RZ/N2H Group

Modbus TCP Server Software

Introduction

This document explains the evaluation of Modbus communication using the RZ microcomputer evaluation board. Modbus protocol is a communication protocol developed by Modicon Inc. (Schneider Electric SA.) for programmable logic controllers (PLCs), and its specifications are open to the public.

For details, refer to the protocol specifications (PI-MBUS-300 Rev.J).

Target Device

RZ/N2H

Contents

1. Overview 3
1.1 Abbreviations / Definitions 3
1.2 Reference 3
1.2.1 About RZ/N2H 3
2. Features 4
3. Requirements (Software and Hardware) 5
3.1 Requirements for this sample package 5
3.2 Hardware 6
3.3 Setting the Board 11
4. Setup a demo tool 12
5. Running the sample application 13
5.1 Setup sample project for e^2 studio 13
5.1.1 Dual core project startup procedure (CR52_0, CR52_1), (CR52_0, CA55_2) 14
5.2 Setup sample project for EWARM 22
5.2.1 Dual-core project startup procedure (CR52_0, CR52_1), (CR52_0, CA55_2) 23
5.3 Demonstration 29
6. Appendix 31
6.1 Appendix A. DHCP mode (lwip) 31
6.2 Appendix B. Multi-client Configuration 32
6.3 Appendix C. Application Programming Interface 33
7. Limitations 48
Revision History 49

1. Overview

This document describes the procedure for testing the Modbus TCP server function using Modbus protocol stack code compatible with the Renesas RZ/N2H platform. Among the multiple cores installed, we will describe examples using Cortex®-R52 and Cortex®-A55.

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
8	Wireshark	A free packet capture tool for monitoring LAN traffic
9	Client	Modbus TCP Master / Requesting device
10	Server	Modbus TCP Slave / Responding device
11	MAC	Media Access Control Address
12	LAN	Local Area Network
13	DHCP	Dynamic Host Configuration Protocol
14	PHY	Physical Layer
15	LwIP	Lightweight IP
16	RS485	EIA-485 / Serial communication standard
17	SBOM	Software Bill of Materials

1.2 Reference

1.2.1 About RZ/N2H

Technical information regarding Modbus can be found on the official Modbus Organization website, while details on the RZ/N2H are provided by Renesas.

Table 1.2 Technical References for RZ/N2H

Document Type	Document Title	Description	Document No.
User's Manual	RZ/T2H and RZ/N2H Group User's Manual Hardware	RZ/T2H and RZ/N2H Group User's Manual Hardware	r01uh1039ej****
Datasheet	RZ/T2H and RZ/N2H Group Datasheet	RZ/T2H and RZ/N2H Group Datasheet	r01ds0426ej****
Getting Started	RZ/T2, RZ/N2 Group FSP Getting Started Guide	RZ/T2, RZ/N2 Group Flexible Software Package Getting Started Guide	r01an6434ej****
User's Manual	RZ/N2H Evaluation Board User's Manual	RZ/N2H Evaluation Board User's Manual	r20ut5522ej****
Reference Guide	Modicon Modbus Protocol Reference Guide Rev.J	Modbus Protocol Reference Guide Rev.J	PI-MBUS-300 Rev. J
Specification	Modbus Application Protocol Specification V1.1b3	Modbus Application Protocol Specification V1.1b3	v1.1b3

2. Features

The Modbus protocol stack for RZ/N2H allows for quick and easy development of the Modbus TCP server. The following nine Modbus function codes are supported by 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

If you use multiple ports, please use Eth0 and Eth1, and make sure IP address does not overlap with other devices.

For more information about Modbus, refer to the following site:
http://www.modbus.org

Note: The version number may differ depending on the update. Refer to the latest manual.

3. Requirements (Software and Hardware)

This project was developed and tested in the environment listed below using the following boards and tools.

3.1 Requirements for this sample package

Table 3.1 Requirements

Category	Name	Version	Description
Board	RZ/N2H Evaluation Board	-	RZ/N2H Evaluation Board Renesas Evaluation Board Kit for RZ/N2H
IDE	Flexible Software Package	V4.1.0	Renesas Flexible Software Package (FSP)
	e^2 studio - GNU ARM Embedded Toolchain - GNU ARM A-Profile (AArch64 bare-metal)	2026-04.1 13.3.1.arm-13-24 for CR52 13.2.1.20231009 for CA55	
	IAR Systems - Embedded Workbench for ARM - patch	9.60.3 EWARM_Patch_for_RZT2H_N2H_rev1.0	
	GNU ARM Embedded Toolchain GNU ARM A-Profile (AArch64 bare-metal)	13.3.1.arm-13-24 for CR52 13.2.1.20231009 for CA55	
Emulator	J-Link ™	8.60	SEGGER SEGGER - The Embedded Experts - Downloads - J-Link / J-Trace
Software	ModbusDemoApplication.exe	V1.3	

3.2 Hardware

This document describes the major hardware. Refer to Renesas Starter Kit+ for RZ/N2H user's manual and schematic for more board details.

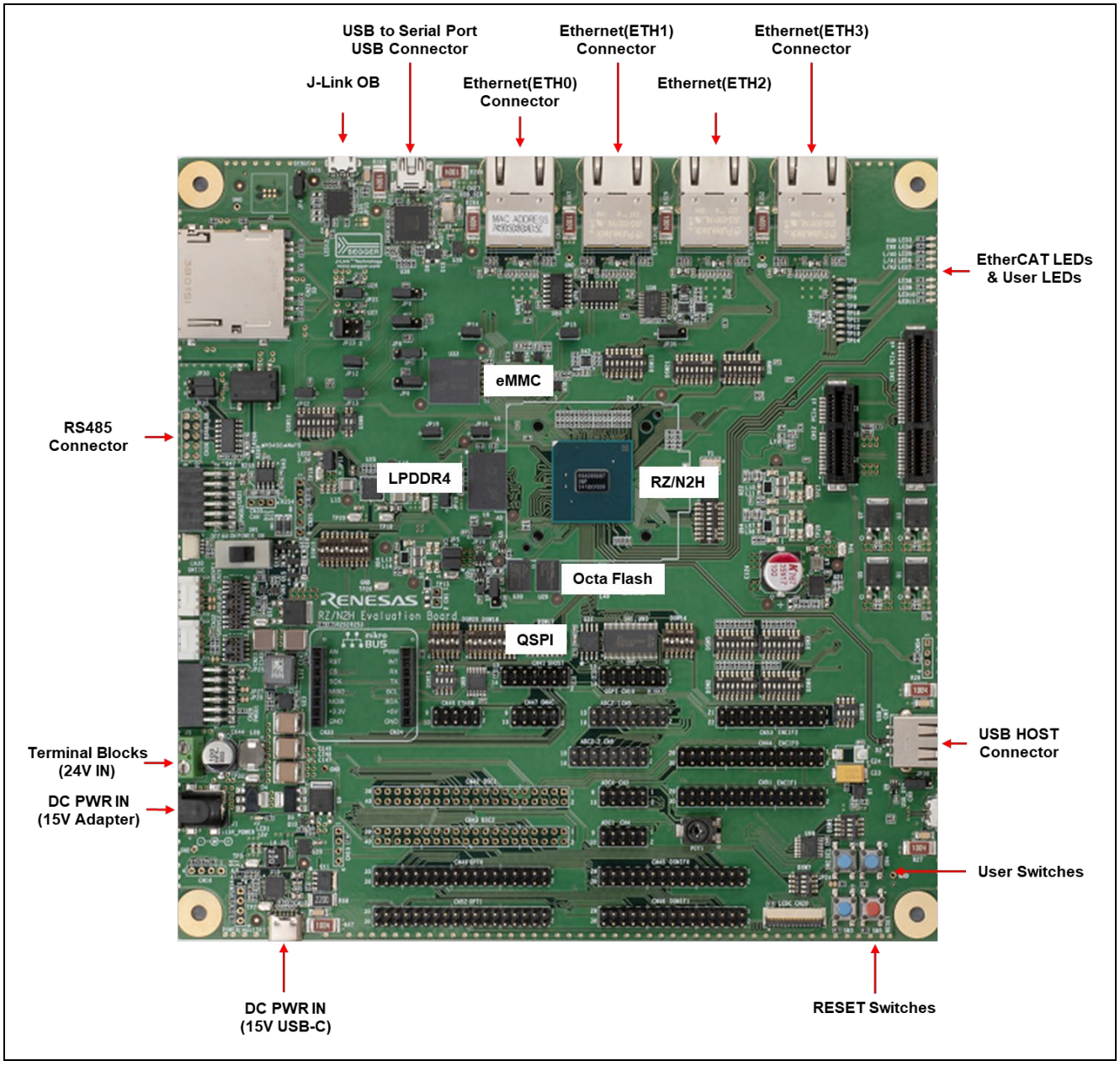


Figure 3.1 RZ/N2H Evaluation board layout.

Table 3.2 Jumper Pin settings (Power Selection)

Reference	Jumper Position	Description
JP6	Short 1-2	1.8-V power is supplied to VCC1833_0. (for Ethernet Port 0)
	Short 2-3	**3.3-V power is supplied to VCC1833_0. (for Ethernet Port 0)**
JP7	Short 1-2	1.8-V power is supplied to VCC1833_1. (for Ethernet Port 1)
	Short 2-3	**3.3-V power is supplied to VCC1833_1. (for Ethernet Port 1)**
JP8	Short 1-2	1.8-V power is supplied to VCC1833_1. (for Ethernet Port 2)
	Short 2-3	**3.3-V power is supplied to VCC1833_1. (for Ethernet Port 2)**
JP9	**Short 1-2**	**1.8-V power is supplied to VCC1833_1. (for Ethernet Port 3)**
	Short 2-3	3.3-V power is supplied to VCC1833_1. (for Ethernet Port 3)
JP21	Short 1-2	3.3-V power is supplied to VCC1833_7. (for SD1)
	Short 2-3	**Power control IC output for SD1 is supplied to VCC1833_7. (for SD1)**
JP23	Short 1-2	3.3-V power is supplied to VCC1833_6. (for SD0)
	Short 2-3	The output from the power-supply control IC for SD0 is supplied to VCC1833_6 (for SD0).
	Short 5-6	**1.8-V power is supplied to VCC1833_6. (for SD0)**
JP35	**Short 1-2**	**VCC1833_0 is supplied to VCC_ETH2_MDIO (DSW5-6 is ON: When P21_4 and P21_5 are selected for MDIO).**
	Short 2-3	VCC1833_2 is supplied to VCC_ETH2_MDIO. (DSW5-6 is OFF: When P30_5 and P30_6 are selected for MDIO)

Table 3.3 Jumper Pin settings (Debugging Function)

Reference	Jumper Position	Description
JP40	**Open - circuit**	**The on-board debugging function J-Link® OB is enabled.**
	Short-circuit	The on-board debugging function J-Link™ OB is disabled. Connect an external emulator to CN23 or CN24 at debugging.

Table 3.4 Jumper Pin settings (RS485 Interface Communication)

Reference	Jumper Position	Description
JP30	1-2 are short-circuit	Full-duplex communication
JP31	**2-3 are short-circuit**	**Half-duplex communication**

Table 3.5 Switch3 settings

SW3	Setting	Description
SW3-1	ON	**RAM Boot mode**
SW3-2	OFF	*** Substitute using "xSPI1 boot mode."**
SW3-3	ON	
SW3-4	OFF	**The number of ATCM wait cycles of CPU0 in the Cortex-R52 is 1 wait cycle.**
SW3-5	OFF	**The number of ATCM wait cycles of CPU1 in the Cortex-R52 is 1 wait cycle.**
SW3-6	OFF	**The power-supply voltage of the boot peripheral is 3.3 V.**
SW3-7	ON	**JTAG mode = Normal mode**
SW3-8	OFF	**(At the time of shipment = OFF)**

Table 3.6 Switch2 settings

SW2	Setting	Description
SW2-1	ON	XTALSEL = 'L' Select oscillator for RZ/N2H clock input.
SW2-2	OFF	(At the time of shipment = OFF)
SW2-3	ON	P27_2, P27_3, P27_6, and P31_3 are used as inputs to user DIP switches.
SW2-4	OFF	P13_4, P13_5, and P14_0 are used as RXD3, TXD3, and DE3 of the RS485.
SW2-5	ON	P00_0 to P00_2 are used as control signals for the USB power-supply IC. In this case, set DSW14-1, DSW14-3, and DSW14-5 to ON and DSW14-2, DSW14-4, and DSW14-6 to OFF.
SW2-6	ON	P01_0, P01_2, P01_4 to P01_7, and P02_0 to P02_3 are used as XSPI1 signals.
SW2-7	OFF	(At the time of shipment = OFF)
SW2-8	OFF	(At the time of shipment = OFF)

Table 3.7 Switch5 settings

SW5	Setting	Description
SW5-1	ON	P12_0 to 7, P13_0 to 2 are connected to eMMC.
SW5-2	ON	
SW5-3	ON	P08_6 and P17_4 are used as SD1 control signal.
SW5-4	OFF	(At the time of shipment = OFF)
SW5-5	OFF	(At the time of shipment = OFF)
SW5-6	ON	MDC and MDIO of Ethernet port 2 are connected to GMAC0 (P21_4 and P21_5).
SW5-7	ON	P29_1 to P29_7, P30_0 to P30_4, P30_7, P31_2, P31_4, and P31_5 are used as control signals for Ethernet Port 2.
SW5-8	ON	P00_0 to P00_2, P33_2 to P33_7, and P34_0 to P34_6 are used as control signals for Ethernet Port 3.

