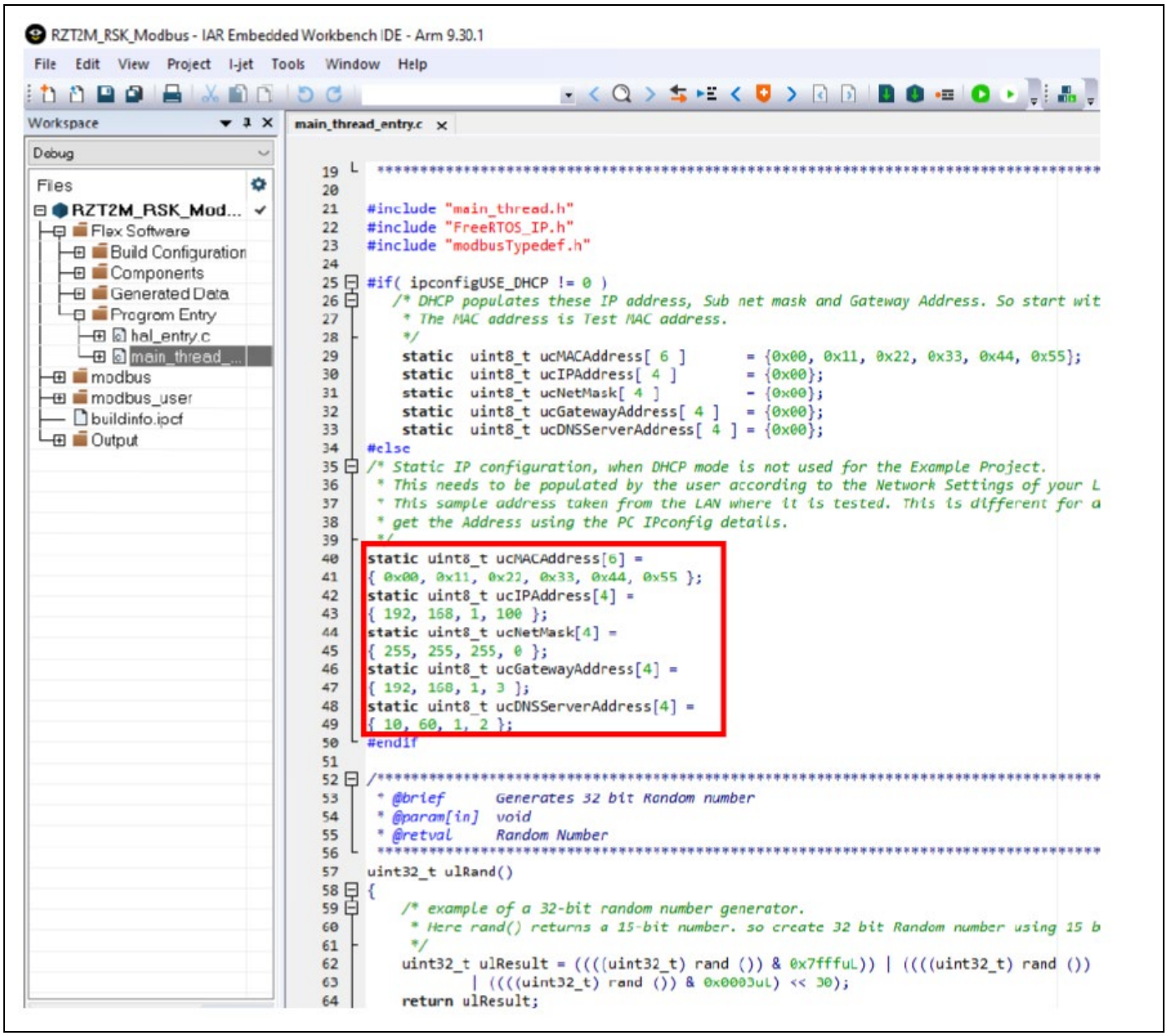

Figure 5.17 Static IP address

2. Set the IP address of the PC used must be in the same domain as the board.

In this example will be used:

- IP address 192.168.1.101
- Subnet mask 255.255.255.0

5.2.3 How to generate source code and how to build

1. Click the “Tool -> FSP Smart Configurator” on tool bar. If you have not set up FSP Smart Configurator yet on EWARM, refer to r01an6434ejxxxx-rzt2m-fsp-getting-started.pdf in which section 5.4 describes how to set up it.

