

RZ/T2H, RZ/N2H Group

Quick Start Guide: lwIP Protocol Stack Sample Program

Introduction

This document describes the setup procedure of the sample program that implements lwIP (lightweight IP) protocol stack on the Renesas RZ/T2H, RZ/N2H platform.

Target Device

RZ/T2H, RZ/N2H

Contents

1. Overview	3
1.1 Abbreviations / Definitions	3
1.2 Reference	3
1.3 Limitation / Known Issue	3
2. Features	4
2.1 Package folder structure	4
3. Requirements (Software&Hardware)	5
3.1 RZ/T2H	5
3.2 RZ/N2H	6
4. Hardware Setup	7
4.1 RZ/T2H Evaluation Board	7
4.2 RZ/N2H Evaluation Board	9
4.3 Connecting the Debugger	11
5. Host PC Settings	12
5.1 Interface Property Configuration	12
5.2 Ethernet Communication Modes	14
6. Running the Sample Application	16
6.1 Setting Up the e ² studio Sample Project	16
6.1.1 Starting e ² studio	16
6.1.2 How to generate source code and how to build.	19
6.1.3 Downloading the application and running the debugger.	23
6.2 Setting Up the EWARM Sample Project	25
6.2.1 Startup EWARM.	25
6.2.2 How to generate source code and how to build.	27
6.2.3 Downloading the application and running the debugger.	30
7. Demonstration of the Sample Application	32

7.1	Connecting to a PC	32
7.2	SNTP Client.....	34
7.3	iperf Client/Server	35
7.3.1	Procedure for measuring communication speed.....	35
7.4	Echo Server.....	37
8.	Software Specifications	38
8.1	Software Structure.....	38
8.1.1	Components Overview.....	39
8.1.2	Drivers Used.....	39
8.2	Task.....	40
8.2.1	Resources	40
8.2.2	Task Operations	41
8.3	Received Packet Mirroring and Transmission Functions.....	42
8.4	Multi-interface	44
8.4.1	Interface Property Table	44
8.4.2	Routing	46
8.4.3	Management port specific tag + Non-routing mode	47
8.5	Memory Allocation	48
8.5.1	e ² studio project	48
8.5.2	EWARM project.....	60
8.6	CPU Utilization	74
	Revision History.....	75

1. Overview

This document describes the setup procedure of the sample program that implements lwIP (lightweight IP) protocol stack on the Renesas RZ/T2H, RZ/N2H platform.

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	lwIP	lightweight IP

1.2 Reference

Technical information about RZ/T2H, RZ/N2H is available via Renesas.

Table 1.2 Technical Inputs for RZ/T2H

Document Type	Description	Document Title	Document No.
User's Manual	Explains the technical details of the RZ/T2H Evaluation Board	RZ/T2H Evaluation Board Kit User's Manual	R20UT5405EJ****
User's Manual	RZ/T2H and RZ/N2H Evaluation Board Specifications	RZ/T2H and RZ/N2H Groups User's Manual: Hardware	R01UH1039EJ****

Table 1.3 Technical Inputs for RZ/N2H

Document Type	Description	Document Title	Document No.
User's Manual	Explains the technical details of the RZ/N2H Evaluation Board	RZ/N2H Evaluation Board Kit User's Manual	R20UT5522EJ****
User's Manual	RZ/T2H and RZ/N2H Evaluation Board Specifications	RZ/T2H and RZ/N2H Groups User's Manual: Hardware	R01UH1039EJ****

1.3 Limitation / Known Issue

For ETH0 and ETH1 on the RZ/T2H and RZ/N2H, the setting is limited to 100 Mbps or lower.

2. Features

lwIP (lightweight IP) is an open-source embedded TCP/IP protocol stack. lwIP can implement full-scale TCP functionality with few resources and is suitable for use in embedded systems. lwIP was originally developed by Adam Dunkels at the Swedish Institute of Computer Science (SICS). It is now developed and maintained by a worldwide network of developers. For information on licensing lwIP, see "`\common\oss\lwip\COPYING`" in the sample program.

2.1 Package folder structure

Here is a folder overview of the main files:

Please refer to the release notes for the folder tree.

Table 2.1. Folder structure

Folder name	overview
common/oss	lwip+freeRTOS package file
common/renesas/application	Applications
common/renesas/module/ether_netif	Ethernet API
common/renesas/module/lwip_port	lwip API
common/renesas/module/serial_io	Serial API for log output
SRC	Entry functions, etc.

3. Requirements (Software&Hardware)

This project was developed and tested in an environment using the following boards and tools:

3.1 RZ/T2H

Table 3.1. RZ/T2H environment

Category	Name	Version	Description
Board	Evaluation Board Kit for RZ/T2H	User's Manual 1.20	Renesas RZ/T2H-EVKIT- Evaluation Board Kit for RZ/T2H
IDE	EWARM	9.60.3	IAR Systems IAR Embedded Workbench for Arm IAR
	e ² studio	2026-04.1	RZ FSP
Configurator	FSP Smart Configurator	2026-04.1	https://github.com/renesas/rz-fsp/releases/tag/v4.1.0
Flexible Software Package	FSP	4.1.0	
GCC Compiler	GNU ARM Embedded Toolchain	13.3.Rel1	
	GNU ARM A-Profile (A-Profile AArch64 bare-metal)	13. 2.Rel1	
Emulator	J-Link TM	8.60	SEGGER SEGGER - The Embedded Experts - Downloads - J-Link / J-Trace
	I-jet		IAR Systems IAR debug probes IAR
	Tera Term		Tera Term Tera Term Open Source Project
Software	Wireshark		WIRESHARK Wireshark • Go Deep Download

3.2 RZ/N2H

Table 3.2. RZ/N2H environment

Category	Name	Version	Description
Board	RZ/N2H Evaluation Board	User's Manual 1.20	Renesas RZ/N2H-EVKIT- Evaluation Board Kit for RZ/N2H
IDE	EWARM	9.60.3	IAR Systems IAR Embedded Workbench for Arm IAR
	e ² studio	2026-04.1	RZ FSP
Configurator	FSP Smart Configurator	2026-04.1	https://github.com/renesas/rz-fsp/releases/tag/v4.1.0
Flexible Software Package	FSP	4.1.0	
GCC Compiler	GNU ARM Embedded Toolchain	13.3.Rel1	
	GNU ARM A-Profile (A-Profile AArch64 bare-metal)	13. 2.Rel1	
Emulator	J-Link TM	8.60	SEGGER SEGGER - The Embedded Experts - Downloads - J-Link / J-Trace
	I-jet		IAR Systems IAR debug probes IAR
	Tera Term		Tera Term Tera Term Open Source Project
Software	Wireshark		WIRESHARK Wireshark • Go Deep Download

4. Hardware Setup

This section explains the main hardware. For details about the board you are using, please refer to the [user's manual](#).

4.1 RZ/T2H Evaluation Board

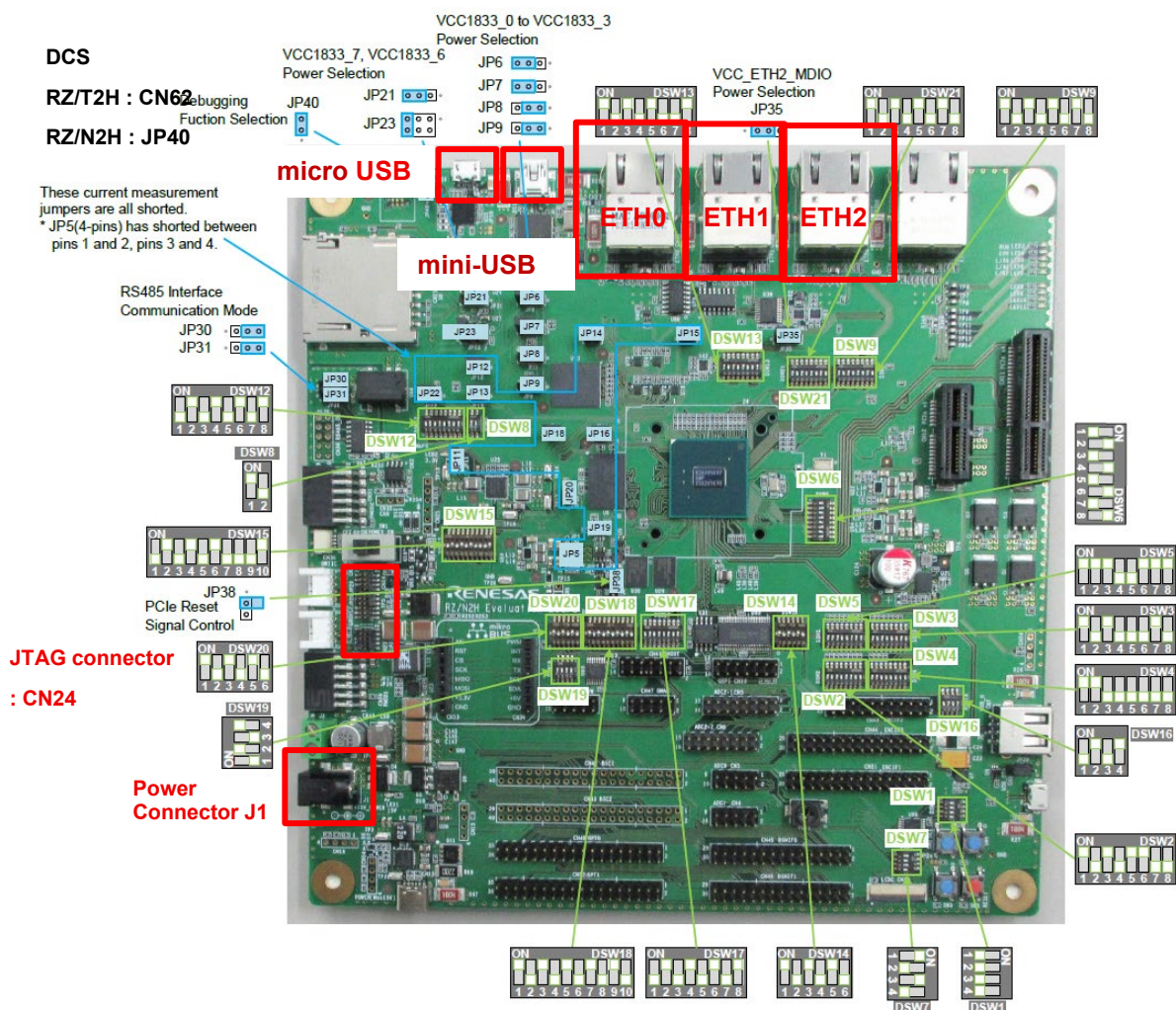


Figure 4.1 RZ/T2H Evaluation Board

1. DIP Switch and Jumper Switch Settings

These settings are described in "6.2 Factory Configuration" in the RZ/T2H Evaluation Board User's Manual.

➤ For RAM boot (RAM execution without flash memory) programs, please note the following:

If the DSW14 is in xSPI1 boot mode, erase the area used by xSPI1.

For erasing instructions, refer to "Appendix: How to Erase Flash Memory" in "Getting Started with Flexible Software Package."

2. If using J-Link OB, connect to the micro-USB connector. If using I-jet, connect to the JTAG connector and switch using the Debugging Control Selection (DCS) setting.
3. Use an AC/DC adapter for power.
 - When using an AC/DC adapter, connect to the Power Connector.
4. You may freely connect the LAN cables to ETH0, ETH1, and ETH2—either simultaneously or individually.

4.2 RZ/N2H Evaluation Board

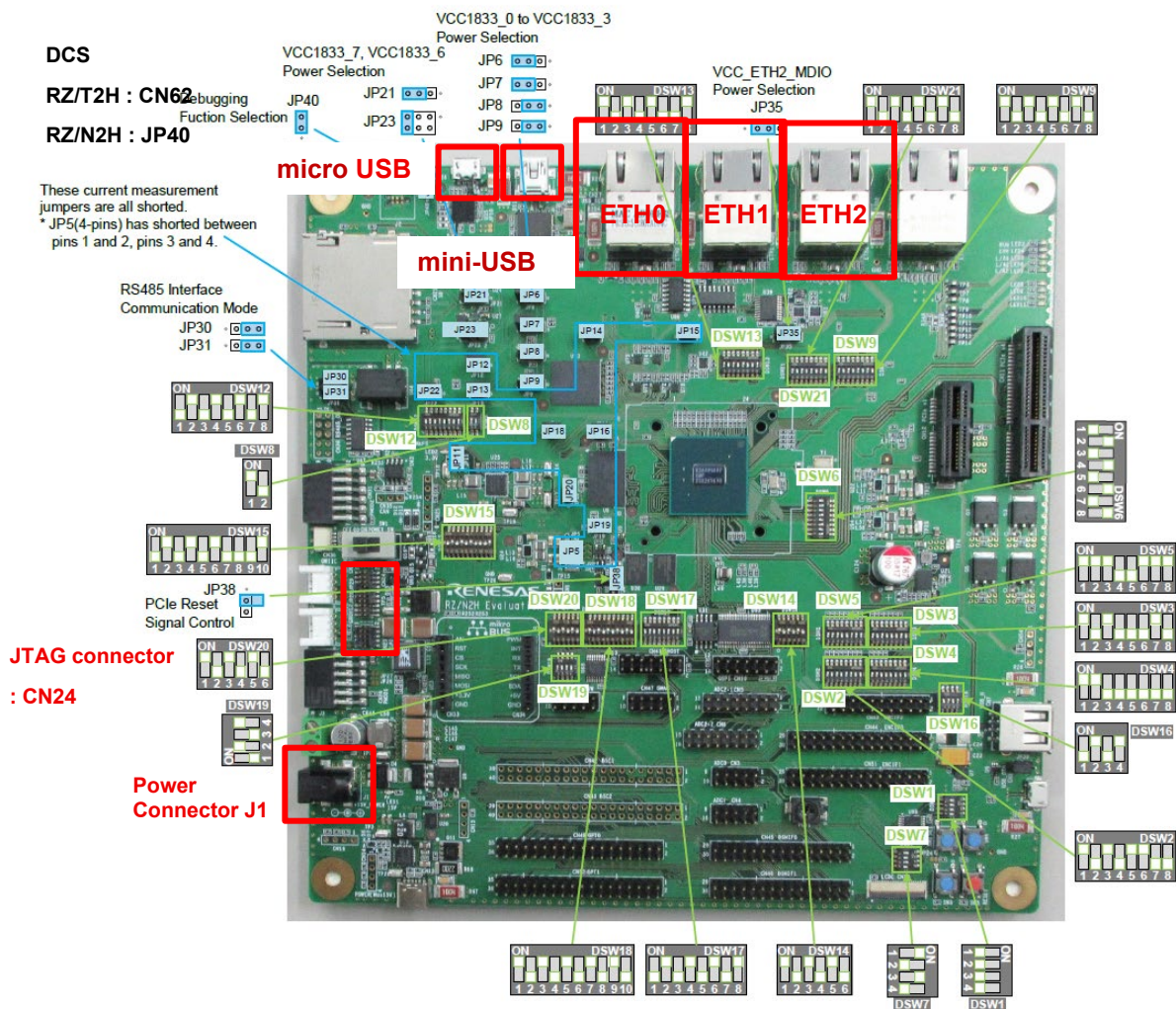


Figure 4.2 RZ/N2H Evaluation Board

1. DIP Switch and Jumper Switch Settings

Settings other than those listed below are the factory settings as described in "6.2 Factory Configuration" in the RZ/N2H Evaluation Board User's Manual.

- DIP Switch Settings to Change from Factory Settings
 - Changing ETH2 Reset Settings: DSW8-1: OFF, DSW8-2: ON
- When booting in RAM boot mode (RAM execution without flash memory), please note the following:
 - If DSW3 is in xSPI1 boot mode, erase the area used by xSPI1.

For instructions on how to erase, see "Appendix: How to Erase Flash Memory" in "Getting Started with Flexible Software Package."

2. If using J-Link OB, connect to the micro-USB. If using I-jet, connect to the JTAG connector and switch using the Debugging Control Selection (DCS).

3. Use an AC/DC adapter to supply power.
 - When using an AC/DC adapter, connect it to the Power Connector.
4. You may freely connect the LAN cables to ETH0, ETH1, and ETH2—either simultaneously or individually.

4.3 Connecting the Debugger

Connect the evaluation board and debugger according to the tool you are using.

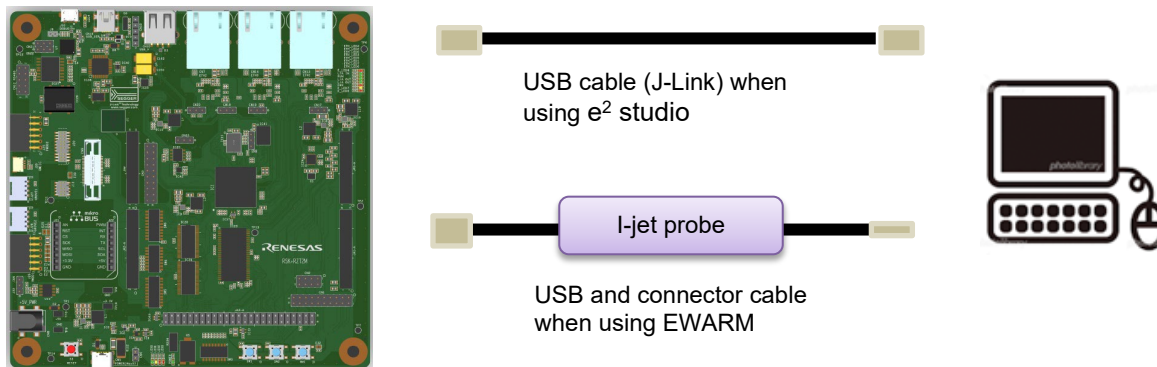


Figure 4.3 Debugger Connection

5. Host PC Settings

5.1 Interface Property Configuration

Following the following steps, configure the properties of the three interfaces, ensuring that the IP addresses and MAC addresses are unique and match the execution environment.

1. Refer to the source code within the project package to verify the IP address (`ip\_address`), subnet mask (`subnet\_mask`), default gateway (`gateway\_address`), MAC address (`hwaddr`), connected LAN port number (`eth\_port`), and other parameters.
2. Echo server host-side port number
Refer to the source code within the project package to verify the host-side port number for the echo server.

The array data in `g\_lwip\_port\_netif\_cfg` defines the configuration for the LAN ports to be used on each evaluation board.

It is not a problem if a configured LAN port remains unconnected.

For details, please refer to "[7. Demonstration of the Sample Application](#)"

There are no differences in the contents of the following source files between e² studio and EWARM.

- src/renesas/application/lwip_port_instance.c
- src/renesas/application/lwip_port_main.c

```

94  /** Initial settings data such as IP address and MAC address */
95  static lwip_port_netif_ip_cfg_t g_lwip_port_netif_cfg[LWIP_NIC_MAX] =
96  {
97      {
98          #if LWIP_NIC_MAX > 1
99              .p_next      = &g_lwip_port_netif_cfg[1],
100          #else
101              .p_next      = NULL,
102          #endif
103              .dhcp        = LWIP_PORT_DHCP_DISABLE,
104              .ip_address   = PP_HTONL(LWIP_MAKEU32(192,168, 10,170)),
105              .subnet_mask  = PP_HTONL(LWIP_MAKEU32(255,255,255, 0)),
106              .gateway_address = PP_HTONL(LWIP_MAKEU32(192,168, 10, 1)),
107              .hwaddr       = {0x00,0x11,0x22,0x33,0x44,0x55},
108              .p_host_name  = gp_lwip_port0_hostname_0,
109              .eth_port     = ETHER_PORT_0,
110          },
111          #if LWIP_NIC_MAX > 1
112          {
113              .p_next      = &g_lwip_port_netif_cfg[2],
114              .dhcp        = LWIP_PORT_DHCP_DISABLE,
115              .ip_address   = PP_HTONL(LWIP_MAKEU32(192,168, 10,171)),
116              .subnet_mask  = PP_HTONL(LWIP_MAKEU32(255,255,255, 0)),
117              .gateway_address = PP_HTONL(LWIP_MAKEU32(192,168, 10, 2)),
118              .hwaddr       = {0x00,0x11,0x22,0x33,0x44,0x77},
119              .p_host_name  = gp_lwip_port0_hostname_1,
120              .eth_port     = ETHER_PORT_1,
121          },
122          #endif
123          #if LWIP_NIC_MAX > 2
124          {
125              .p_next      = NULL,
126              .dhcp        = LWIP_PORT_DHCP_DISABLE,
127              .ip_address   = PP_HTONL(LWIP_MAKEU32(192,168, 10,172)),
128              .subnet_mask  = PP_HTONL(LWIP_MAKEU32(255,255,255, 0)),
129              .gateway_address = PP_HTONL(LWIP_MAKEU32(192,168, 10, 3)),
130              .hwaddr       = {0x00,0x11,0x22,0x33,0x44,0x88},
131              .p_host_name  = gp_lwip_port0_hostname_2,
132              .eth_port     = ETHER_PORT_2,
133          },
134          #endif
135      };
136

```

Figure 5.1 Static IP address

```

41  * Macro definitions
42
43  #define TCP_SERVER_PORT      (65000)
44  #define TCP_SERVER_TASK_PRIORITY (3 + LWIP_TASK_PRIORITY_BASE)
45  #define TCP_SERVER_TASK_STACK_SIZE (1024)
46  #define TCP_SERVER_TASK_NAME "TCP Server task"
47  #define TCP_SERVER_CLIENT_MAX 10
48  #define TCP_SNTP_TASK_PRIORITY (2 + LWIP_TASK_PRIORITY_BASE)

```

Figure 5.2 Port number

5.2 Ethernet Communication Modes

The following communication modes are available between the PC and the evaluation board; please select the appropriate mode or combination based on your specific use case.