Table 3.8 Switch8 settings

SW8	Setting	Description
SW8-1	ON	P03_1_GMAC_RESETOUT2# is used as RESET for Ethernet Port 2.
SW8-2	OFF	

Table 3.9 Switch12 settings

SW12	Setting	Description
SW12-1	OFF	P00_3 is used as P00_3_ETH3_COL of Ethernet Port 3.
SW12-2	ON	
SW12-3	OFF	P11_0 is used as P11_0_ESC_RESETOUT# of Ethernet Port 0 and 1.
SW12-4	ON	
SW12-5	OFF	P03_2 is used as P03_2_GMAC_RESETOUT3# of Ethernet Port 3.
SW12-6	ON	
SW12-7	OFF	P03_1 is used as P03_1_GMAC_RESETOUT2# of Ethernet Port 2. In this case, set DSW8-1 to ON and DSW8-2 to OFF.
SW12-8	ON	

Table 3.10 Switch13 settings

SW13	Setting	Description
SW13-1	ON	P26_7 is used as P26_7_ETH1_RXER of Ethernet Port 1.
SW13-2	OFF	
SW13-3	ON	P27_0 is used as P27_0_ETH1_CRS of Ethernet Port 1.
SW13-4	OFF	
SW13-5	ON	P27_1 is used as P27_1_ETH1_COL of Ethernet Port 1.
SW13-6	OFF	
SW13-7	OFF	P13_7 is used as MDINT of Ethernet Port 2.
SW13-8	ON	

Table 3.11 Switch15 settings

SW15	Setting	Description
SW15-1	OFF	P22_6 is used as P22_6_SD0_WP.
SW15-2	ON	
SW15-3	OFF	P22_5 is used as P22_5_SD0_CD.
SW15-4	ON	
SW15-5	OFF	P14_7 is used as P14_7_USER_LED1(LED9).
SW15-6	ON	
SW15-7	OFF	- (At the time of shipment = OFF)
SW15-8	OFF	P14_6 is used as P14_6_USER_LED0(LED8).
SW15-9	OFF	
SW15-10	ON	

Table 3.12 Switch17 settings

SW17	Setting	Description
SW17-1	OFF	P03_0 is used as USER_LED3(LED11).
SW17-2	ON	
SW17-3	OFF	P02_7 is used as USER_LED2(LED10).
SW17-4	ON	
SW17-5	OFF	P02_6 is used as P02_6_SD0_IOVS.
SW17-6	ON	
SW17-7	OFF	- (At the time of shipment = OFF)
SW17-8	ON	P02_5 is used as P02_5_SD0_PWEN.

Table 3.13 Switch18 settings

SW18	Setting	Description
SW18-1	ON	**P22_7 is used as ESC_LINKACT0(LED5).**
SW18-2	OFF	
SW18-3	ON	**P23_0 is used as ESC_LINKACT1(LED6).**
SW18-4	OFF	
SW18-5	ON	**P14_3 is used as ESC_LINKACT2(LED7).**
SW18-6	OFF	
SW18-7	ON	**P31_6 is used as ESC_LEDRUN(LED3).**
SW18-8	OFF	
SW18-9	ON	**P18_1 is used as ESC_LEDERR(LED4).**
SW18-10	OFF	

3.3 Setting the Board

Setting the board for running sample program is shown below.
Build and run the sample code on the RZ/N2H Evaluation 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 debugger to the header "CN24" on the RZ/N2H Evaluation board.

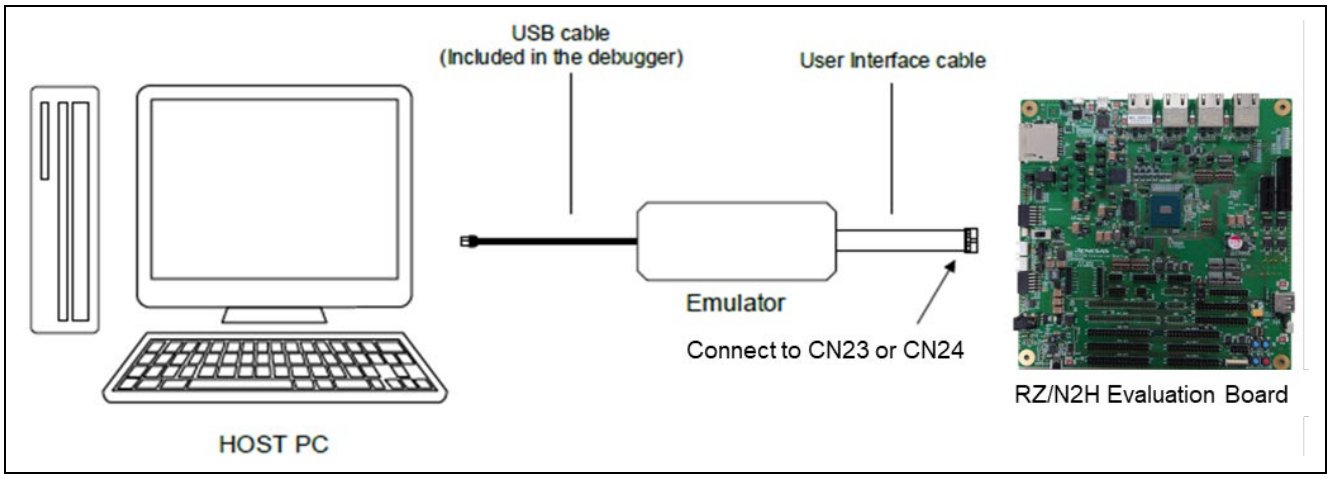


Figure 3.2 RZ/N2H Evaluation board debug connection diagram.

When using J-Link OB, connect the USB cable to the header "CN26" and set "JP40" to open.

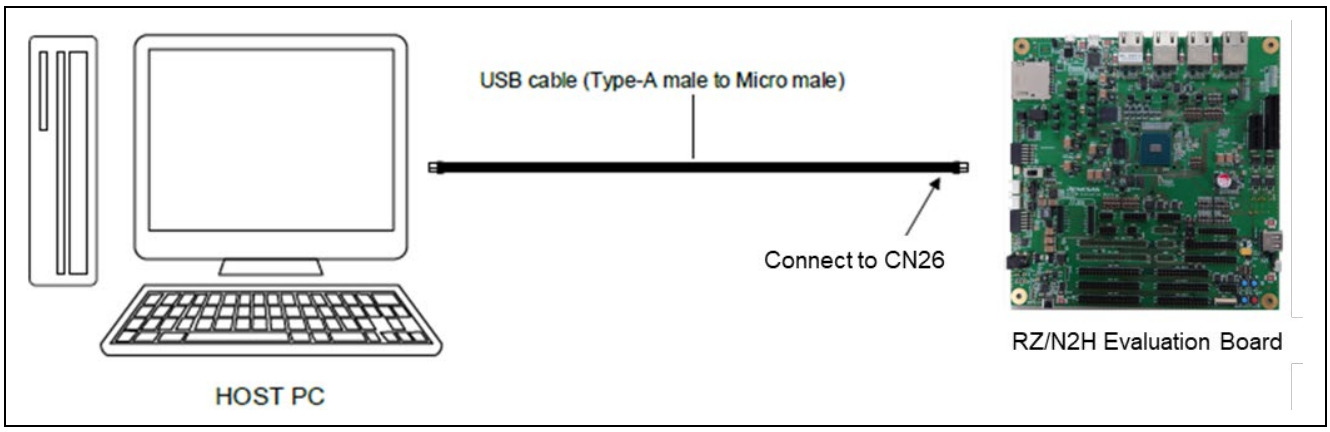


Figure 3.3 RZ/N2H Evaluation board debug connection diagram (J-Link OB)

2. When connecting to the AC/DC adapter, connect it to the connector "J1" on the RZ/N2H Evaluation board.

3. Connect Ethernet Cable to the Ethernet Connector “ETH0” or “ETH1”.

4. Setup a demo tool

1. Set the IP address of the client (PC) in advance.
   Set it to the default setting of "**192.168.1.101**".

2. Open “**ModbusDemoApplication.exe**” which is included in this package.

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

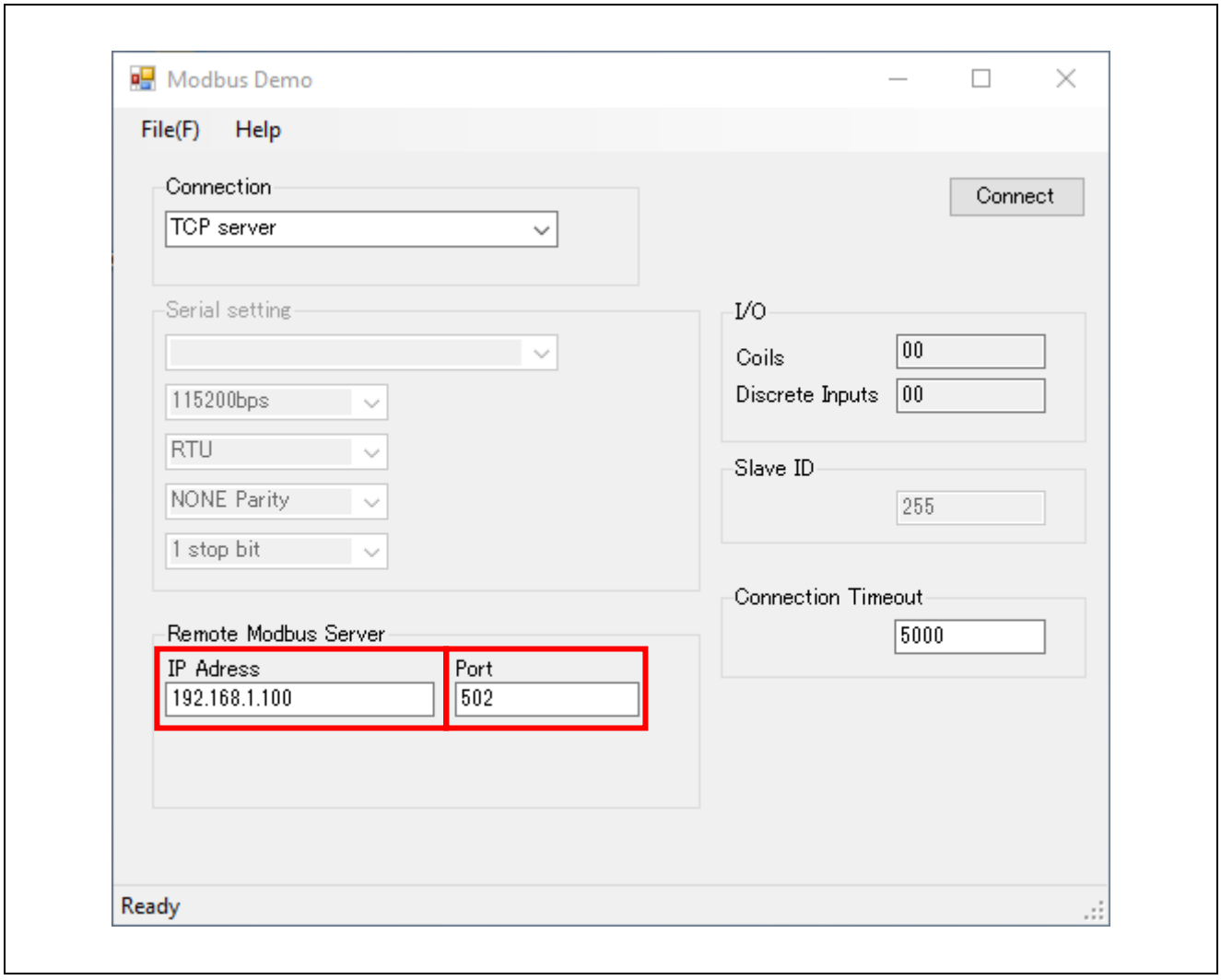


Figure 4.1 ModbusDemoApplication remote server setting.

5. Running the sample application

Refer to Section 3.3 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_v4.1.0 in advance.
Refer to the latest getting started guide. (r01an6434ejxxxx-rzt2-rzn2-fsp-getting-started.pdf)

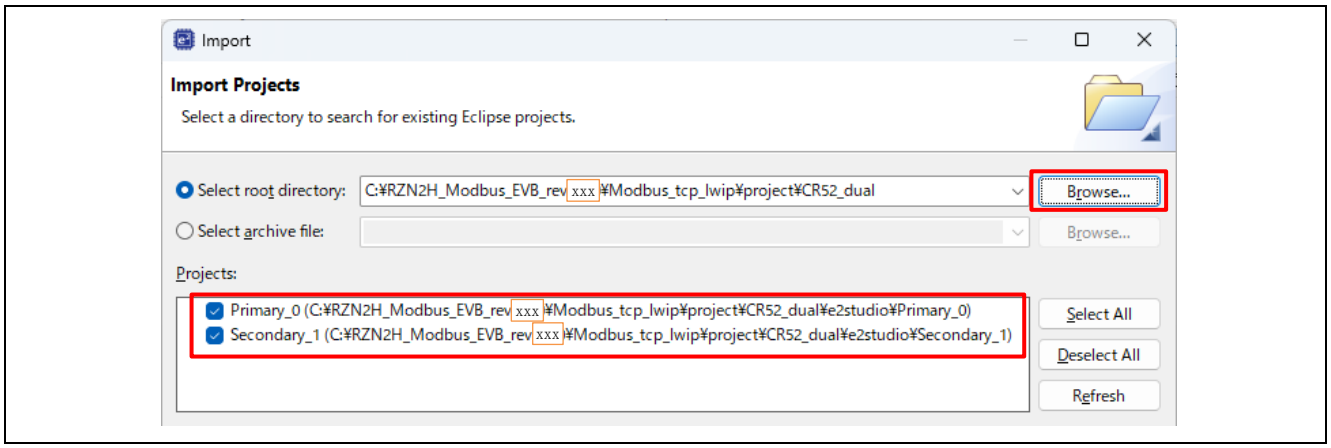
This package includes the following projects.