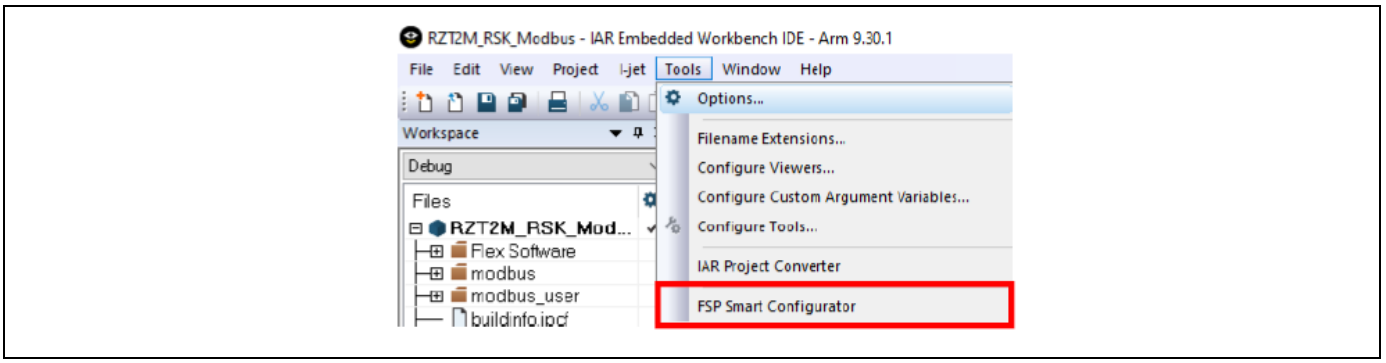

Figure 5.18 Tools tab

2. Click ‘Generate Project Content’ button then will be generate rzt, rzt_gen, rzt_cfg folder.

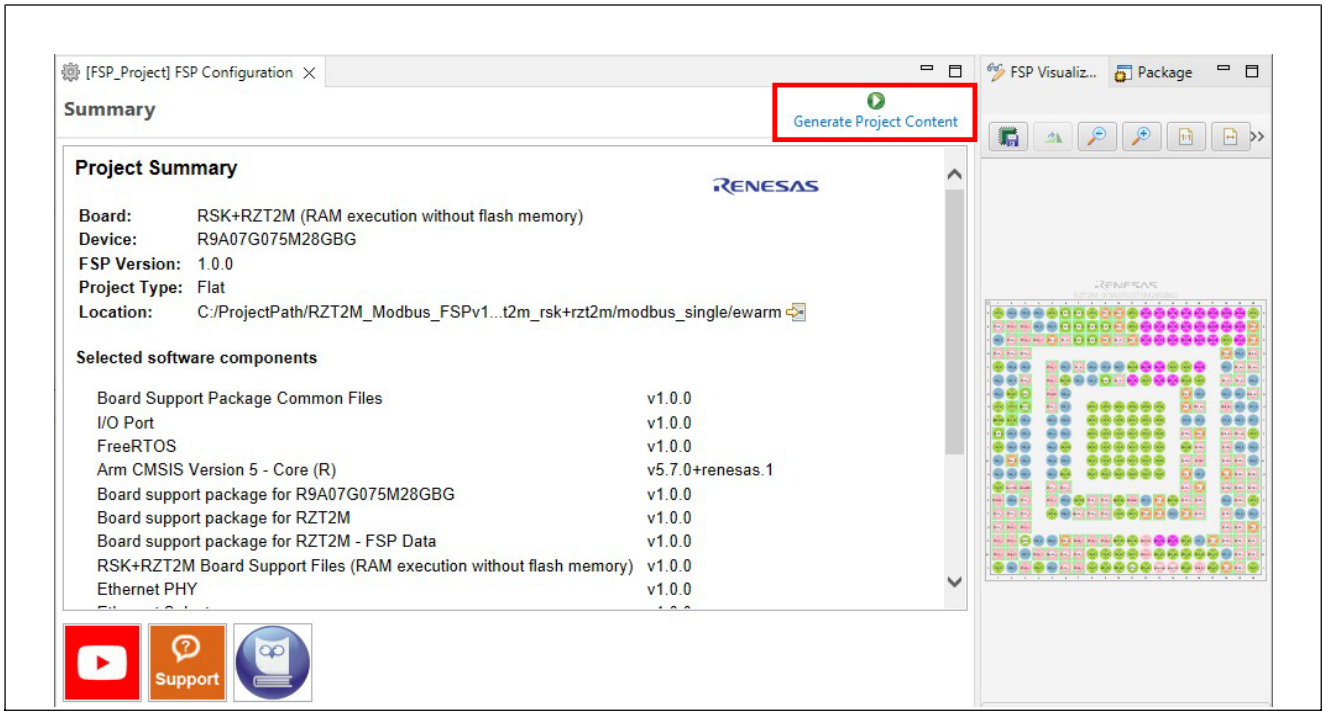

Figure 5.19 FSP SC Smart Configurator

3. Click on Project -> Make from menu bar or Make button on tool bar to build. Once the build is completed, the build message is displayed in the Build Console window that displays compilation target files and the number of error/warnings.

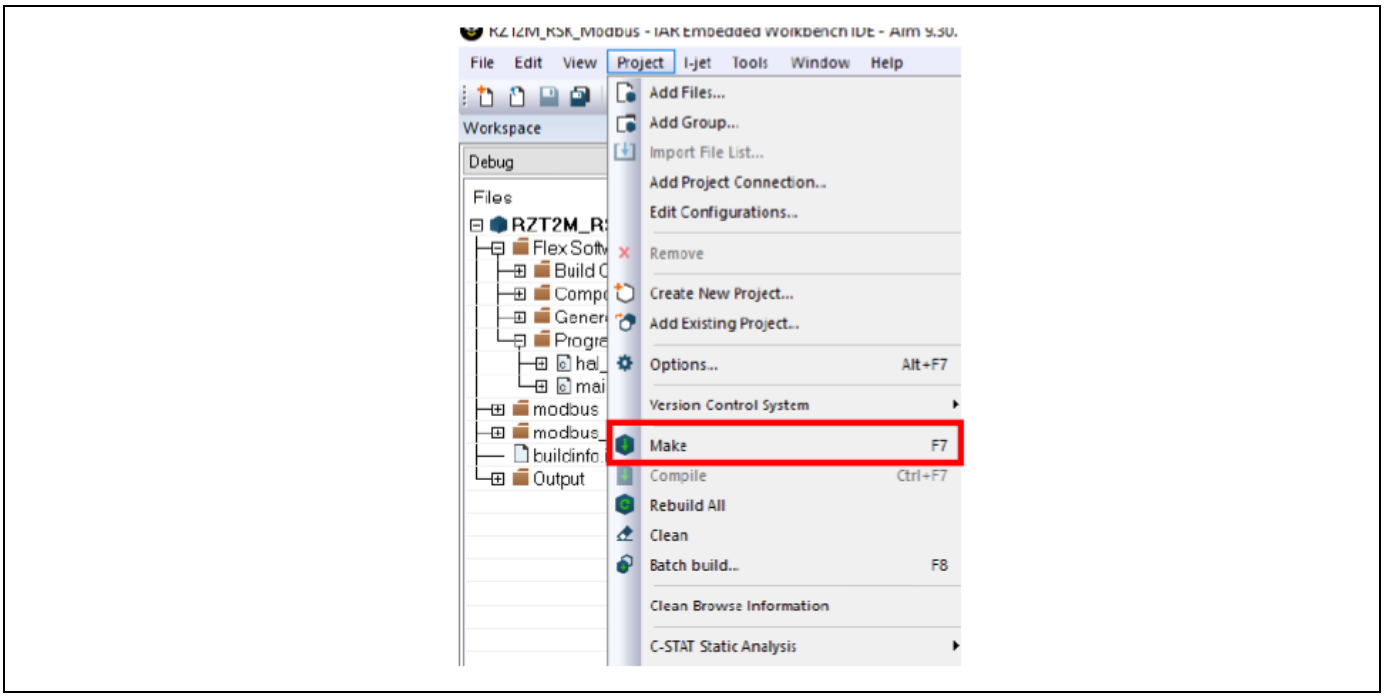

Figure 5.20 Make button1

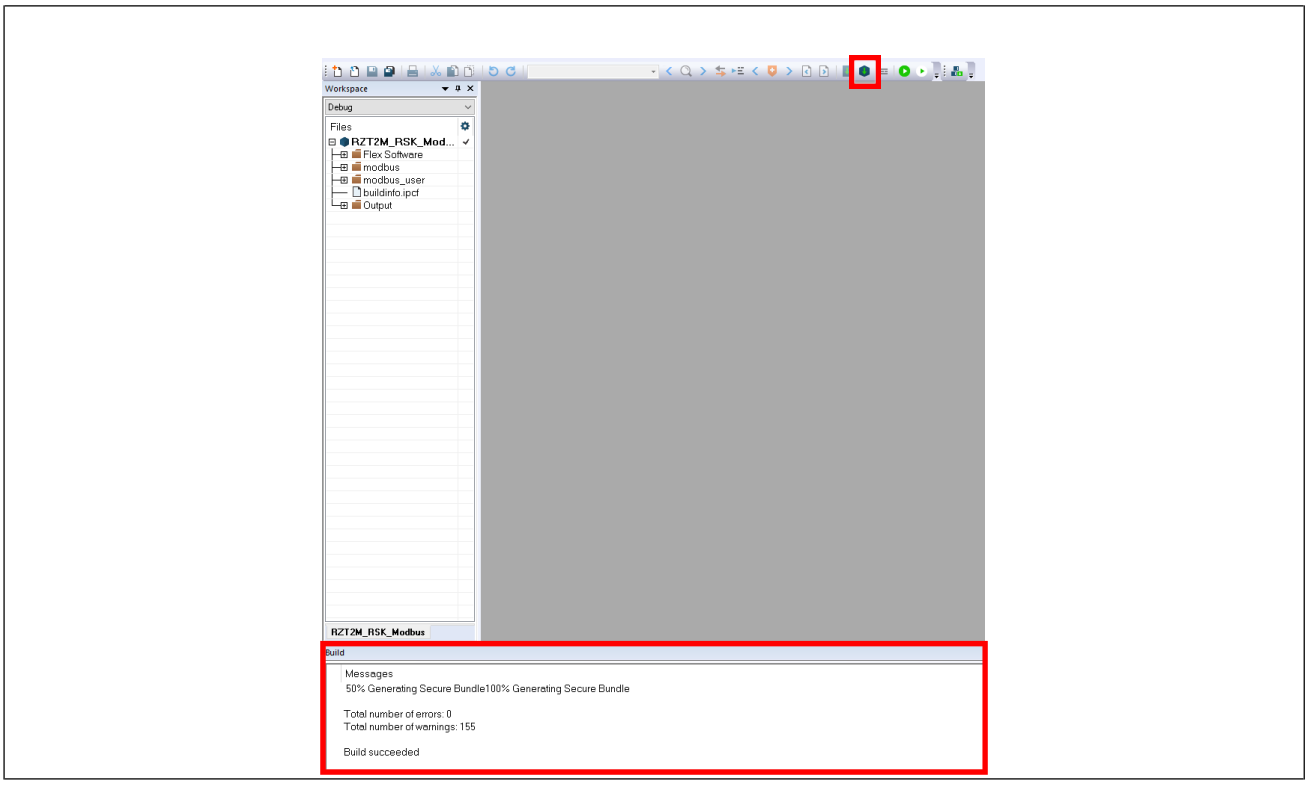

Figure 5.21 Make button2 and Build console

5.2.4 Download application and run debugger

1. Click the Debug button in tool bar to download the built application program and launch the debugger. The program will break at the first code in main function.