1. Routing Mode

In this mode, when communicating with the PC, the evaluation board associates the MAC address of frames received via the internal Ethernet driver with the NETIF and notifies the lwIP stack of the reception.

2. "Management port specific tag"

In the "Stacks" tag of FSP_SC, within "g_ether0 Ethernet(r_ethsw)" of "g_ether0 Ethernet(r_gmac)", the "Management port specific tag" setting will be as follows:

➤ Enable

While this allows for the configuration of the EtherSW and the viewing of settings, data copying occurs for every frame during transmission and reception, resulting in significant overhead.

Additionally, this setting makes it possible to disable the aforementioned routing mode, resulting in the following changes.

Source file: src/renesas/module/lwip_port/um_lwip_port_cfg.h

Macro name: UM_LWIP_PORT_ROUTING_SET

Setting value: 1 (Enabled), 0 (Disabled)

Source file: src/renesas/oss_deps/lwip/lwipopts.h

Macro name: ETH_PAD_SIZE 8

When disabled, the lwIP's NETIF and LAN port will be connected via P2P.

For details, please refer to "[8. Software Specifications.](#)"

➤ Disabled

This setting is the default unless you need to change the configuration, such as forcing port-to-port forwarding on the EtherSW.

Note that if this is set to "Disabled," routing mode is automatically enabled.

Also, please check the number of macros below.

Source file: src/renesas/oss_deps/lwip/lwipopts.h

Macro name: ETH_PAD_SIZE 0

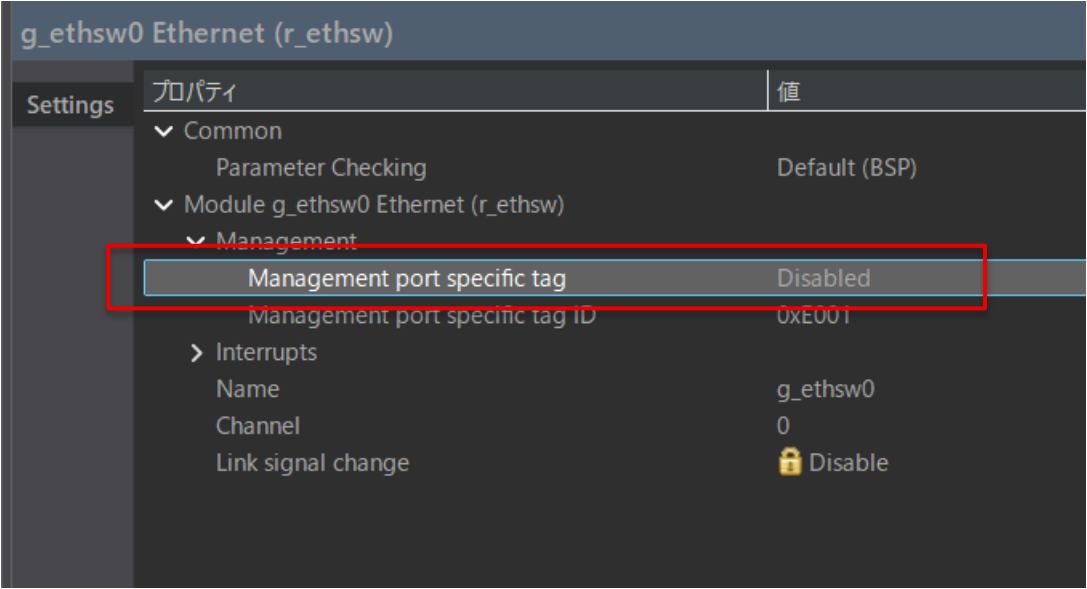


Figure 5.3 Management port specific tag

```
34  /** Routing packets to lwIP based on MAC address. 1 = Enable 0 = Disable */
35  #define UM_LWIP_PORT_ROUTING_SET (1)
36
```

Figure 5.4 Routing mode

```
47  /* If the MAC frame mode containing EtherSW management data is ETHSW_SPECIFIC_TAG_ENABLE, this data will be 8. */
48  #define ETH_PAD_SIZE 0
49
```

Figure 5.5 ETH_PAD_SIZE

6. Running the Sample Application

To run an EWARM project, proceed to [“6.2. Setting Up the EWARM Sample Project.”](#)

6.1 Setting Up the e² studio Sample Project

Since both the RZ/T2H and RZ/N2H are multi-core processors, projects run on both the primary CPU (hereinafter referred to as CPU0) and the secondary CPU (hereinafter referred to as CPU1).

If the project workspace contains a "Debug" folder, do not delete the folder or the files within it.

6.1.1 Starting e² studio

1. Open e² studio and select the workspace directory.
2. Click "Open Projects from File System..." on the File tab.

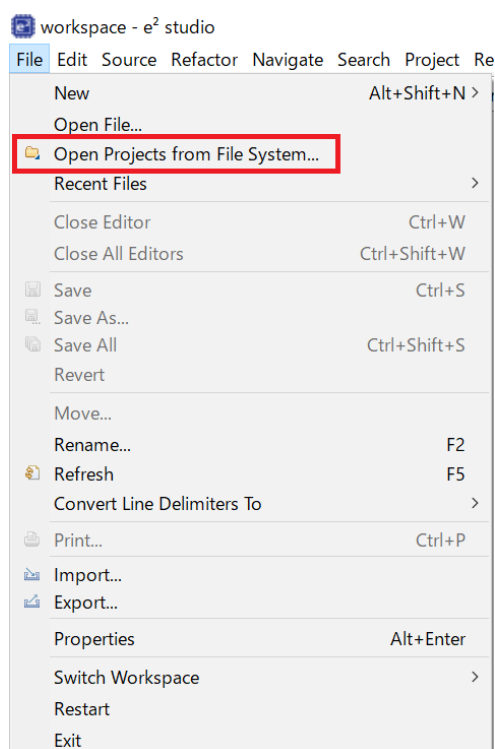


Figure 6.1 e²studio File tab

3. Import the two projects as shown below, depending on the board and CPU you are using.

■ For CR52 and CR52 projects

➤ Primary Project (CPU0)

This project has no functionality other than blinking the LED and initial startup settings.

- When using RZ/T2H
project\rzt2h_evb\cr52_cr52\studio\RZT2H_EVB_lwIP_cpu0
- When using RZ/N2H
project\rzn2h_evb\cr52_cr52\studio\RZN2H_EVB_lwIP_cpu0

➤ Secondary Project (CPU1)

This project includes functions related to "[7. Demonstration of the Sample Application](#)".

- When using RZ/T2H
project\rzt2h_evb\cr52_cr52\studio\RZT2H_EVB_lwIP_cpu1
- When using RZ/N2H
project\rzn2h_evb\cr52_cr52\studio\RZN2H_EVB_lwIP_cpu1

■ For CR52 and CA55 projects

➤ Primary Project (CPU0)

This project has no functionality other than blinking the LED and initial startup settings.

- When using RZ/T2H
project\rzt2h_evb\cr52_0-ca55_2\studio\RZT2H_EVB_lwIP_cpu0
- When using RZ/N2H
project\rzn2h_evb\cr52_0-ca55_2\studio\RZN2H_EVB_lwIP_cpu0

➤ Secondary Project (CPU1)

This project includes functions related to "[7. Demonstration of the Sample Application](#)".

- When using RZ/T2H
project\rzt2h_evb\cr52_0-ca55_2\studio\RZT2H_EVB_lwIP_cpu1
- When using RZ/N2H
project\rzn2h_evb\cr52_0-ca55_2\studio\RZN2H_EVB_lwIP_cpu1

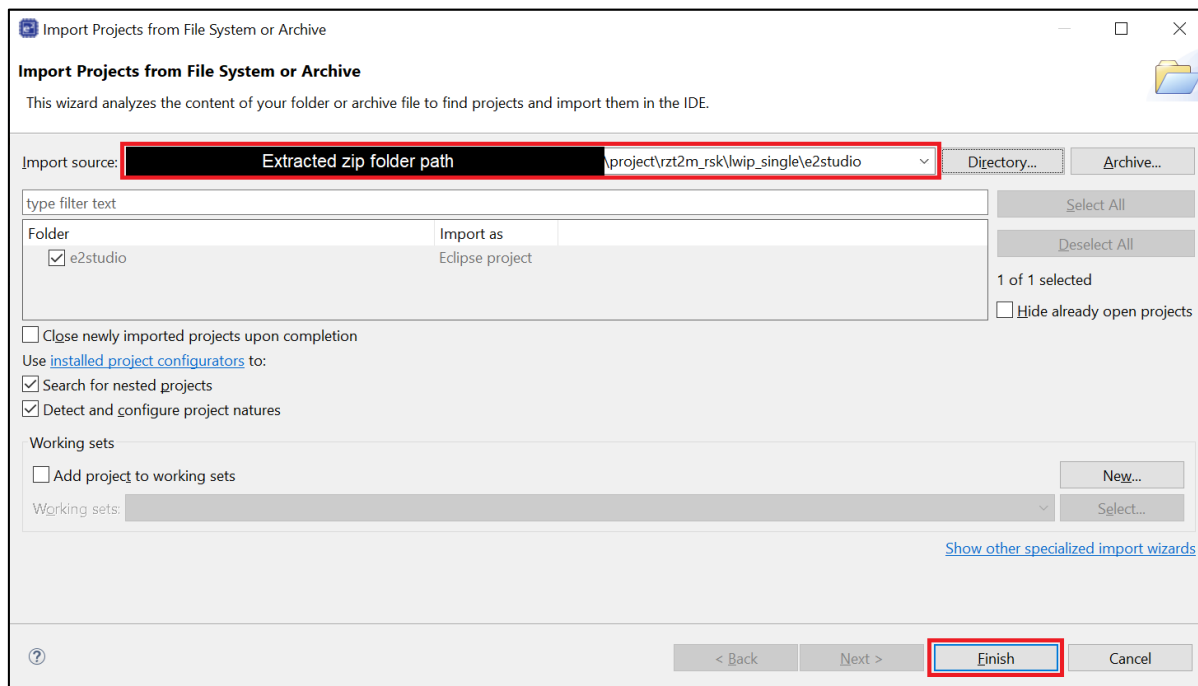


Figure 6.2 Import project on e²studio

6.1.2 How to generate source code and how to build.

Here too, please generate the code for the CPU0 and CPU1 projects in order, starting with CPU0.

If there is a script folder in the Project Explorer window before code generation, the following precautions are required.

- If an "ld" file exists in the "script" folder, using a file other than those in that folder may result in unexpected outcomes, such as build errors.
 - If you change the boot mode or FSP version, the "ld" file will be overwritten during code generation; therefore, please save it beforehand and restore the saved "ld" file after code generation is complete.
- If there is no change, there is no rewriting.

1. Click the "Configuration.xml".

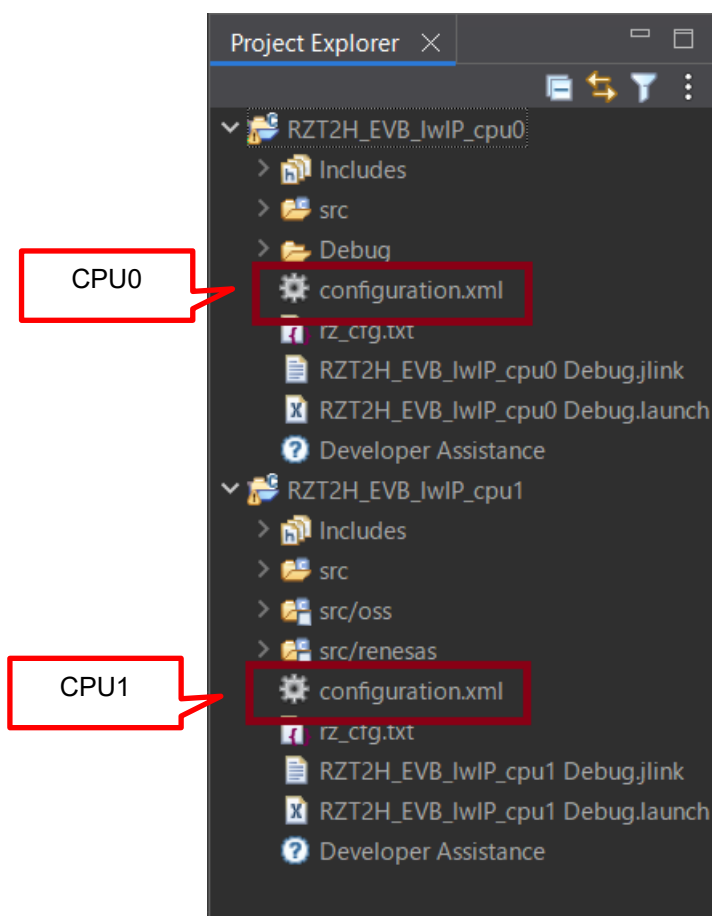


Figure 6.3 Configuration

2. After clicking "Configuration.xml", check the following settings from the "BSP" tag in the FSP settings screen that appears, then click "Generate Project Content". The generated folder will be added to your workspace.

Note that the following settings can only be changed from the CPU0 project.

- RAM boot (RAM execution without flash memory)
 - RZ/T2H
 - Board : RZT2H Evaluation Board (RAM execution without flash memory)
 - RZ/N2H
 - Board : RZN2H Evaluation Board (RAM execution without flash memory)

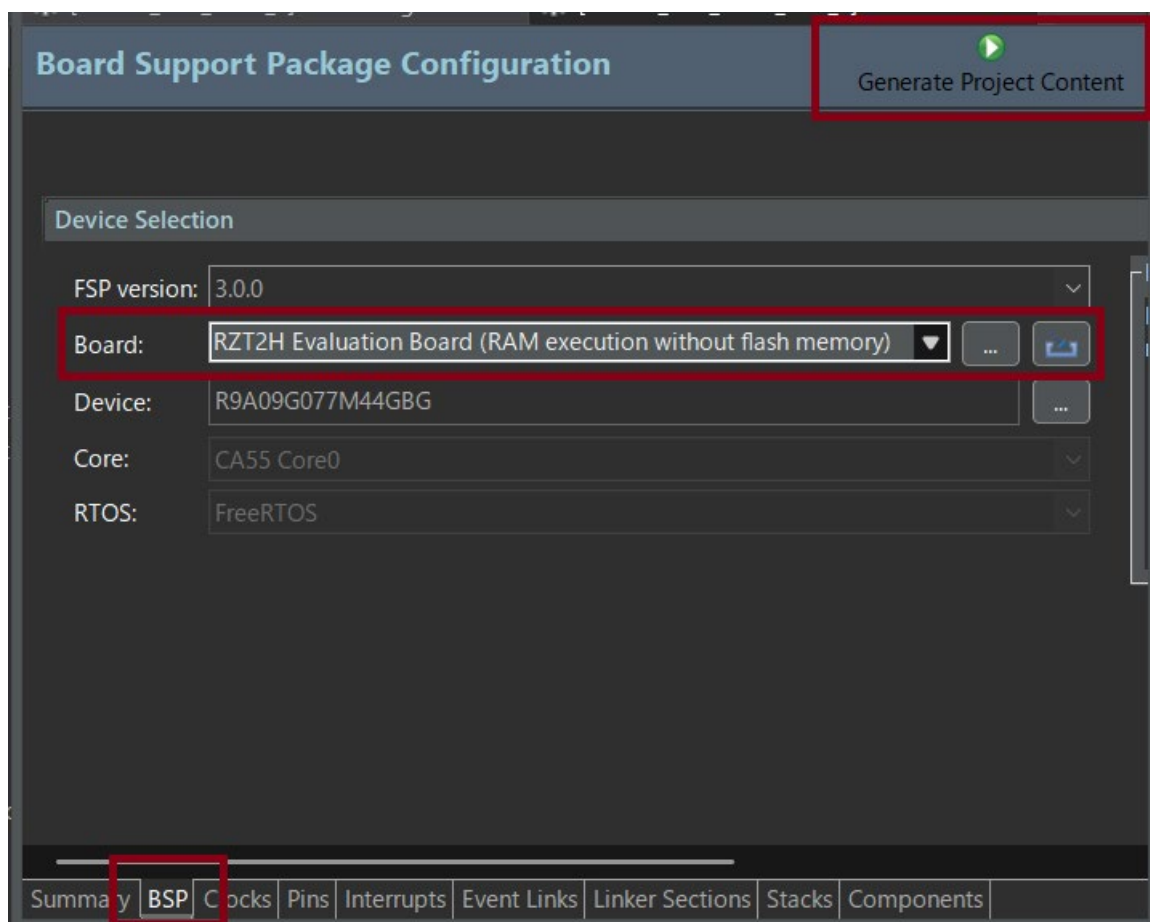


Figure 6.4 Generate Project Content

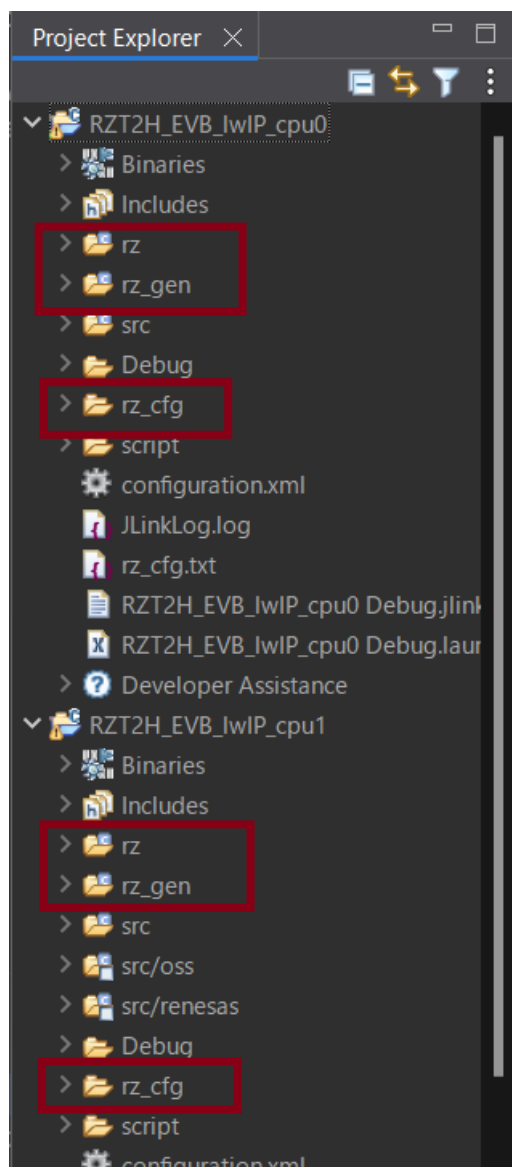


Figure 6.5 Generate project folder

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

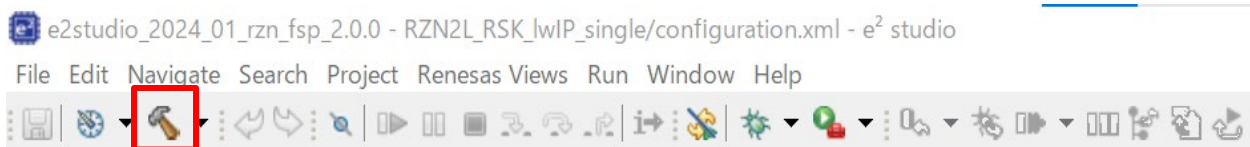
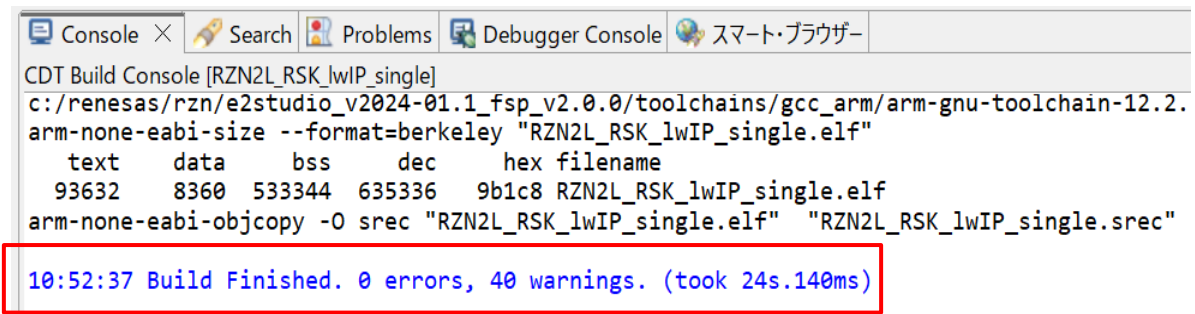


Figure 6.6 Build button



```
CDT Build Console [RZN2L_RSK_lwIP_single]
c:/renesas/rzn/e2studio_v2024-01.1_fsp_v2.0.0/toolchains/gcc_arm/arm-gnu-toolchain-12.2.
arm-none-eabi-size --format=berkeley "RZN2L_RSK_lwIP_single.elf"
  text    data    bss     dec     hex filename
 93632    8360   533344   635336   9b1c8 RZN2L_RSK_lwIP_single.elf
arm-none-eabi-objcopy -O srec "RZN2L_RSK_lwIP_single.elf" "RZN2L_RSK_lwIP_single.srec"

10:52:37 Build Finished. 0 errors, 40 warnings. (took 24s.140ms)
```

Figure 6.7 Build message

6.1.3 Downloading the application and running the debugger.

Please follow these steps for each project.

1. Focus on the CPU0 project and click the Debug button on the toolbar to download the CPU0 program and launch the debugger.