Project		CPU
Dual-core project (CR52_dual)	Primary_0	CR52_0
	Secondary_1	CR52_1
Dual-core project (CR52-CA55_dual)	Primary_0	CR52_0
	Secondary_1	CA55_2

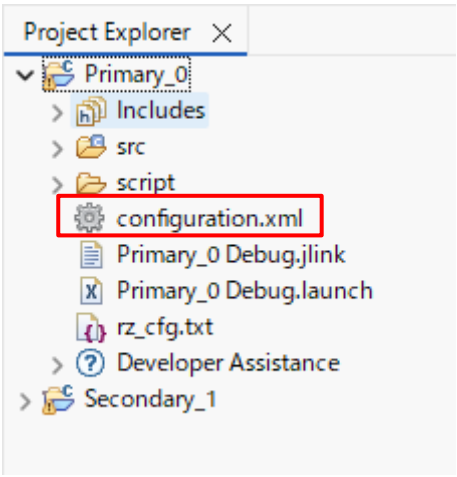
5.1.1 Dual core project startup procedure (CR52_0, CR52_1), (CR52_0, CA55_2)

1. Import the sample project. After the program is started, by selecting [File] → [Import] → [Existing Projects into Workspace]. Check the "select root directory" and click “Browse…” button.
   →Select “**RZN2H_Modbus_EVB_rev410\Modbus_tcp_lwip\project\xxxx_dual**” folder. →
   Check “**Primary_0**” and “**Secondary_1**” → [Finish].

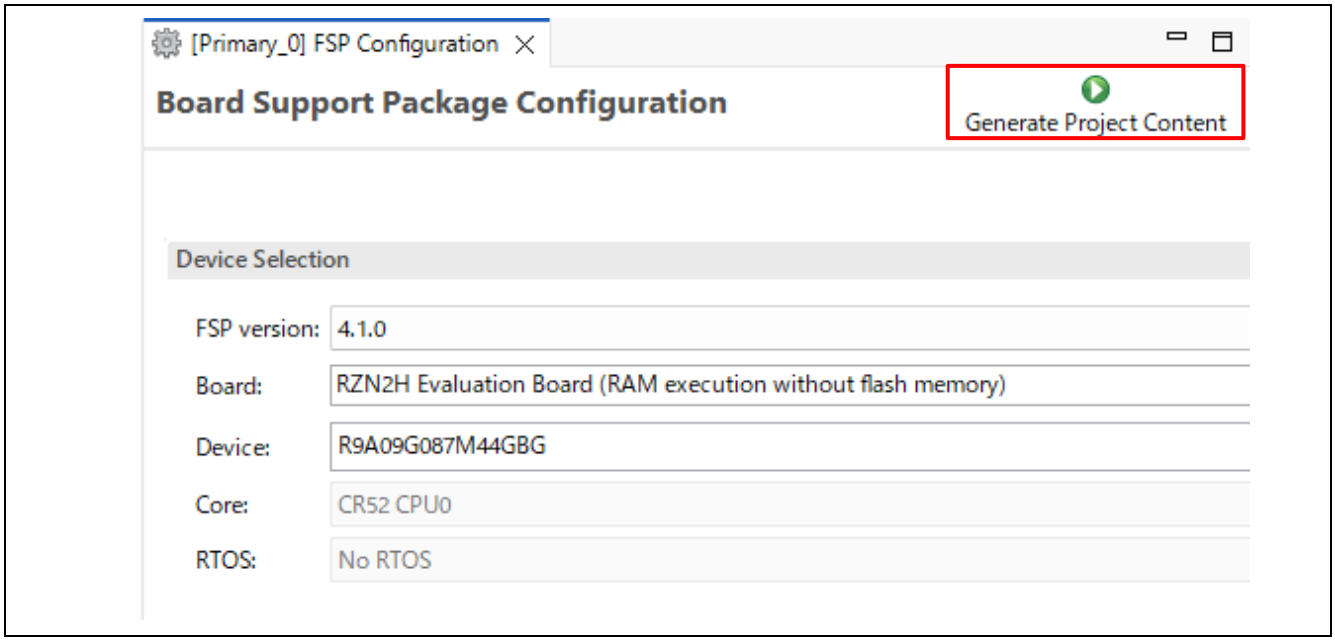
   Note), **xxxx** is either "**CR52**" or "**CR52-CA55**", select one.



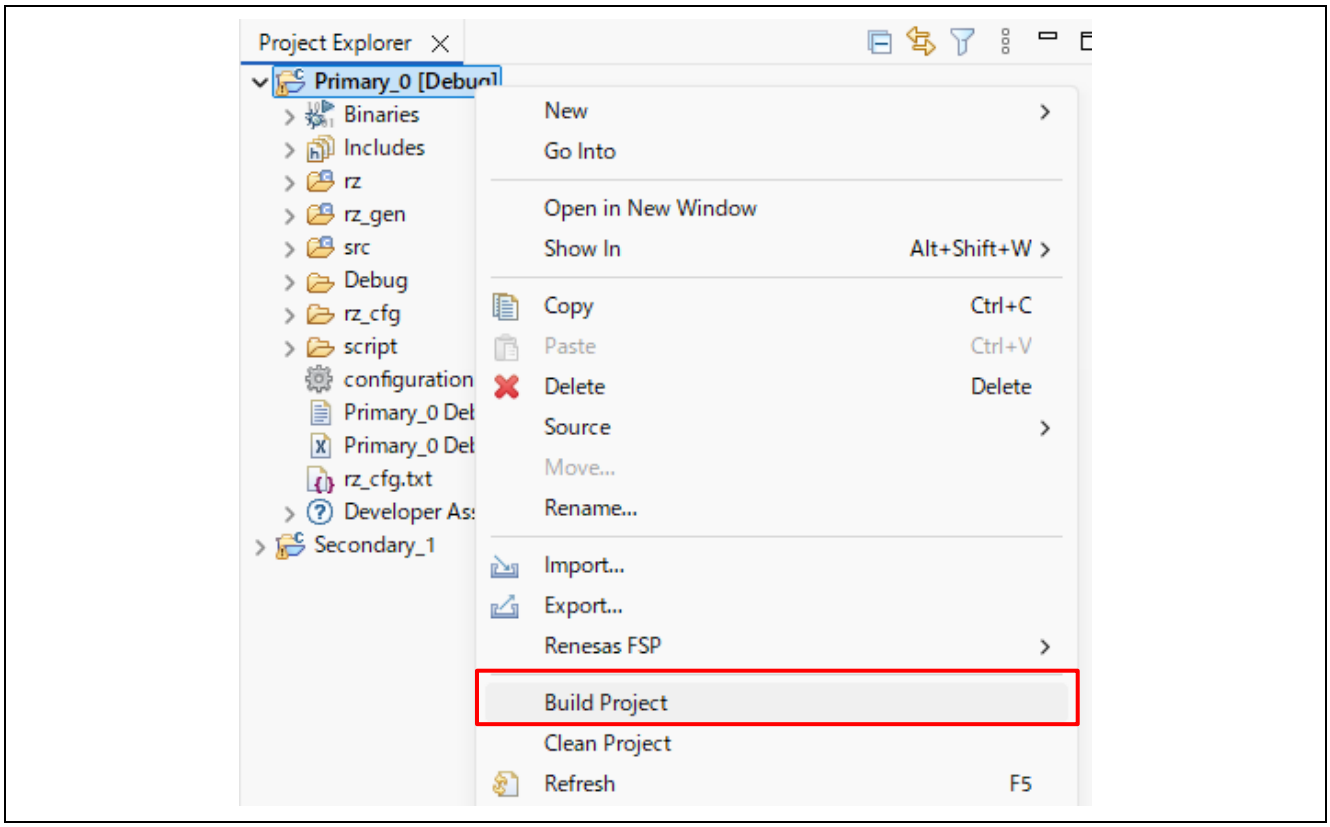
2. Open "configuration.xml" in the "**Primary_0**" project



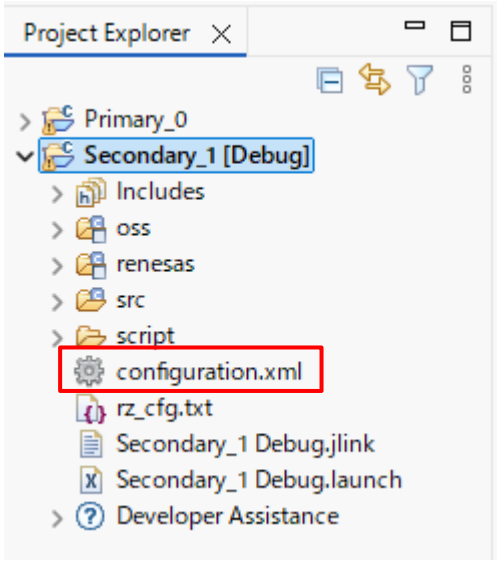
3. Check the boot mode project from BSP tab (following is RAM boot mode). And generate the code with "**Generate Project Content**".



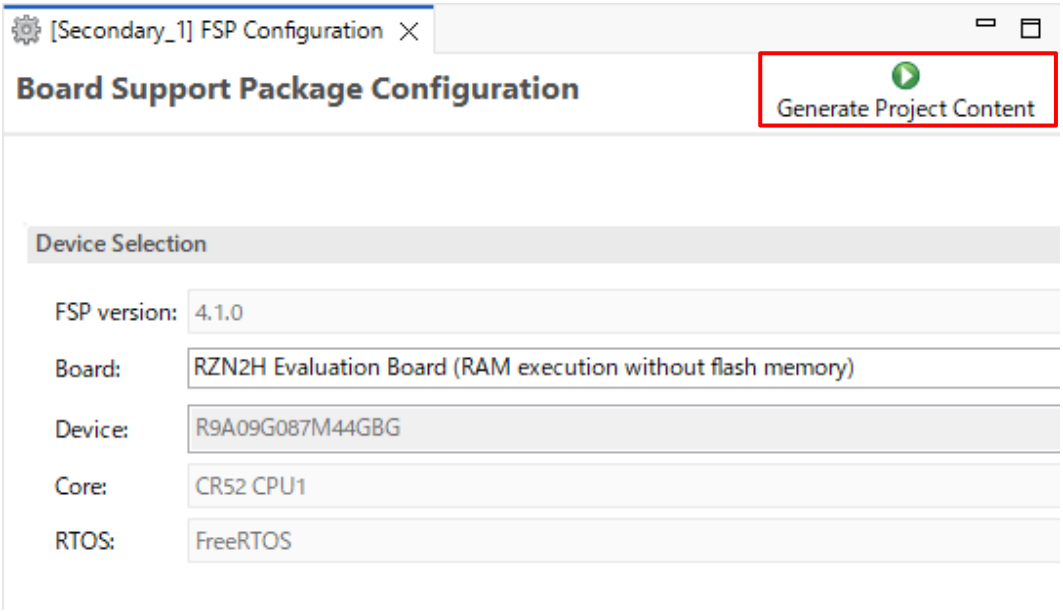
4. Select the "**Primary_0**" project and execute the build.



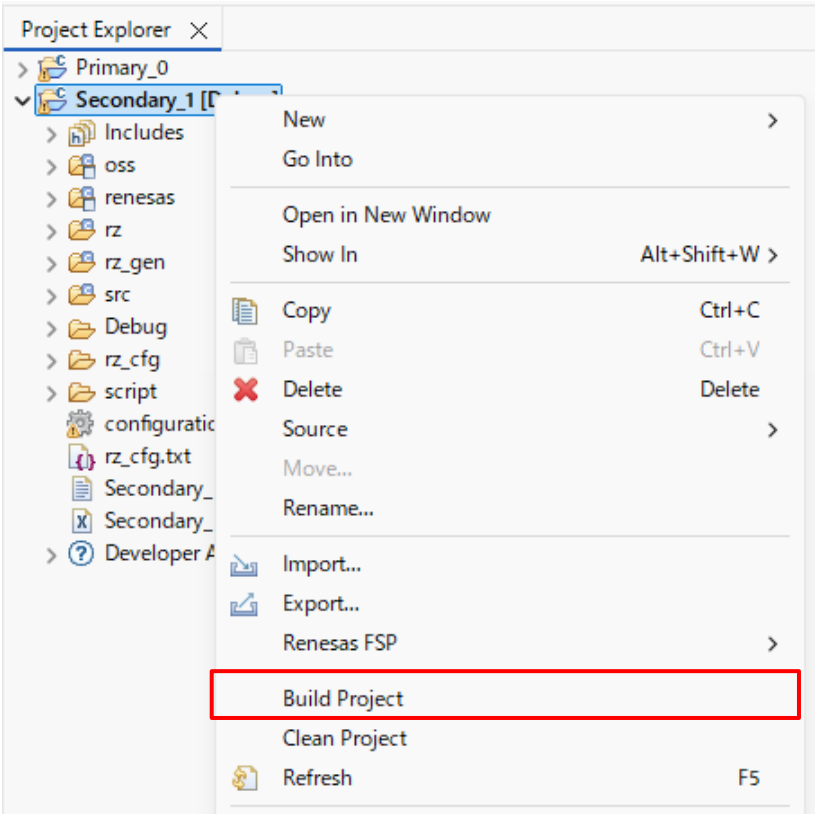
5. Open "configuration.xml" in the "**Secondary_1**" project.