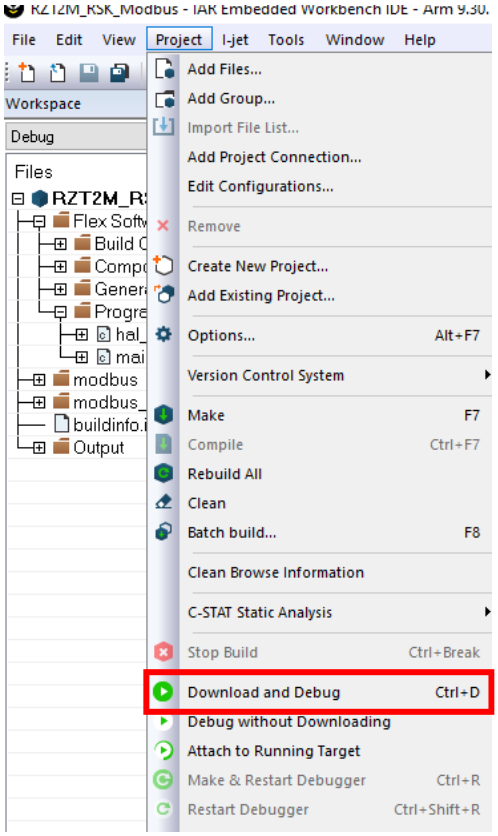

Figure 5.22 Download and Debug button

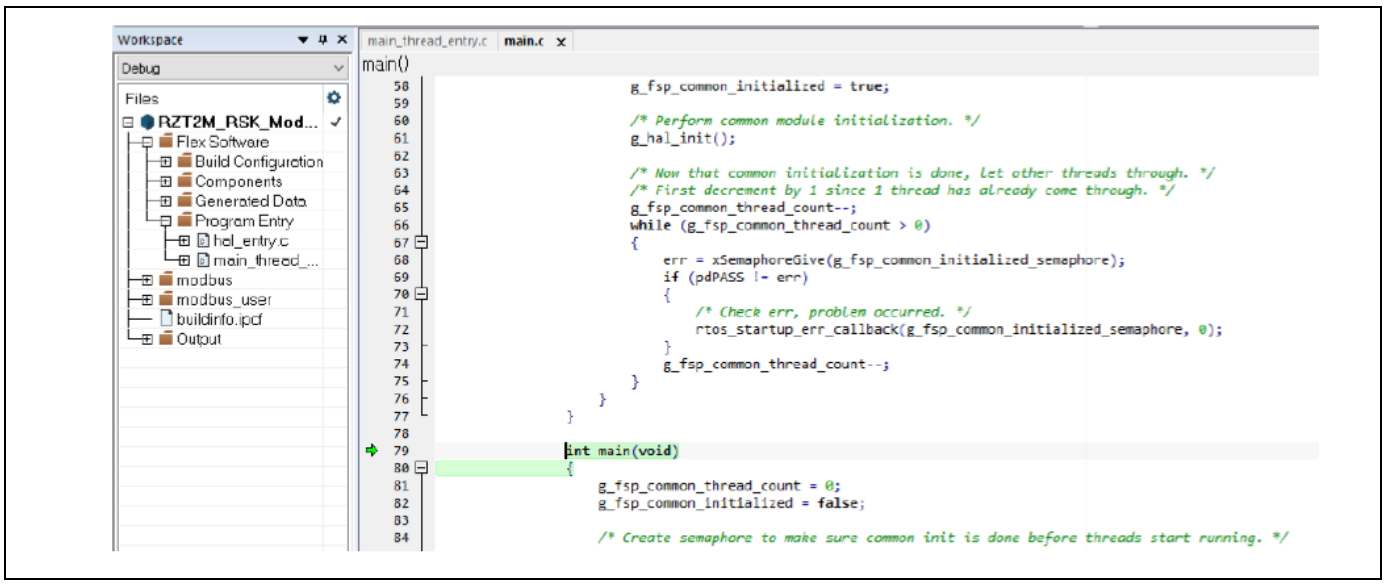

Figure 5.23 Break point

2. Click the Go button. If the program is working properly, it will be waiting for the TCP/IP connection request.

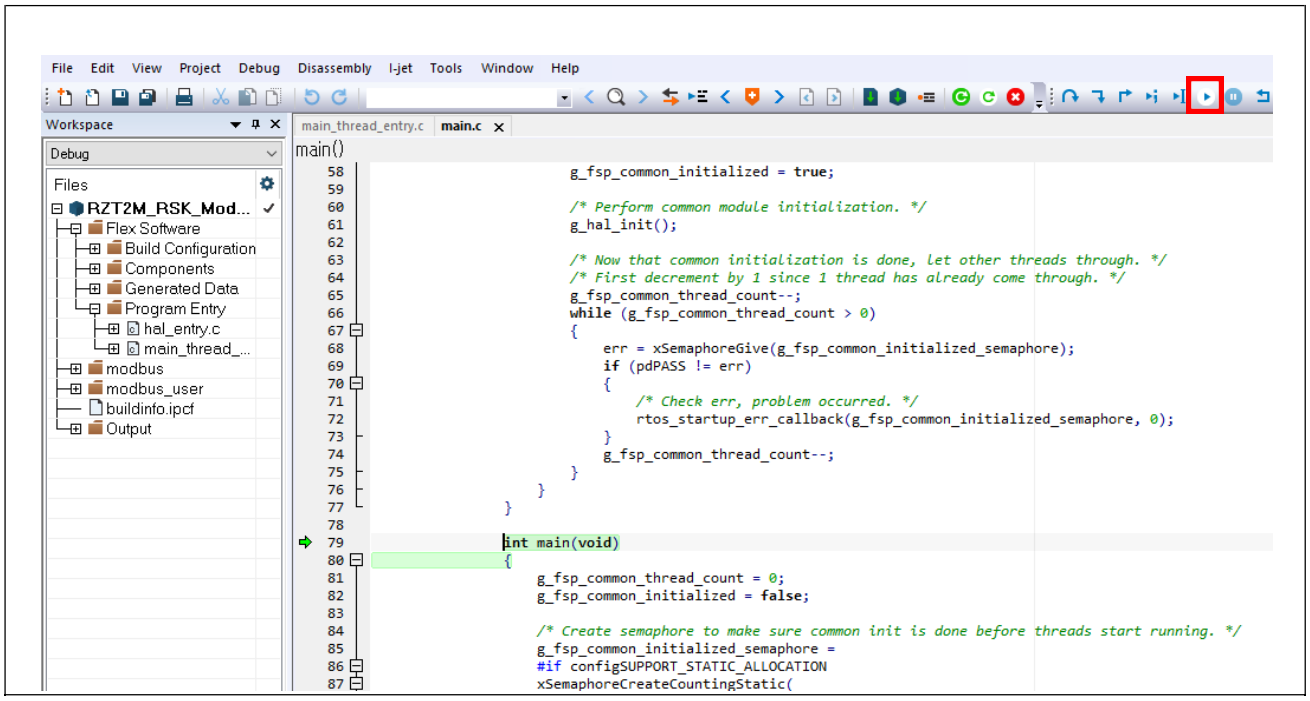

Figure 5.24 Go button

5.3 Demonstration

Users can see the simple demonstration with using this Modbus protocol stack in this sample project.