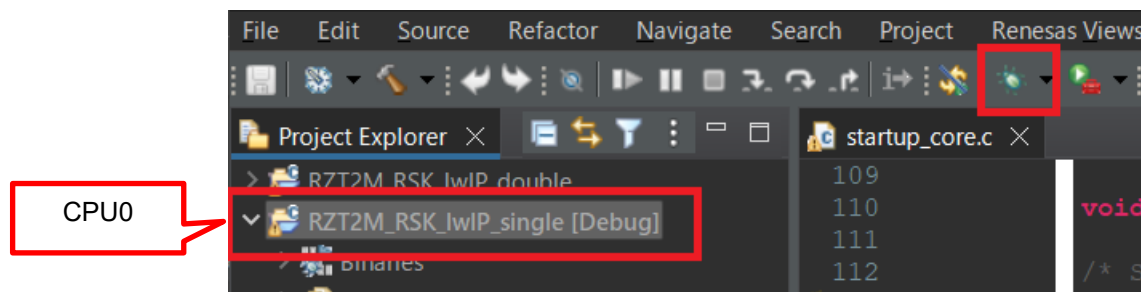


Figure 6.8 CPU0 Debug button

2. Dialog response
When starting the debugger, if a dialog box other than an error message appears due to the operating environment, etc., close it by selecting "Switch," "Yes," or "OK."
3. The program will stop at the breakpoint set at startup, so simply click the Run button on the toolbar.
Before running, connect your PC as described in "[7. Sample Application Demonstration.](#)"

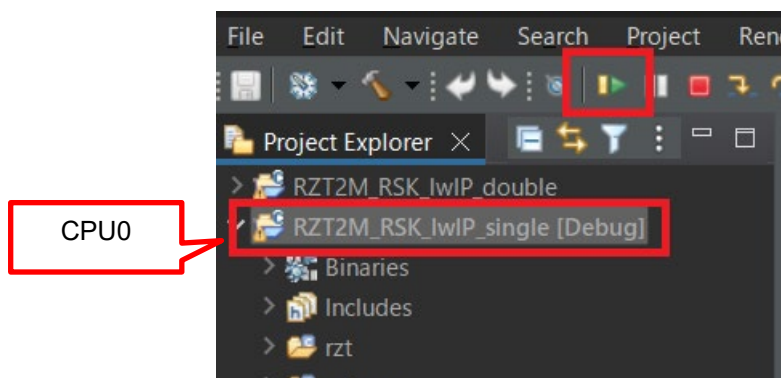


Figure 6.9 CPU0 Run button

- Next, focus on the CPU1 project and click the Debug button on the toolbar to download the CPU1 program and launch the debugger.

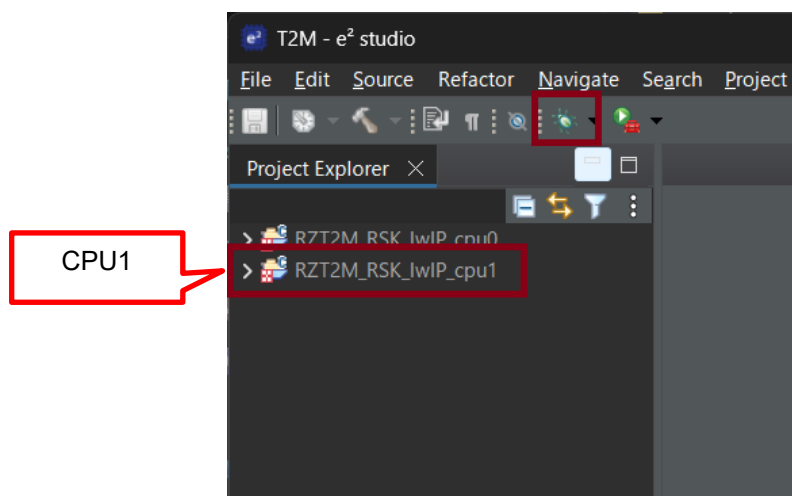


Figure 6.10 CPU1 Debug button

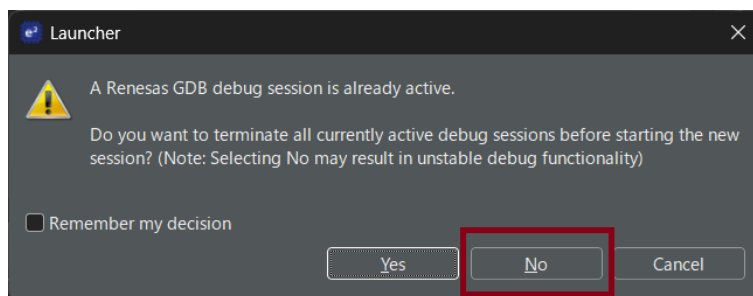


Figure 6.11 CPU1 Duplicate confirmation

5. Other Dialogs

If a dialog other than an error message appears due to your environment or other reasons, close it by selecting "Switch," "Yes," or "OK."

- The program will stop at the breakpoint set at startup, so simply click the Run button on the toolbar.

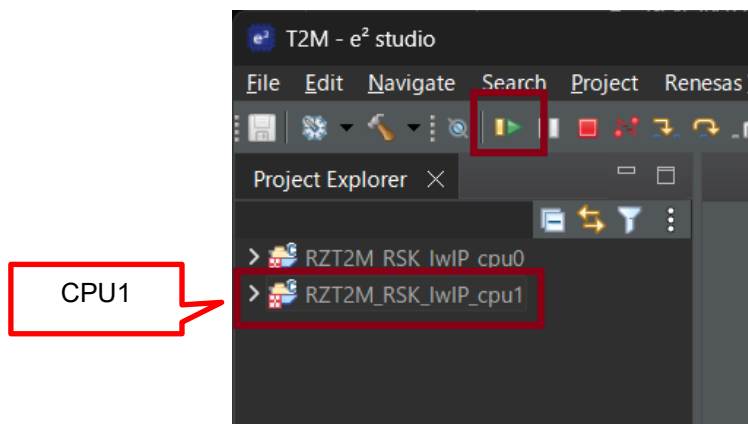


Figure 6.12 CPU1 Run button

6.2 Setting Up the EWARM Sample Project

6.2.1 Startup EWARM.

For each of the project files listed below, please perform steps 1 through 3 below.

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

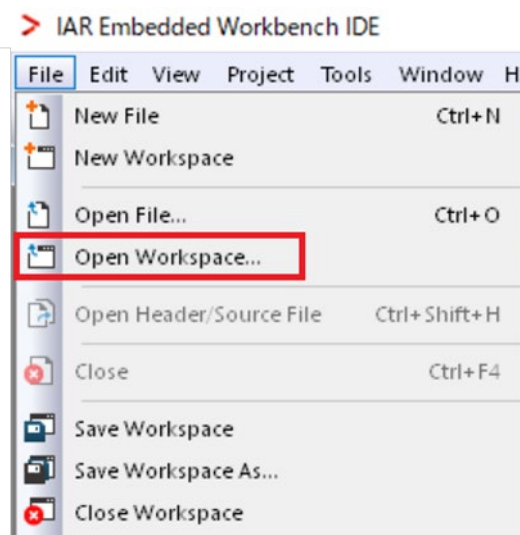


Figure 6.13 EWARM file tab

3. Select the Workspace File(.eww) and click the Open button.

■ For CR52 and CR52 projects

➤ Primary Project (CPU0)

This project has no functionality other than blinking the LED and initial startup settings.

- When using RZ/T2H
project\rzt2h_evb\cr52_cr52\ewarm\RZT2H_EVB_lwIP_cpu0
- When using RZ/N2H
project\rzn2h_evb\cr52_cr52\ewarm\RZN2H_EVB_lwIP_cpu0

➤ Secondary Project (CPU1)

This project will implement the functionality related to "[7. Demonstration of Sample Application](#)".

- When using RZ/T2H
project\rzt2h_evb\cr52_cr52\ewarm\RZT2H_EVB_lwIP_cpu1
- When using RZ/N2H
project\rzn2h_evb\cr52_cr52\ewarm\RZN2H_EVB_lwIP_cpu1

■ For CR52 and CA55 projects

➤ Primary Project (CPU0)

This project has no functionality other than blinking the LED and initial startup settings.

- When using RZ/T2H
project\rzt2h_evb\cr52_0-ca55_2\ewarm\RZT2H_EVB_lwIP_cpu0
- When using RZ/N2H
project\rzn2h_evb\cr52_0-ca55_2\ewarm\RZN2H_EVB_lwIP_cpu0

➤ Secondary Project (CPU1)

This project will implement the functionality related to "[7. Demonstration of Sample Application](#)".

- When using RZ/T2H
project\rzt2h_evb\cr52_0-ca55_2\ewarm\RZT2H_EVB_lwIP_cpu1
- When using RZ/N2H
project\rzn2h_evb\cr52_0-ca55_2\ewarm\RZN2H_EVB_lwIP_cpu1

6.2.2 How to generate source code and how to build.

1. Notes on Code Generation.

- If an "icf" file exists in the "script" folder, using a file other than those in that folder may result in unexpected outcomes, such as build errors.
- If you change the boot mode or FSP version, the "icf" file will be overwritten during code generation; therefore, please save it beforehand and restore the saved "icf" file after code generation is complete.
- Please build the project using both CPU0 and CPU1.
- Do not open "FSP Smart Configurator" for both projects simultaneously.

2. Launching the FSP Smart Configurator

If you have not set up the Configurator, see "5.4.1 Launch FSP Smart Configurator from IAR EWARM" in "Getting Started with Flexible Software Package."

If you are unable to start FSP_SC when installing it because you specified an installation destination other than the default, please correct it using the following method according to your environment.

- If you want to always use the same version of EWARM and FSP_SC

In "5.4.1 Launch FSP Smart Configurator from IAR EWARM," delete "\$RASC_EXE_PATH\$" from the "Command" field and use "Browse..." to specify the installed "rasc.exe."

- If you want to link and launch FSP_SC for each project

In the workspace's "buildinfo.ipcf," change the environment variables as follows:

```
<group name="RA Smart Configurator">
```

```
<argVar>
```

```
<name>RASC_EXE_PATH</name>
```

```
<value>C:\Renesas\fsp_sc\ eclipse\rasc.exe</value>
```

👉 Change this path to your installation location.

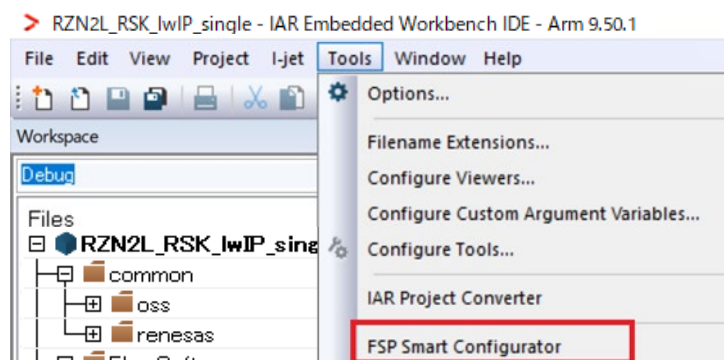


Figure 6.14 Tools tab

3. Code Generation Procedure

After clicking "Configuration.xml," the FSP settings screen will appear. From the "BSP" tab, confirm the following settings and click "Generate Project Content." The generated folder will be added.

Please note that the following settings can only be changed from the CPU0 project.

- RZ/T2H
Board : : RZT2H Evaluation Board (RAM execution without flash memory)
- RZ/N2H
Board : : RZN2H Evaluation Board (RAM execution without flash memory)

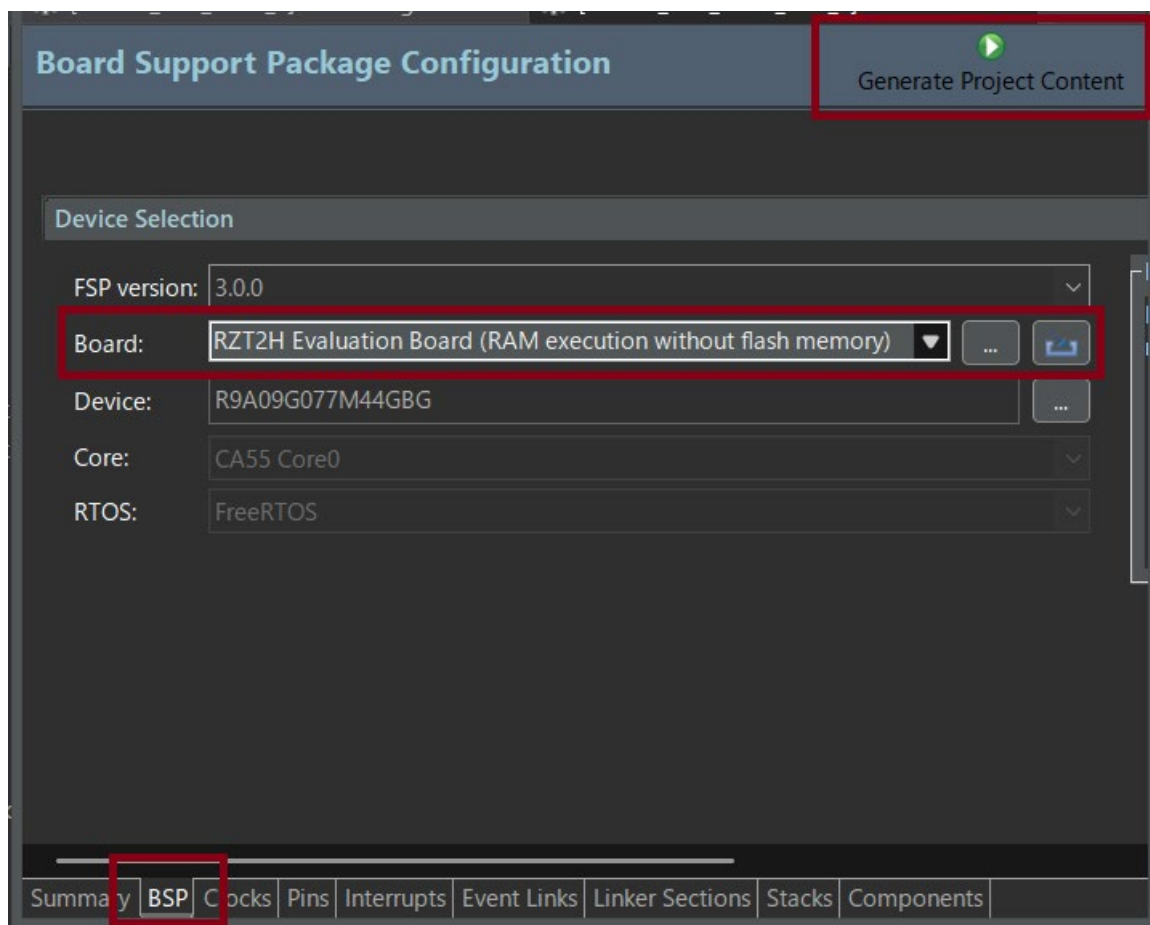


Figure 6.15 FSP SC Smart Configurator

4. Build Procedure

Click "Make" on the Project tab or click the button to build. When the build is complete, build messages will be displayed in the Build Console window, displaying the compilation target files and the number of errors/warnings.

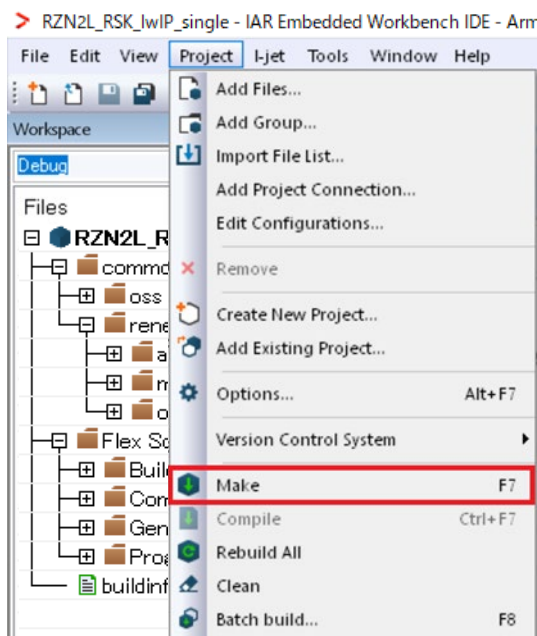


Figure 6.16 Make button 1



Figure 6.17 Make button 2

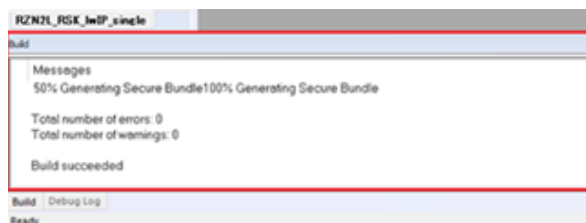


Figure 6.18 Make Build console

6.2.3 Downloading the application and running the debugger.

Please follow these steps.

1. Close EWARM for the CPU1 project.
2. Click "Download and Debug" in the "Project" menu bar, or click the Debug button to download the application program. After the debugger starts, the program will stop at the set breakpoint.

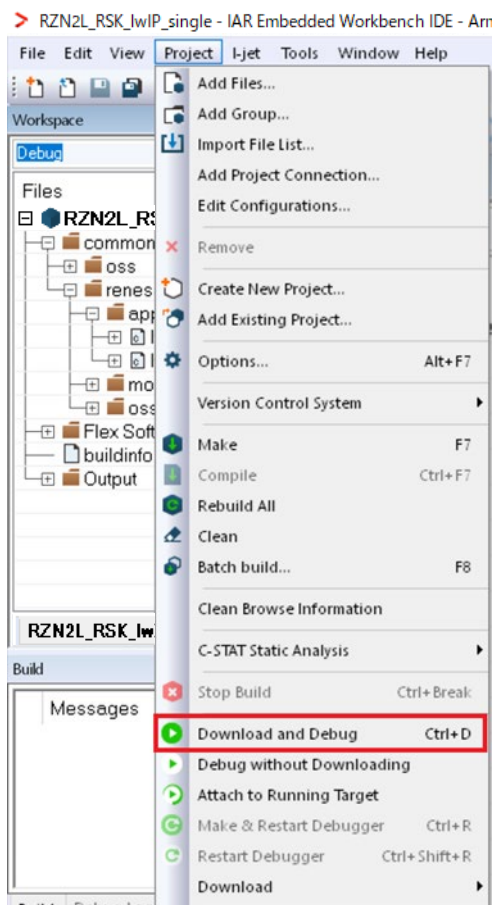


Figure 6.19 Download and Debug

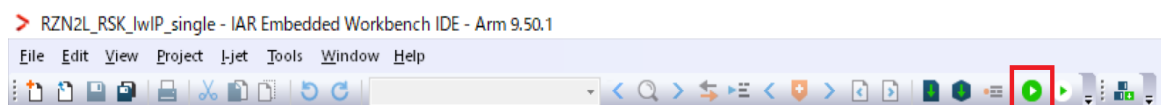


Figure 6.20 Debug button

- As CPU0 downloads, CPU0 will become the master and the CPU1 project will automatically start as the slave, and both projects will stop at the set breakpoints.
- Click the Run button in the CPU0 project, then run the CPU1 project in the same way. Before running, connect via UART to a PC running terminal software.

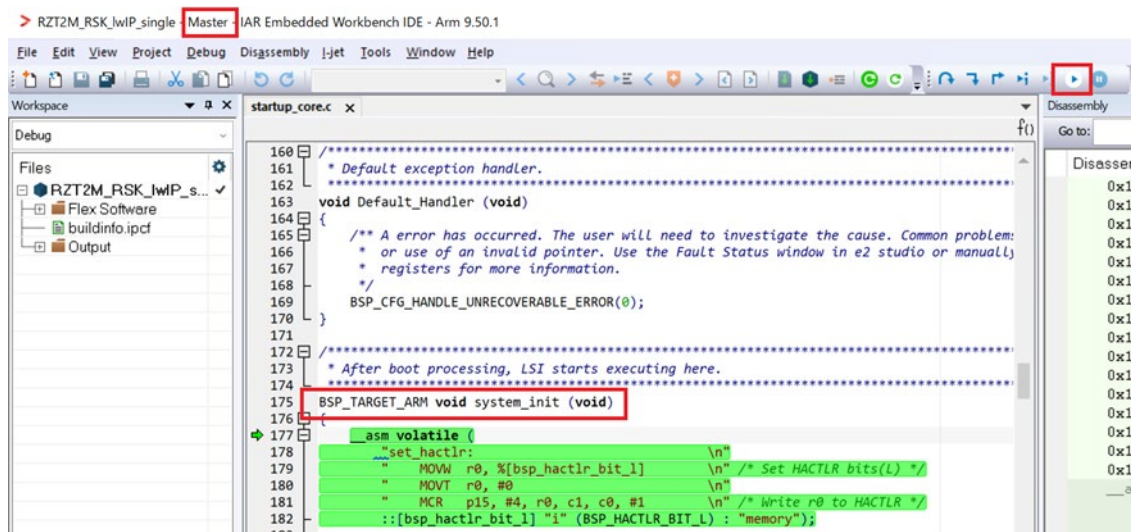


Figure 6.21 CPU0 Go button

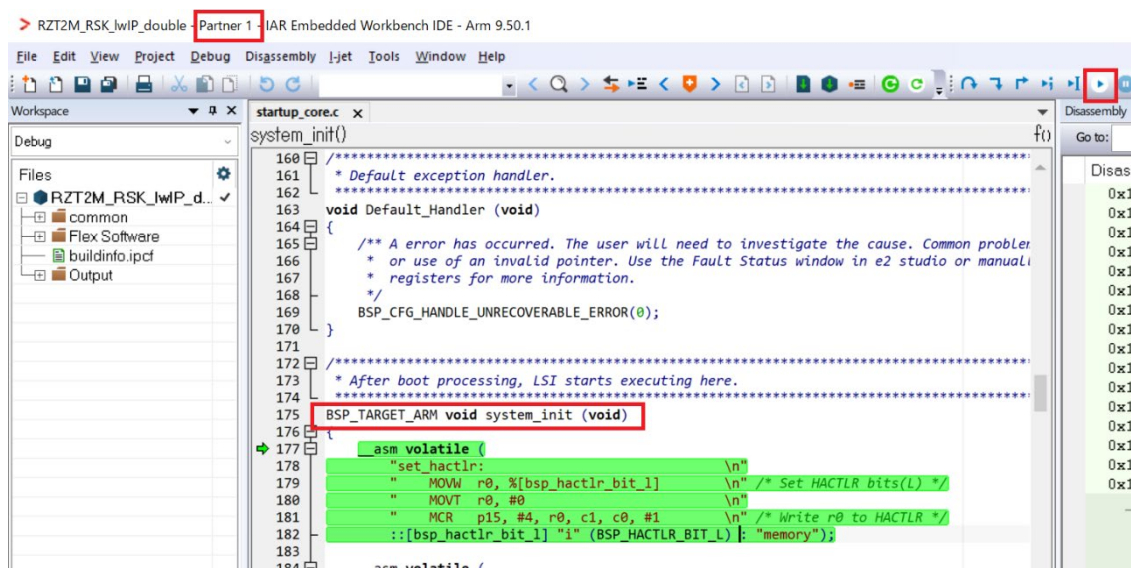


Figure 6.22 CPU1 Go button

7. Demonstration of the Sample Application

The sample program demonstrates the echo server function and the lwIP add-on, "iperf" and "SNTP," which are located in the "apps" folder and can be executed.