6. Check the boot mode project from BSP tab (following is RAM boot mode). And generate the code with "**Generate Project Content**".

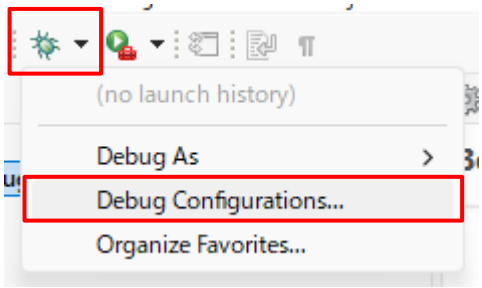


7. Select the “**Secondary_1**” project and execute the build.

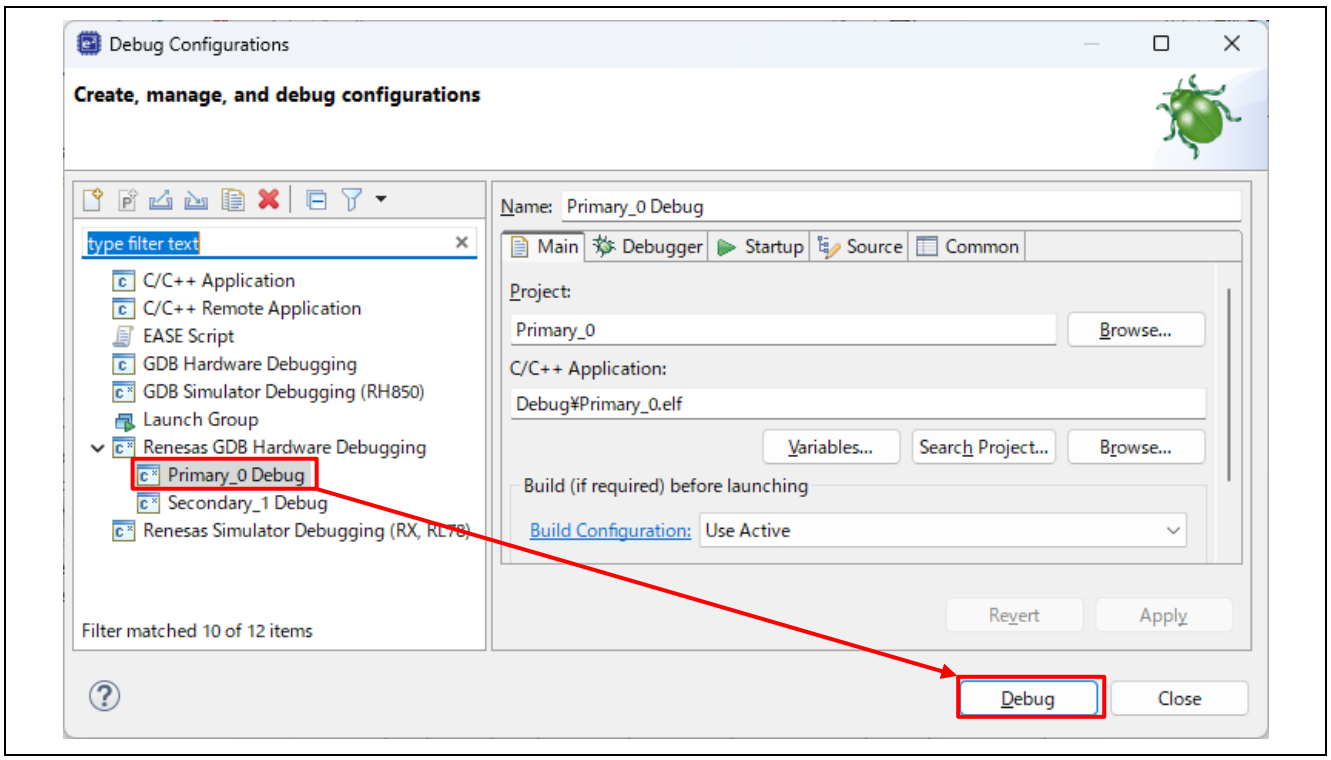


8. Press the “**RESET**” switch of the RSK + RZN2H board.

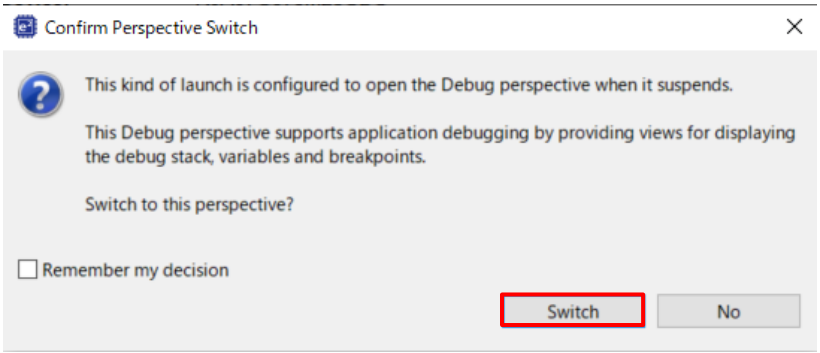
9. After connecting the board and J-Link, start debugging in the following procedure. Select the drop-down menu next to the bug icon and selecting “**Debug Configurations…**”.



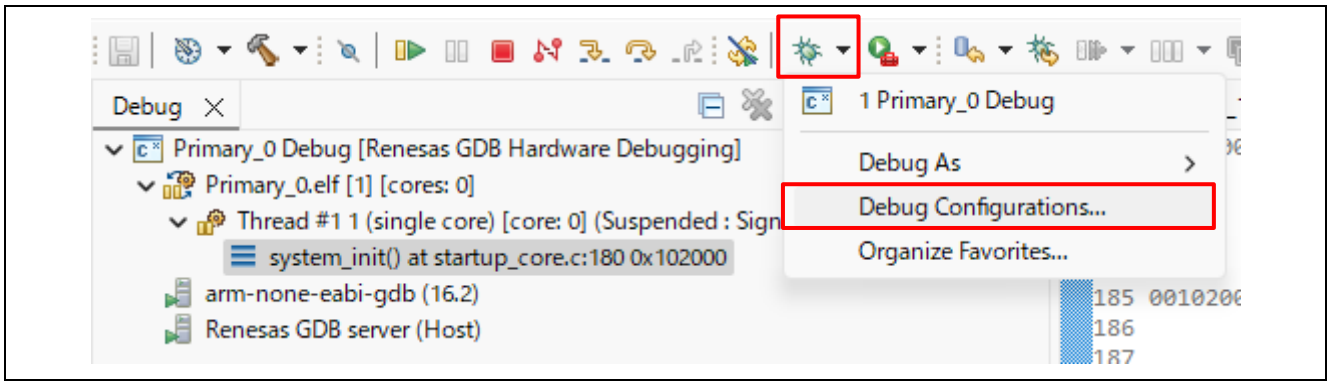
[Renesas GDB Hardware Debugging] → [**Primary_0 Debug**] item, then press [Debug].



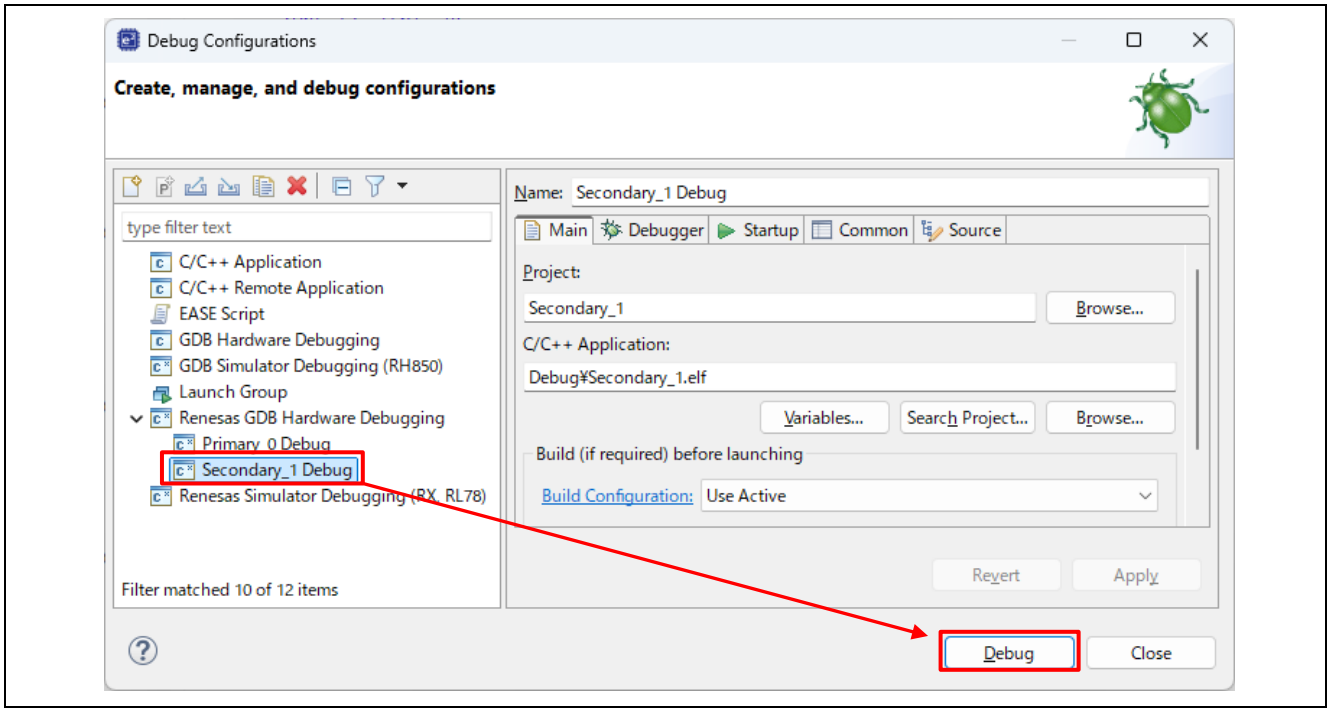
Following dialog will appear, so switch to the debug screen.



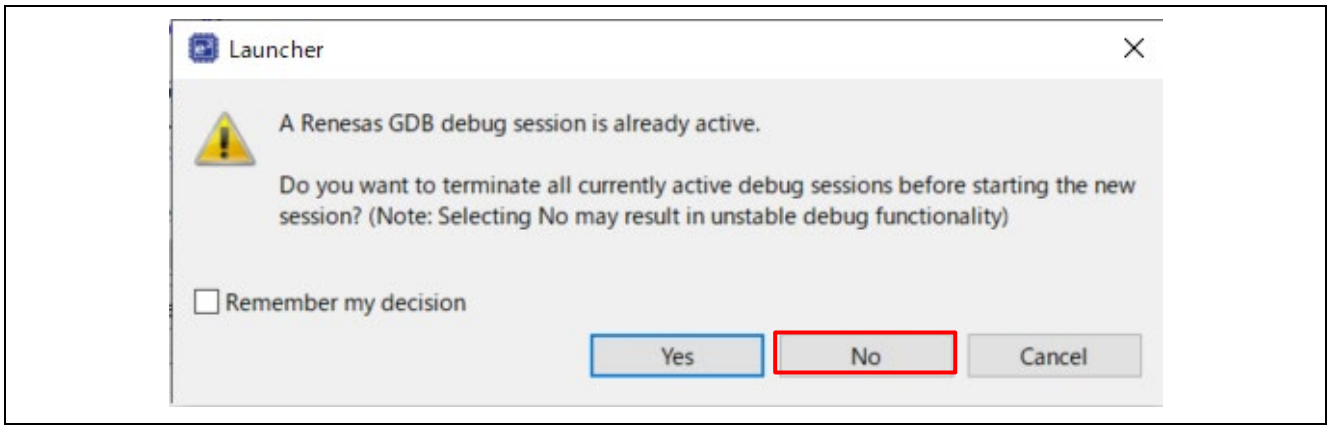
10. Select the drop-down menu next to the bug icon and selecting “**Debug Configurations…**”.