5.3.1 Specification of demonstration

By communicating with PC through the Modbus TCP protocol, LED blinking speed is controlled dynamically.

For this control, "Read_Discrete_Inputs" and "Write_Single_Coil" function codes are used. Specifically, the following sequence is executed.

1. PC application checks the state of the switch (J6), by using Modbus "Read_Discrete_Inputs" function code. The [SW setting value] is the 8-bits of data calculated by the state of J6.

   8-bits SW setting value

Bit number	7	6	5	4	3	2	1	0
Value	J6-4	J6-3	J6-2	J6-1	0	0	0	0

2. According to the states of the switch, the states of the output ports, which are connected to LED, is updated periodically.

   - When [SW setting value] is less than 0x7F

   Update span = ([SW setting value] +1) * 10 [msec]

   - When [SW setting value] is equal to or greater than 0x7F

   Update span = 10 [msec]

   ex. J6-1, J6-3 = ON (1) J6-2, J6-4 = OFF (0)

   SW setting value = 0101 0000b = 0x50 = 80

   Update span = (80 +1) + 10 = 810 [msec]

5.3.2 Connect TCP communication

1. Refer to Chapter 4 Setup a master tool for Maser tool setup.
2. Click the **Connect** button to start TCP communication and the LED will start blinking.

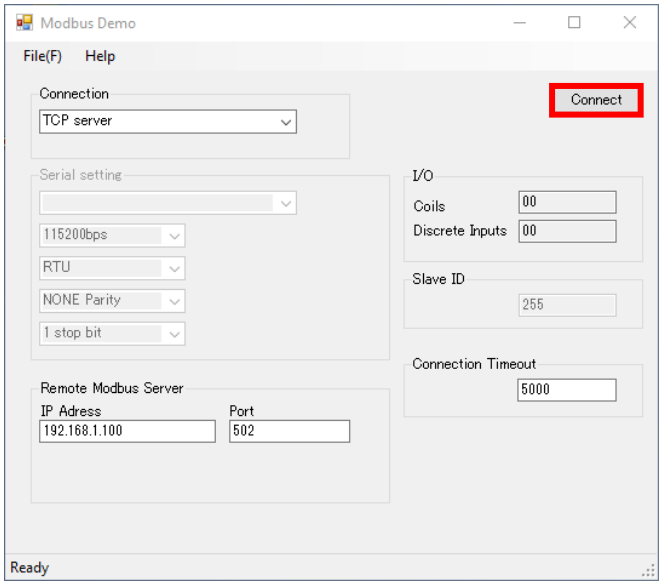

Figure 5.25 Connect button

3. Check the coils status and SW setting value.

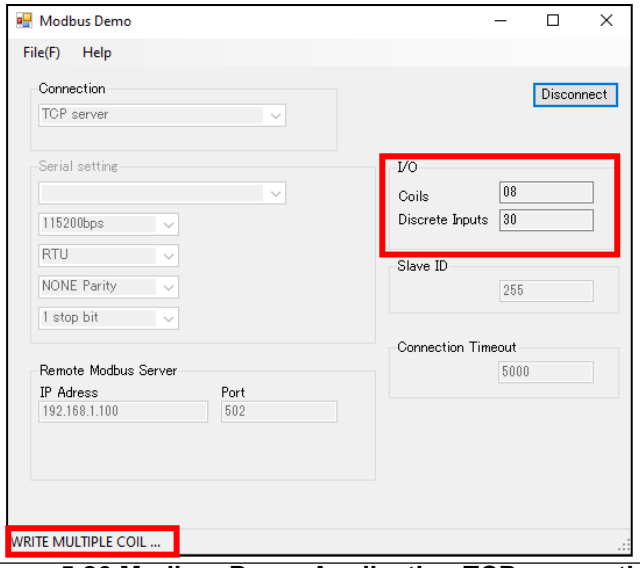

Figure 5.26 Modbus Demo Application TCP connection

Appendix A. DHCP mode

1. Open configuration.xml
2. Click “Stacks” tab to open the Stacks Configuration pane and select the “FreeRTOS + TCP” in the left threads pane.
3. Open the properties, change “Use DHCP” to “Enable” and click “Generate Project Content” button.

