

RZ/T2L Group

Quick Start Guide: Modbus TCP Server Software

Introduction

This document explains Sample Program setup procedures for Modbus TCP functionalities with the adapted Modbus protocol stack code for Renesas RZ/T2L platform.

Target Device

RZ/T2L

Contents

1. Overview	3
1.1 Abbreviations / Definitions	3
1.2 Reference	3
2. Features	3
3. Project Setup	4
3.1 Requirements	4
3.2 Hardware	5
3.3 Setting the Board	9
4. Setup a Client tool	10
5. Running the sample application	11
5.1 Setup sample project for e ² studio	11
5.2 Setup sample project for EWARM	14
5.3 Demonstration	18
6. Appendix	20
6.1 Appendix A. DHCP mode	20
6.2 Appendix B. Multi-client Configuration	21
6.3 Appendix C. Application Programming Interface	22
7. Limitations	37
Revision History	38

1. Overview

This document describes the procedure for testing the Modbus TCP server function using Modbus protocol stack code compatible with the Renesas RZ/T2L platform. Run the standalone variant using only one core.

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/T2L is available via Renesas.

Table 1.2 Technical Inputs for RZ/T2L

Index	Technical Inputs
1	r01an6434ejxxxx-rzt2-rzn2-fsp-getting-started.pdf
2	r01ds0409ejxxxx-rzt2l.pdf
3	r01uh0985ejxxxx-rzt2l.pdf
4	r20ut5164ejxxxx-rskrzt2l.pdf
5	Modicon Modbus Protocol Reference Guide Rev.J
6	Modbus Application Protocol Specification V1.1b3

2. Features

The Modbus protocol stack for RZ/T2L 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 RZ/T2L Modbus protocol stack project has been developed and tested on these environments using the following boards and tools.

Table 3.1 RZ/T2L Requirements

Item	Description
Board	Renesas Electronics RZ/T2L RSK Board
IDE	IAR Systems - IAR Embedded Workbench® for ARM Version 9.60.3 Renesas Electronics - e² studio 2025-12 GCC toolchain GNU ARM Embedded Toolchain (version 13.3.1.arm-13-24 for CR52) - FSP Smart Configurator 2025-12
Emulator	IAR Systems I-jet SEGGER J-Link 8.60
Client demo tool	Renesas Electronics ModbusDemoApplication.exe (Included in this package)

3.2 Hardware

This document describes the major hardware. Refer to Renesas Starter Kit+ for RZ/T2L user's manual and schematic for more board details.

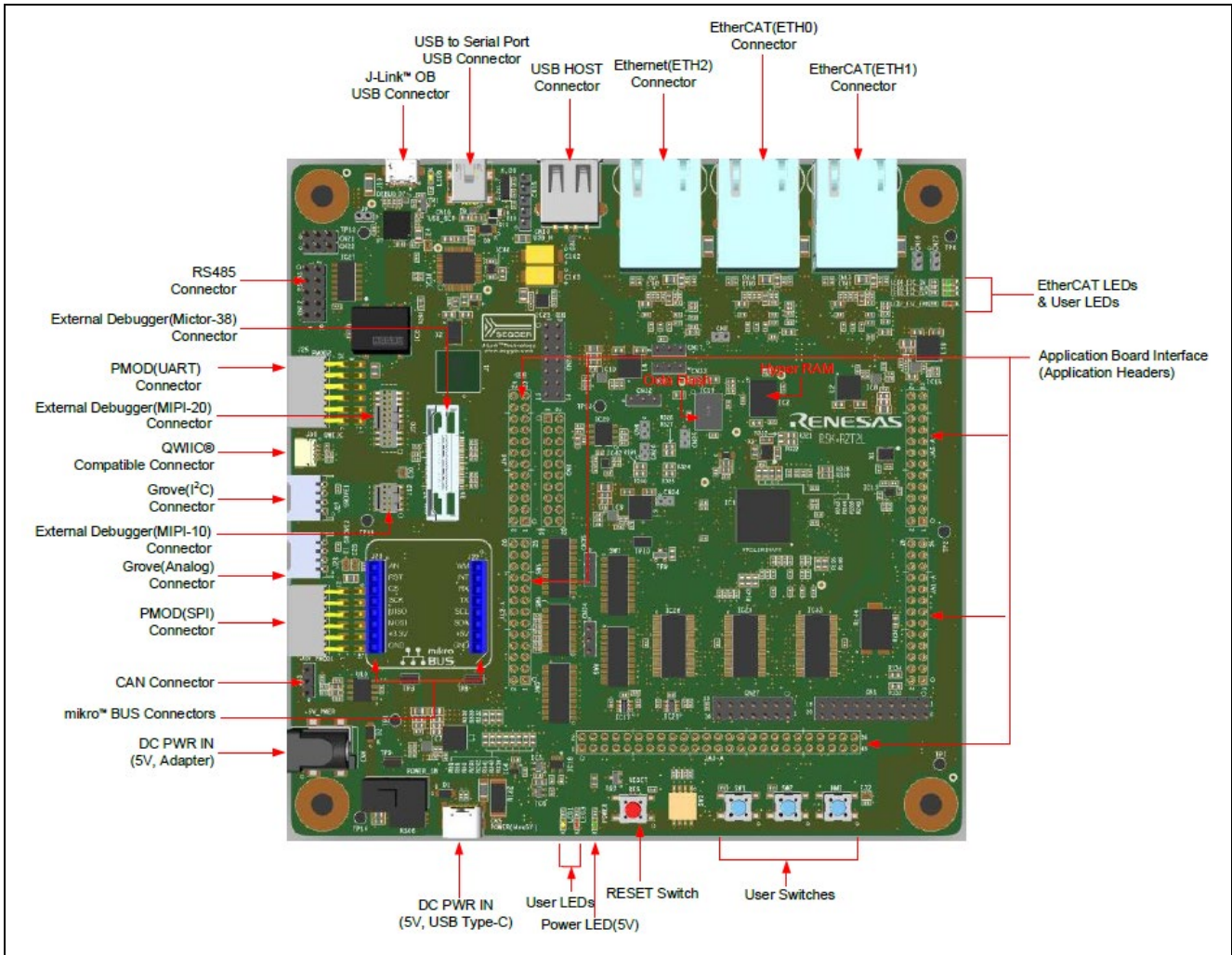


Figure 3.1 RZT2L RSK board layout

Note), When operating Modbus TCP on the RZT2L RSK board, the GMAC port is assigned to Port2, so connect the Ether cable here.

Table 4. Jumper Pin settings

Reference	Jumper Position	Description
CN2	Short	VCC1833_2 current measurement
CN4	Short	VCC1833_3 current measurement
CN24	Short	VCC11_RVCORE current measurement
CN25	Short	CPU1V8 current measurement
CN26	Short	CPU3V3 current measurement
CN17	Short 1-2	VCC1833_2 Use power supply at 3.3V
	Short 2-3	VCC1833_2 Use power supply at 1.8V
CN32	Short 1-2	VCC1833_3 Use power supply at 3.3V
	Short 2-3	VCC1833_3 Use power supply at 1.8V
CN21	Short 1-2	Full-duplex communication
	Short 2-3	Half-duplex communication
CN22	Short 1-2	Full-duplex communication
	Short 2-3	Half-duplex communication
CN34	Short 1-2	Use TXD0_P16_0_JA1 as transmit data signal and RXD0_P16_1_JA1 as receive data signal (use SCI ch0)
	Short 2-3	Use TXD1_MDAT5 as transmit data signal and RXD1_P24_0_JA1 as receive data signal (use SCI ch1)
CN35	Short 1-2	Use TXD0_P16_0_JA1 as transmit data signal and RXD0_P16_1_JA1 as receive data signal (use SCI ch0)
	Short 2-3	Use TXD1_MDAT5 as transmit data signal and RXD1_P24_0_JA1 as receive data signal (use SCI ch1)
CN33	Short 1-2	Connect XSPI1_CS0# to CS# of QuadSPI FLASH (IC3)
	Short 2-3	Connect XSPI1_CS0# to CS# of extended SPI connector (CN28)

Table 5. Switch4 settings

SW4	Setting	Description
SW4-1	ON	16-bit bus boot mode (NOR flash)
SW4-2	OFF	* Refer to "r20ut5164ejxxx-rskrzt2l.pdf" and set according to the mode to be used.
SW4-3	ON	
SW4-4	OFF	ATCM wait cycle = 1 wait
SW4-5	ON	JTAG mode = Normal mode
SW4-6	ON	VCC1833_2 = 1.8V
SW4-7	ON	VCC1833_3 = 3.3V
SW4-8	OFF	Not used

Note), RAM mode is shared between xSPI0 and x1.

Therefore, if there is data in xSPI0, please delete it as it may cause a problem.