7.1 Connecting to a PC

Follow the steps below to connect the evaluation board to the communication PC.

1. Connect the PC to the "mini-USB" port indicated in [4. Hardware Settings](#) using a USB cable.
2. To display log data from the evaluation board, launch terminal software such as "Tera Term" using the following protocol.

We will use "Tera Term" here.

- Port: Port added by connecting to the evaluation board
- Speed: 115200
- Other: DATA8 / NONE / 1 bit / Local Echo OFF

3. Connect the Ethernet port shown in [4. Hardware Settings](#) to your PC's NIC using a LAN cable.

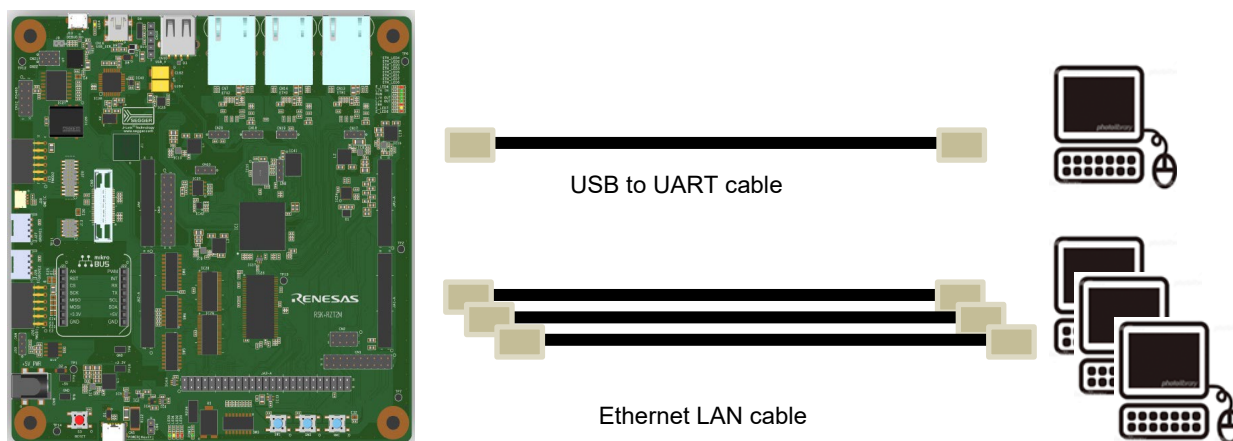


Figure 7.1 Communication terminal connection

4. Startup Screen

As shown in the figure above, when you connect the UART to the PC and start up the evaluation board, the following screen appears in the PC's terminal software.

```
>>Info : Started Serial I/O interface.
>>Info : lwIP initialization successful
>>Info : Initialized Ethernet network interface.
>>Info : Successfully registered lwIP as a callback to Ethernet.
>>Info : Started Ethernet network interface.
>>Info : <<< Started sample application (v2.0) for lwIP Port. >>>
>>Info : =====
>>Info : lwIP API routing mode ENABLED
>>Info : lwIP ETH_PAD_SIZE = 0
>>Info : ETHSW_SPECIFIC_TAG_DISABLE
>>Info : =====
>>Info : Please connect the IPERF LAN cable to the port.
>>Info : link-up port( NIC_0 )
>>Info : link-up port( NIC_1 )
>>Info : link-up port( NIC_2 )
>>Info : << Start the SNTP client for 15 seconds. >>
>>Info : << SNTP client application completed [ 25.07.2026 07:08:18 ] >>
>>Info : << IPERF2 server application start >>
>>Info : << The echo server will be start. >>
```

Figure 7.2 Startup screen

7.2 SNTP Client

When this sample program is launched, if it can connect to an NTP server, it will acquire and display time data for 15 seconds. If data cannot be retrieved within 15 seconds, a log message indicating this is displayed, and the program proceeds to the next process.

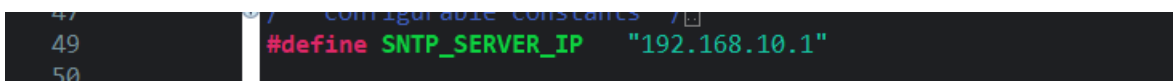
1. SNTP Configuration

- Verifying the NTP server IP address

Please modify the NTP server IP address to match your operating environment, referring to the settings in the source file below.

Source file: src/renesas/application/lwip_add_on_main.c

Note: There is no difference between e² studio and EWARM.



```
47 // Configurable constants
49 #define SNTP_SERVER_IP "192.168.10.1"
50
```

Figure 7.3 NTP server IP address

2. NTP Server Configuration

For this sample program, operation was verified using the Windows NTP server function. The NTP server is disabled by default but can be enabled by modifying the registry.

Please note that NTP servers are not supported, so please check the web or other resources for setup instructions.

7.3 iperf Client/Server

This sample program uses iperf to measure the Ethernet baud rate.

- PC-side application: iperf2-1-9win (freeware)
- Evaluation board: Add-on located in the lwIP "apps" folder

7.3.1 Procedure for measuring communication speed

Perform the measurement according to the following steps:

- Start the lwIP sample program.
The following message appears on the startup screen:
`"<< IPERF2 server application start >>"`
- Start the server from the PC's terminal software using the following command:
`"iperf-2-1-9-win.exe -s"`
- From the PC, verify the destination IP address and transmit data for 10 seconds using the following command:
`"iperf-2-1-9-win.exe -c 192.168.10.170"`
- Once communication from the PC completes, the evaluation board automatically transmits data back to the source (PC) address for 10 seconds.

Transmission and reception communication speeds are not displayed on the evaluation board's terminal; please check them on the PC's terminal.

The iperf parameters are as follows:

- PC
 - Client: `iperf-2-1-9-win.exe -c 192.168.10.170`
 - Server: `iperf-2-1-9-win.exe -s`
- Evaluation Board
 - Client: Uses the source address from the received data; other settings are default.
 - Server: IP address set to ANY; other settings are default.

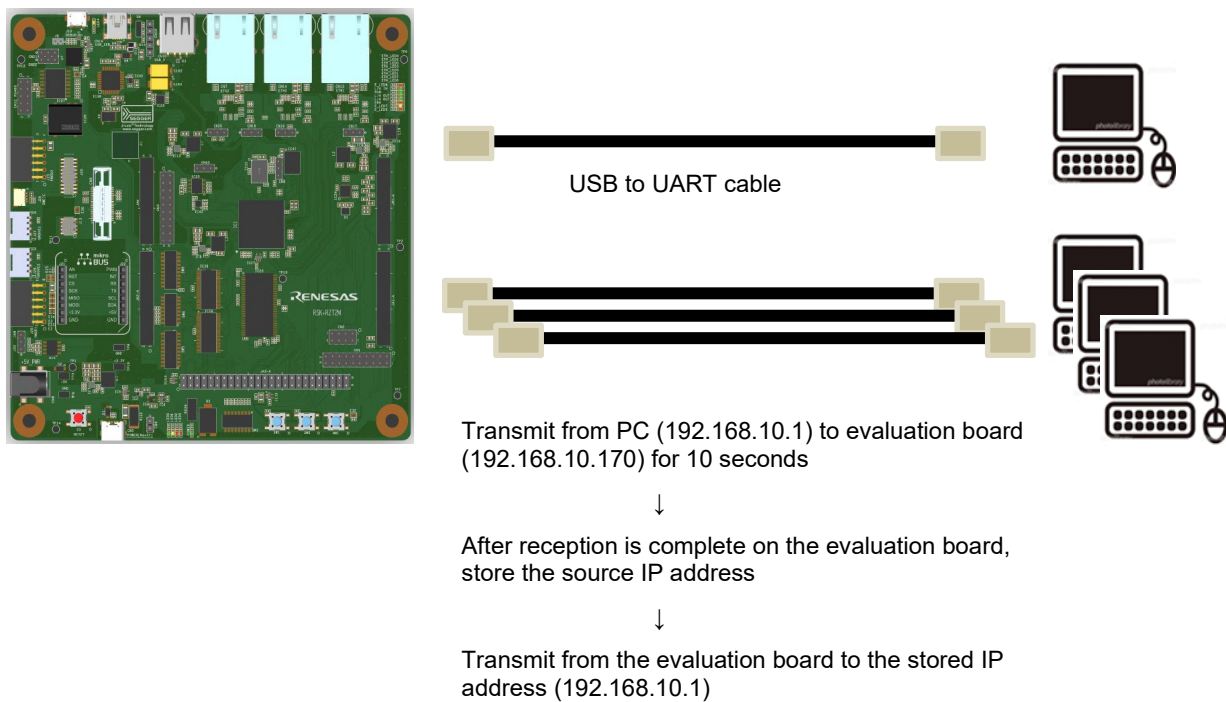


Figure 7.4 Communication sequence

7.4 Echo Server

In this sample program, the echo server is always running, so after startup, you can communicate via TELNET from your PC's terminal software.

- TELNET port number: As configured in "[5.1. Ethernet Property Settings](#)"
- IP address: One of the IP addresses configured in "[5.1. Ethernet Property Settings](#)"
- Maximum number of connections: 10

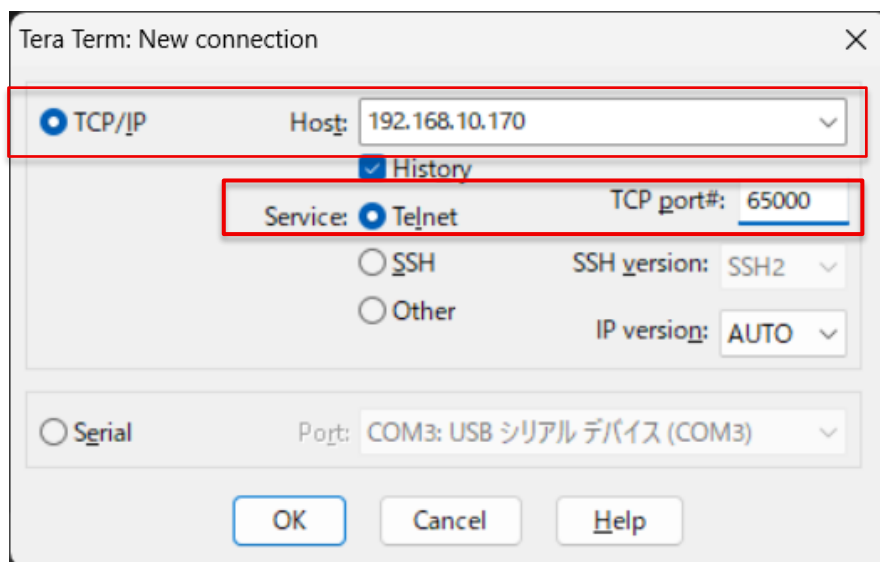


Figure 7.5 Tera Term Configuration

8. Software Specifications

8.1 Software Structure

An overview of the software is shown below.

Legend in the figure: () Destination of the corresponding source file

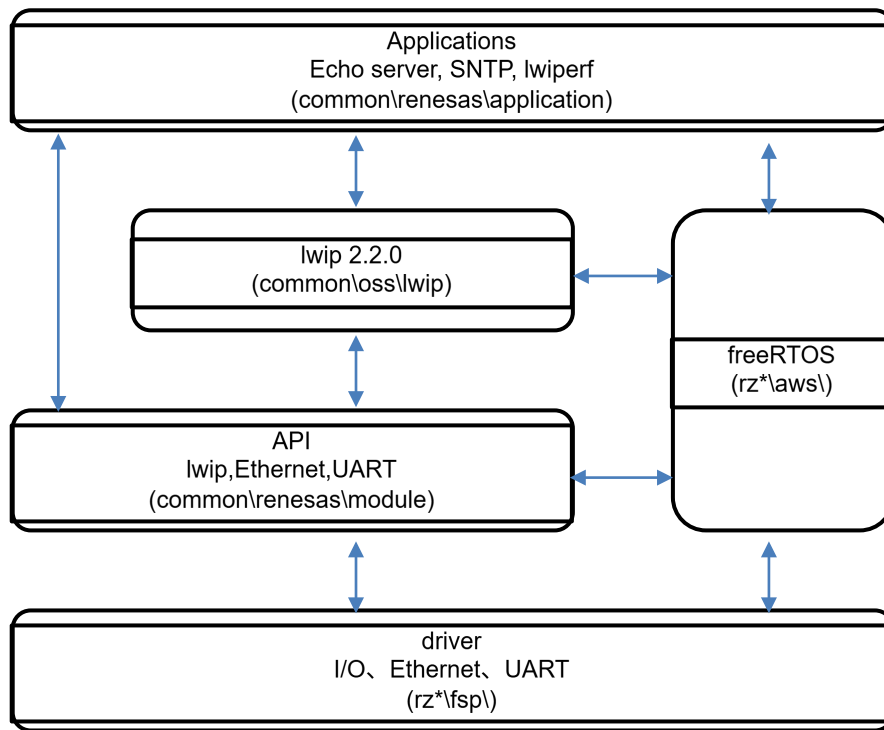


Figure 8.1. Software Architecture Diagram

8.1.1 Components Overview.

Functionality overview in the overview diagram

Table 8.1. Overview of Each Component

Group	Folder Name	Summary
Applications	common/renesas/application	Demonstration App
	SRC	Entry functions etc.
API	common/renesas/module/ether_netif	Ethernet API
	common/renesas/module/lwip_port	lwip API
	common/renesas/module/serial_io	UART API
OS	Rz*/aws/	freeRTOS
driver	rz*/fsp/	Ethernet driver
	rz*/fsp/	UART driver

8.1.2 Drivers Used

The stacks used in FSP are shown below.

Table 8.2. Sample Software Support Status

Stack Name	function
g_ioport I/O Port Driver	Pin Settings, Control
g_ether0 Ethernet	Ethernet communication
g_uart0 UART	Serial communication
Heap 4	Allocating heap for FreeRTOS

8.2 Task

The task properties used in the sample program are shown below.

Table 8.3. Task List

Module	Task name	Priority	Stack size [bytes]	Synchronous mode	Overview
Main	Main thread	5	1024	Pause after initialization	Main task
Ethernet Port	Ethernet Monitor Task	3	1024	Wake up every 1 second	Ethernet link status monitoring
	Ethernet Reader Task	4	2048	Received events	Waiting for received Ethernet data
lwIP Port	iperf_client_task	4	1024	Waiting for lperf send event	lperf send event
lwIP Port	lwIP	4	2048	Queue notification events	LINK status notification processing
application	TCP Server task	3	1024	Received events	Reception processing within lwIP
	TCP Server task(No.)*	3	1024	Waiting for reception	Reception and transmission

* This is a client connection task; it can connect up to 10 lines, and the (No.) indicated will be a number from 1 to 10. the "(No.)" in the name corresponds to a number from 1 to 10. Note that the specifications for all tasks (1 through 10) are identical.

Table 8.4. RTOS Max Priority

Item	Value
Max Priority	10

8.2.1 Resources

The resources used are shown below.

Table 8.5. Used Resource

Resources used	Format	Module	Use
Heap4	Heap memory	All modules	Ethernet transmission and reception data
queue	Queue	Ethernet port	Link status notification

8.2.2 Task Operations

The following diagram shows the relationships between tasks.

1. Overview

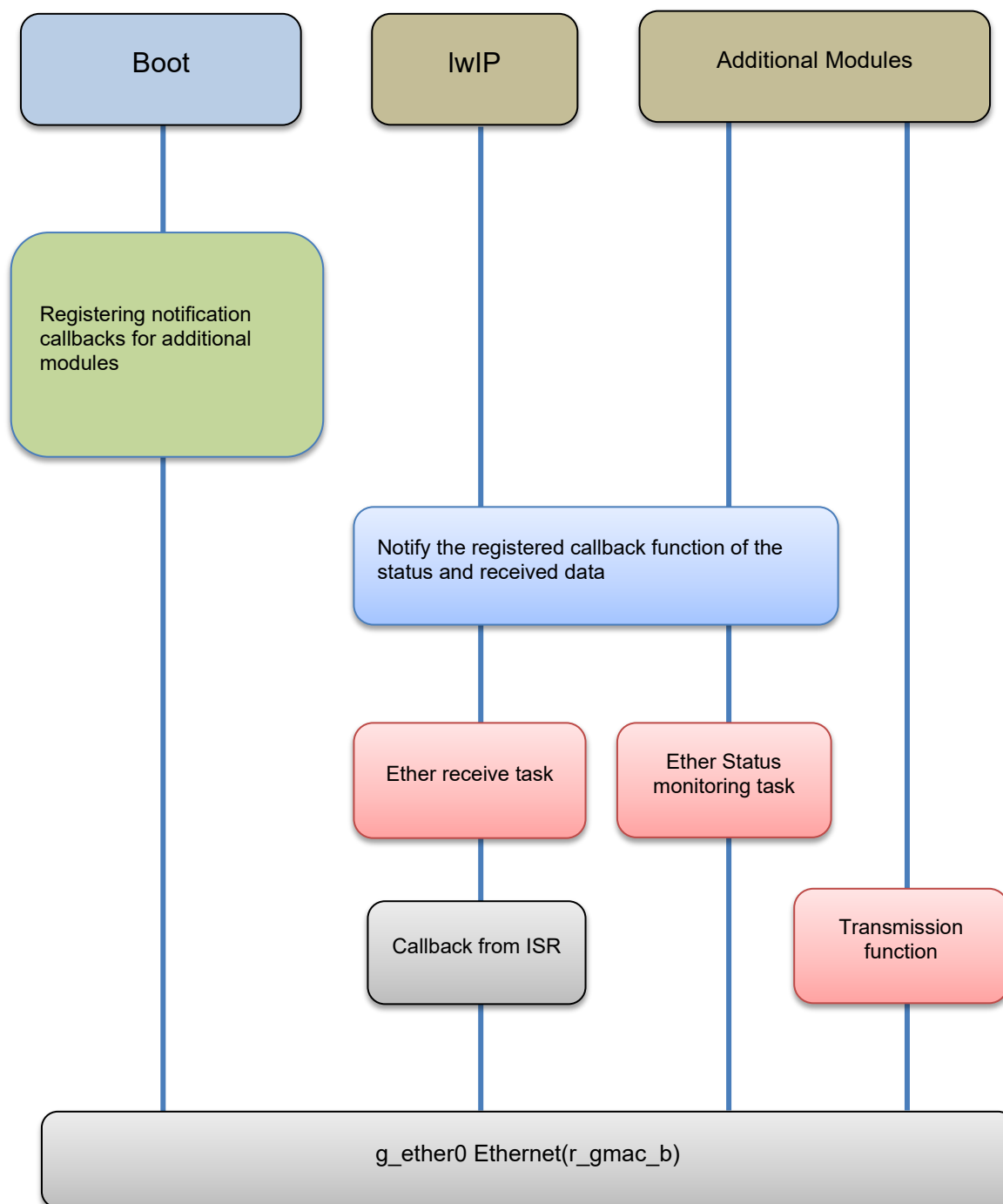


Figure 8.2. sequence

8.3 Received Packet Mirroring and Transmission Functions

Please refer to the following documentation when directly capturing received data from the driver or performing direct transmission without going through lwIP..

1. Overview



Legend :



Added processing



lwIP API

Figure 8.3. sequence

2. Mirroring

To enable the mirroring function, register a callback function using the Ethernet API to receive the LINK status and received data (MAC frames).

➤ Parameters when registering a callback function

Please create it as follows, referring to the existing process.

Table 8.6. Callback function parameters

Data Format	Overview
struct st_ether_netif_callback_link_node * p_next;	Initial value: NULL; writing is prohibited except for the initial value.
void (* p_func) (ether_netif_callback_args_t *);	Callback function (function that receives notifications).
ether_netif_callback_args_t * p_memory;	Data (ether_netif_callback_args_t) set when the callback notification occurs.
request_port_t Request_port;	Specifies the target port(s) for the callback notification. To specify multiple ports, combine the values for each port using a bitwise OR operation. 1: ETH0, 2: ETH1, 4: ETH2
void const * p_context;	Data to be added to the callback parameters.

➤ Callback registration function

```
usr_err = gp_ether_netif0->p_api->callbackAddstart(gp_ether_netif0->p_ctrl,
ether_netif_callback_link_node_t *);
```

➤ Parameters for the callback function

Table 8.7. Callback parameters

ether_netif_callback_args_t	Data is set during the callback.
ether_netif_callback_event_t event	Callback type
ether_netif_frame_t * p_frame_packet	Received data
ether_netif_monitor_queue_t * p_port_status	Port status at the time of the event
void const * p_context;	Data specified during callback registration

3. Transmission Function

To transmit a MAC frame without going through lwIP, use the function as follows:

➤ Transmission function parameters

Create a new `static ether\_netif\_frame\_t` and initialize it to zero.