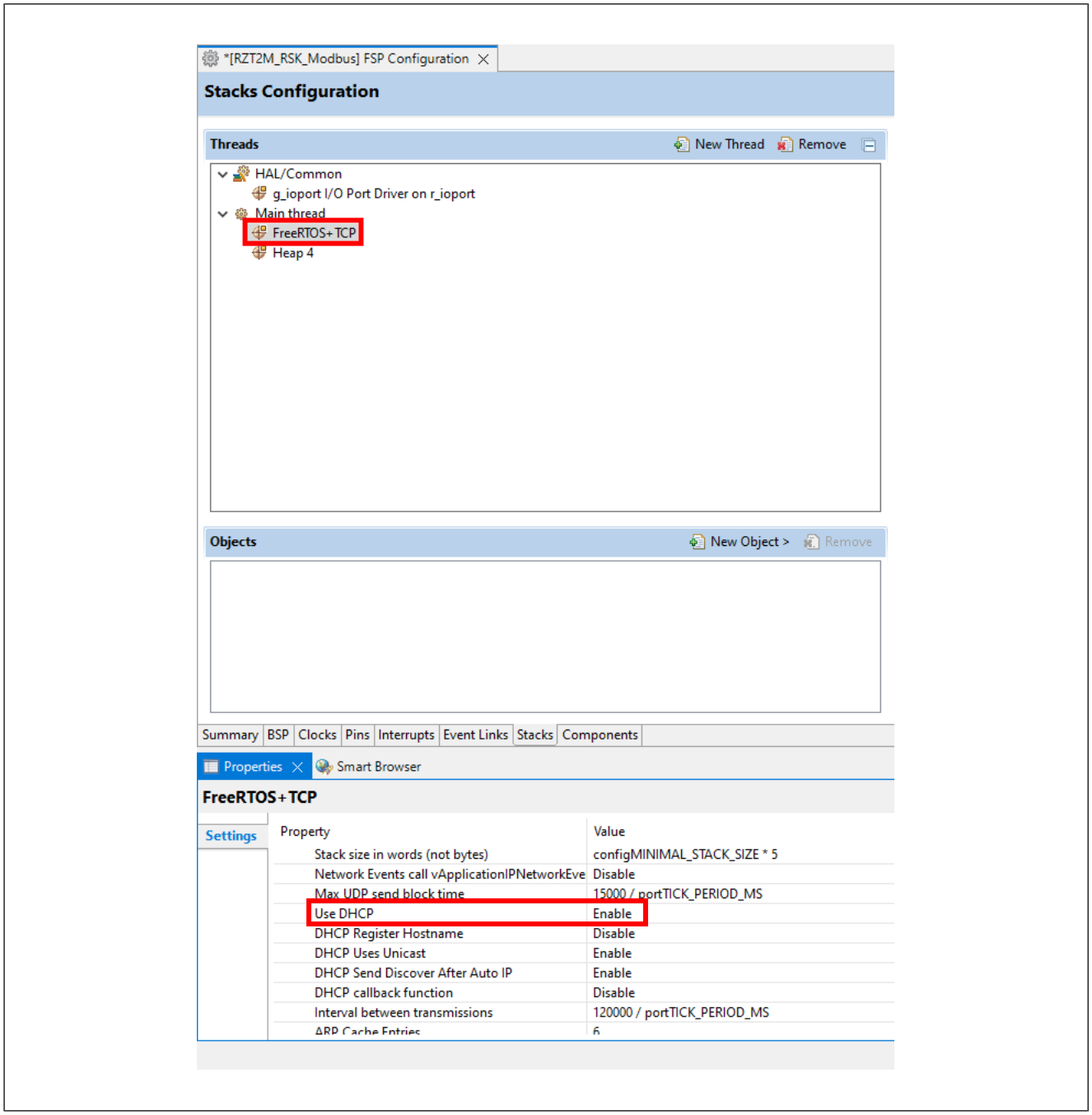

Figure A.1 Use DHCP mode

4. Build and debug.
5. If you want to check the IP address etc. in DHCP mode, use the arp command on command line.
6. Check the physical Address “00-11-22-33-44-55”.

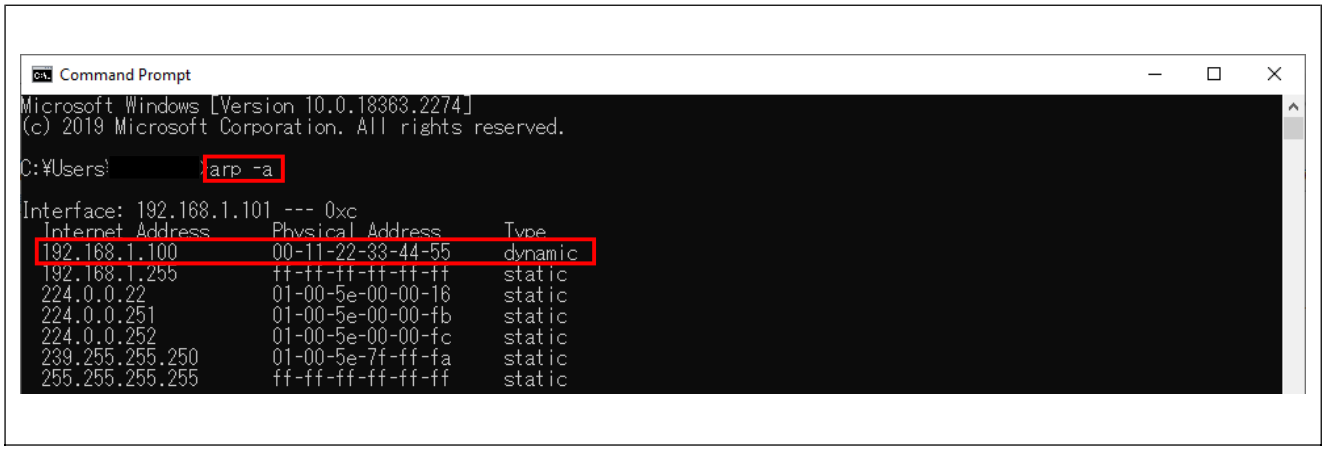

Figure A.2 Check the IP address

Appendix B. Application Programming Interface

Function

· Modbus_tcp_init_ip_table

[Description]

Modbus set host IP list properties

[Format]

```
void
Modbus_tcp_init_ip_table(
                    ENABLE_FLAG e_flag,
                    TABLE_MODE e_mode
);
```

[Parameter]

ENABLE_FLAG	e_flag	Status is whether the connection table enabled or disabled
TABLE_MODE	e_mode	Status indicating the list contain IP to be accepted or rejected

[Return value]

None

[Error code]

None

· Modbus_tcp_add_ip_addr

[Description]

Modbus add an IP address to host IP list

[Format]

```
uint32_t
Modbus_tcp_add_ip_addr(
                    pchar_t pu8_add_ip
)
```

[Parameter]

pchar_t	pu8_add_ip	Host IP address in numbers and dots notation. ex. 192.168.1.100

[Return value]

Error code

[Error code]

ERR_OK	On success
ERR_IP_ALREADY_PRESENT	If address already present in list
ERR_MAX_CLIENT	If maximum connections reached.
ERR_TABLE_DISABLED	If IPlist is disabled

· **Modbus_tcp_delete_ip_addr**

[Description]

Modbus delete an IP address to host IP list

[Format]

uint32_t

Modbus_tcp_delete_ip_addr(

pchar_t pu8_del_ip

)

[Parameter]

pchar_t	pu8_del_ip	Host IP address in numbers and dots notation

ex. 192.168.1.100

[Return value]

Error code

[Error code]

ERR_OK	On success
ERR_IP_NOT_FOUND	IPlist is not found
ERR_TABLE_EMPTY	IF the list is empty
ERR_TABLE_DISABLED	If the IPlist is disabled

· **Modbus_slave_map_init**

[Description]

Modbus function code mapping API

[Format]

uint32_t

Modbus_slave_map_init(

p_slave_map_init_t pt_slave_func_tbl

)

[Parameter]

p_slavemap_init	pt_slave_func_tbl	Structure pointer to function code mapping table

[Return value]

Error code

[Error code]

ERR_OK	On success
ERR_INVALID_STACK_INIT_PARAMS	If parameter is null
ERR_MEM_ALLOC	If memory allocation failed

· **Modbus_tcp_server_init_stack**

[Description]

Modbus TCP stack initialization API

[Format]

uint32_t

Modbus_tcp_server_init_stack(

uint32_t u32_additional_port,

uint8_t u8_tcp_multiple_client

)

[Parameter]

Uint32_t	u32_additonal_port	Additional port configured by user
Uint8_t	u8_tcp_multiple_client	Status whether multiple clients is enabled

[Return value]

Error code

[Error code]

ERR_OK	On successful initialization of the task or mailbox
ERR_STACK_INIT	If initialization of the task or mailbox failed

Structure

· **slave_map_init_t**

Member variables type	Member variables	Description
fp_function_code1_t	fp_function_code1	Callback function pointer for Modbus function code 1 (Read coils) operation.
fp_function_code2_t	fp_function_code2	Callback function pointer for Modbus function code 2 (Read Discrete Inputs) operation.
fp_function_code3_t	fp_function_code3	Callback function pointer for Modbus function code 3 (Read Holding Registers) operation.
fp_function_code4_t	fp_function_code4	Callback function pointer for Modbus function code 4 (Read Input RegisterRead coils) operation.
fp_function_code5_t	fp_function_code5	Callback function pointer for Modbus function code 5 (Write Single Coil) operation.
fp_function_code6_t	fp_function_code6	Callback function pointer for Modbus function code 6 (Write Single Register) operation.
fp_function_code15_t	fp_function_code15	Callback function pointer for Modbus function code 15 (Write Multiple Coils) operation.
fp_function_code16_t	fp_function_code16	Callback function pointer for Modbus function code 16 (Write Multiple Registers) operation.
fp_function_code23_t	fp_function_code23	Callback function pointer for Modbus function code 23 (Read/Write Multiple Registers) operation.