Table 6. Switch5 settings

SW5	Setting	Description
SW5-1	ON	Use P01_7 as CAN_RX_OB of CAN interface
SW5-2	OFF	* Refer to" r20ut5164ejxxxx-rskrzt2l.pdf" and set according to the mode to be used.
SW5-3	OFF	
SW5-4	OFF	
SW5-5	OFF	
SW5-6	OFF	Use P02_0 as CAN_TX1_JA5 of JA5-A
SW5-7	OFF	* Refer to" r20ut5164ejxxxx-rskrzt2l.pdf" and set according to the mode to be used.
SW5-8	OFF	
SW5-9	ON	
SW5-10	OFF	Not used

Table 7. Switch6 settings

SW6	Setting	Description
SW6-1	ON	Use P02_2 as CAN_TX_OB of CAN interface
SW6-2	OFF	* Refer to" r20ut5164ejxxxx-rskrzt2l.pdf" and set according to the mode to be used.
SW6-3	OFF	
SW6-4	OFF	
SW6-5	OFF	
SW6-6	OFF	Use P02_3 as BSC_A15 of JA3-A
SW6-7	OFF	* Refer to" r20ut5164ejxxxx-rskrzt2l.pdf" and set according to the mode to be used.
SW6-8	ON	
SW6-9	OFF	
SW6-10	OFF	Not used

Table 8. Switch7 settings

SW7	Setting	Description
SW7-1	OFF	Use P05_5 as HDSL1_LINK of pin header CN1
SW7-2	OFF	* Refer to" r20ut5164ejxxxx-rskrzt2l.pdf" and set according to the mode to be used.
SW7-3	ON	
SW7-4	OFF	Use P17_6 as LED1 for user LED control
SW7-5	OFF	* Refer to" r20ut5164ejxxxx-rskrzt2l.pdf" and set according to the mode to be used.
SW7-6	ON	
SW7-7	OFF	Use P18_1 as LED3 for user LED control
SW7-8	OFF	* Refer to" r20ut5164ejxxxx-rskrzt2l.pdf" and set according to the mode to be used.
SW7-9	OFF	
SW7-10	ON	

Table 9. Switch8 settings

SW8	Setting	Description
SW8-1	ON	Use P22_3 as Ethernet Port GMAC_RESETOU#
SW8-2	OFF	* Refer to" r20ut5164ejxxxx-rskrzt2l.pdf" and set according to the mode to be used.
SW8-3	OFF	
SW8-4	ON	Use P22_1 as serial host interface,
SW8-5	OFF	HSPI_IO7_M2POE_BSC_D08 for JA5-A and JA3-A (When SW8-10=OFF:TRACE_OPTION_SEL=H) * Refer to" r20ut5164ejxxxx-rskrzt2l.pdf" and set according to the mode to be used.
SW8-6	OFF	Not used
SW8-7	ON	ECAT0_OPTION_SEL = 'L' Select ETH0-related signals with bus switch IC33
SW8-8	ON	ECAT1_OPTION_SEL = 'L' Select ETH1-related signals with bus switch IC41
SW8-9	OFF	XSPI1_OPTION_SEL = 'H' Select signals other than XSPI1 related signals with bus switch IC37
SW8-10	OFF	TRACE_OPTION_SEL = 'H' Select signals other than TRACE-related signals with bus switch IC12

3.3 Setting the Board

Setting the board for running sample program is shown below.

Build and run the sample code on the RZ/T2L RSK board by following the steps below.

Both loading into RAM and flash can be done using IAR Embedded Workbench or e² studio.

1. Connect the decoder to the header "J20" on the RZ/T2L RSK board.

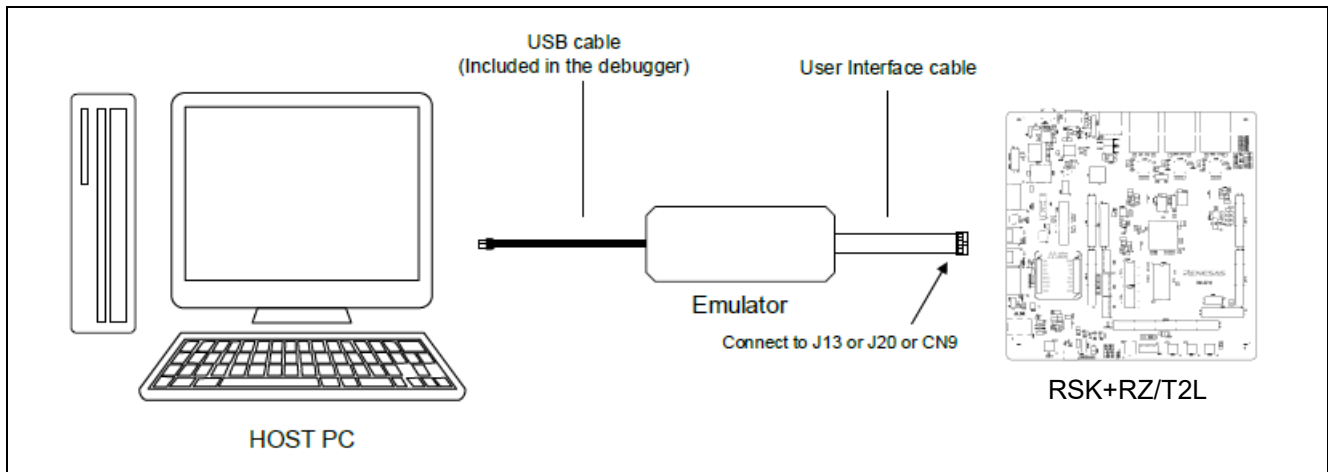


Figure 3.2: RZ/T2L RSK board debug connection diagram

When using J-Link OB, connect the USB cable to the header "J10" and set "J9" to open.

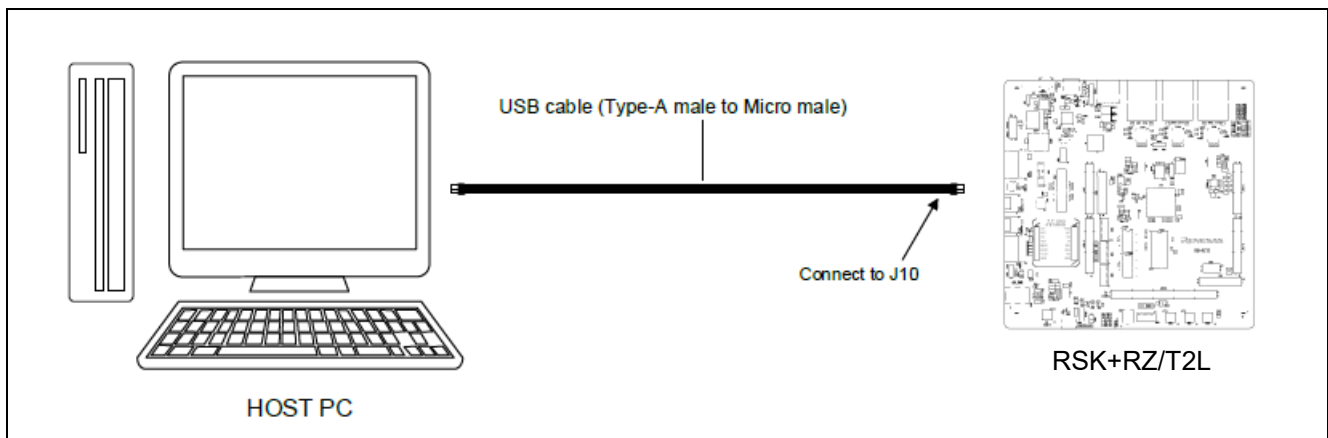


Figure 3.3: RZ/T2L RSK board debug connection diagram (J-Link OB)

2. Power is supplied using a USB cable (Type-C) or an AC / DC adapter. When using a USB cable (Type-C), connect it to the USB connector "CN5" of the RZ/T2L RSK board. When connecting to the AC/DC adapter, connect it to the connector "CN6" of the RZ/T2L RSK board.
3. Connect the Ethernet LAN cable to ETH2 on the RZ/T2L RSK board as shown on Fig. 3.1.