- buffer[] : Copy the data to be transmitted here.
- length : Transmission length
- port : Destination port number (1: ETH0, 2: ETH1, 4: ETH2)

➤ Transmission function

```
usr_err = gp_ether_netif0->p_api->send(gp_ether_netif0->p_ctrl, ether_netif_frame_t *);
```

8.4 Multi-interface

8.4.1 Interface Property Table

```

94  /** Initial settings data such as IP address and MAC address */
95  static lwip_port_netif_ip_cfg_t g_lwip_port_netif_cfg[LWIP_NIC_MAX] =
96  {
97      {
98  #if LWIP_NIC_MAX > 1
99          .p_next      = &g_lwip_port_netif_cfg[1],
100  #else
101          .p_next      = NULL,
102  #endif
103          .dhcp        = LWIP_PORT_DHCP_DISABLE,
104          .ip_address   = PP_HTONL(LWIP_MAKEU32(192,168, 10,170)),
105          .subnet_mask  = PP_HTONL(LWIP_MAKEU32(255,255,255, 0)),
106          .gateway_address = PP_HTONL(LWIP_MAKEU32(192,168, 10, 1)),
107          .hwaddr       = {0x00,0x11,0x22,0x33,0x44,0x55},
108          .p_host_name  = gp_lwip_port0_hostname_0,
109          .eth_port     = ETHER_PORT_0,
110      },
111  #if LWIP_NIC_MAX > 1
112      {
113          .p_next      = &g_lwip_port_netif_cfg[2],
114          .dhcp        = LWIP_PORT_DHCP_DISABLE,
115          .ip_address   = PP_HTONL(LWIP_MAKEU32(192,168, 10,171)),
116          .subnet_mask  = PP_HTONL(LWIP_MAKEU32(255,255,255, 0)),
117          .gateway_address = PP_HTONL(LWIP_MAKEU32(192,168, 10, 2)),
118          .hwaddr       = {0x00,0x11,0x22,0x33,0x44,0x77},
119          .p_host_name  = gp_lwip_port0_hostname_1,
120          .eth_port     = ETHER_PORT_1,
121      },
122  #endif
123  #if LWIP_NIC_MAX > 2
124      {
125          .p_next      = NULL,
126          .dhcp        = LWIP_PORT_DHCP_DISABLE,
127          .ip_address   = PP_HTONL(LWIP_MAKEU32(192,168, 10,172)),
128          .subnet_mask  = PP_HTONL(LWIP_MAKEU32(255,255,255, 0)),
129          .gateway_address = PP_HTONL(LWIP_MAKEU32(192,168, 10, 3)),
130          .hwaddr       = {0x00,0x11,0x22,0x33,0x44,0x88},
131          .p_host_name  = gp_lwip_port0_hostname_2,
132          .eth_port     = ETHER_PORT_2,
133      }
134  #endif
135  };
136

```

Figure 8.4. sequence

1. Property Details

Details of the property items are provided below.

Table 8.8. InterfaceProperties

Variables	overview
.p_next	Pointer to an array of interface structures Set this data to "NULL" to limit the number of property data entries; any subsequent data will be ignored.
.dhcp	Presence or absence of DHCP function
.ip_address	IP address
.subnet_mask	Subnet mask
.gateway_address	Default gateway (connected host PC)
.hwaddr	MAC address
.p_host_name	Network name
.eth_port	LAN port(s) used: 1: ETH0, 2: ETH1, 4: ETH2 (use a logical OR for multiple ports) This enables callbacks for received data and LINK status for the LAN ports specified by the bitmask.

2. Notes on Properties

- IP addresses and MAC addresses cannot duplicate those of other interface properties.
- If multiple LAN ports are specified in the LAN port settings with the intention of allowing cable swapping, connect the LAN cable to only one of the specified ports.

If different IP and MAC addresses are required, do not simply swap the LAN cable; instead, configure two interfaces and use routing to direct traffic intended for a single IP address.

8.4.2 Routing

With multi-interface configuration, the Ethernet driver and lwIP associate received frames with the corresponding NETIF based on their MAC addresses and pass them to lwIP.

1. Overview Diagram

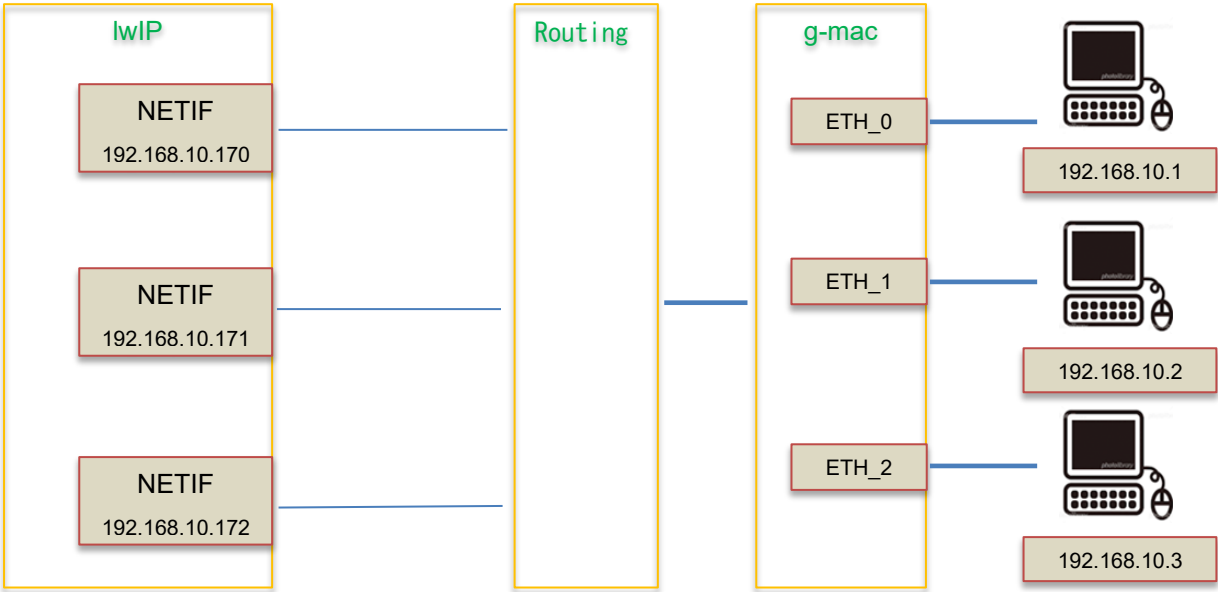


Figure 8.5. Routing Overview Diagram

2. Examples of IP address combinations that allow communication via routing

Table 8.9. IP address combinations

PC 192.168.10.1 → 192.168.10.170 → 192.168.10.171 → 192.168.10.172	lwIP 192.168.10.170 → 192.168.10.1 → 192.168.10.2 → 192.168.10.3
PC 192.168.10.2 → 192.168.10.170 → 192.168.10.171 → 192.168.10.172	lwIP 192.168.10.171 → 192.168.10.1 → 192.168.10.2 → 192.168.10.3
PC 192.168.10.3 → 192.168.10.170 → 192.168.10.171 → 192.168.10.172	lwIP 192.168.10.172 → 192.168.10.1 → 192.168.10.2 → 192.168.10.3

8.4.3 Management port specific tag + Non-routing mode

When "Management port specific tag" is set to "Enable" and routing is disabled, communication proceeds as follows.

1. Overview Diagram

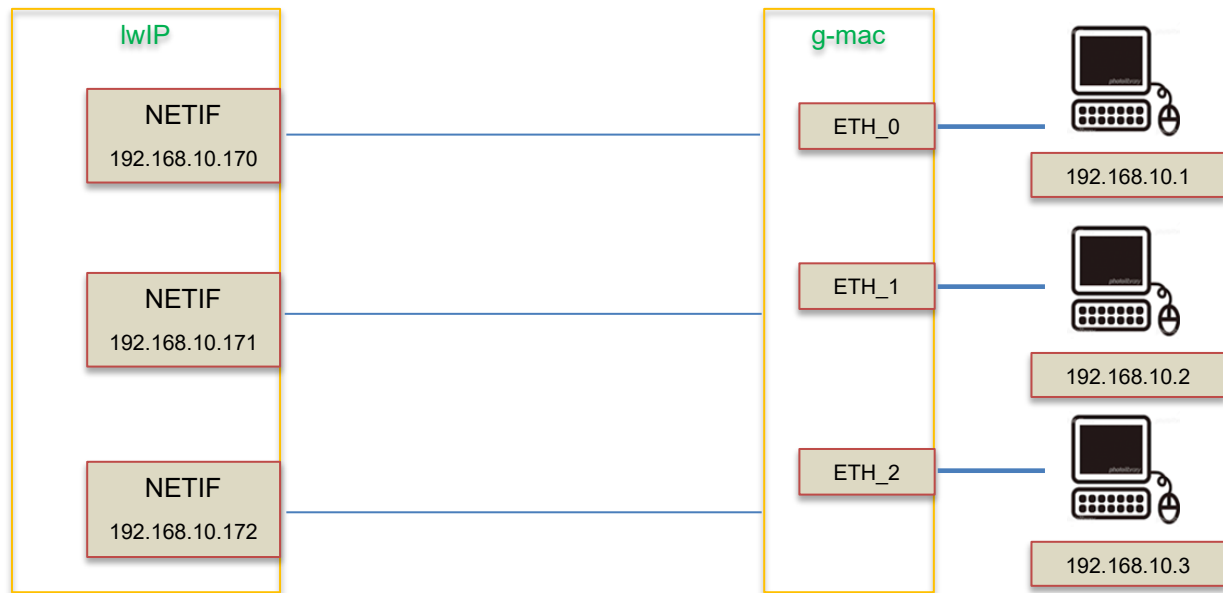


Figure 8.6. Non-routing overview diagram

2. Examples of communicable IP address combinations

Table 8.10. IP address combinations

PC 192.168.10.1 → 192.168.10.170	lwIP 192.168.10.170 → 192.168.10.1
PC 192.168.10.2 → 192.168.10.171	lwIP 192.168.10.171 → 192.168.10.2
PC 192.168.10.3 → 192.168.10.172	lwIP 192.168.10.172 → 192.168.10.3

8.5 Memory Allocation

The following three groups are placed in a project.

- Loader Program
Startup processing and initialization of the clock, interrupts, IO, etc. are placed in the BTCM.
- User Program
Processing other than the loader program, such as the RTOS, interrupt vectors, drivers, and applications, is placed in the ATCM or system RAM.
- Shared Non-Cache
Used as a storage location for Ethernet transmission and reception data, etc., this is placed in the mirror area of system RAM.

8.5.1 e²studio project

These definition files are located in the "script" folder.

- xxxxxxxx.ld
This is a link script file; the file name changes depending on the boot mode, etc.
- xxxxxxxx.h
This is a physical memory definition file, and defines the address of the implemented device, etc.

1. RZ/T2H

The memory usage by group is shown below.

■ For CR52 and CR52 projects

➤ CPU0(CR52_0)

The only application that runs on CPU0 is the LED blinking function.

RZT2H Memory Map

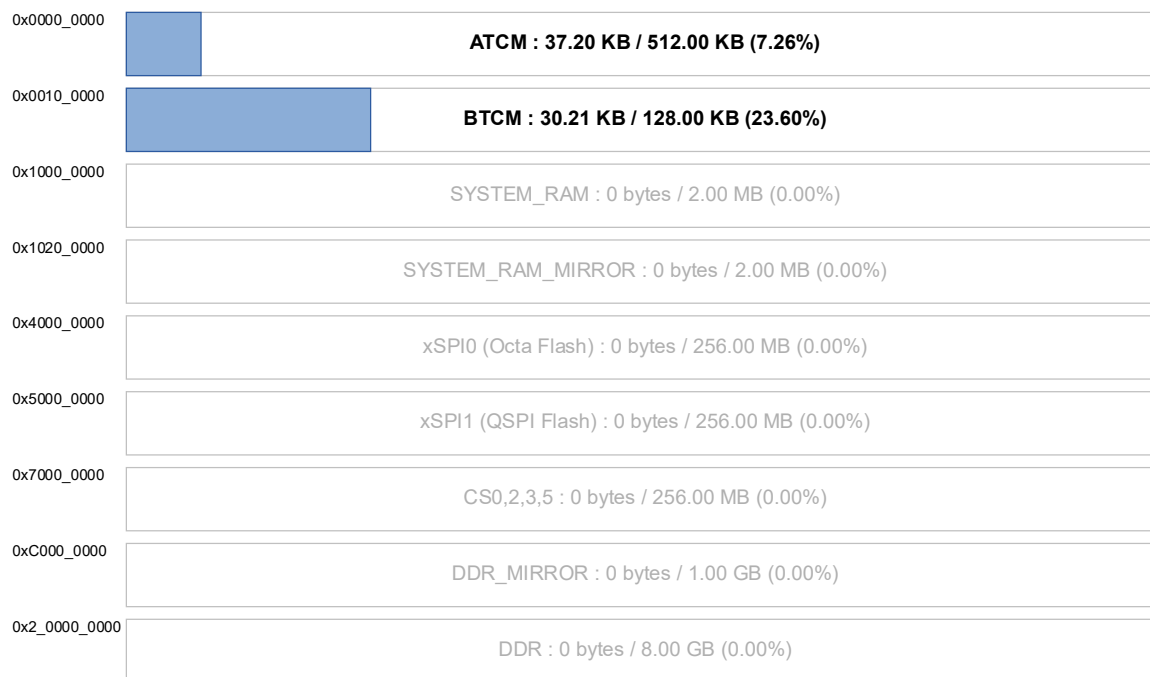


Figure 8.7. Memory footprint of the CPU0 project (e² studio)

ATCM

Total Usage : 37.20 KB

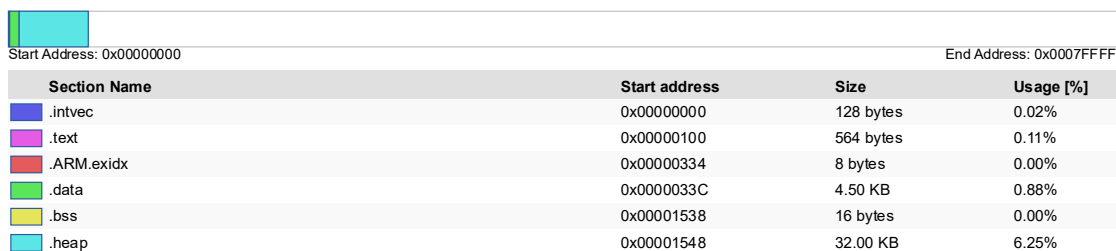


Figure 8.8. ATCM footprint of the CPU0 project (e² studio)

BTCM

Total Usage : 30.21 KB

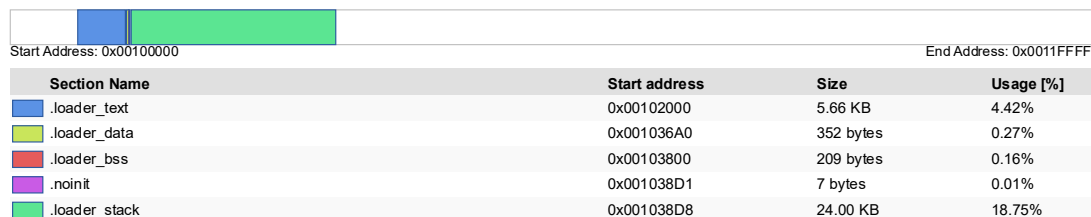


Figure 8.9. BTCM footprint of the CPU0 project (e² studio)

► CPU1(CR52_1)

The application running on CPU1 will perform functions other than blinking the LED.

RZT2H Memory Map

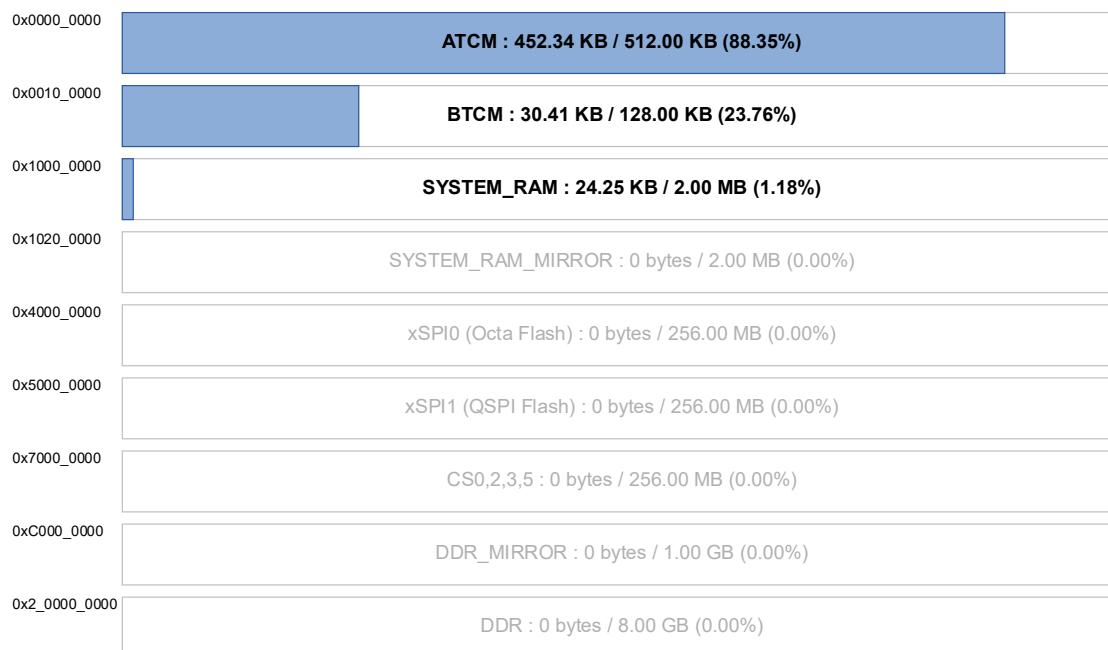


Figure 8.10. Memory footprint of the CPU1 project (e² studio)

ATCM

Total Usage : 452.34 KB

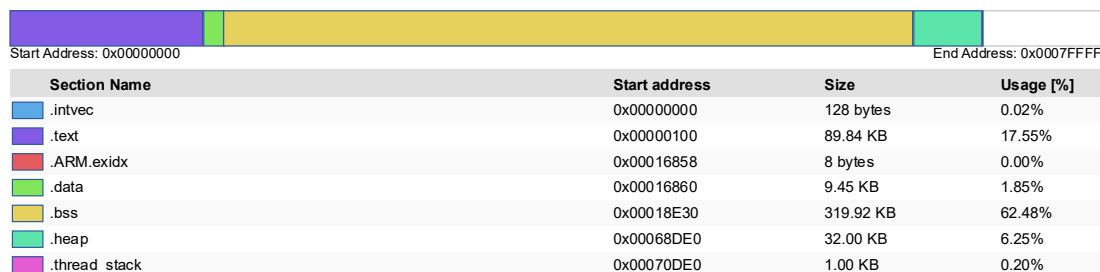


Figure 8.11. ATCM footprint of the CPU1 project (e² studio)

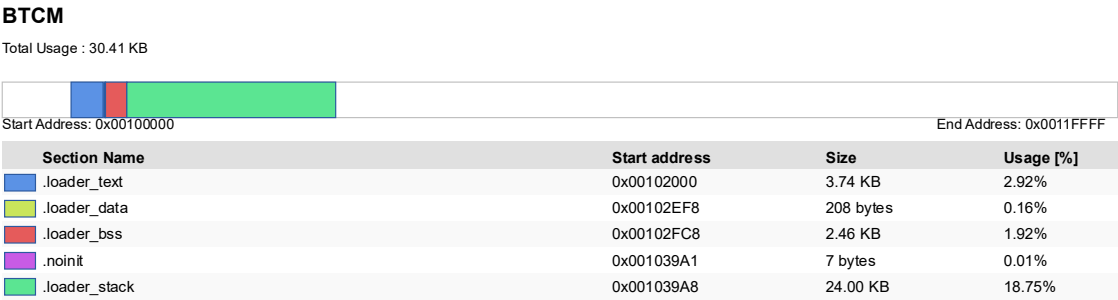


Figure 8.12. BTCM footprint of the CPU1 project (e² studio)

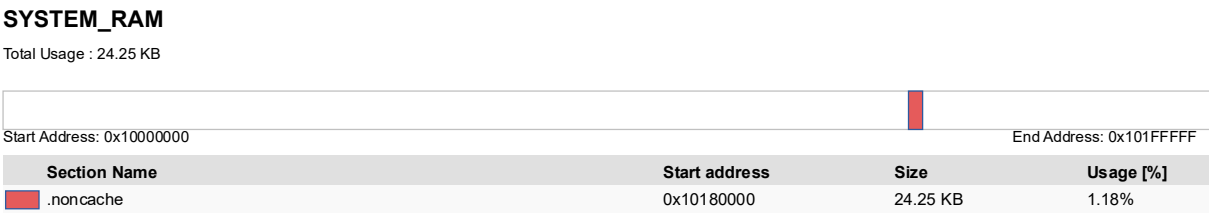


Figure 8.13. SYSTEM_RAM footprint of the CPU1 project (e² studio)

- For CR52 and CA55 projects
 - CPU0(CR52_0)
 - The only application that runs on CPU0 is the LED blinking function.