· **p_req_read_coils_t**

Member variables type	Member variables	Description
uint16_t	u16_transaction_id	Specifies the transaction ID
uint16_t	u16_protocol_id	Specifies the protocol ID
uint8_t	u8_slave_id	Identification of a remote slave connected
uint16_t	u16_start_addr	Specifies address of the first coil
uint16_t	u16_num_of_coils	Specifies the number of coils to be read

· **p_req_read_inputs_t**

Member variables type	Member variables	Description
uint16_t	u16_transaction_id	Specifies the transaction ID
uint16_t	u16_protocol_id	Specifies the protocol ID
uint8_t	u8_slave_id	Identification of a remote slave connected
uint16_t	u16_start_addr	Specifies address of the first discrete input
uint16_t	u16_num_of_inputs	Specifies the number of discrete inputs to be read

· **p_req_read_holding_reg_t**

Member variables type	Member variables	Description
uint16_t	u16_transaction_id	Specifies the transaction ID
uint16_t	u16_protocol_id	Specifies the protocol ID
uint8_t	u8_slave_id	Identification of a remote slave connected
uint16_t	u16_start_addr	Specifies address of the first holding register
uint16_t	u16_num_of_reg	Specifies the number of registers to be read

· **p_req_read_input_reg_t**

Member variables type	Member variables	Description
uint16_t	u16_transaction_id	Specifies the transaction ID
uint16_t	u16_protocol_id	Specifies the protocol ID
uint8_t	u8_slave_id	Identification of a remote slave connected
uint16_t	u16_start_addr	Specifies address of the first input register
uint16_t	u16_num_of_reg	Specifies the number of registers to be read

· **p_req_write_single_coil_t**

Member variables type	Member variables	Description
uint16_t	u16_transaction_id	Specifies the transaction ID
uint16_t	u16_protocol_id	Specifies the protocol ID
uint8_t	u8_slave_id	Identification of a remote slave connected
uint16_t	u16_output_addr	Specifies address of the coil
uint16_t	u16_output_value	Data to be written

· **p_req_write_single_reg_t**

Member variables type	Member variables	Description
uint16_t	u16_transaction_id	Specifies the transaction ID
uint16_t	u16_protocol_id	Specifies the protocol ID
uint8_t	u8_slave_id	Identification of a remote slave connected
uint16_t	u16_register_addr	Specifies address of the register
uint16_t	u16_register_value	Data to be written

· **p_req_write_multiple_coils_t**

Member variables type	Member variables	Description
uint16_t	u16_transaction_id	Specifies the transaction ID
uint16_t	u16_protocol_id	Specifies the protocol ID
uint8_t	u8_slave_id	Identification of a remote slave connected
uint16_t	u16_start_addr	Specifies address of the first coil
uint16_t	u16_num_of_outputs	Specifies the number of coils to be written
uint8_t	u8_num_of_bytes	Specifies the number of bytes of data
uint8_t	aru8_data[MAX_DISCRETE_DATA]	Data to be written

* MAX_DISCRETE_DATA is defined in 251

· **p_req_write_multiple_reg _t**

Member variables type	Member variables	Description
uint16_t	u16_transaction_id	Specifies the transaction ID
uint16_t	u16_protocol_id	Specifies the protocol ID
uint8_t	u8_slave_id	Identification of a remote slave connected
uint16_t	u16_start_addr	Specifies address of the first register
uint16_t	u16_num_of_reg	Specifies the number of registers to be written
uint8_t	u8_num_of_bytes	Specifies the number of bytes of data
uint16_t	aru16_data[MAX_REG_DATA]	Data to be written

* MAX_REG_DATA is defined in 125

· **p_req_read_write_multiple_reg_t**

Member variables type	Member variables	Description
uint16_t	u16_transaction_id	Specifies the transaction ID
uint16_t	u16_protocol_id	Specifies the protocol ID
uint8_t	u8_slave_id	Identification of a remote slave connected
uint16_t	u16_read_start_addr	Specifies address of the first register to be read from
uint16_t	u16_num_to_read	Specifies the number of registers to be read
uint16_t	u16_write_start_addr	Specifies address of the first register to be written to
uint16_t	u16_num_to_write	Specifies the number of registers to be written
uint8_t	u8_write_num_of_bytes	Specifies the number of bytes of data
uint16_t	aru16_data[MAX_REG_DATA]	Data to be written

* MAX_REG_DATA is defined in 125

· **p_resp_read_coils_t**

Member variables type	Member variables	Description
uint16_t	u16_transaction_id	Specifies the transaction ID
uint16_t	u16_protocol_id	Specifies the protocol ID
uint8_t	u8_slave_id	Identification of a remote slave connected (Own ID)
uint8_t	u8_exception_code	Error detected during processing the request. On success the exception code should be zero, if the exception code is nonzero the aru8_data will be ignored
uint8_t	u8_num_of_bytes	Specifies the number of bytes of data
uint8_t	aru8_data[MAX_DISCRETE_DATA]	Data to be read

* MAX_DISCRETE_DATA is defined in 251

· **p_resp_read_inputs_t**

Member variables type	Member variables	Description
uint16_t	u16_transaction_id	Specifies the transaction ID
uint16_t	u16_protocol_id	Specifies the protocol ID
uint8_t	u8_slave_id	Identification of a remote slave connected (Own ID)
uint8_t	u8_exception_code	Error detected during processing the request. On success the exception code should be zero, if the exception code is nonzero the aru8_data will be ignored
uint8_t	u8_num_of_bytes	Specifies the number of bytes of data
uint8_t	aru8_data[MAX_DISCRETE_DATA]	Buffer to store the read data

* MAX_DISCRETE_DATA is defined in 251

· **p_resp_read_holding_reg_t**

Member variables type	Member variables	Description
uint16_t	u16_transaction_id	Specifies the transaction ID
uint16_t	u16_protocol_id	Specifies the protocol ID
uint8_t	u8_slave_id	Identification of a remote slave connected (Own ID)
uint8_t	u8_exception_code	Error detected during processing the request. On success the exception code should be zero, if the exception code is nonzero the aru16_data will be ignored
uint8_t	u8_num_of_bytes	Specifies the number of bytes of data
uint16_t	aru16_data[MAX_REG_DATA]	Buffer to store the read data

* MAX_REG_DATA is defined in 125

· **p_resp_read_input_reg_t**

Member variables type	Member variables	Description
uint16_t	u16_transaction_id	Specifies the transaction ID
uint16_t	u16_protocol_id	Specifies the protocol ID
uint8_t	u8_slave_id	Identification of a remote slave connected (Own ID)
uint8_t	u8_exception_code	Error detected during processing the request. On success the exception code should be zero, if the exception code is nonzero the aru16_data will be ignored
uint8_t	u8_num_of_bytes	Specifies the number of bytes of data
uint16_t	aru16_data[MAX_REG_DATA]	Buffer to store the read data

* MAX_REG_DATA is defined in 125

· **p_resp_write_single_coil_t**

Member variables type	Member variables	Description
uint16_t	u16_transaction_id	Specifies the transaction ID
uint16_t	u16_protocol_id	Specifies the protocol ID
uint8_t	u8_slave_id	Identification of a remote slave connected (Own ID)
uint8_t	u8_exception_code	Error detected during processing the request. On success the exception code should be zero
uint16_t	u16_output_addr	Specifies address of the coil
uint16_t	u16_output_value	Data to be written

· **p_resp_write_single_reg_t**

Member variables type	Member variables	Description
uint16_t	u16_transaction_id	Specifies the transaction ID
uint16_t	u16_protocol_id	Specifies the protocol ID
uint8_t	u8_slave_id	Identification of a remote slave connected (Own ID)
uint8_t	u8_exception_code	Error detected during processing the request. On success the exception code should be zero
uint16_t	u16_register_addr	Specifies address of the register
uint16_t	u16_register_value	Data to be written