4. Setup a Client 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").

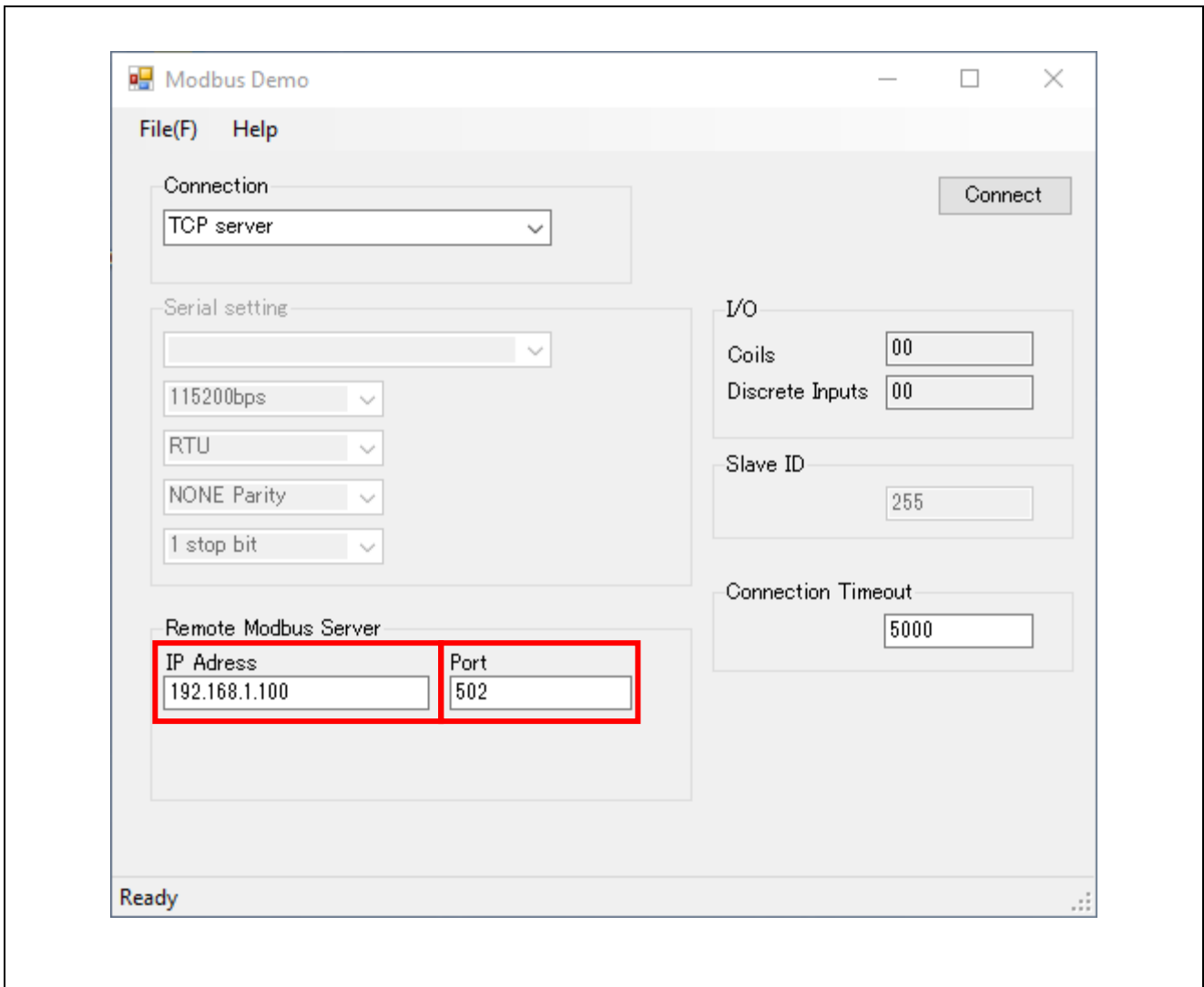


Figure 4.1 ModbusDemoApplication remote server setting

This sample project, IP address of the accepted client is set to "192.168.1.101", so please adjust the network settings of your PC accordingly.

5. Running the sample application

Refer to Section 3.3 Setting the Board for board settings.

The setup differs depending on the IDE.

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

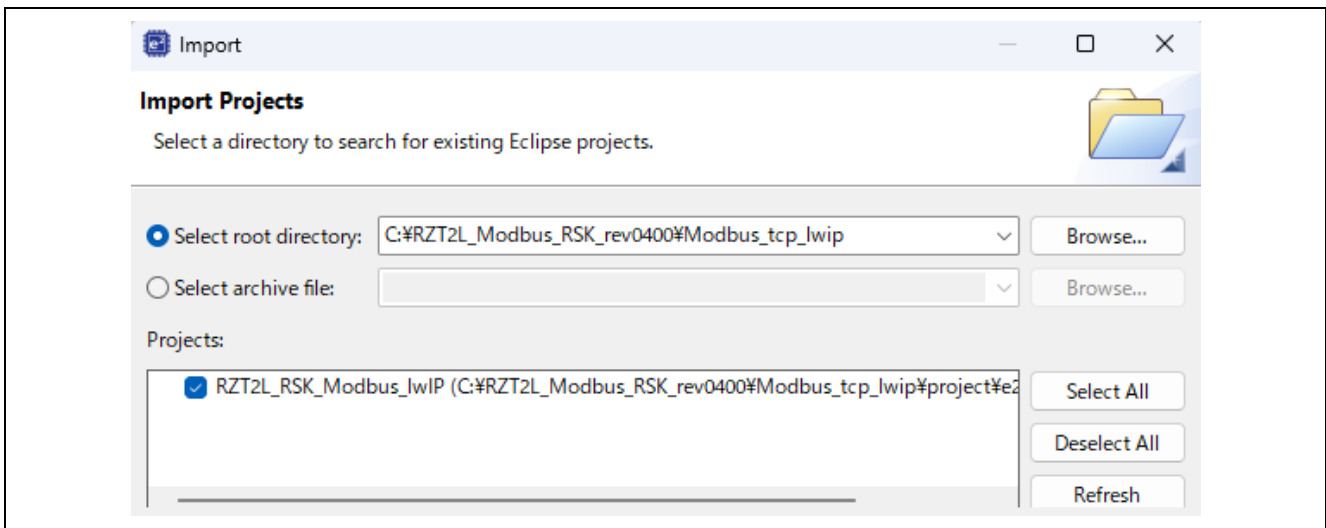
5.1 Setup sample project for e² studio

Build the sample code and load it into RAM using Renesas Electronics e² studio.

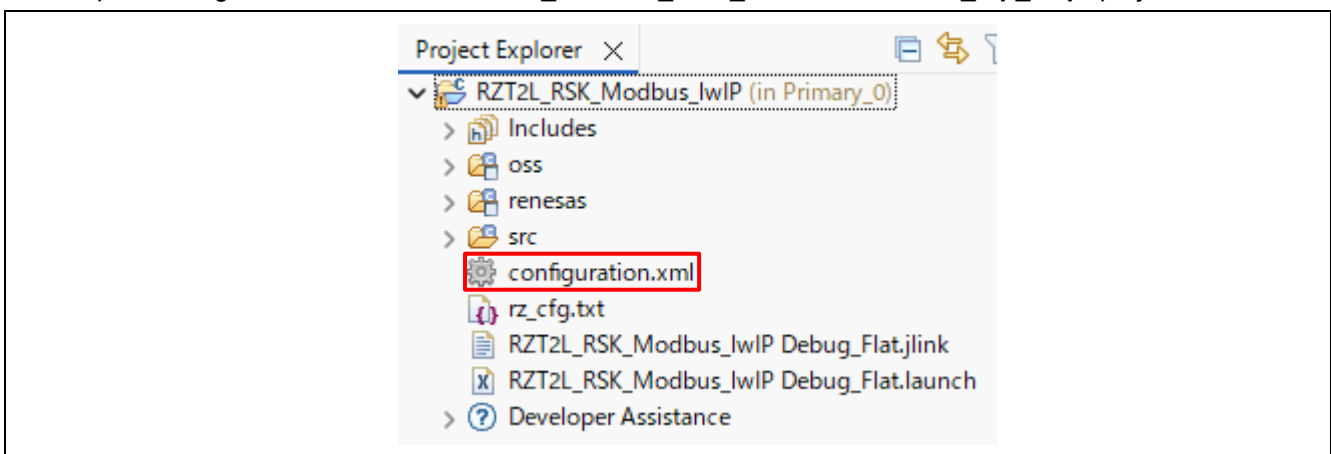
Note). Please install e² studio and adapt the RZ_FSP_Packs_v4.0.0 in advance.

Refer to the latest getting started guide. (r01an6434ejxxxx-rzt2-rzn2-fsp-getting-started.pdf)

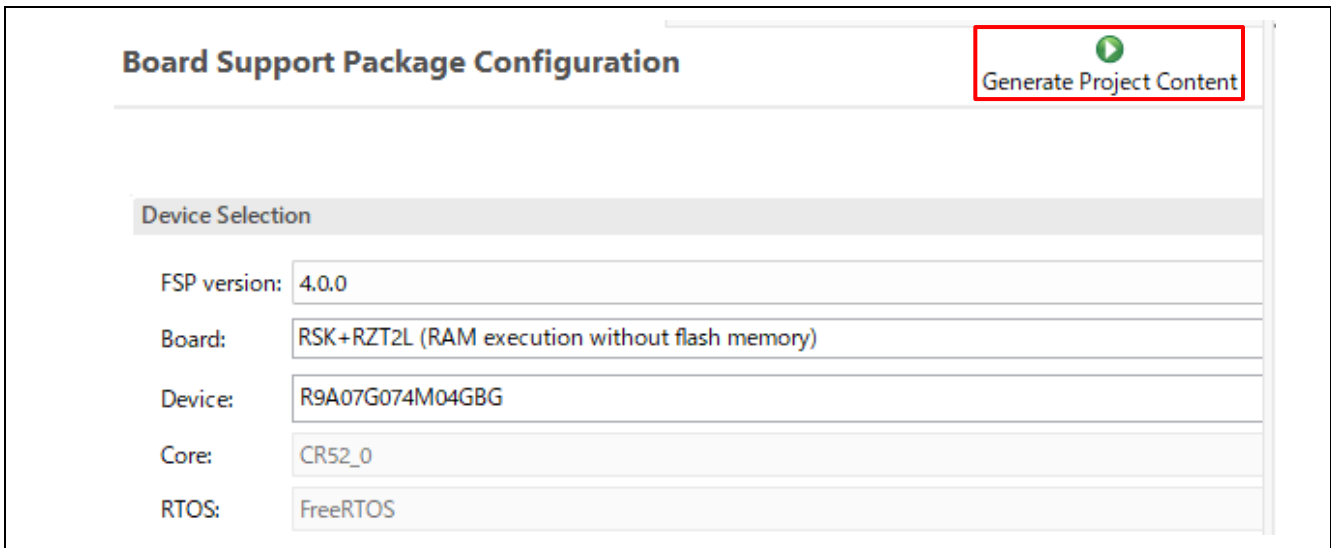
1. Import the sample project. After the program is started, by selecting [File] → [Import] → [Existing Projects into Workspace]. Check the "select root directory" and select "RZT2L_Modbus_RSK_rev0400\Modbus_tcp_lwip" folder →[Finish].