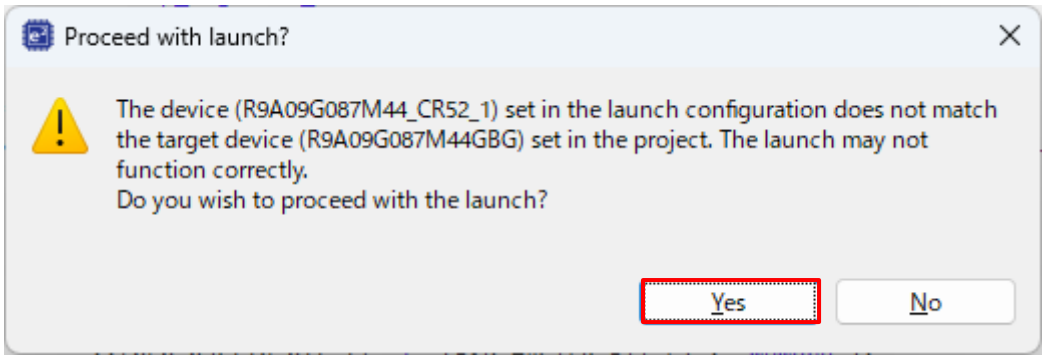
[Renesas GDB Hardware Debugging] → [**Secondary_1 Debug**] item, then press [Debug]



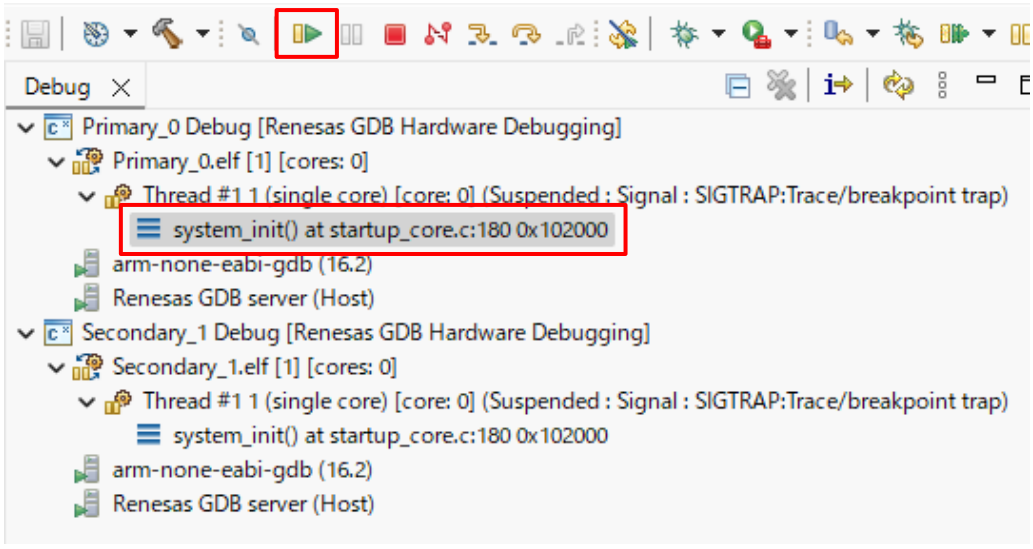
11. Message will appear asking if you want to close all currently active programs. Select "**No.**"

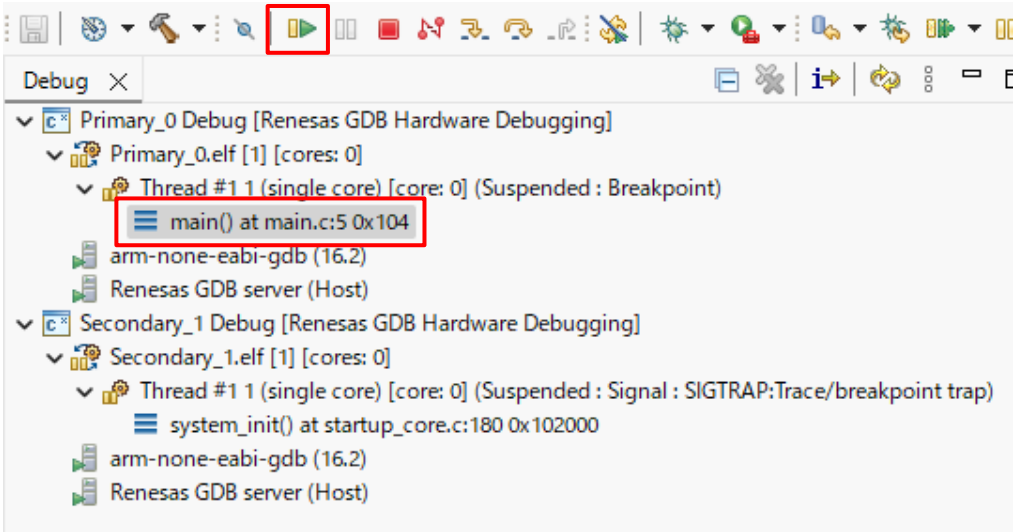


12. Message will appear asking if you want to continue launching, select "**Yes**".

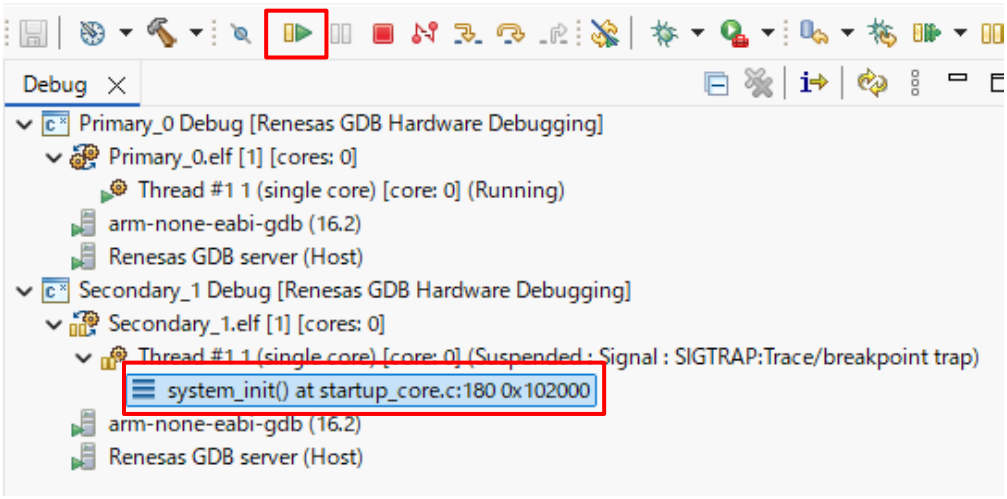


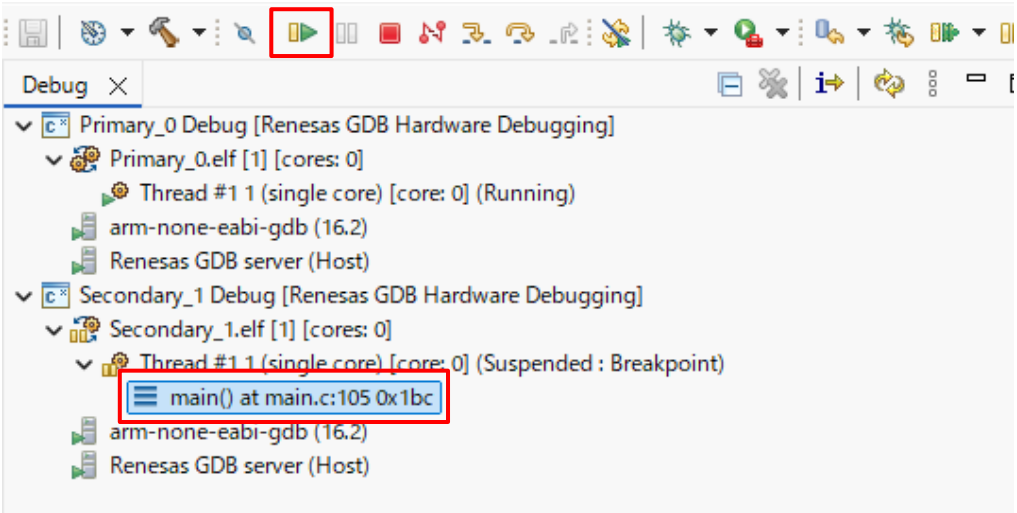
13. Press the "**Resume**" button. "**Primary_0**" debug will start, and the program will be suspended in main.c. Press the "**Resume**" button again. Program will run.



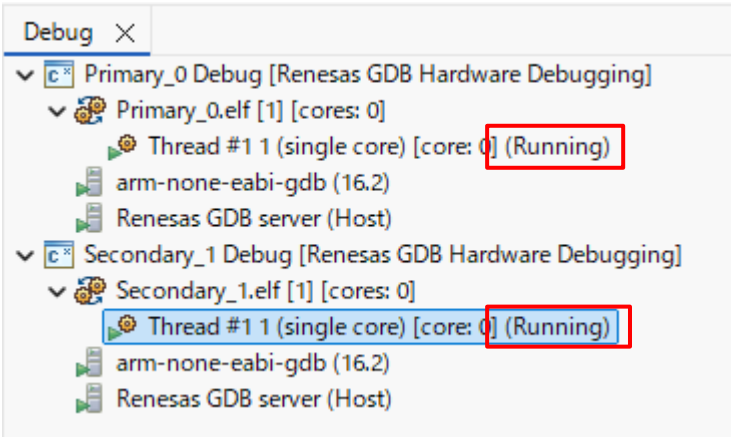


14. Press the "**Resume**" button. "**Secondary_1**" debug will start, and the program will be suspended in main.c. Press the "**Resume**" button again. Program will run.





15. Finally, "**Primary_0**" and "**Secondary_1**" cores will start up and the program will run.



5.2 Setup sample project for EWARM

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

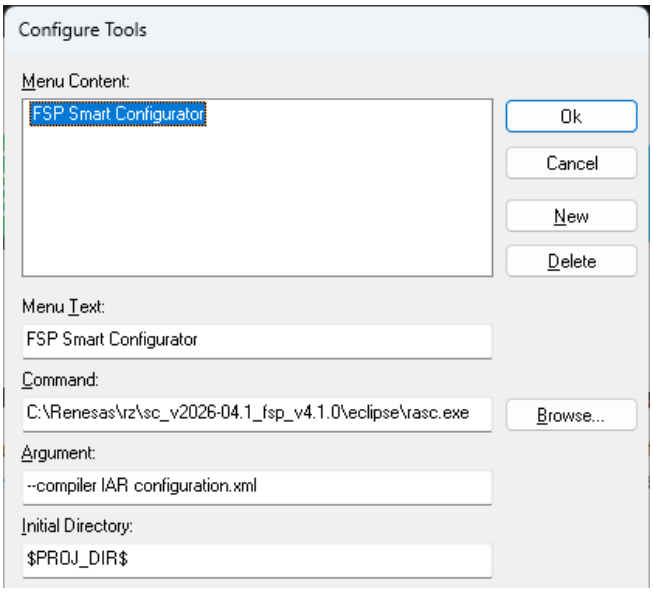
Note). Please install EWARM and adapt the RZ_FSP_Packs_v4.1.0 in advance.
Refer to the latest getting started guide. (r01an6434ejxxxx-rzt2-rzn2-fsp-getting-started.pdf)

This package includes the following projects.

Project		CPU
Dual-core project (CR52_dual)	Primary_0	CR52_0
	Secondary_1	CR52_1
Dual-core project (CR52-CA55_dual)	Primary_0	CR52_0
	Secondary_1	CA55_2

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_v2026-04.1_fsp_v4.1.0\eclipse\rasc.exe")
- Argument: **--compiler IAR configuration.xml**
- Initial Directory: $**PROJ_DIR$**



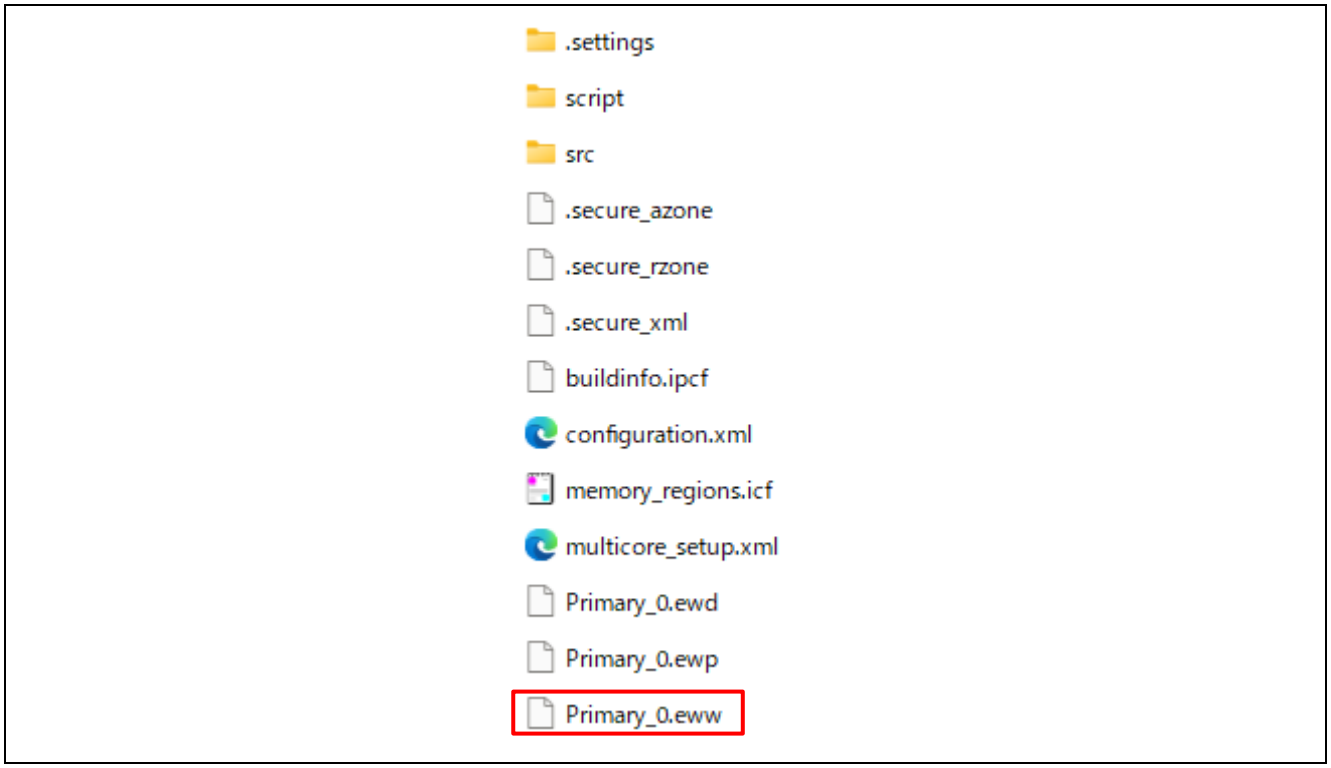
Edit "**buildinfo.ipcf**" file. Open the buildinfo.ipcf file and edit "**RASC_EXE_PATH**" to match the installation path of "**rasc.exe**".

```
<customArgVars>
    <group name="RA Smart Configurator">
        <argVar>
            <name>RASC_EXE_PATH</name>
            <value>C:¥Renesas¥rz¥sc_v2026-04.1_fsp_v4.1.0¥eclipse¥rasc.exe</value>
        </argVar>
        <argVar>
            <name>RASC_DEVICE_FAMILY</name>
            <value>rz</value>
        </argVar>
```