· **p_resp_write_multiple_coils_t**

Member variables type	Member variables	Description
uint16_t	u16_transaction_id	Specifies the transaction ID
uint16_t	u16_protocol_id	Specifies the protocol ID
uint8_t	u8_slave_id	Identification of a remote slave connected (Own ID)
uint8_t	u8_exception_code	Error detected during processing the request. On success the exception code should be zero
uint16_t	u16_start_addr	Specifies address of the first coil
uint16_t	u16_num_of_outputs	Specifies the number of coils to be written

· **p_resp_write_multiple_reg_t**

Member variables type	Member variables	Description
uint16_t	u16_transaction_id	Specifies the transaction ID
uint16_t	u16_protocol_id	Specifies the protocol ID
uint8_t	u8_slave_id	Identification of a remote slave connected (Own ID)
uint8_t	u8_exception_code	Error detected during processing the request. On success the exception code should be zero
uint16_t	u16_start_addr	Specifies address of the first register
uint16_t	u16_num_of_reg	Specifies the number of registers to be written

· **p_resp_read_write_multiple_reg_t**

Member variables type	Member variables	Description
uint16_t	u16_transaction_id	Specifies the transaction ID
uint16_t	u16_protocol_id	Specifies the protocol ID
uint8_t	u8_slave_id	Identification of a remote slave connected (Own ID)
uint8_t	u8_exception_code	Error detected during processing the request. On success the exception code should be zero, if the exception code is nonzero the aru16_read_data will be ignored
uint8_t	u8_num_of_bytes	Specifies the number of complete bytes of data
uint16_t	aru16_read_data[MAX_REG_DATA]	Data to be read

* MAX_REG_DATA is defined in 125

· **p_resp_invalid_function_code_t**

Member variables type	Member variables	Description
uint16_t	u16_transaction_id	Specifies the transaction ID
uint16_t	u16_protocol_id	Specifies the protocol ID
uint8_t	u8_slave_id	Identification of a remote slave connected (Own ID)
uint8_t	u8_exception_code	Error detected during processing the request. On success the exception code should be zero
uint8_t	u8_num_of_bytes	Specifies the number of complete bytes of data

Callback function

· fp_function_code1

[Description]

Callback function pointer for Modbus function code 1 (Read coils) operation.

[Format]

```
uint32_t
(*fp_function_code1_t)(
                        p_req_read_coils_t pt_req_read_coils,
                        p_resp_read_coils_t pt_resp_read_coils
);
```

[Parameter]

p_req_read_coils_t	pt_req_read_coils	structure pointer from stack to user with read coils request information
p_resp_read_coils_t	pt_resp_read_coils	structure pointer to stack from user with read coils response data

[Return value]

0: success

1: failure

· fp_function_code2

[Description]

Callback function pointer for Modbus function code 2 (Read Discrete Inputs) operation.

[Format]

```
uint32_t
(*fp_function_code2_t)(
                        p_req_read_inputs_t pt_req_read_inputs,
                        p_resp_read_inputs_t pt_resp_read_inputs
);
```

[Parameter]

p_req_read_inputs_t	pt_req_read_inputs	structure pointer from stack to user wirh read discreate inputs request information
p_resp_read_inputs_t	pt_resp_read_inputs	structure pointer from stack to user with read discrete inputs response data

[Return value]

0: success

1: failure

· fp_function_code3

[Description]

Callback function pointer for Modbus function code 3 (Read Holding Registers) operation.

[Format]

uint32_t

(*fp_function_code3_t(

p_req_read_holding_reg_t pt_req_read_holding_reg,

p_resp_read_holding_reg_t pt_resp_read_holding_reg

);

[Parameter]

p_req_read_holding_reg_t	pt_req_read_holding_reg	structure pointer from stack to user wirh read holding registers request information
p_resp_read_inputs_t	pt_resp_read_inputs	structure pointer to stack from user wirh read holding registers response data

[Return value]

0: success

1: failure

· fp_function_code4

[Description]

Callback function pointer for Modbus function code 4 (Read Input RegisterRead coils) operation.

[Format]

uint32_t

(*fp_function_code4_t(

p_req_read_input_reg_t pt_req_read_input_reg,

p_resp_read_input_reg_t pt_resp_read_input_reg

);

[Parameter]

p_req_read_input_reg_t	pt_req_read_input_reg	structure pointer from stack to user with read input registers request information
p_resp_read_input_reg_t	pt_resp_read_input_reg	structure pointer to stack from user with read input registers response data

[Return value]

0: success

1: failure

· fp_function_code5

[Description]

Callback function pointer for Modbus function code 5 (Write Single Coil) operation.

[Format]

```
uint32_t
(*fp_function_code5_t(
                        p_req_write_single_coil_t pt_req_write_single_coil,
                        p_resp_write_single_coil_t pt_resp_write_single_coil
);
```

[Parameter]

p_req_write_single_coil_t	pt_req_write_single_coil	structure pointer from stack to user with write single coil request information
p_resp_write_single_coil_t	pt_resp_write_single_coil	structure pointer to stack from user with write single coil response

[Return value]

0: success

1: failure

· fp_function_code6

[Description]

Callback function pointer for Modbus function code 6 (Write Single Register) operation.

[Format]

```
uint32_t
(*fp_function_code6_t(
                        p_req_write_single_reg_t pt_req_write_single_reg,
                        p_resp_write_single_reg_t pt_resp_write_single_reg
);
```

[Parameter]

p_req_write_single_reg_t	pt_req_write_single_reg	structure pointer from stack to user wirh write single register request information
p_resp_write_single_reg_t	pt_resp_write_single_reg	structure pointer to stack from user wirh write single register response

[Return value]

0: success

1: failure

· fp_function_code15

[Description]

Callback function pointer for Modbus function code 15 (Write Multiple Coils) operation.

[Format]

```
uint32_t
(*fp_function_code15_t(
                    p_req_write_multiple_coils_t pt_req_write_multiple_coils,
                    p_resp_write_multiple_coils_t pt_resp_write_multiple_coils
);
```

[Parameter]

p_req_write_multiple_coils_t	pt_req_write_multiple_coils	structure pointer from stack to user wirh write multiple coils request information
p_resp_write_multiple_coils_t	pt_resp_write_multiple_coils	structure pointer to stack from user wirh write multiple coils response

[Return value]

0: success

1: failure

· fp_function_code16

[Description]

Callback function pointer for Modbus function code 16 (Write Multiple Registers) operation.

[Format]

```
uint32_t
(*fp_function_code16_t(
                    p_req_write_multiple_reg_t pt_req_write_multiple_reg,
                    p_resp_write_multiple_reg_t pt_resp_write_multiple_reg
);
```

[Parameter]

p_req_write_multiple_reg_t	pt_req_write_multiple_reg	structure pointer from stack to user wirh write multiple registers request information
p_resp_write_multiple_reg_t	pt_resp_write_multiple_reg	structure pointer to stack from user wirh write multiple registers response

[Return value]

0: success

1: failure

· **fp_function_code25**

[Description]

Callback function pointer for Modbus function code 23 (Read/Write Multiple Registers) operation.

[Format]

```
uint32_t
(*fp_function_code23_t(
                        p_req_read_write_multiple_reg_t pt_req_read_write_multiple_reg,
                        p_resp_read_write_multiple_reg_t pt_resp_read_write_multiple_reg
);
```

[Parameter]

p_req_read_write_multiple_reg_t	pt_req_read_write_multiple_reg	structure pointer from stack to user with read/write multiple registers request information
p_resp_read_write_multiple_reg_t	pt_resp_read_write_multiple_reg	structure pointer to stack from user with read/write mltiple response

[Return value]

0: success

1: failure

Enumeration type

Enumeration type	Enumerator	Description
ENABLE_FLAG	DISABLE	IPlist is disabled.
	ENABLE	IPlist is enabled.
TABLE_MODE	REJECT	Reject the connection.
	ACCEPT	Accept the connection.
ERR_CODE	ERR_OK	On success.
	ERR_STACK_INIT	In stack initialization failure.
	ERR_MEM_ALLOC	Memory allocation failure.
	ERR_INVALID_STACK_INIT_PARAMS	Specifies invalid stack init information from user.
	ERR_TCP_IP_TABLE_DISABLED	IPlist is disabled.
	ERR_TCP_IP_TABLE_IP_ALREADY_PRESENT	Address already present in list.
	ERR_TCP_IP_TABLE_MAX_CLIENT	Maximum connections reached.
	etc.	Other error codes used in internal function.