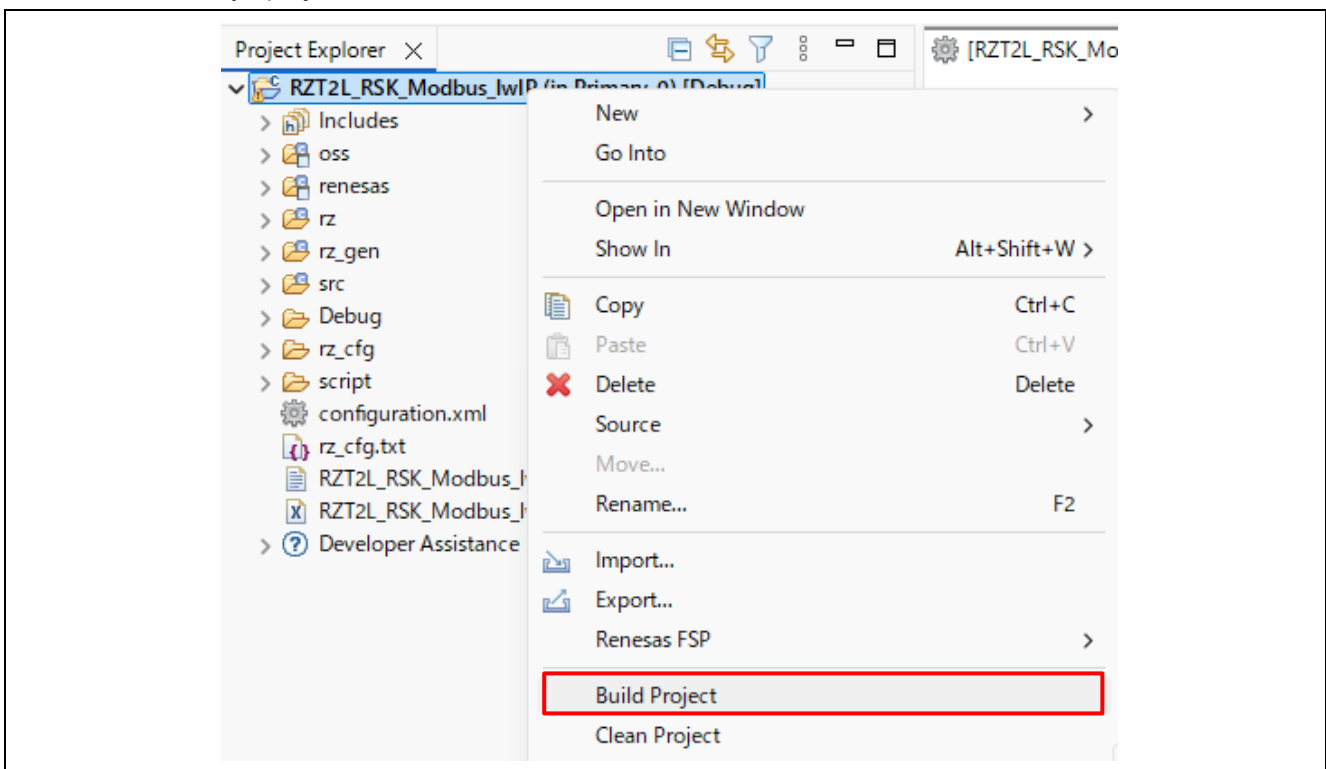
2. Open "configuration.xml" in the "RZT2L_Modbus_RSK_rev0400\ Modbus_tcp_lwip" project



3. Generate the code with "Generate Project Content".

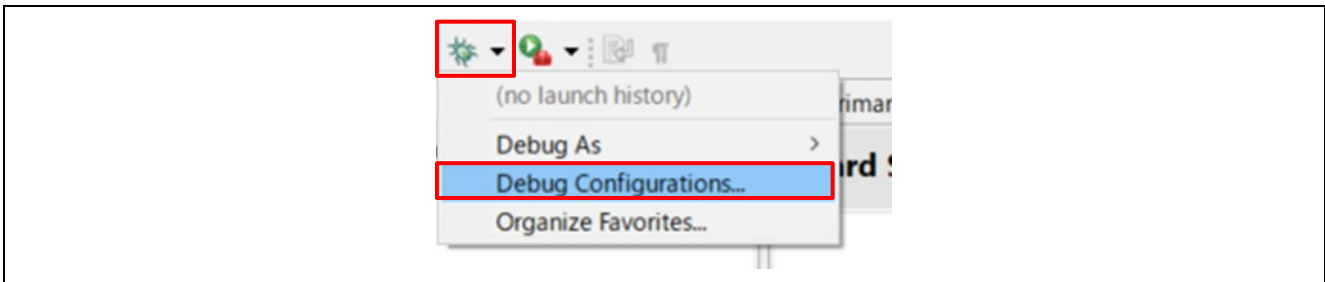


4. Select Primary0 project and execute the build.

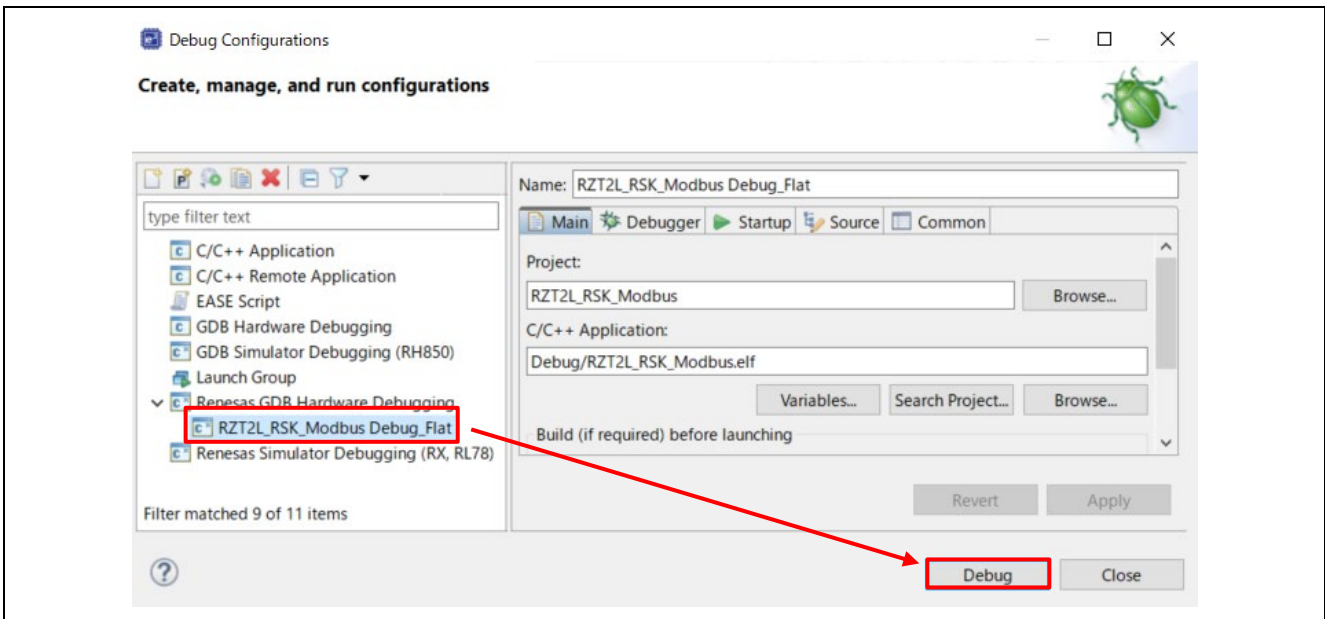


5. Press the "RESET" switch of the RSK + RZT2L board.

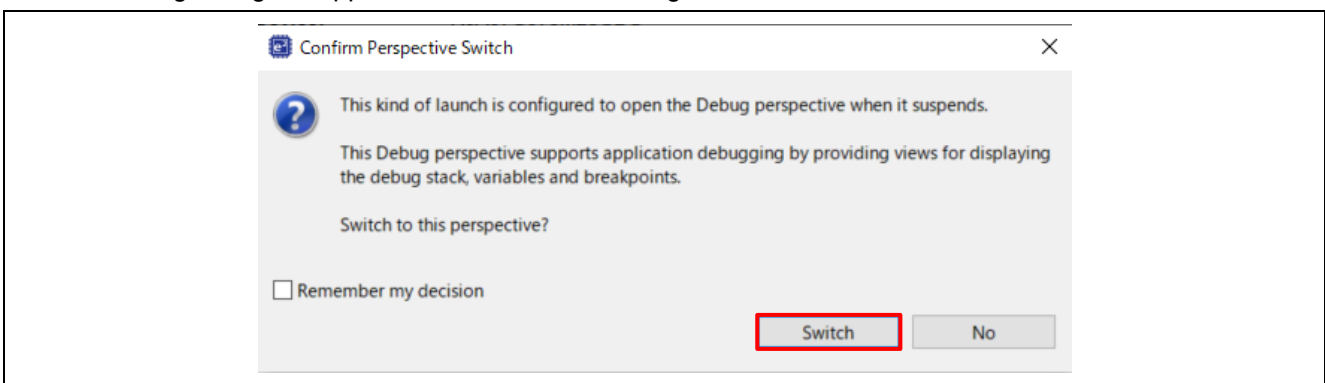
6. Select the drop-down menu next to the bug icon and selecting "Debugger Configurations ...".



[Renesas GDB Hardware Debugging] → [RZT2L_RSK_Modbus_IWIP Debug_Flat] item, then press [Debug].



Following dialog will appear, so switch to the debug screen.



7. Press the "Resume" button.
 When debugging CPU0 is started, the program is interrupted at "hal_entry ();" in main.c.
 Press the "Resume" button again. Program will run.

5.2 Setup sample project for EWARM

Build the sample code and load it into RAM using IAR Systems EWARM.