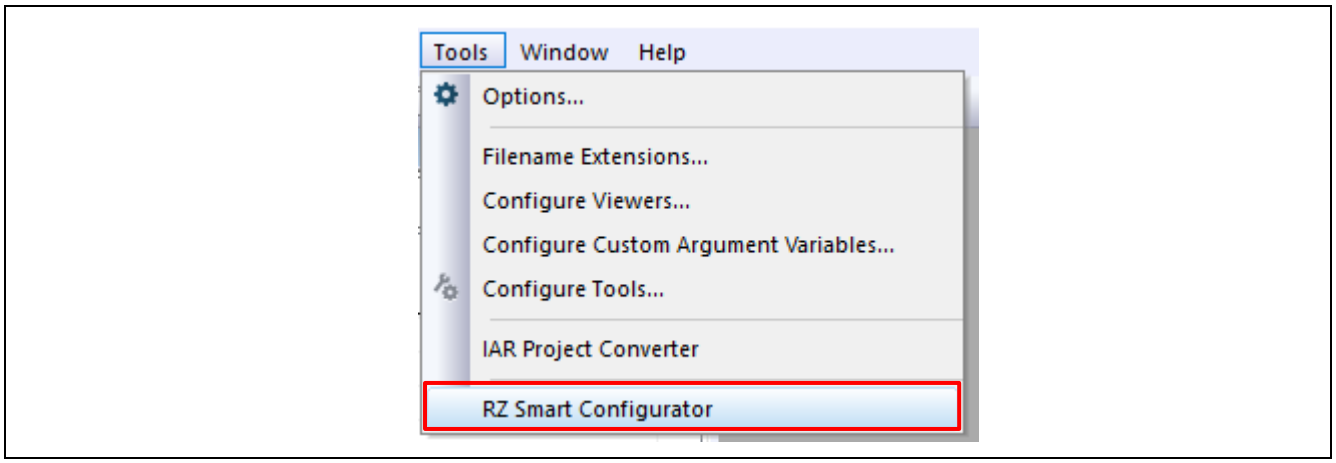
5.2.1 Dual-core project startup procedure (CR52_0, CR52_1), (CR52_0, CA55_2)

1. Open the sample project.
   "**RZN2H_Modbus_EVB_rev410\Modbus_tcp_lwip\project\xxxx_dual\ewarm\Primary_0\Primary_0.eww**"

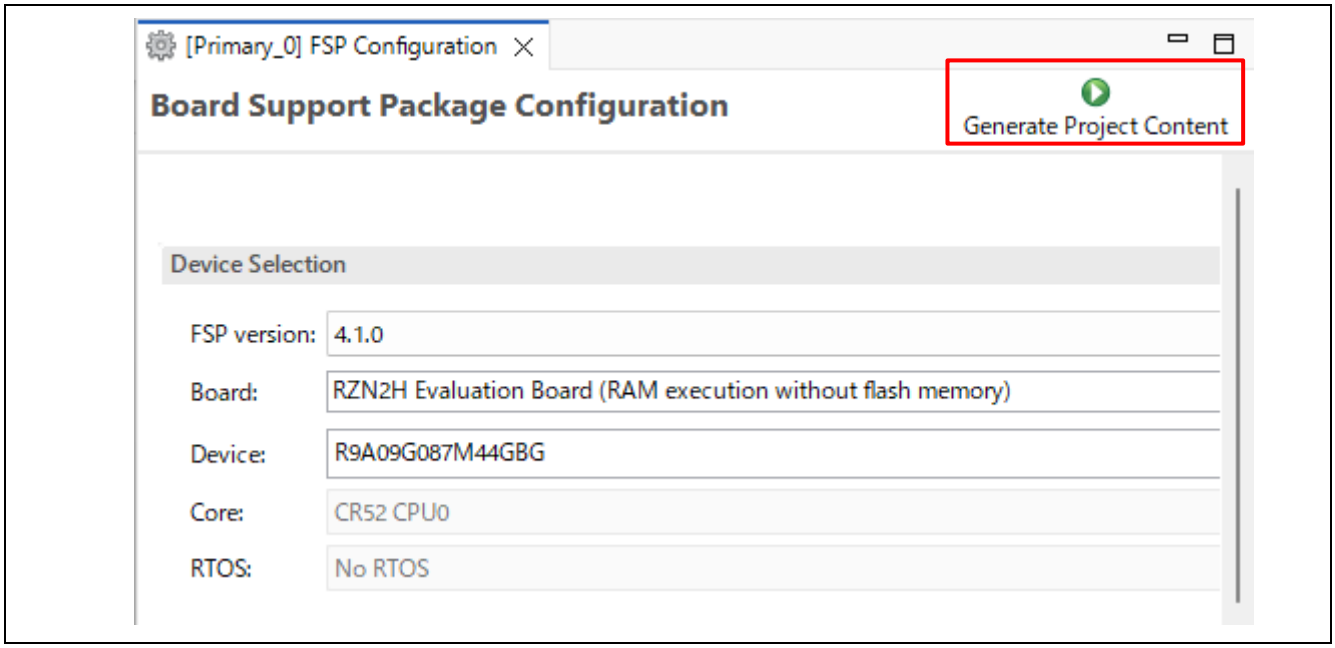
   Note), **xxxx** is either "**CR52**" or "**CR52-CA55**", select one.



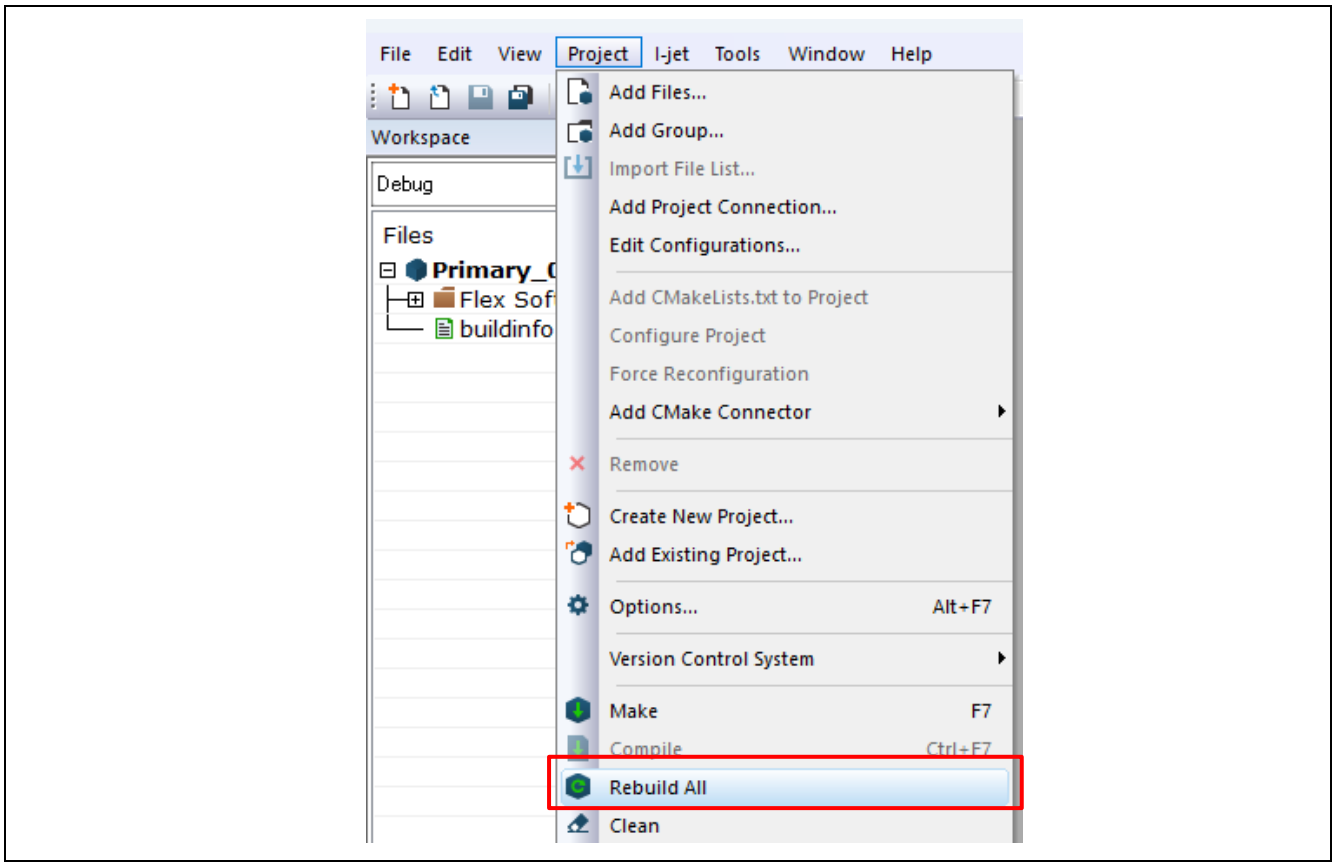
2. Open the "RZ Smart Configurator"



3. Check the boot mode project from BSP tab (following is RAM boot mode). And generate the code with "**Generate Project Content**".



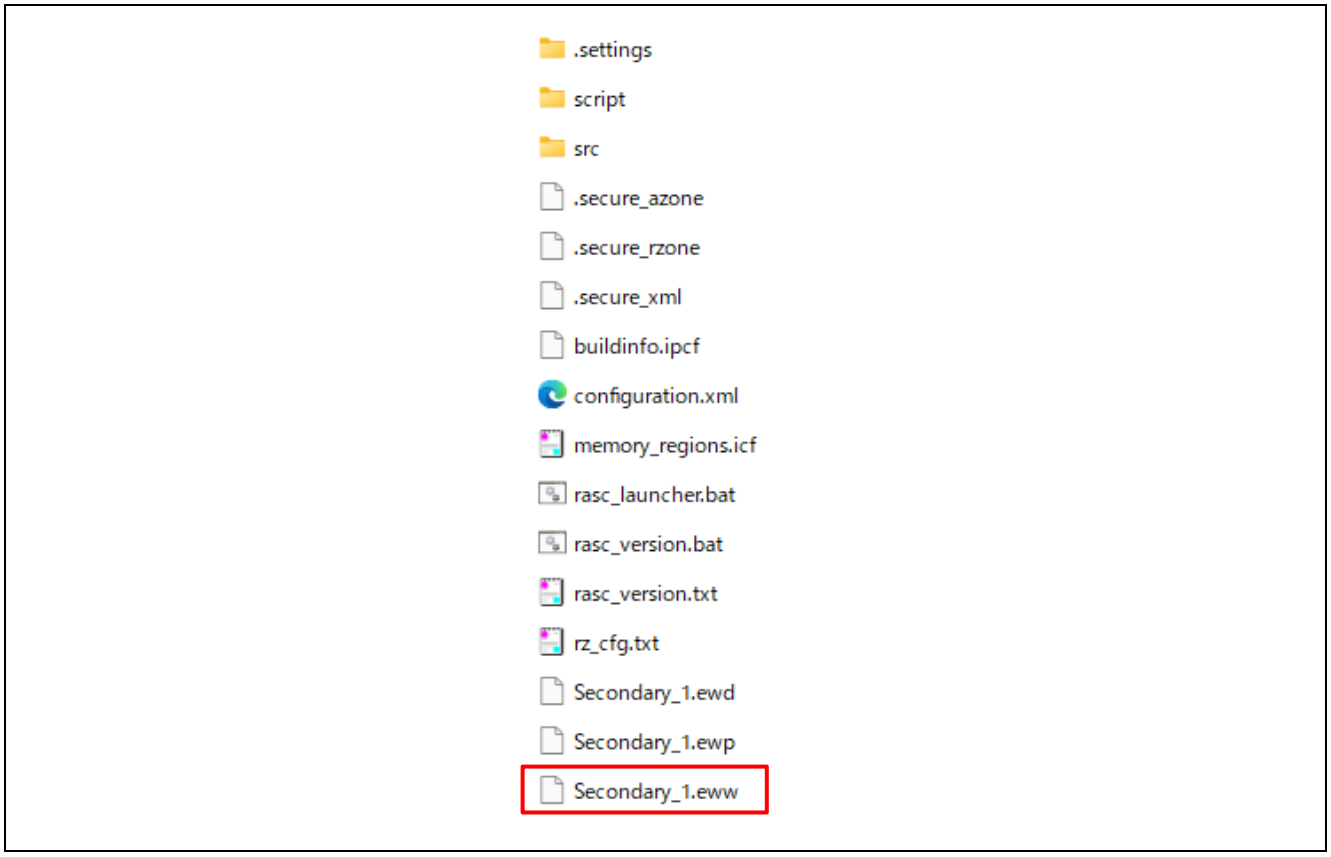
4. Select the "**Rebuild All**" item from the "Project" menu to rebuild the project.