RZT2H Memory Map

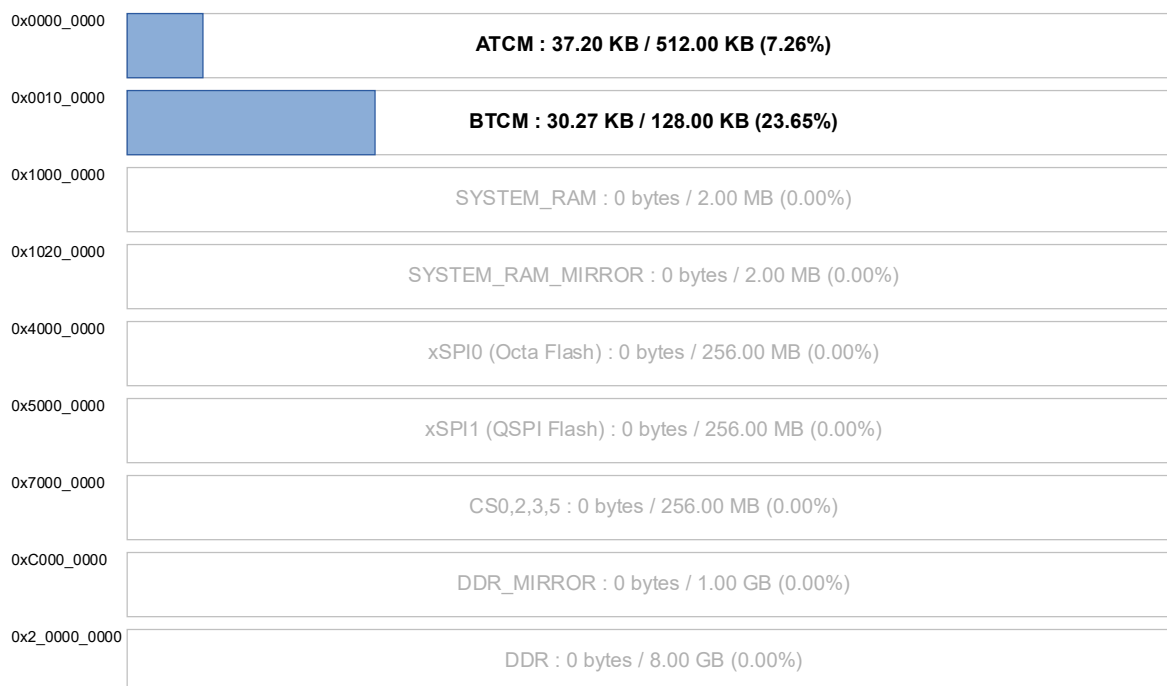


Figure 8.14. Memory footprint of the CPU0 project (e² studio)

ATCM

Total Usage : 37.20 KB

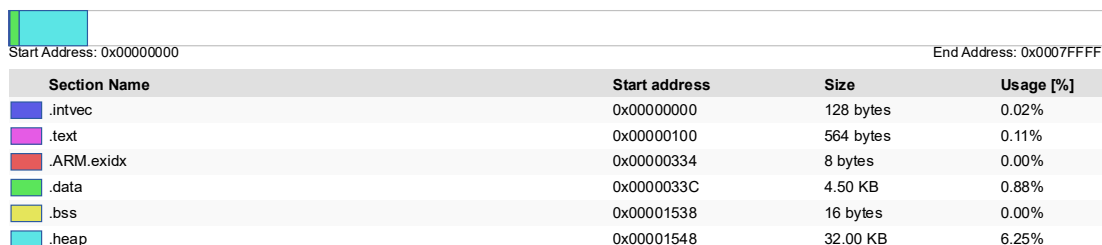


Figure 8.15. ATCM footprint of the CPU0 project (e² studio)

BTCM

Total Usage : 30.27 KB

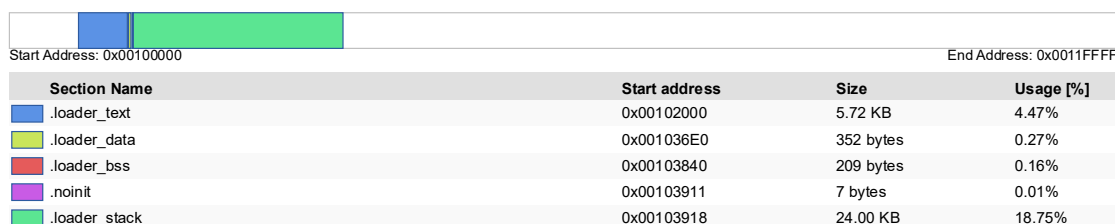


Figure 8.16. BTCM footprint of the CPU0 project (e² studio)

► CPU1(CA55_2)

The application running on CPU1 will perform functions other than blinking the LED.

RZT2H Memory Map

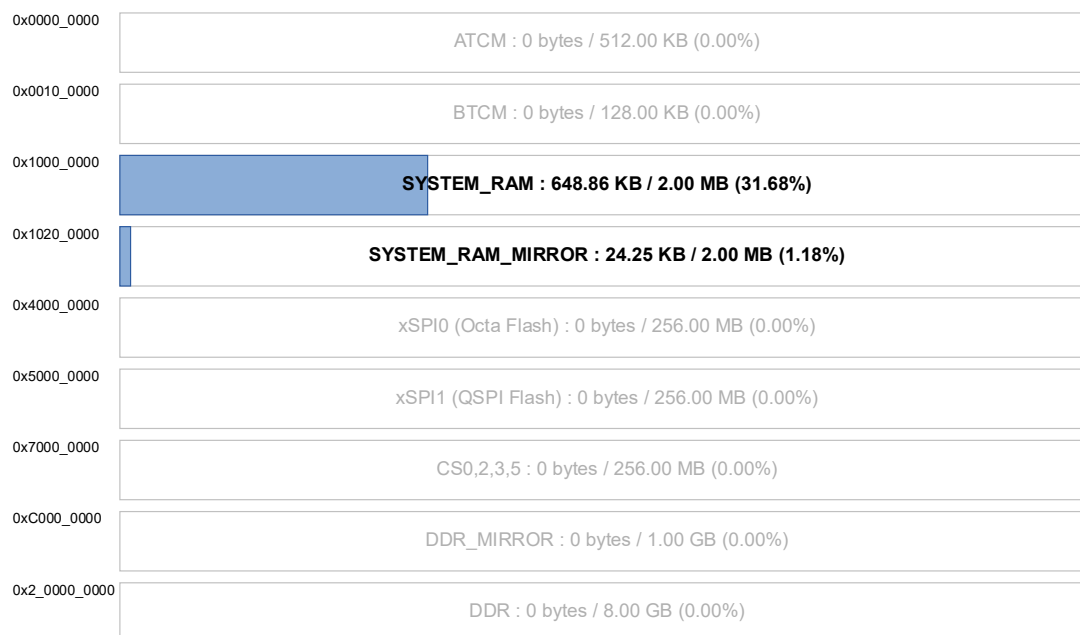


Figure 8.17. Memory footprint of the CPU1 project (e² studio)

SYSTEM_RAM

Total Usage : 648.86 KB

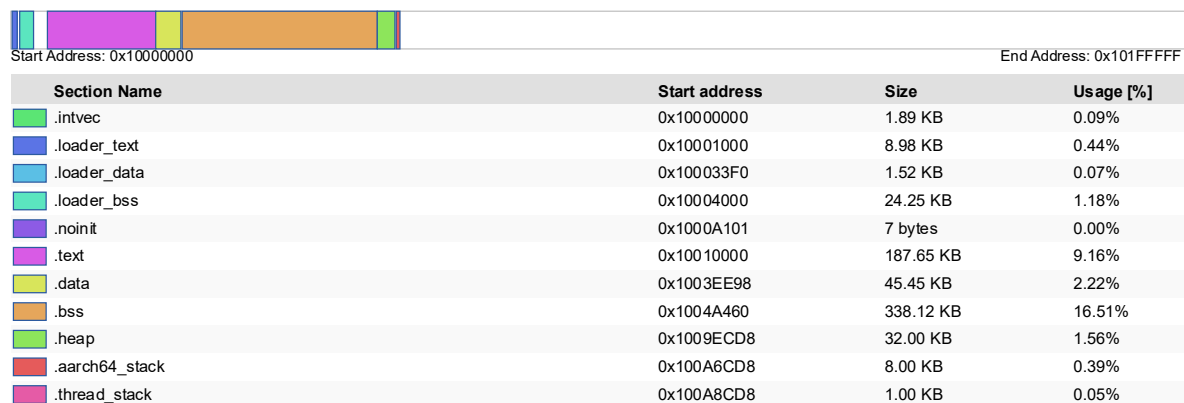


Figure 8.18. SYSTEM_RAM footprint of the CPU1 project (e² studio)

SYSTEM_RAM_MIRROR

Total Usage : 24.25 KB



Figure 8.19. SYSTEM_RAM(MIRROR) footprint of the CPU1 project (e² studio)

2. RZ/N2H

The memory usage by group is shown below.

- For CR52 and CR52 projects

- CPU0(CR52_0)

The only application that runs on CPU0 is the LED blinking function.

RZN2H Memory Map

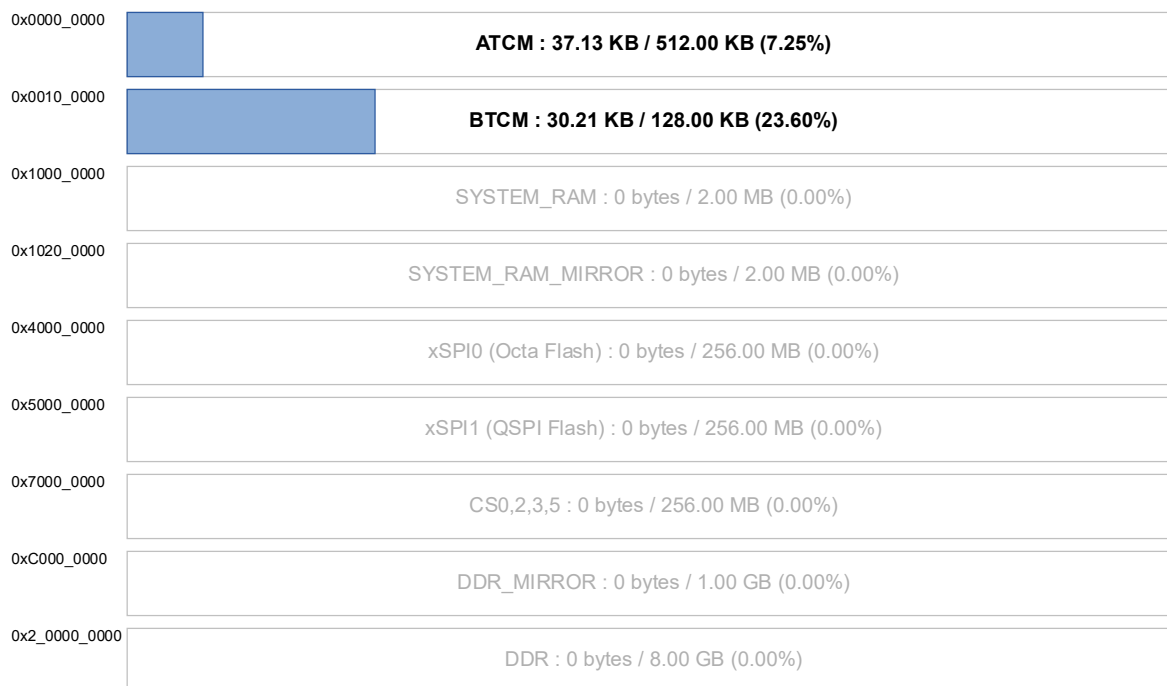


Figure 8.20. Memory footprint of the CPU0 project (e² studio)

ATCM

Total Usage : 37.13 KB

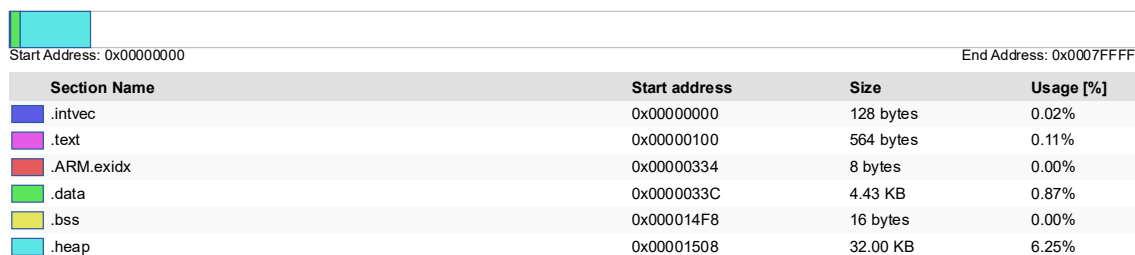


Figure 8.21. ATCM footprint of the CPU0 project (e² studio)

BTCM

Total Usage : 30.21 KB

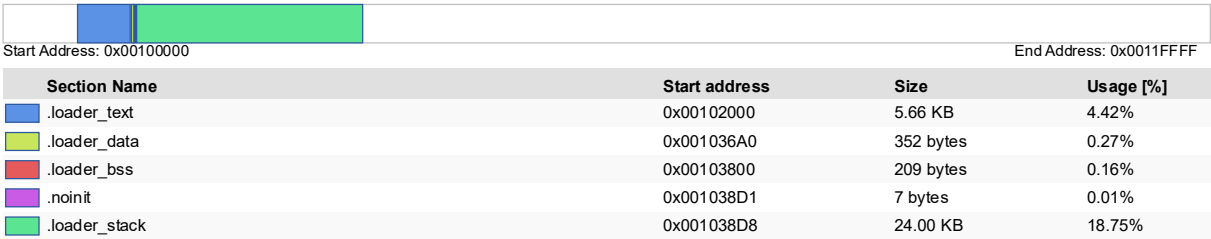


Figure 8.22. BTCM footprint of the CPU0 project (e² studio)

► CPU1(CR52_1)

The application running on CPU1 will perform functions other than blinking the LED.

RZN2H Memory Map

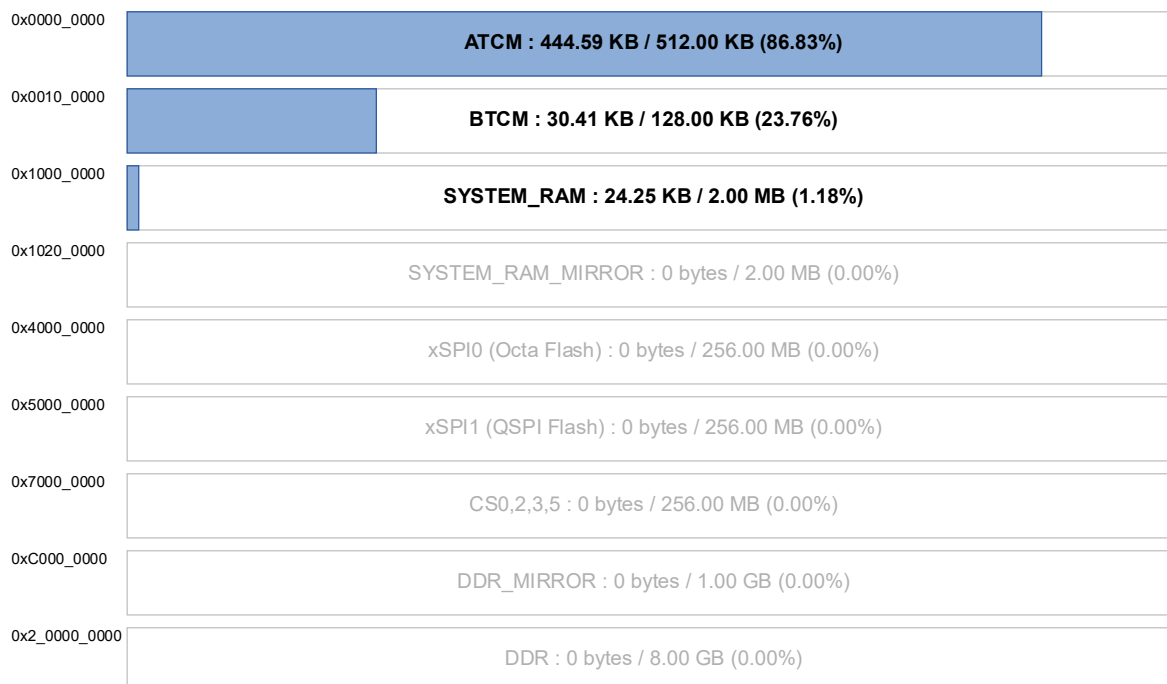


Figure 8.23. Memory footprint of the CPU1 project (e² studio)

ATCM

Total Usage : 444.59 KB

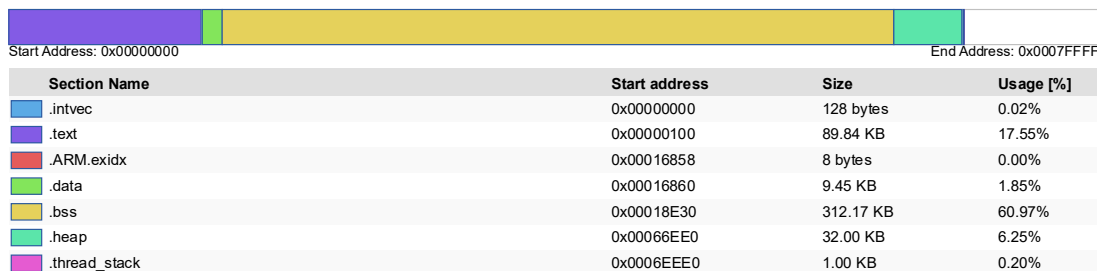


Figure 8.24. ATCM footprint of the CPU1 project (e² studio)

BTCM

Total Usage : 30.41 KB

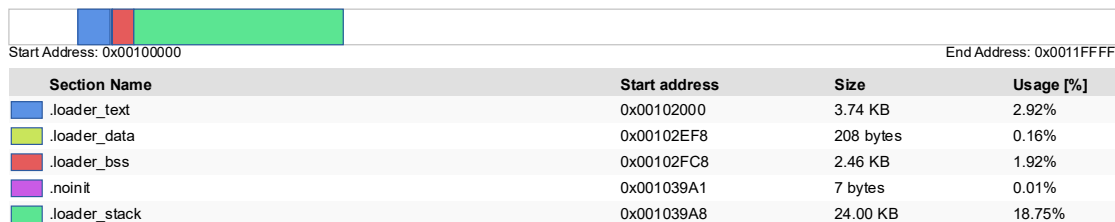


Figure 8.25. BTCM footprint of the CPU1 project (e² studio)

SYSTEM_RAM

Total Usage : 24.25 KB

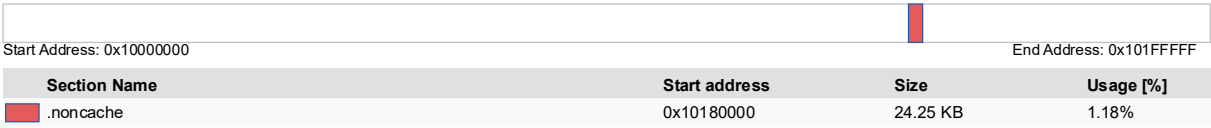


Figure 8.26. SYSTEM_RAM footprint of the CPU1 project (e² studio)

- For CR52 and CA55 projects
 - CPU0(CR52_0)
 - The only application that runs on CPU0 is the LED blinking function.

RZN2H Memory Map

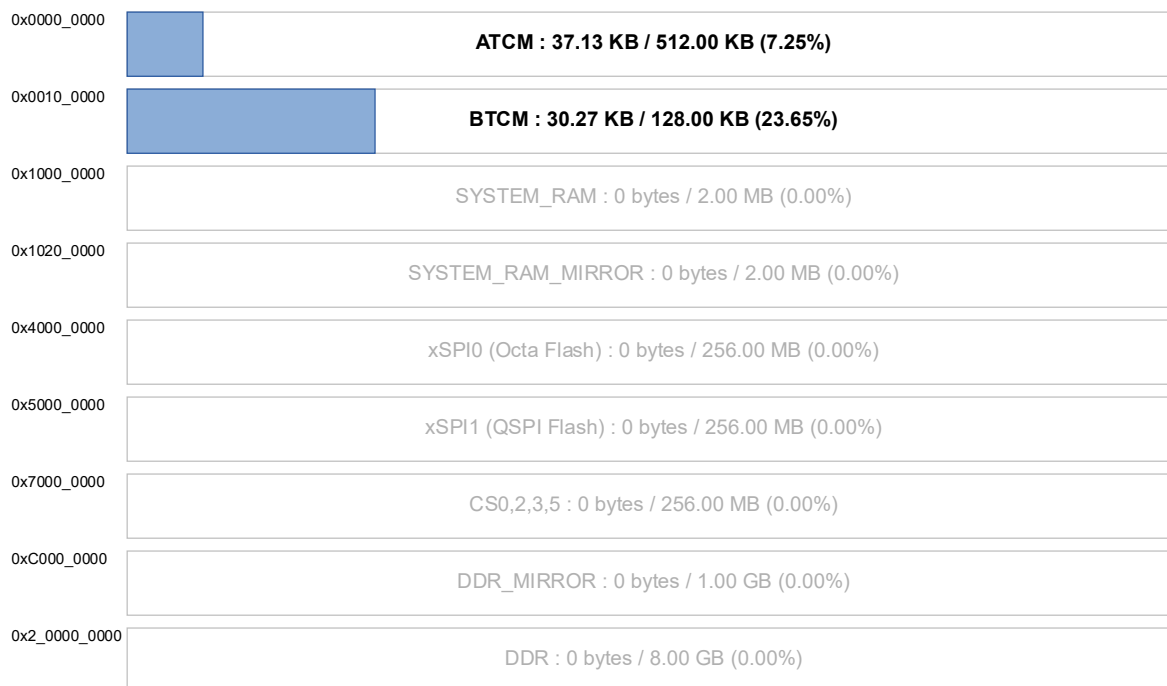


Figure 8.27. Memory footprint of the CPU0 project (e² studio)

ATCM

Total Usage : 37.13 KB

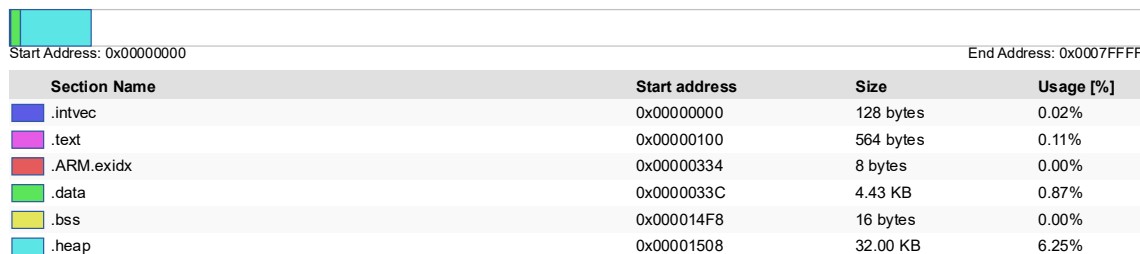


Figure 8.28. ATCM footprint of the CPU0 project (e² studio)

BTCM

Total Usage : 30.27 KB

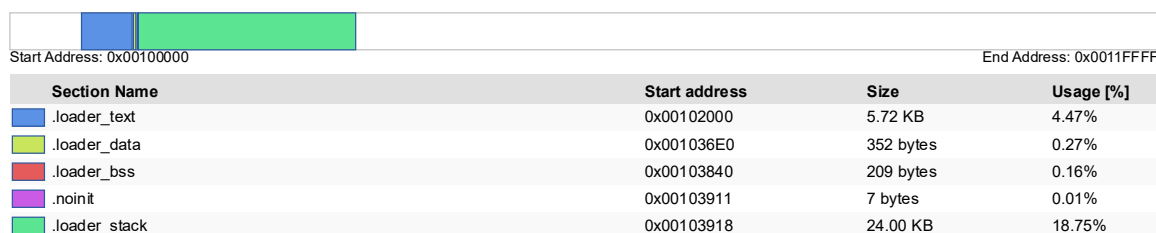


Figure 8.29. BTCM footprint of the CPU0 project (e² studio)

► CPU1(CA55_2)

The application running on CPU1 will perform functions other than blinking the LED.