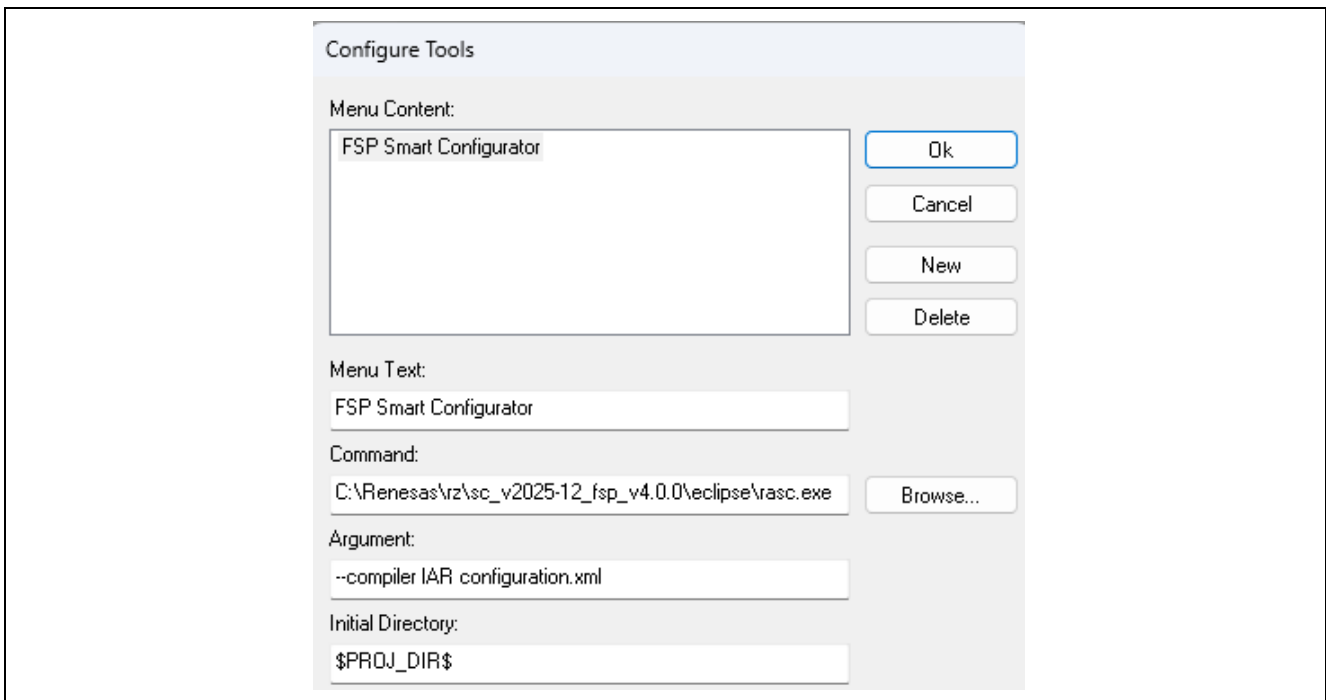
Note). Please install EWARM and adapt the RZ_FSP_Packs_v4.0.0 in advance.

Refer to the latest getting started guide. (r01an6434ejxxxx-rzt2-rzn2-fsp-getting-started.pdf)

Note). In EWARM, set the Smart Configurator startup settings as follows.

[Tools] -> [Configure Tools]

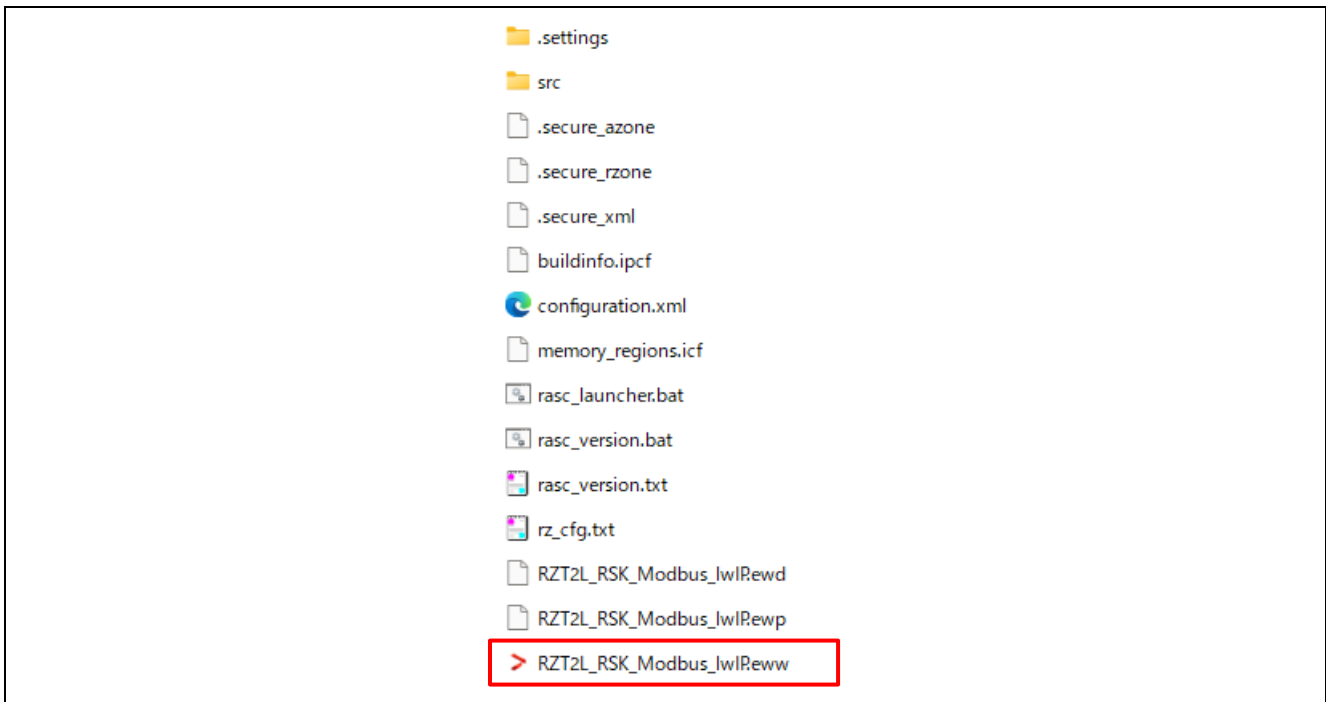
- Menu Text: **FSP Smart Configurator**
- Command: **\$RASC_EXE_PATH\$** (The following example is "C:\Renesas\rz\sc_v2025-12_fsp_v4.0.0\eclipse\rasc.exe")
- Argument: **--compiler IAR configuration.xml**
- Initial Directory: **\$PROJ_DIR\$**



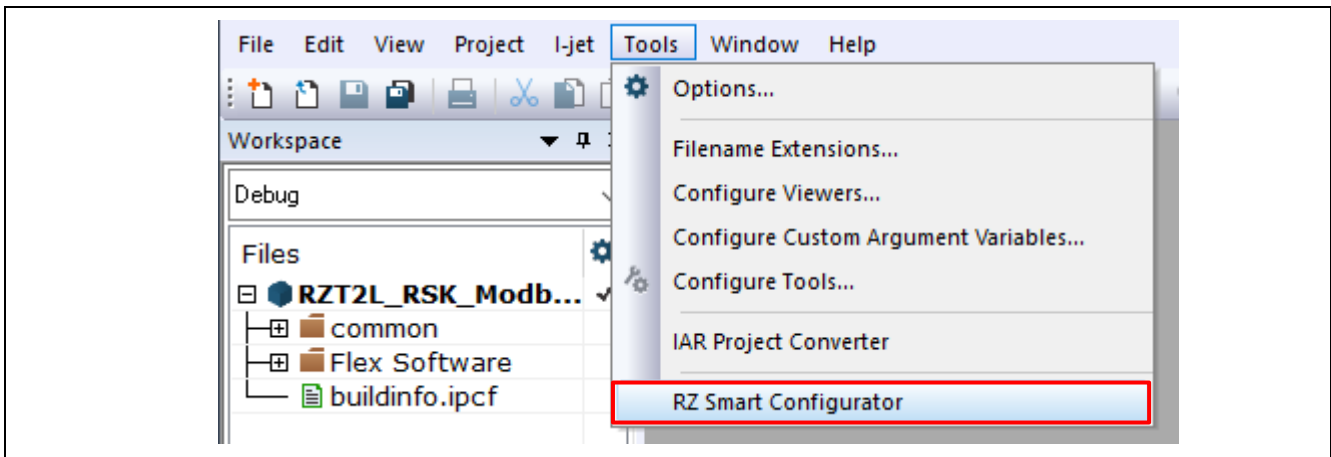
Edit "buildinfo.ipcf" file. Open the buildinfo.ipcf file and edit "RASC_EXE_PATH" to match the installation path of "rasc.exe".

```
<customArgVars>↓
  <group name="RA Smart Configurator">↓
    <argVar>↓
      <name>RASC_EXE_PATH</name>↓
      <value>C:\Renesas\rz\sc_v2025-12_fsp_v4.0.0\eclipse\rasc.exe</value>↓
    </argVar>↓
```