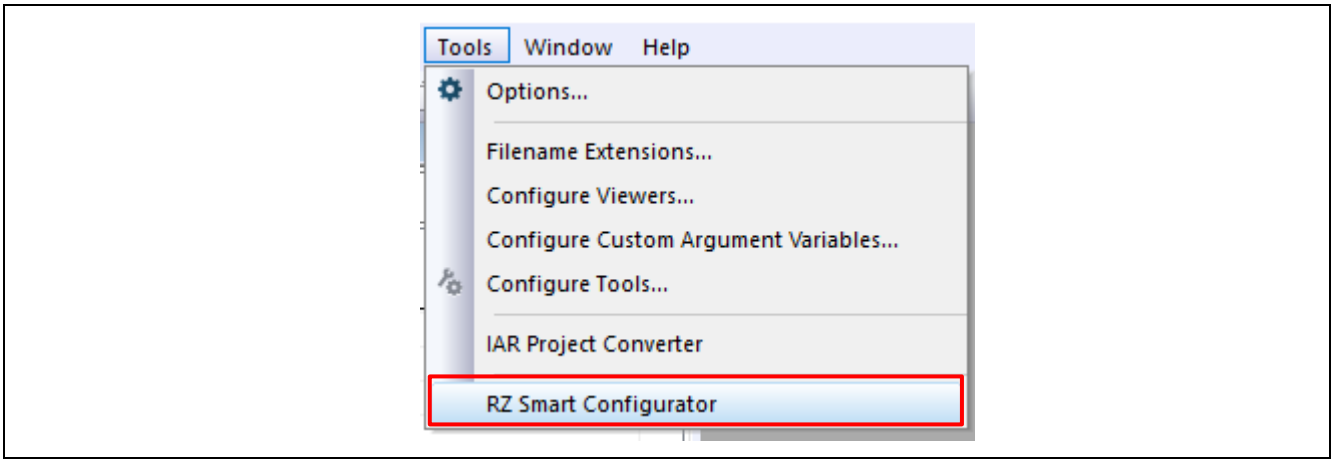
5. When the building is completed, close the "**Primary_0**" project.

6. Open the sample project.
   “**RZN2H_Modbus_EVB_rev410\Modbus_tcp_lwip\project\xxxx_dual\ewarm\Secondary_1\Secondary_1.eww**”

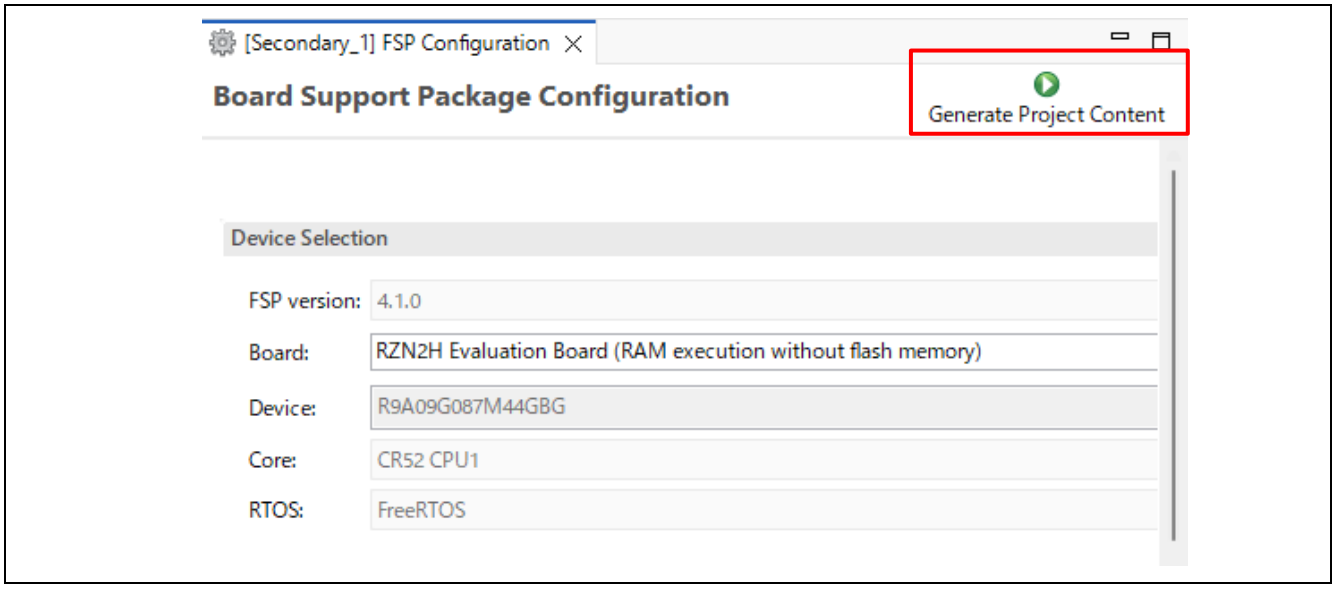
   Note), **xxxx** is either "**CR52**" or "**CR52-CA55**", select one.



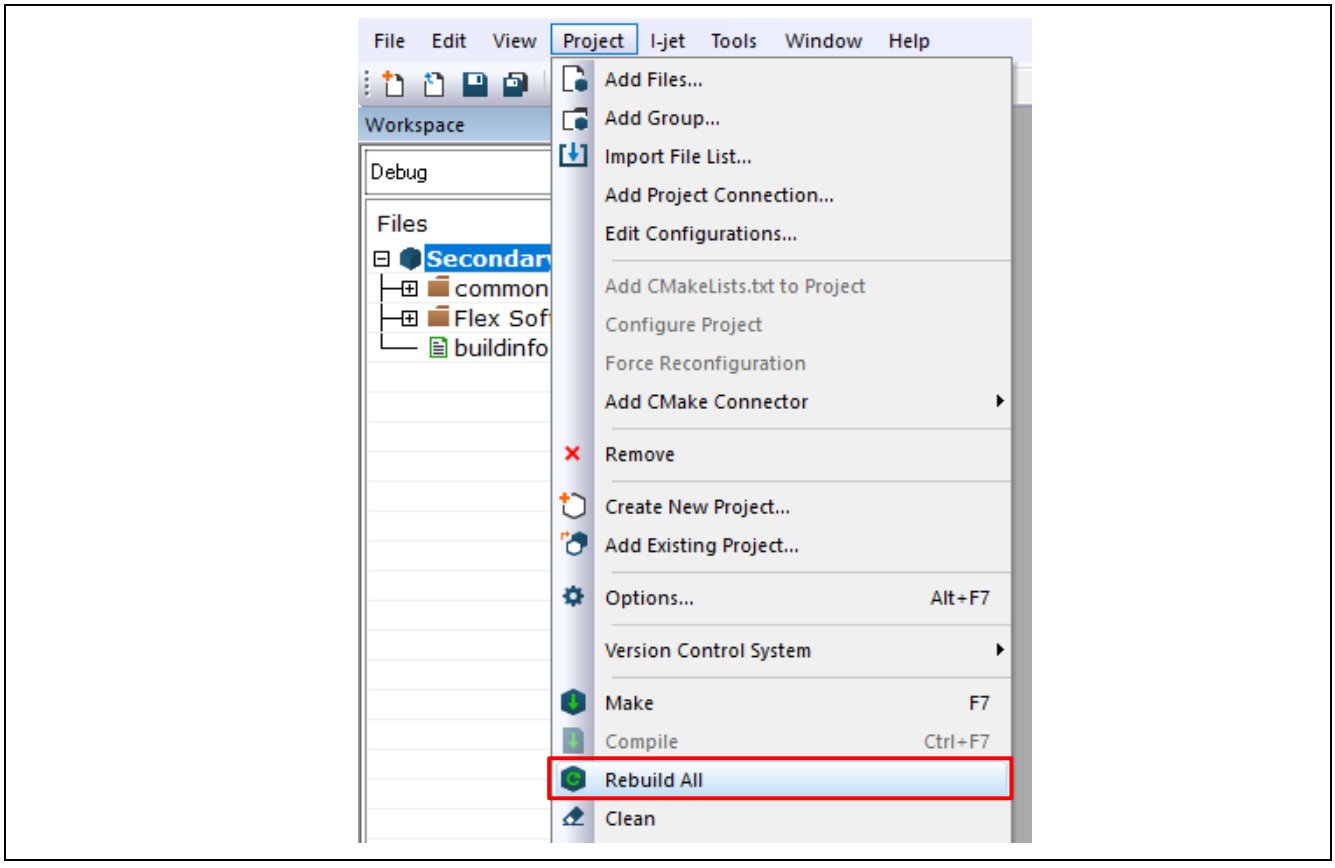
7. Open the “RZ Smart Configurator”



8. Check the boot mode project from BSP tab (following is RAM boot mode). And generate the code with "**Generate Project Content**".



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

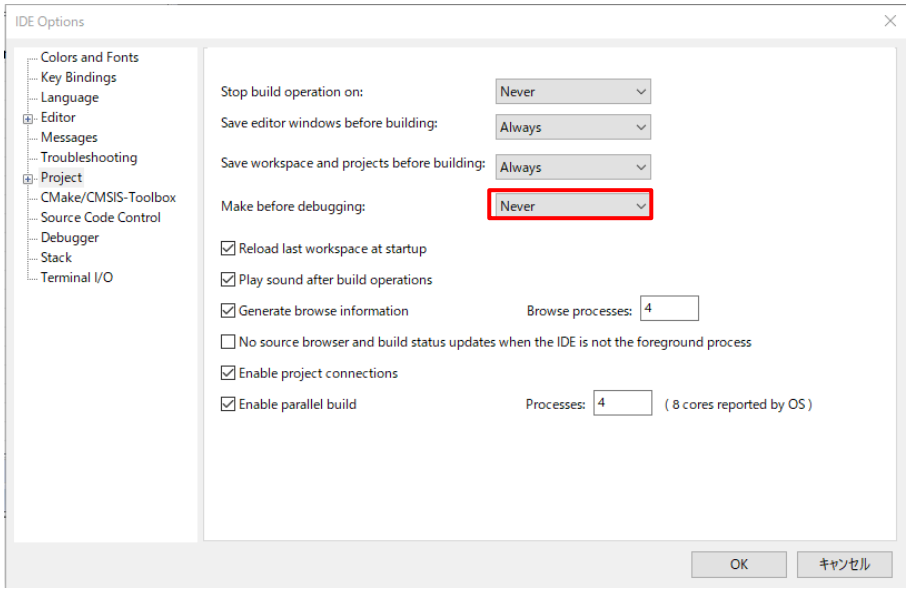


10. When the building is completed, close the "**Secondary_1**" project.

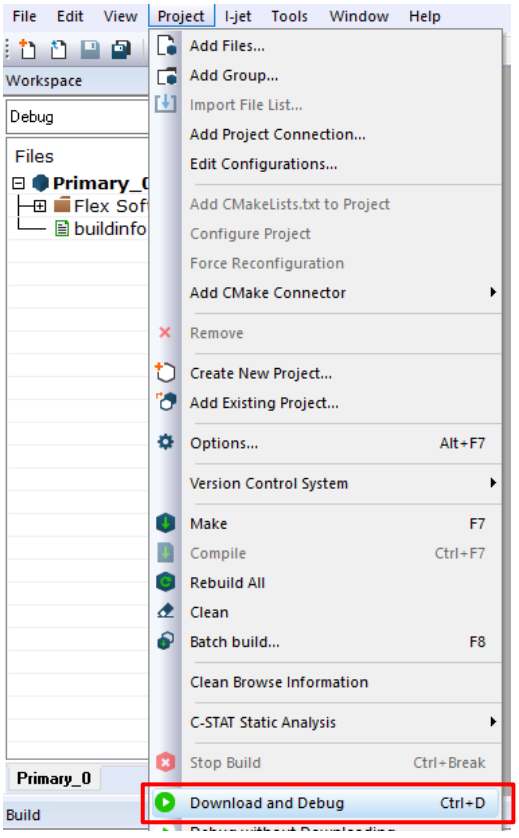
11. Press the "**RESET**" switch of the RSK + RZN2H board.