RZN2H Memory Map

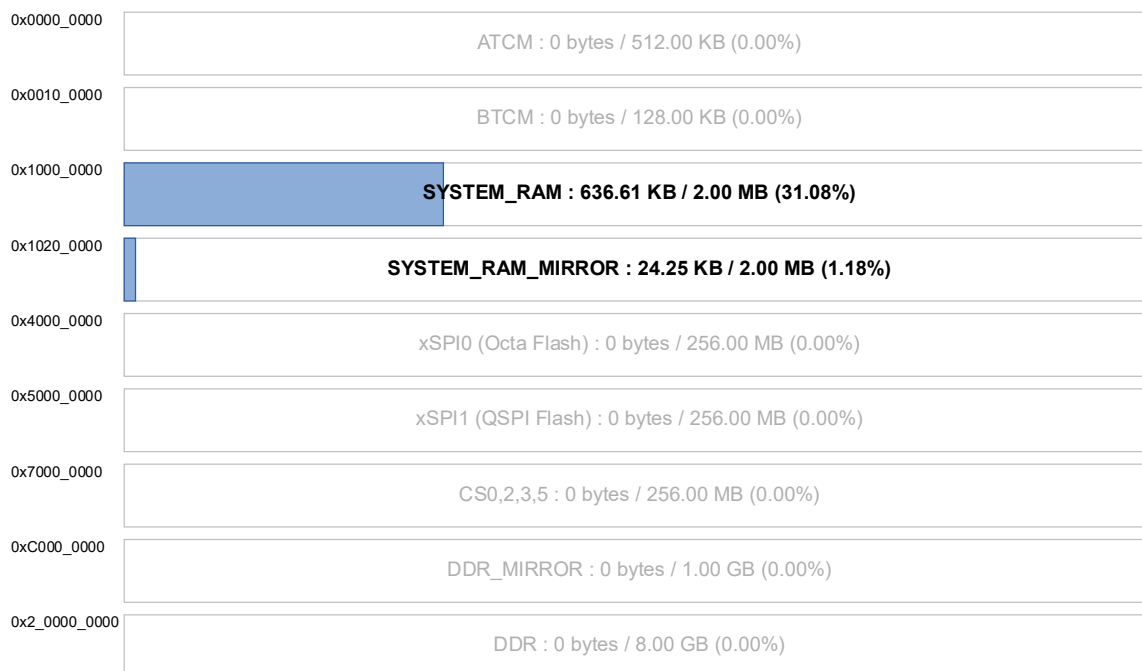


Figure 8.30. Memory footprint of the CPU1 project (e² studio)

SYSTEM_RAM

Total Usage : 636.61 KB

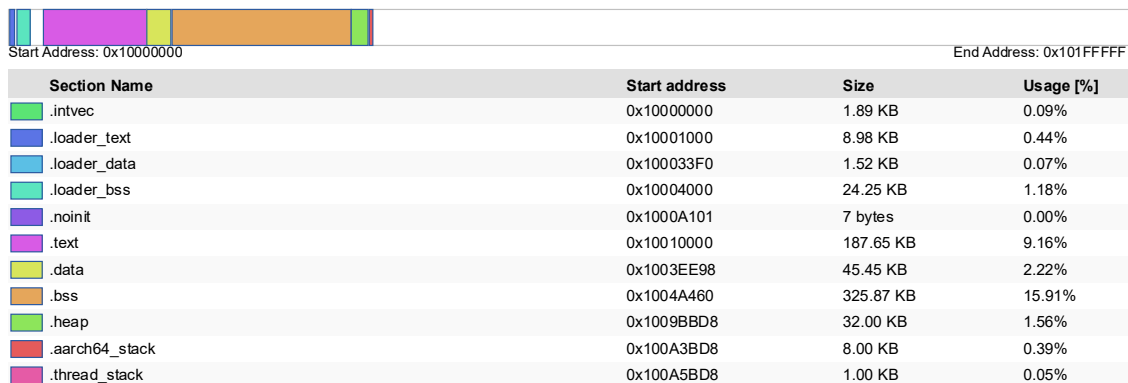


Figure 8.31. SYSTEM_RAM footprint of the CPU1 project (e² studio)

SYSTEM_RAM_MIRROR

Total Usage : 24.25 KB



Figure 8.32. SYSTEM_RAM (MIRROR) footprint of the CPU1 project (e² studio)

8.5.2 EWARM project

It is placed in the definition file below in the "script" folder.

► xxxxxxxx.icf

This is a link script file. The file name changes depending on the boot mode, etc.

► xxxxxxxxx.h

This is the physical memory definition file, and defines the addresses of the implemented devices.

1. RZ/T2H

The memory usage by group is shown below.

- For CR52 and CR52 projects
 - CPU0(CR52_0)

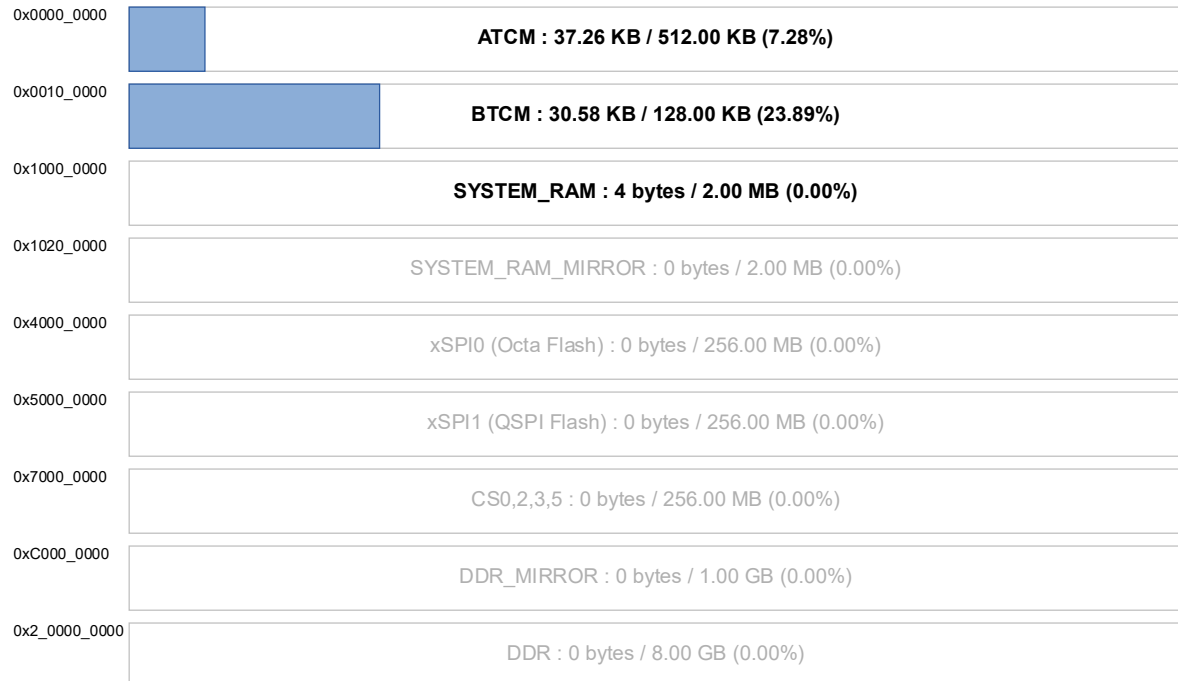
RZN2H Memory Map

Figure 8.33. Memory footprint of the CPU0 project (EWARM)

ATCM

Total Usage : 37.26 KB

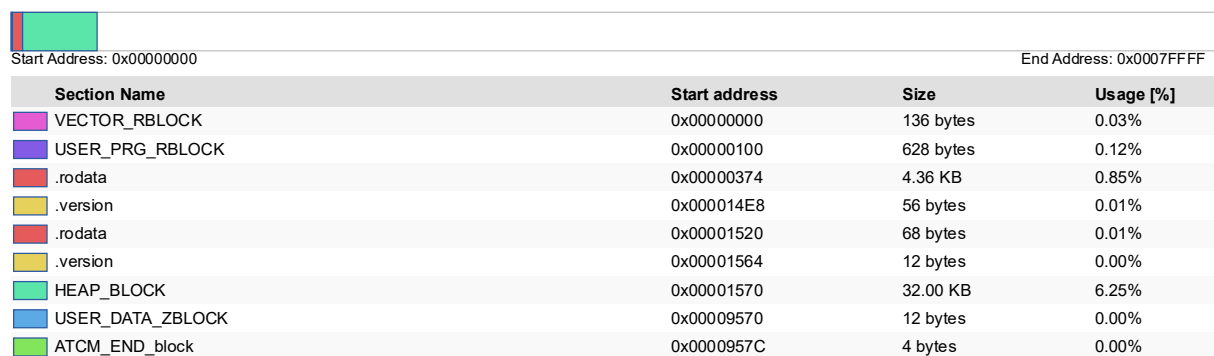


Figure 8.34. ATCM footprint of the CPU0 project (EWARM)

BTCM

Total Usage : 30.58 KB

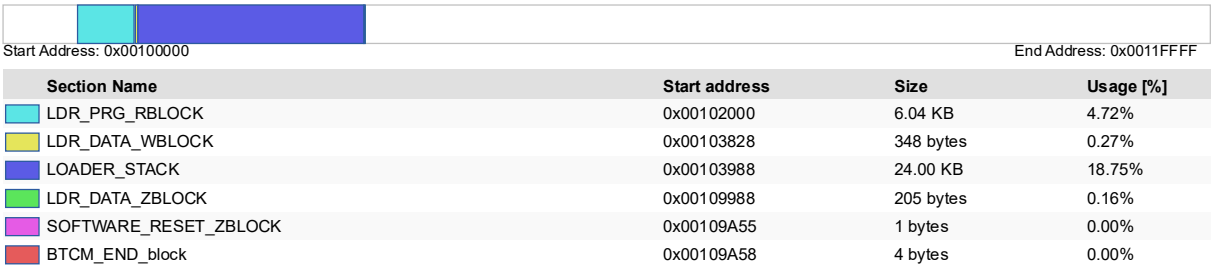


Figure 8.35. BTCM footprint of the CPU0 project (EWARM)

► CPU1(CR52_1)

The application running on CPU1 will perform functions other than blinking the LED.

RZN2H Memory Map

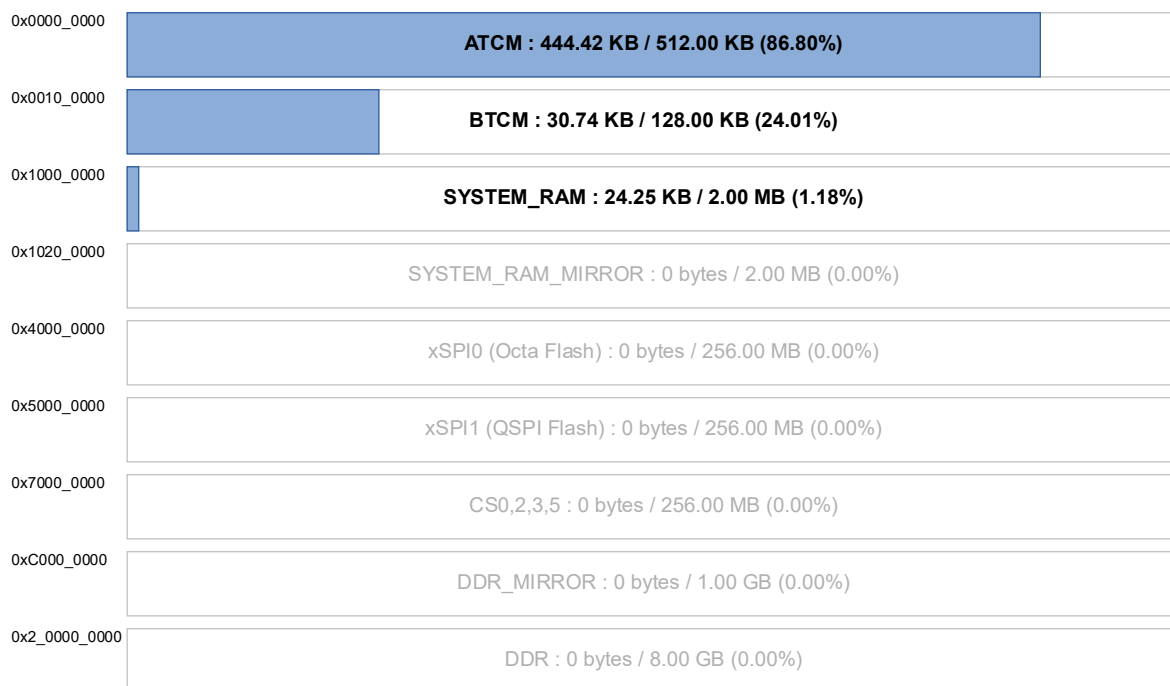


Figure 8.36. Memory footprint of the CPU1 project (EWARM)

ATCM

Total Usage : 444.42 KB

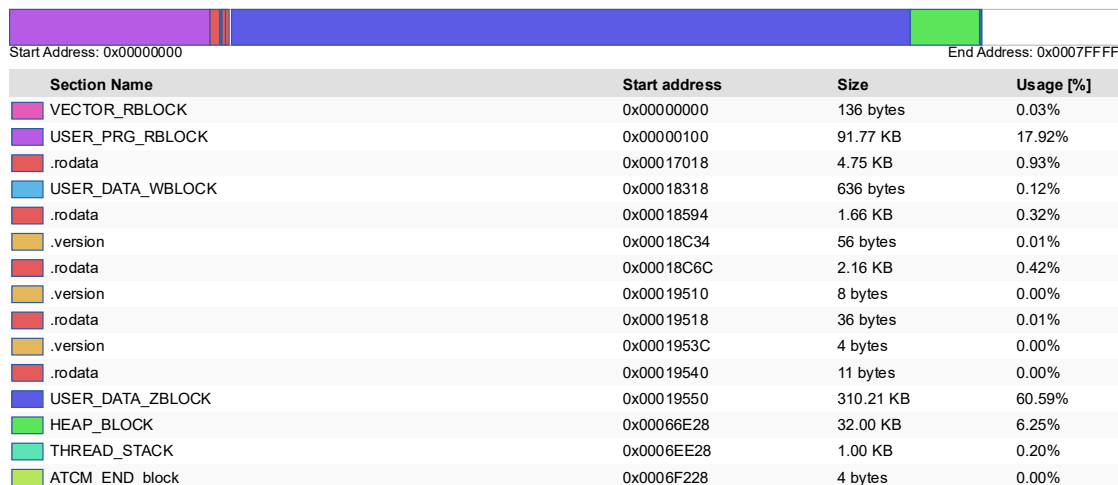
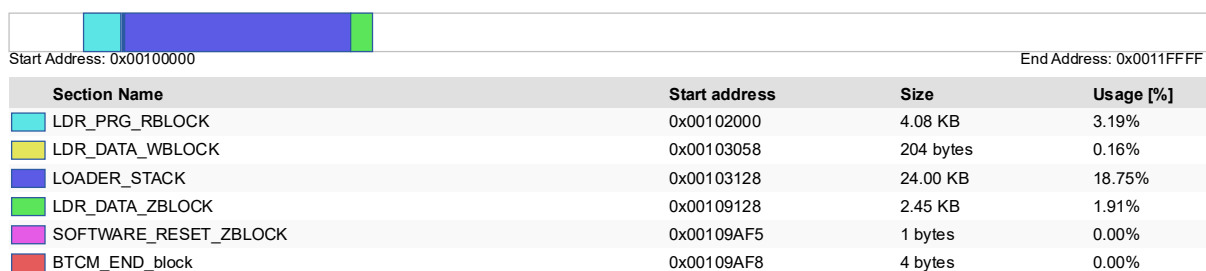


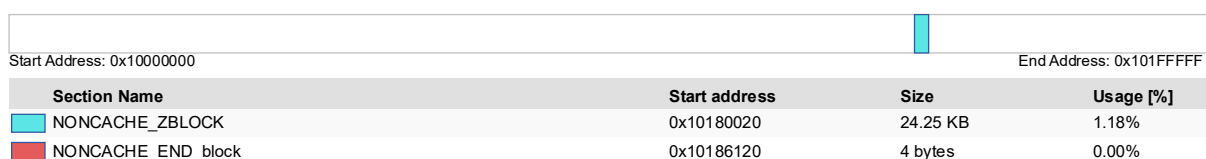
Figure 8.37. ATCM footprint of the CPU1 project (EWARM)

BTCM

Total Usage : 30.74 KB

**Figure 8.38. BTCM footprint of the CPU1 project (EWARM)****SYSTEM_RAM**

Total Usage : 24.25 KB

**Figure 8.39. SYSTEM_RAM footprint of the CPU1 project (EWARM)**

- For CR52 and CA55 projects
 - CPU0(CR52_0)
 - The only application that runs on CPU0 is the LED blinking function.

RZN2H Memory Map

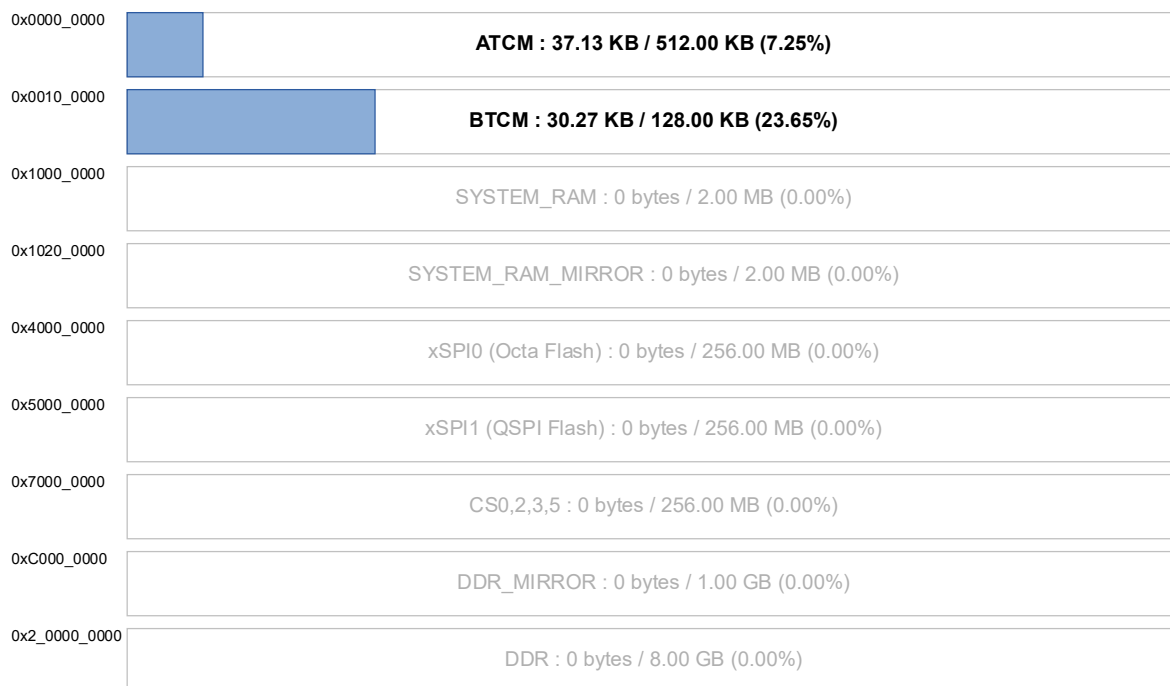


Figure 8.40. Memory footprint of the CPU0 project (EWARM)

ATCM

Total Usage : 37.13 KB

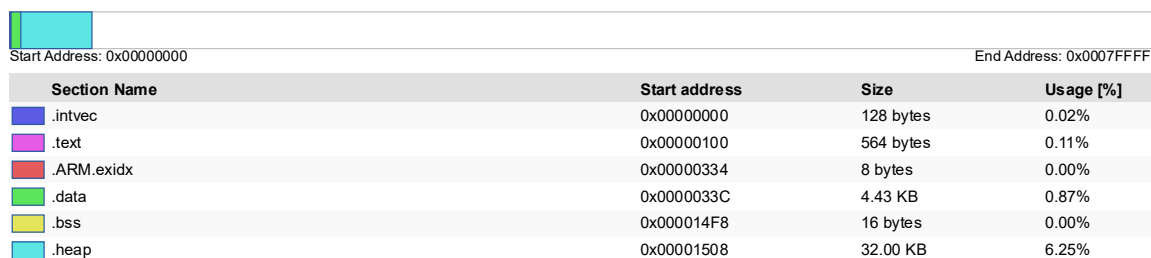


Figure 8.41. ATCM footprint of the CPU0 project (EWARM)

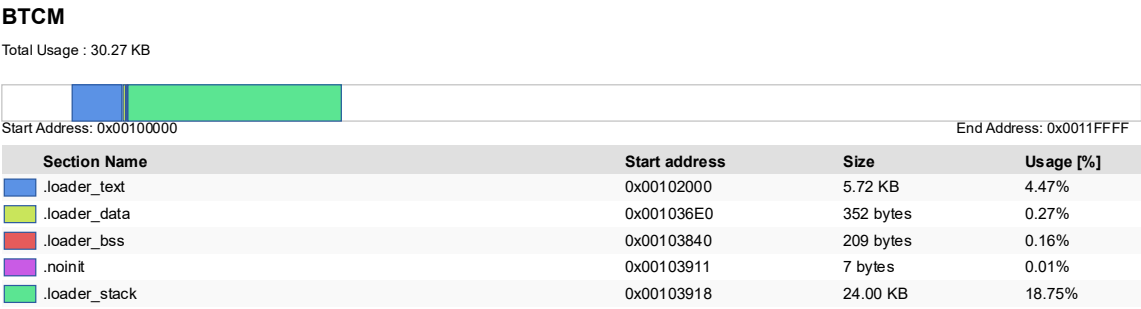


Figure 8.42. BTM footprint of the CPU0 project (EWARM)

► CPU1(CA55_2)

The application running on CPU1 will perform functions other than blinking the LED.