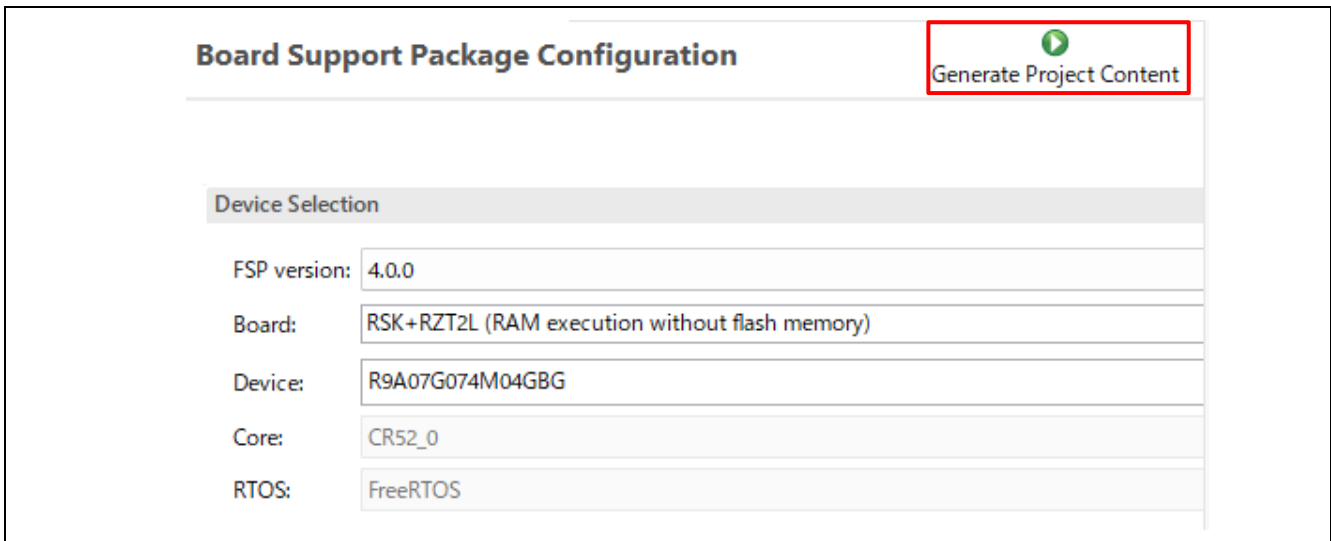
1. Open the sample project.
"RZT2L_Modbus_RSK_rev0400\ModbusTCP\modbus_tcp\projectlewarm
\RZT2L_RSK_Modbus.eww"



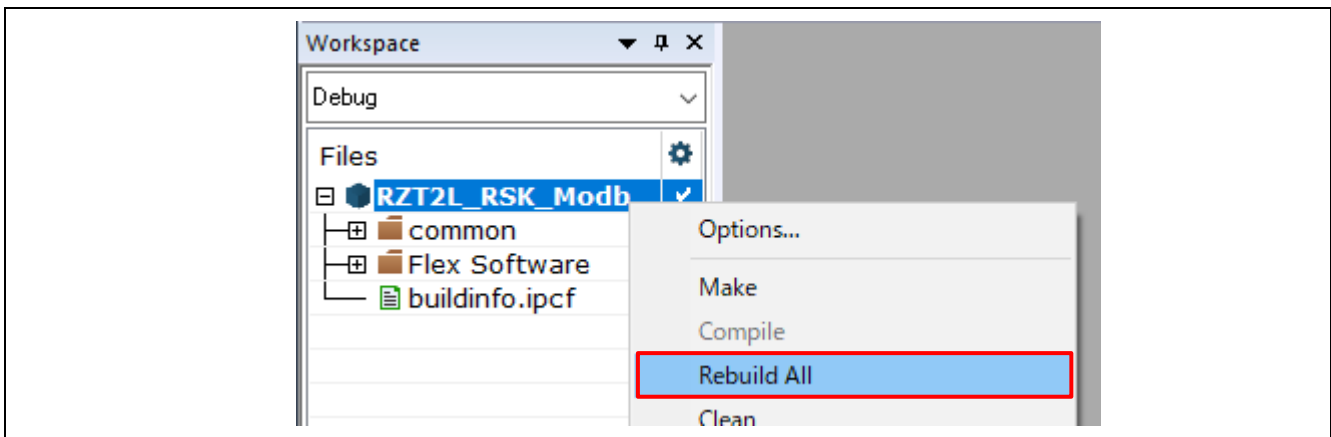
2. Open the "RZ Smart Configurator"