Appendix C. FSP Configuration for VSC8531

RZ/N2L Industrial Network SOM Kit has VSC8531 as PHY chip.
If reconfiguring by latest FSP, FSP configuration and source code needs to change from default.

(1) Regenerate source files by lates FSP

Remove the following four folders. After that, open the project according to section 5.

- When using e2studio, \project\rzn2l_som\modbus_single\e2studio
- When using EWARM, \project\rzn2l_som\modbus_single\ewarm

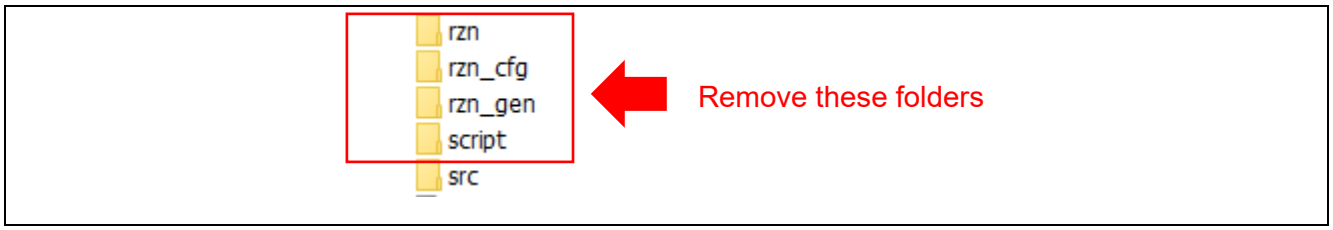

Figure 6-1 Remove folder generated by FSP

(2) Change ethernet driver configuration for VSC8531

Configure g_ether_phy0 Ethernet Driver on r_ether_phy for VSC8531.
Configuration value for VSC8531 shows in Table 6-1.

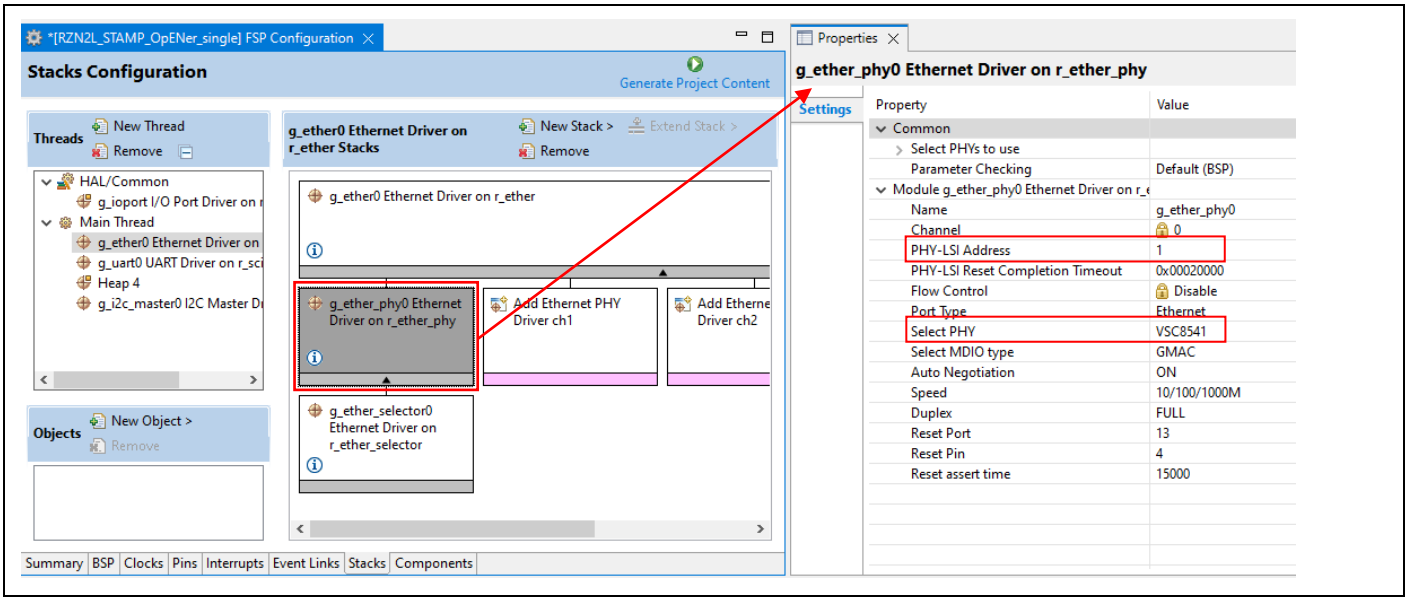

Figure 6-2 Ethernet Driver Configuration for VSC8531 (e.g. ETH0)

Table 6-1 FSP Configuration Value for VSC8531

Items	Default value	Config value for VSC8531	
		ETH0	ETH1
PHY-LSI Address	0	0	1
Select PHY	Default	VSC8541	VSC8541

(3) Add initialization code for VSC8531

The following code for VSC8531 initialization should be added to "ether_phy_targets_initialize_vsc8541" function in rzn/fsp/src/r_ether_phy/r_ether_phy.c. The inclusion of "board_som.h" is aiso required for code activation.

```
#include "board_som.h"
                                        ~~ Omission ~~

void ether_phy_targets_initialize_vsc8541 (ether_phy_instance_ctrl_t * p_instance_ctrl)
{
                                        ~~ Omission ~~

    /* LED Behavior */
    reg  = ether_phy_read(p_instance_ctrl, ETHER_PHY_REG_LED_BEHAVIOR);
    reg &= ~(1U << ETHER_PHY_REG_LED0_FEATURE_DISABLE_OFFSET);
    reg |= 1U << ETHER_PHY_REG_LED1_FEATURE_DISABLE_OFFSET;
    ether_phy_write(p_instance_ctrl, ETHER_PHY_REG_LED_BEHAVIOR, reg);
#if defined(BOARD_RZN2L_SOM_KIT)  /* for VSC8531 */
    /* select extended page 2 register */
    ether_phy_write(p_instance_ctrl,   ETHER_PHY_REG_EXTEND_GPIO_PAGE, 0x02);

    /* read WoL and MAC Interface Control */
    reg  = ether_phy_read(p_instance_ctrl, 0x1b);

    /* set control to slow */
    reg &= 0xFF9F;
    ether_phy_write(p_instance_ctrl, 0x1b, reg);

    /* Configure RX_CLK delay and TX_CLK delay to 2.0ns */
    ether_phy_write(p_instance_ctrl, ETHER_PHY_REG_EXPAGE2_RGMII_CTRL, 0x0044);

    /* select extended page 0 register */
    ether_phy_write(p_instance_ctrl,   ETHER_PHY_REG_EXTEND_GPIO_PAGE, 0x00);
#endif

}                                   /* End of function ether_phy_targets_initialize() */
```

Revision History

Rev.	Date	Description	
		Page	Summary
1.0	Aug. 7, 2023	-	First Edition issued

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 system-evaluation test for the given product.

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/.