12. Start the “**Primary_0**” project again.

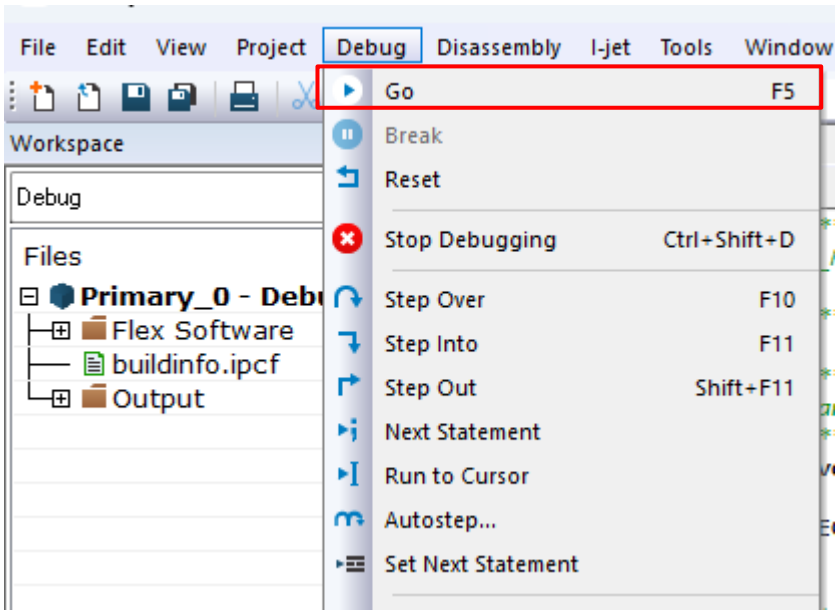
13. Check your debugger options settings.
[Tools] → [Options] → [Project] item, Set "Make before debugging" to "Never".



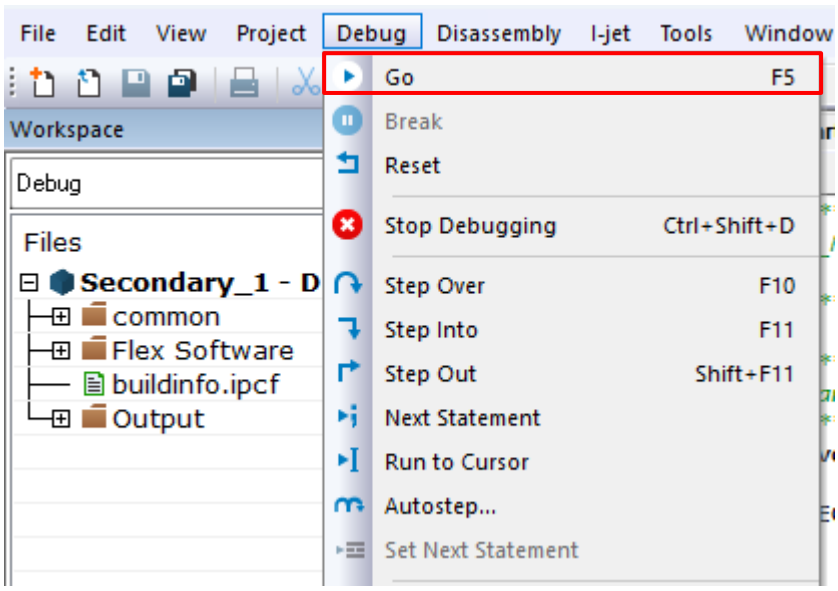
14. After connecting the board and I-jet, click the “Download and Debug button” on the Project toolbar.
When run “**Download and Debug**”, the “**Secondary_1**” debug screen will also launch at the same time.



15. First, press the "**Resume**" button for the “**Primary_0**” project. Program will run, and “**Secondary_1**” project will open.



16. Next, press the "**Resume**" button for the “**Secondary_1**” project. The program will run.

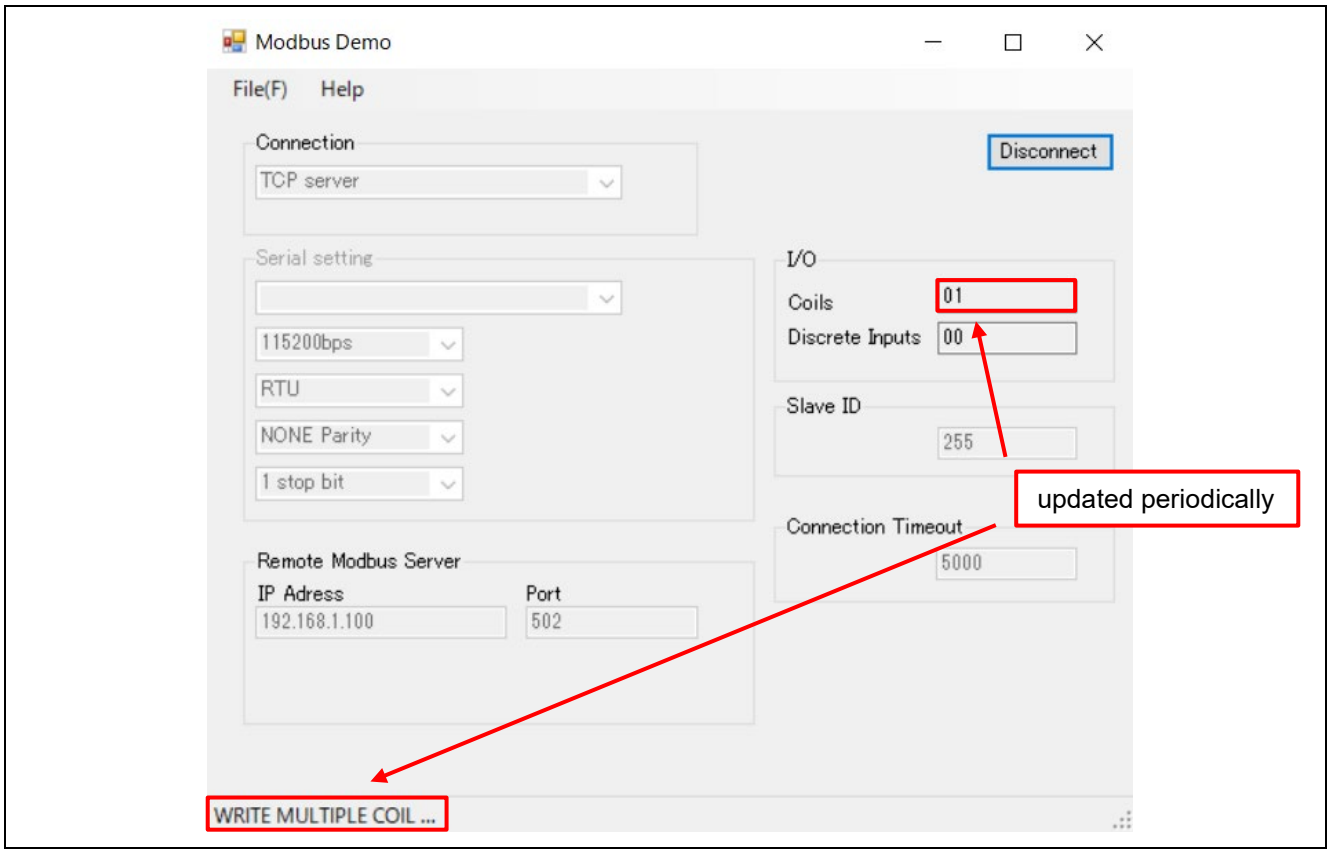


5.3 Demonstration

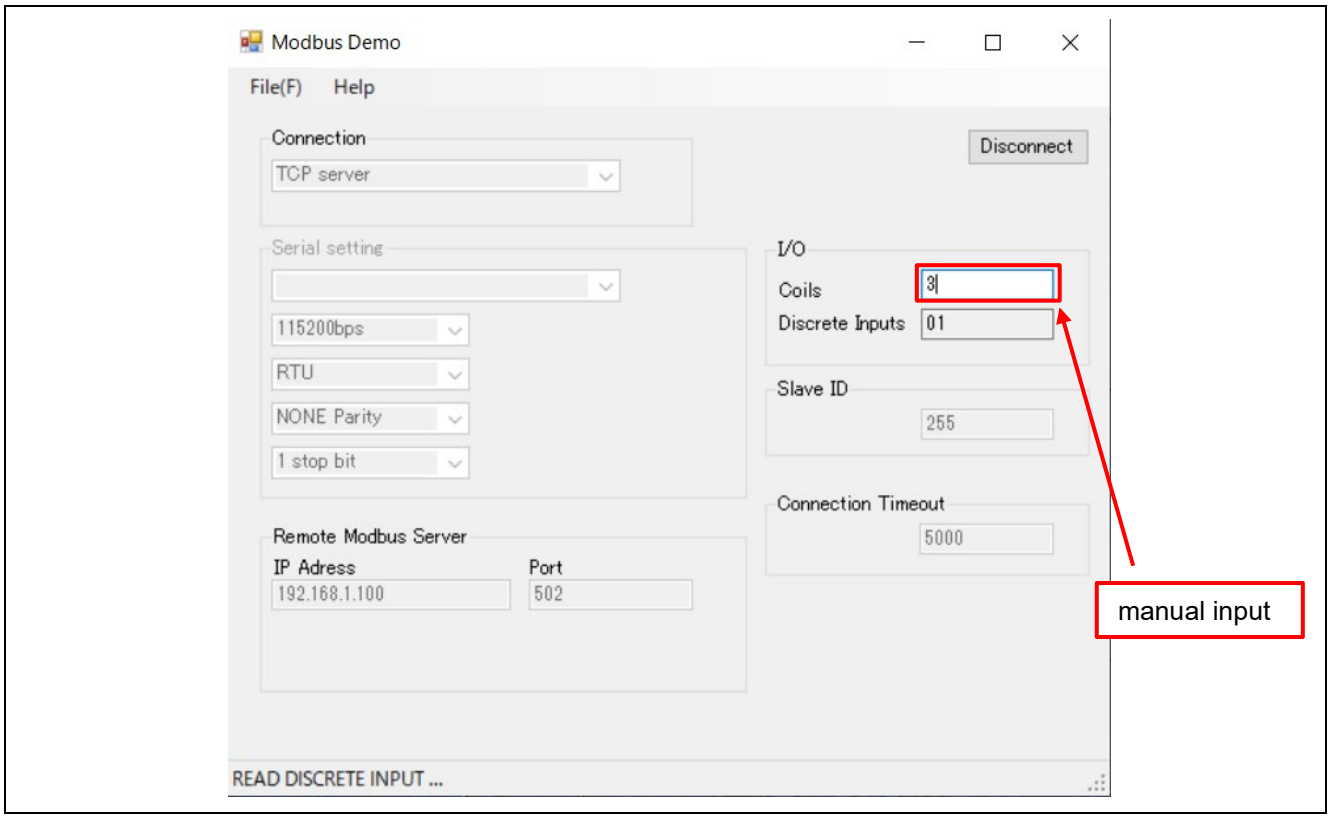
Users can see the simple demonstration using this Modbus protocol stack in this sample project.
By communicating with PC through the Modbus TCP protocol, LED blinking speed is controlled dynamically.
For this control, "Read_Discrete_Inputs" and "Write_Single_Coil" function codes are used. Specifically, the following sequence is executed.
PC application checks the state of the switch (DSW1), by using Modbus "Read_Discrete_Inputs" function code. The [SW setting value] is the 8-bits of data calculated by the state of DSW1.
According to the state of the switch, the states of the output ports, which are connected to LED, are updated periodically.

Note) The LED used depends on the CPU running the project. For example, LED4 blinks when CR52_1 is used, and LED10 blinks when CA55_2 is used.

1. When DSW1-1 is OFF, LED flash periodically.



2. When DSW1-1 is ON, LED lights up depending on the manual input 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.

RZN2H_Modbus_EVB_rev410\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 (lwip)

1. When operating with DHCP, change the member "dhcp" of the variable "g_lwip_port0_netif_cfg" in "**lwip_port_instance.c**" to "**LWIP_PORT_DHCP_ENABLE**" to enable DHCP operation.

2. Build and debug.

*lwip_port_instance.c

```
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,
```

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".

RZN2H_Modbus_EVB_rev0410\Modbus_tcp_lwip\common\renesas\modbus_user\modbus_init.c

```
uint32_t modbus_init(void);

/**
 ***************************************************************************
 @brief  Initialize MODBUS protocol stack
 @param  none
 @retval 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	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. 192.168.1.101
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_ALREADY_PRESENT If address already present in list. ERR_MAX_CLIENT If maximum connections reached. ERR_TABLE_DISABLED If 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 (uint16_t u16_additional_port, uint8_t u8_tcp_multiple_client)
Parameter	uint16_t u16_additional_port Additional port configured by user. uint8_t u8_tcp_multiple_client Status whether multiple clients are 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 Register) 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	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 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	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 with read holding registers request information.
	p_resp_read_holding_reg_t	pt_resp_read_holding_reg	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	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 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	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 with write multiple coils request information.
	p_resp_write_multiple_coils_t	pt_resp_write_multiple_coils	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	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 with write multiple registers request information.
	p_resp_write_multiple_reg_t	pt_resp_write_multiple_reg	structure pointer to stack from user with write multiple registers response.
Return value	0: success 1: failure		

· **fp_function_code23**

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 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_NONE	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

There is no limitation.

Revision History

Rev.	Date	Description	
		Page	Summary
1.00	Dec 23, 2024	-	First Edition issued
2.11	Jun 20, 2025	-	- Added FSP2.2 compatibility method - Change the switch settings for the MP board
3.00	Nov 28, 2025	-	Supports RZ/N2 FSP3.0.0
4.00	Apr 10, 2026	-	Supports RZ FSP4.0.0
4.10	Jul 31, 2026	-	Supports RZ FSP4.1.0

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.

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/.