4. Generate the code with "Generate Project Content".

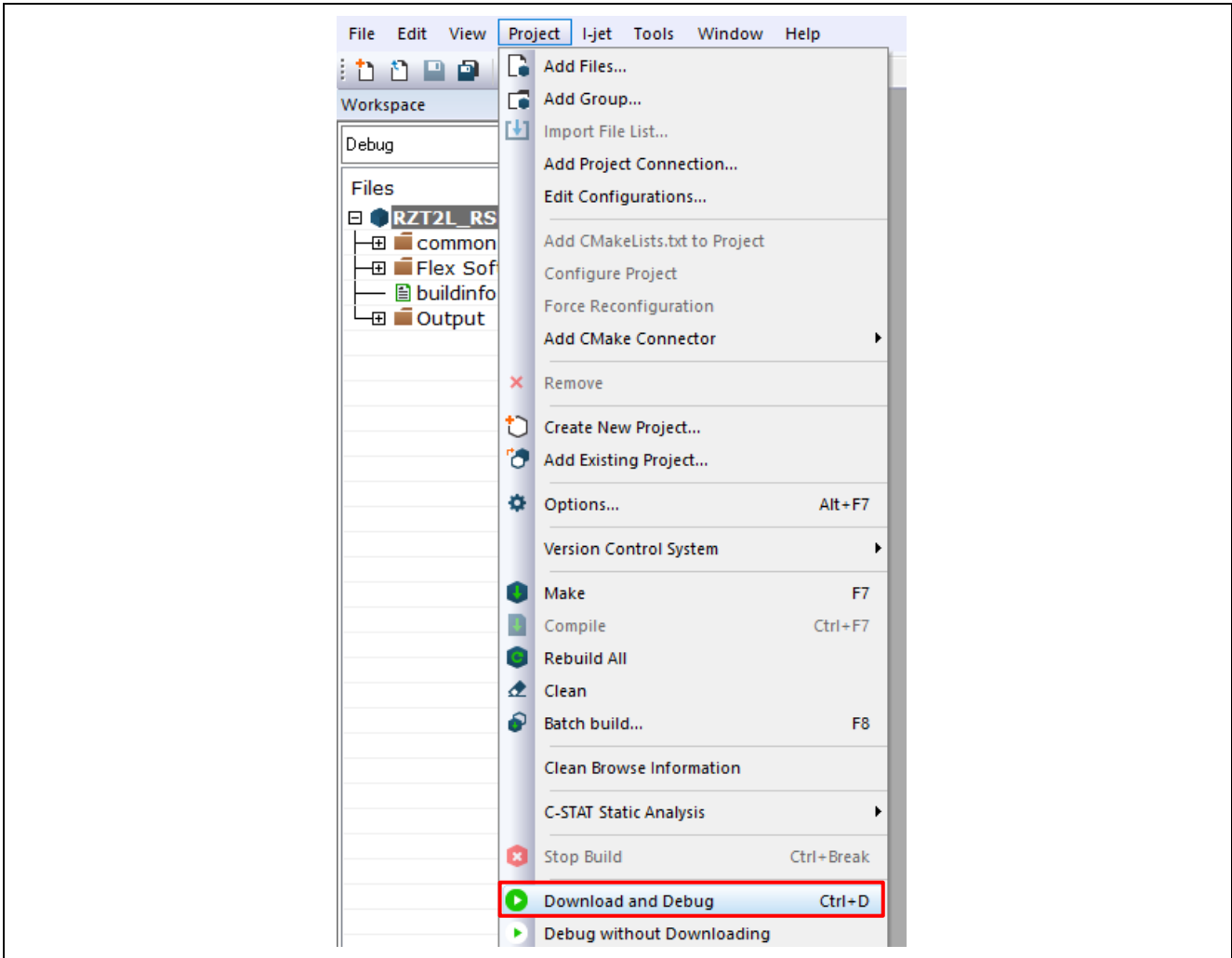


5. Select the "Rebuild All" item from the "Project" menu to rebuild the project.

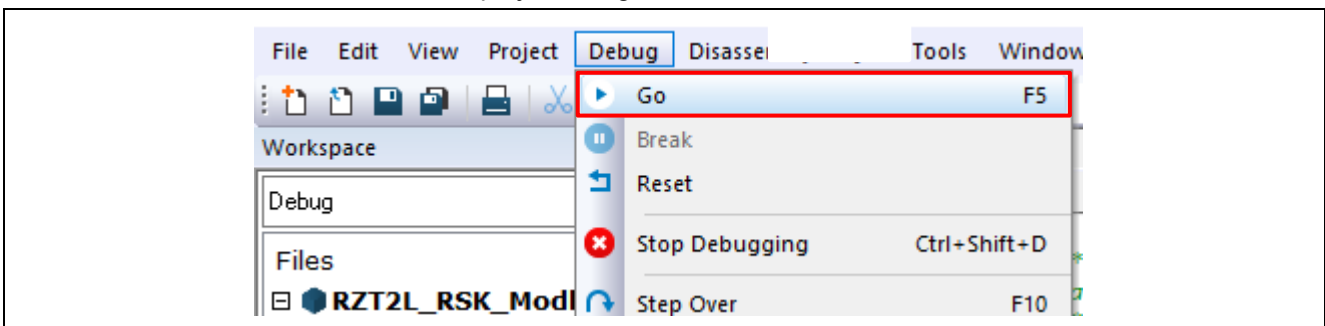


6. Press the "RESET" switch of the RSK + RZT2 board.

7. After connecting the board and I-jet, click the Download and Debug button on the Project toolbar.



8. Press the "Resume" button for the project. Program will run.



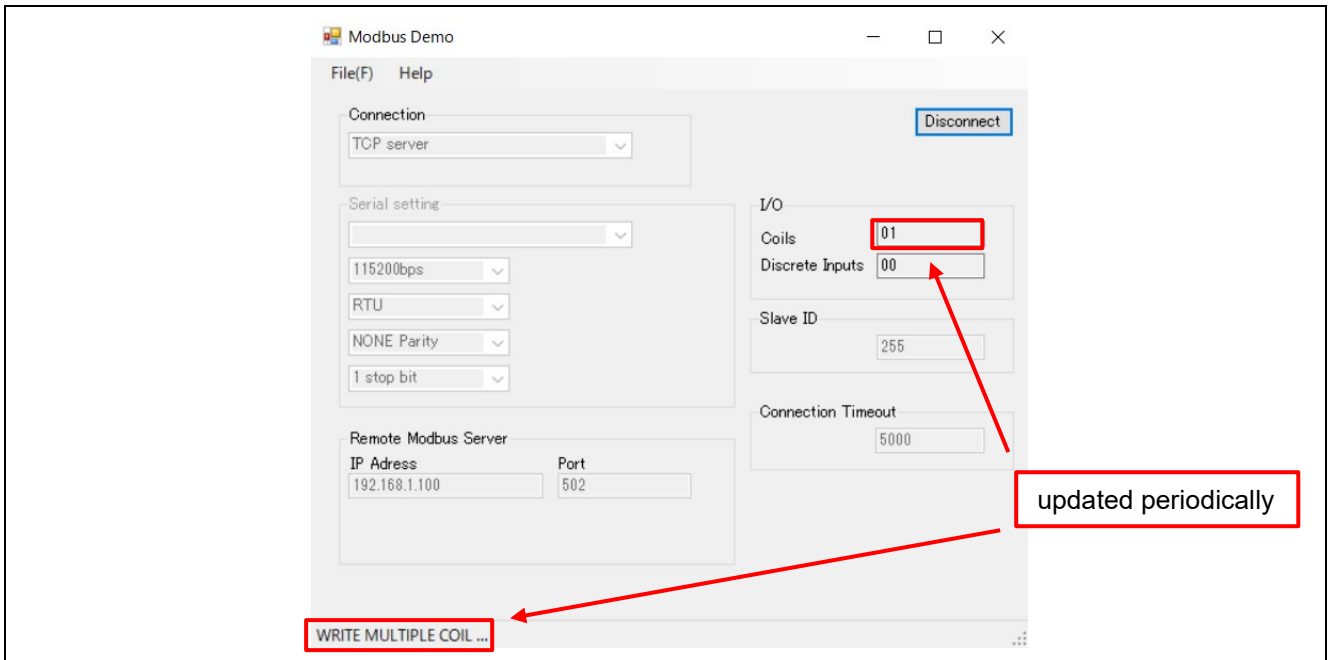
5.3 Demonstration

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

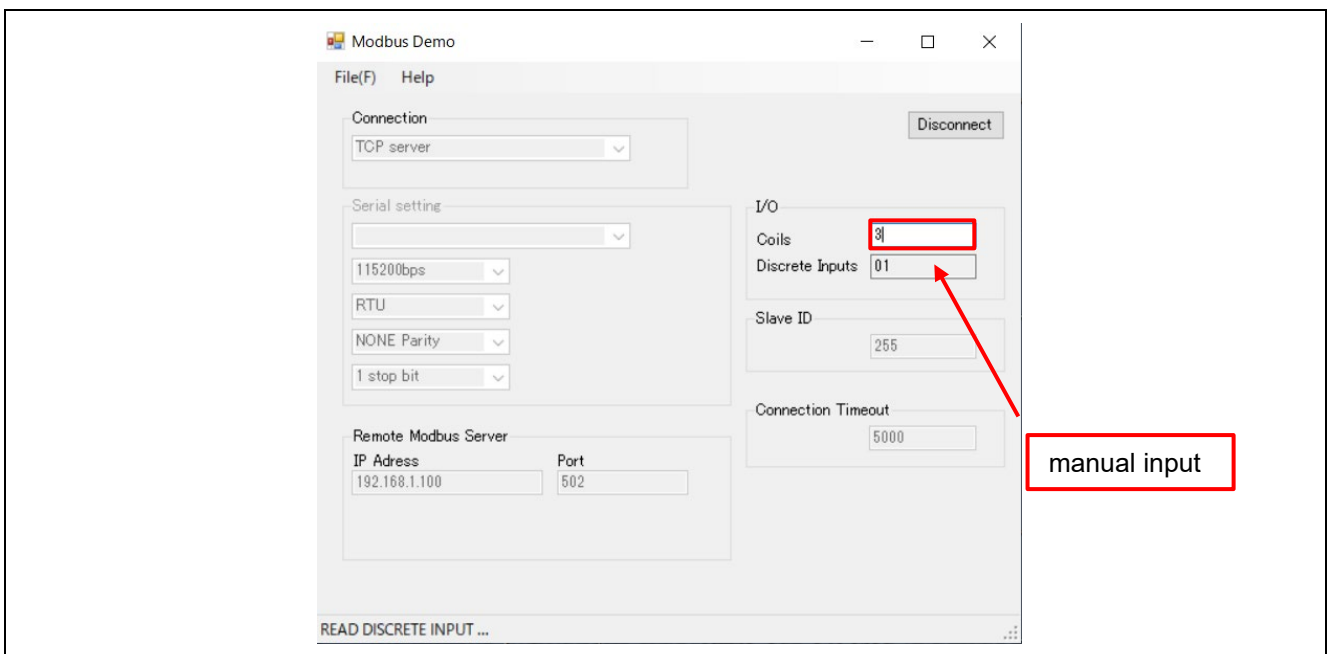
PC application checks the state of the switch (SW3), by using Modbus "Read_Discrete_Inputs" function code. The [SW setting value] is the 8-bits of data calculated by the state of SW3.

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

1. When SW3-1 is OFF, LED0,1 flash periodically.



2. When SW3-1 is ON, LED0,1 lights up depending on the input value of the coil value of the demo application's I/O.



3. Board IP address setting.

Set a fixed IP address by changing the following sections. This setting is not necessary if you use the default IP.

Set the required server network address in the following files included in the lwip projects.

lwip: **Modbus_tcp_lwip\common\renesas\application\lwip_port_instance.c**

```
* lwIP network interface module instance
*/
static lwip_port_instance_ctrl_t g_lwip_port0_ctrl;
static lwip_port_common_ctrl_t g_lwip_port0_common_ctrl;
static uint8_t gp_lwip_port0_hostname[16] = "LWIP_NETIF0";
static lwip_port_netif_cfg_t g_lwip_port0_netif_cfg =
{
    .dhcp          = LWIP_PORT_DHCP_DISABLE,
    .ip_address    = PP_HTONL(LWIP_MAKEU32(192,168, 1,100)),
    .subnet_mask   = PP_HTONL(LWIP_MAKEU32(255,255,255, 0)),
    .gateway_address = PP_HTONL(LWIP_MAKEU32(192,168, 1, 1)),
    .p_host_name   = gp_lwip_port0_hostname
};
```

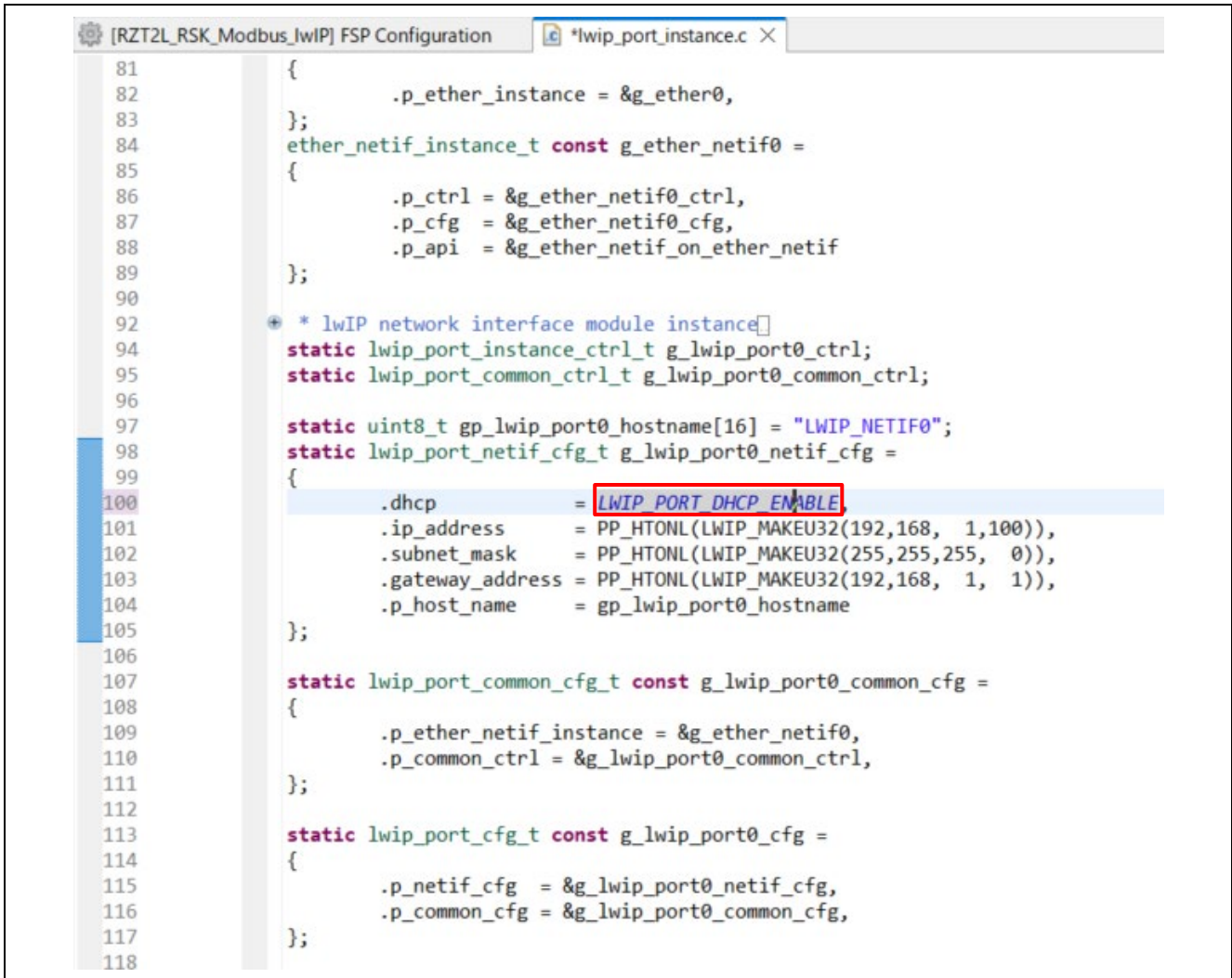
The default settings are as follows:

- IP address **192.168.1.100**
- Subnet mask **255.255.255.0**

6. Appendix

6.1 Appendix A. DHCP mode

1. When operating with DHCP, change the member "dhcp" of the variable "g_lwio_port0_netif_cfg" in "lwip_port_instance.c" to "LWIP_PORT_DHCP_ENABLE" to enable the mode and enable operation.
2. Build and debug.



```
[RZT2L_RSK_Modbus_LwIP] FSP Configuration | *lwip_port_instance.c X
81      {
82          .p_ether_instance = &g_ether0,
83      };
84      ether_netif_instance_t const g_ether_netif0 =
85      {
86          .p_ctrl = &g_ether_netif0_ctrl,
87          .p_cfg = &g_ether_netif0_cfg,
88          .p_api = &g_ether_netif_on_ether_netif
89      };
90
92      * lwIP network interface module instance
94      static lwip_port_instance_ctrl_t g_lwip_port0_ctrl;
95      static lwip_port_common_ctrl_t g_lwip_port0_common_ctrl;
96
97      static uint8_t gp_lwip_port0_hostname[16] = "LWIP_NETIF0";
98      static lwip_port_netif_cfg_t g_lwip_port0_netif_cfg =
99      {
100         .dhcp = LWIP_PORT_DHCP_ENABLE,
101         .ip_address = PP_HTONL(LWIP_MAKEU32(192,168, 1,100)),
102         .subnet_mask = PP_HTONL(LWIP_MAKEU32(255,255,255, 0)),
103         .gateway_address = PP_HTONL(LWIP_MAKEU32(192,168, 1, 1)),
104         .p_host_name = gp_lwip_port0_hostname
105     };
106
107     static lwip_port_common_cfg_t const g_lwip_port0_common_cfg =
108     {
109         .p_ether_netif_instance = &g_ether_netif0,
110         .p_common_ctrl = &g_lwip_port0_common_ctrl,
111     };
112
113     static lwip_port_cfg_t const g_lwip_port0_cfg =
114     {
115         .p_netif_cfg = &g_lwip_port0_netif_cfg,
116         .p_common_cfg = &g_lwip_port0_common_cfg,
117     };
118
```

6.2 Appendix B. Multi-client Configuration

This project supports multiple clients.

The initial state is enabled, and clients that can receive will be able to register their IP addresses. Please add your IP address in the section below.

In the initial state, the only valid IP address is "**192.168.1.101**"

Example of adding "192.168.1.102" and "192.168.1.103".

RZT2L_Modbus_RSK_rev0400\Modbus_tcp_lwip\common\renesas\modbus_user\modbus_init.c

```
uint32_t modbus_init(void);

/**
*****
@brief Initialize MODBUS protocol stack
@param none
@return error code
*****
*/
uint32_t modbus_init(void)
{
    uint32_t ercd;
    slave_map_init_t st_slave_map;

    /* Enable IP table */
    Modbus_tcp_init_ip_table (ENABLE, ACCEPT);

    /* register IP address */
    ercd = Modbus_tcp_add_ip_addr ("192.168.1.101");
    ercd = Modbus_tcp_add_ip_addr ("192.168.1.102");
    ercd = Modbus_tcp_add_ip_addr ("192.168.1.103");
    if (ercd != ERR_OK)
    {
        return ercd;
    }
}
```

Note), The number of clients is limited to 8.

6.3 Appendix C. Application Programming Interface

Function

· Modbus_tcp_init_ip_table

Description	Modbus set host IP list properties.		
Format	<pre>void Modbus_tcp_init_ip_table (ENABLE_FLAG e_flag, TABLE_MODE e_mode);</pre>		
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	<pre>uint32_t Modbus_tcp_add_ip_addr (pchar_t pu8_add_ip.);</pre>		
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 IP list is disabled.

· Modbus_tcp_delete_ip_addr

Description	Modbus delete an IP address to host IP list		
Format	<pre>uint32_t Modbus_tcp_delete_ip_addr (pchar_t pu8_del_ip)</pre>		
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 ERR_IP_ALREADY_PRESENT ERR_MAX_CLIENT ERR_TABLE_DISABLED		On success If address already present in list. If maximum connections reached. If IPlist is disabled.

· Modbus_slave_map_init

Description	Modbus function code mapping API		
Format	<pre>uint32_t Modbus_slave_map_init (p_slave_map_init_t pt_slave_func_tbl)</pre>		
Parameter	p_slavemap_init	pt_slave_func_tbl	Structure pointer to function code mapping table
Return value	Error code		
Error code	ERR_OK ERR_INVALID_STACK_INIT_PARAMS ERR_MEM_ALLOC		On success. If parameter is null. If memory allocation failed.

· Modbus_tcp_server_init_stack

Description	Modbus TCP stack initialization API		
Format	<pre>uint32_t Modbus_tcp_server_init_stack (uint32_t u32_additional_port, uint8_t u8_tcp_multiple_client)</pre>		
Parameter	Uuint32_t	u32_additonal_port	Additional port configured by user.
	Uuint8_t	u8_tcp_multiple_client	Status whether multiple clients are enabled.
Return value	Error code		
Error code	ERR_OK ERR_STACK_INIT		On successful initialization of the task or mailbox. 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 Register Read 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 [253] will be null
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 [253] will be null
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 aru8_data [253] will be null
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 aru8_data [253] will be null
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, if the exception code is nonzero the aru8_data [253] will be null
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 aru8_data [253] will be null
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, if the exception code is nonzero the aru8_data [253] will be null
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	<pre>uint32_t (fp_function_code1_t) (p_req_read_coils_t pt_req_read_coils, p_resp_read_coils_t pt_resp_read_coils.);</pre>		
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	<pre>uint32_t (fp_function_code2_t) (p_req_read_inputs_t pt_req_read_inputs, p_resp_read_inputs_t pt_resp_read_inputs.);</pre>		
Parameter	p_req_read_inputs_t	pt_req_read_inputs	structure pointer from stack to user with read discrete 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	<pre>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.);</pre>		
Parameter	p_req_read_holding_reg_t	pt_req_read_holding_reg	structure pointer from stack to user with read holding registers request information.
	p_resp_read_inputs_t	pt_resp_read_inputs	structure pointer to stack from user with 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 Register) operation.		
Format	<pre>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.);</pre>		
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	<pre>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.);</pre>		
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	<pre>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.);</pre>		
Parameter	p_req_write_single_reg_t	pt_req_write_single_reg	structure pointer from stack to user with write single register request information.
	p_resp_write_single_reg_t	pt_resp_write_single_reg	structure pointer to stack from user with 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	<pre>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.));</pre>		
Parameter	<code>p_req_write_multiple_coils_t</code>	<code>pt_req_write_multiple_coils</code>	structure pointer from stack to user with write multiple coils request information.
	<code>p_resp_write_multiple_coils_t</code>	<code>pt_resp_write_multiple_coils</code>	structure pointer to stack from user with 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	<pre>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.));</pre>		
Parameter	<code>p_req_write_multiple_reg_t</code>	<code>pt_req_write_multiple_reg</code>	structure pointer from stack to user with write multiple registers request information.
	<code>p_resp_write_multiple_reg_t</code>	<code>pt_resp_write_multiple_reg</code>	structure pointer to stack from user with write multiple registers response.
Return value	0: success 1: failure		

· **fp_function_code25**

Description	Callback function pointer for Modbus function code 25 (Read/Write Multiple Registers) operation.
Format	uint32_t (*fp_function_code25_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 multiple 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	IP list is disabled.
	ERR_TCP_IP_TABLE_IP_ALREADY_PRESENT	Address already presents in list.
	ERR_TCP_IP_TABLE_MAX_CLIENT	Maximum connections reached.
	etc.	Other error codes used in internal function.

7. Limitations

None

Revision History

Rev.	Date	Description	
		Page	Summary
1.00	Nov 30, 2023	-	First Edition issued
2.00	Aug 31, 2024	-	Modbus TCP updated to FSP2.0.0
2.20	Apr 4, 2025	-	Modbus TCP updated to FSP2.2.0
3.00	Nov 7, 2025	-	Supports RZ/T2 FSP3.0.0
4.00	Apr 3, 2026	-	Supports RZ FSP4.0.0

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.

<ul style="list-style-type: none">•Arm® and Cortex® are registered trademarks of Arm Limited (or its subsidiaries) in the EU and/or elsewhere. All rights reserved.•Ethernet is a registered trademark of Fuji Xerox Co. Ltd.•Modbus is a registered trademark of Schneider Electric, licensed to the Modbus Organization, Inc.•IEEE is a registered trademark of the Institute of Electrical and Electronics Engineers Inc•Additionally all product names and service names in this document are a trademark or a registered trademark which belongs to the 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/.