RZN2H Memory Map

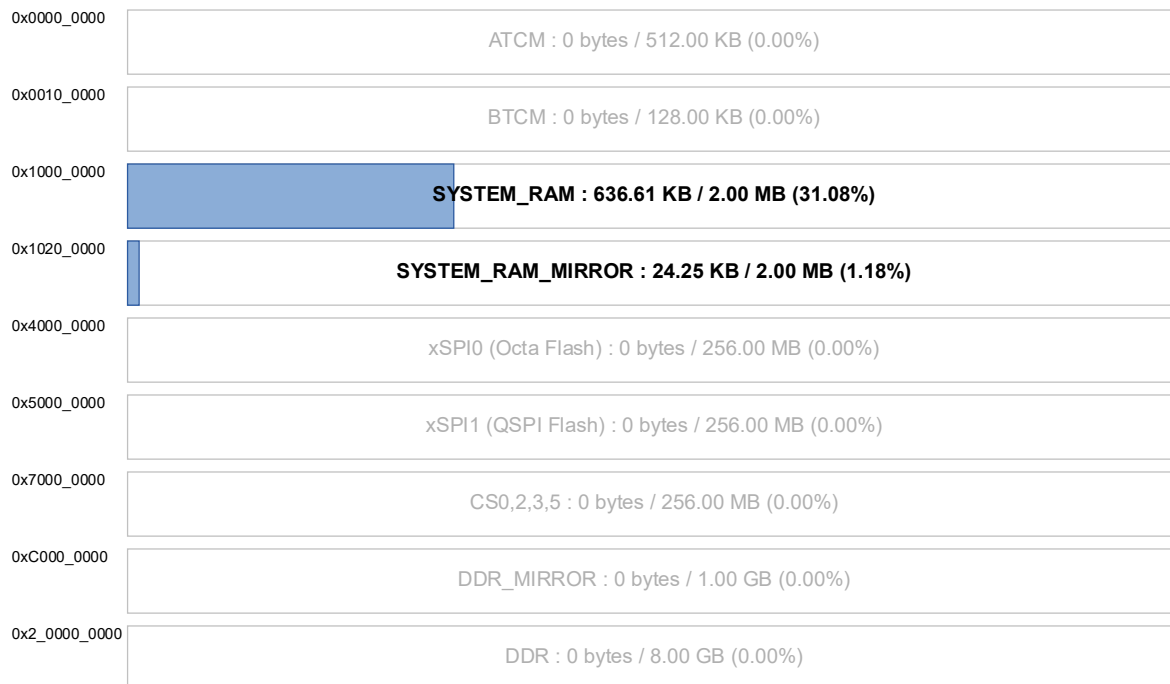


Figure 8.43. Memory footprint of the CPU1 project (EWARM)

SYSTEM_RAM

Total Usage : 636.61 KB

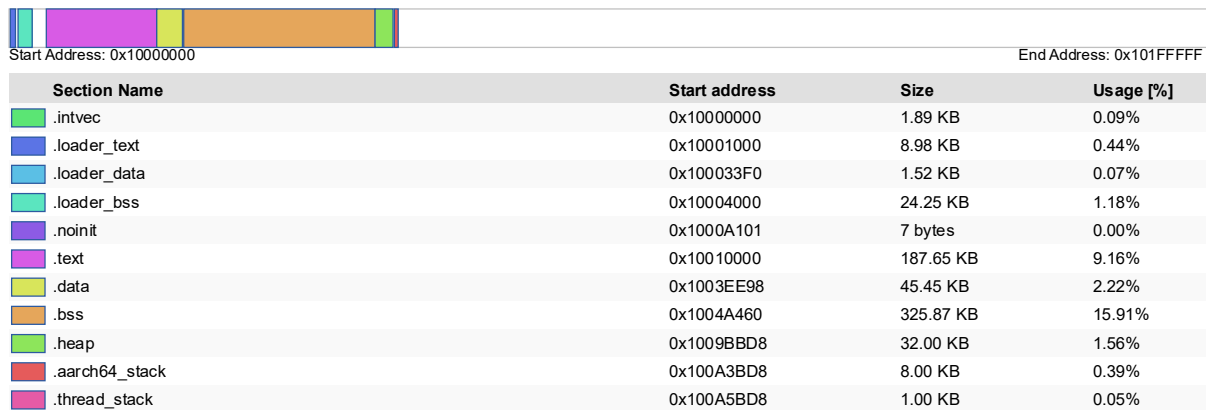


Figure 8.44. SYSTEM_RAM footprint of the CPU1 project (EWARM)

SYSTEM_RAM_MIRROR

Total Usage : 24.25 KB



Figure 8.45. SYSTEM_RAM (MIRROR) footprint of the CPU1 project (EWARM)

2. RZ/N2H

The memory usage by group is shown below.

- For CR52 and CR52 projects

- CPU0(CR52_0)

The only application that runs on CPU0 is the LED blinking function.

RZN2H Memory Map

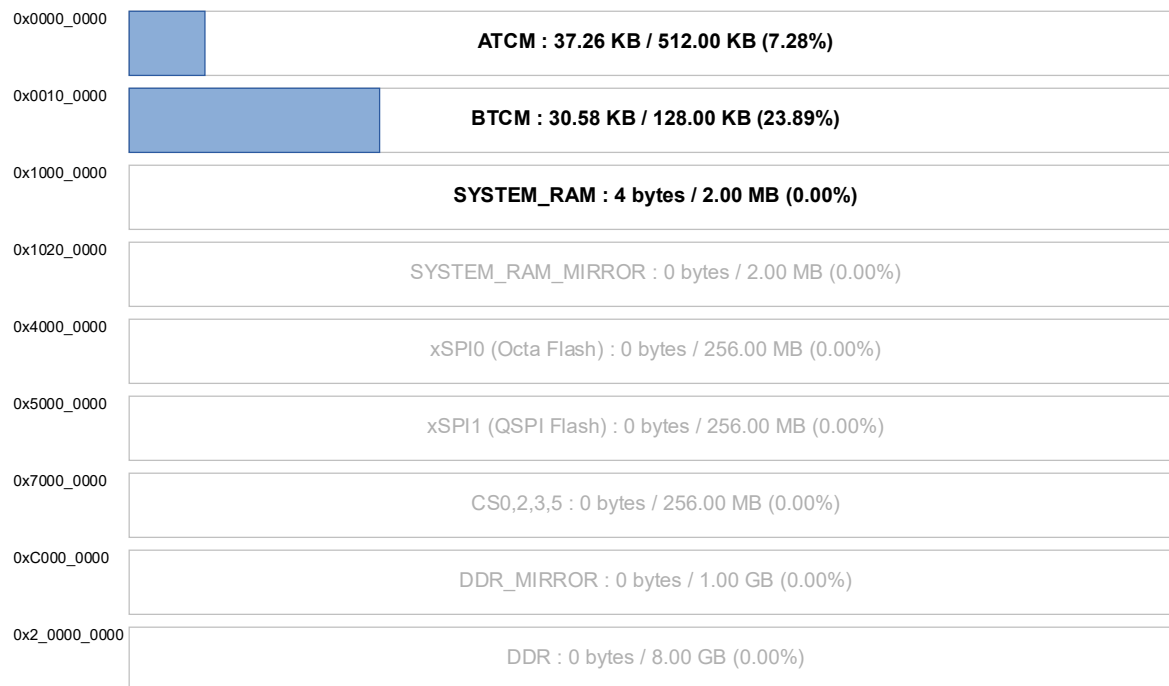


Figure 8.46. Memory footprint of the CPU0 project (EWARM)

ATCM

Total Usage : 37.26 KB



Figure 8.47. ATCM footprint of the CPU0 project (EWARM)

BTCM

Total Usage : 30.73 KB

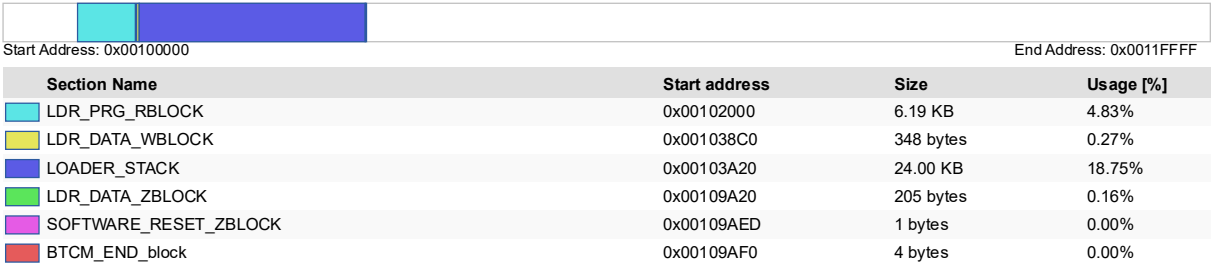


Figure 8.48. BTCM footprint of the CPU0 project (EWARM)

► CPU1(CR52_1)

The application running on CPU1 will perform functions other than blinking the LED.

RZN2H Memory Map

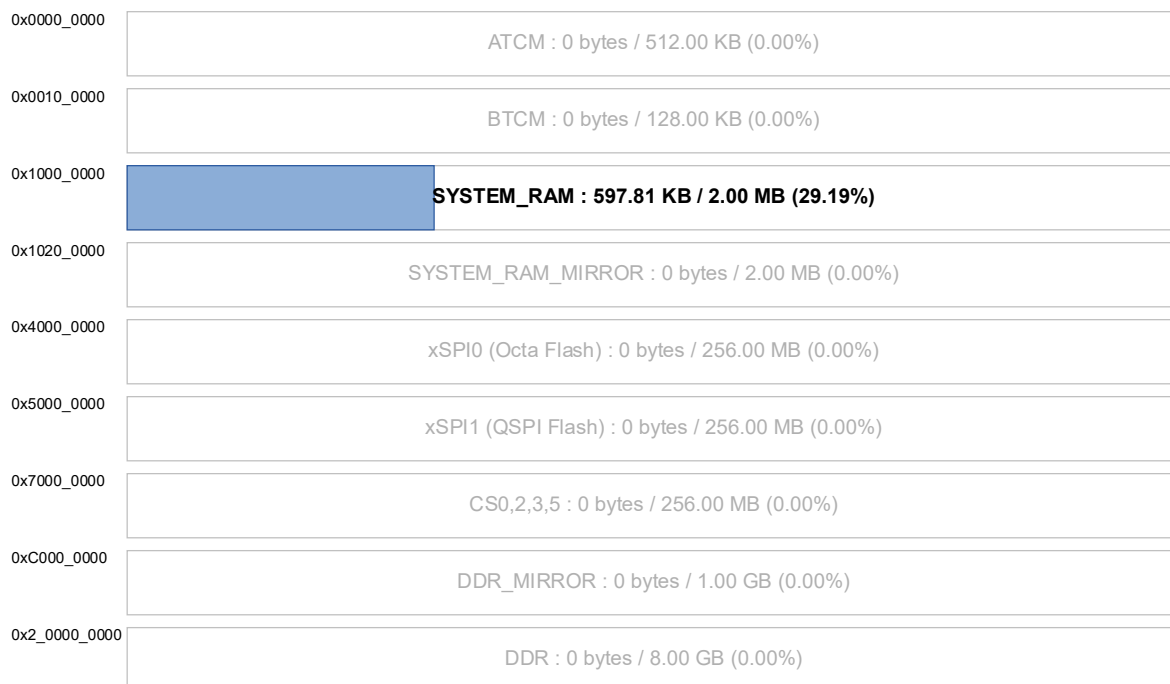


Figure 8.49. Memory footprint of the CPU1 project (EWARM)

SYSTEM_RAM

Total Usage : 597.81 KB

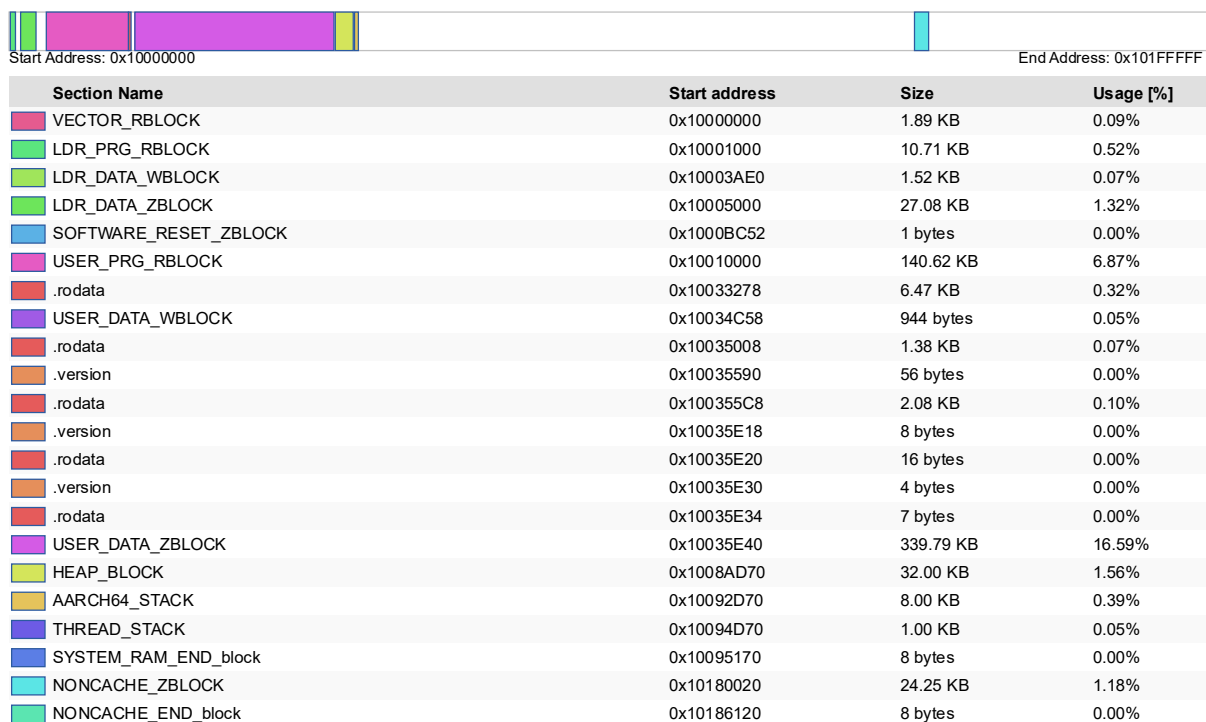
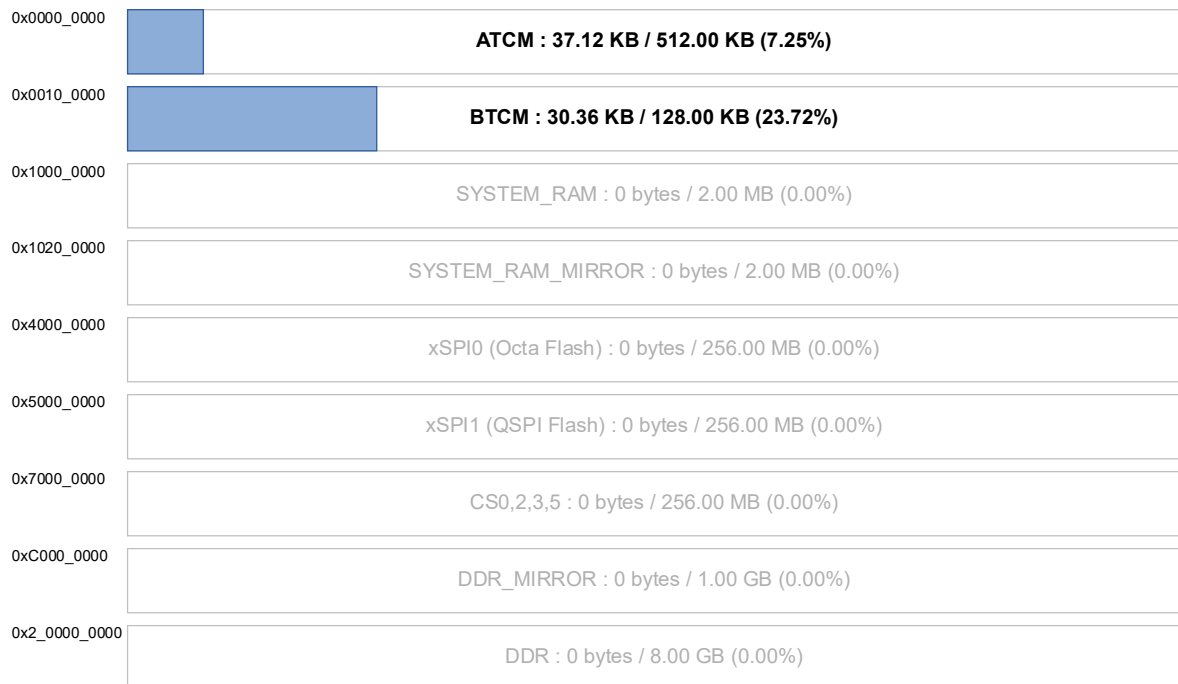
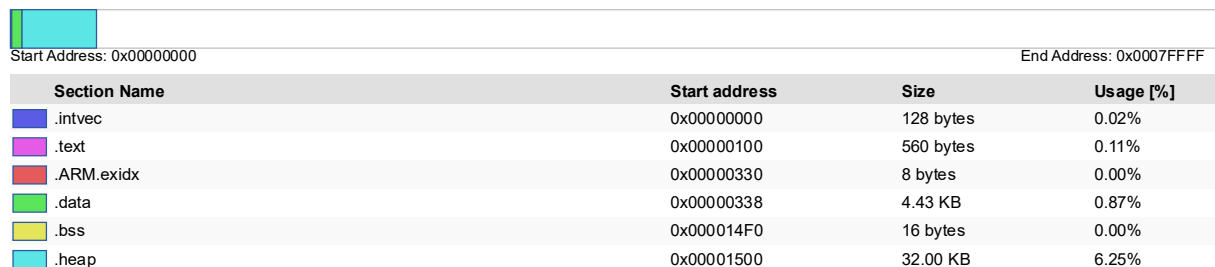


Figure 8.50. SYSTEM_RAM footprint of the CPU1 project (EWARM)

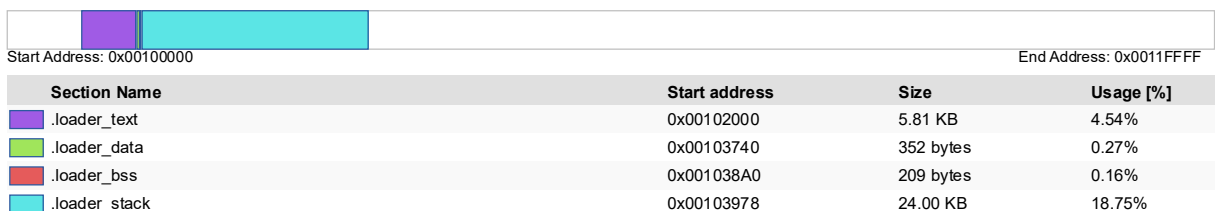
- For CR52 and CA55 projects
 - CPU0(CR52_0)

RZN2H Memory Map**Figure 8.51. Memory footprint of the CPU0 project (EWARM)****ATCM**

Total Usage : 37.12 KB

**Figure 8.52. ATCM footprint of the CPU0 project (EWARM)****BTCM**

Total Usage : 30.36 KB

**Figure 8.53. BTCM footprint of the CPU0 project (EWARM)**

► CPU1(CA55_2)

The application running on CPU1 will perform functions other than blinking the LED.

RZN2H Memory Map

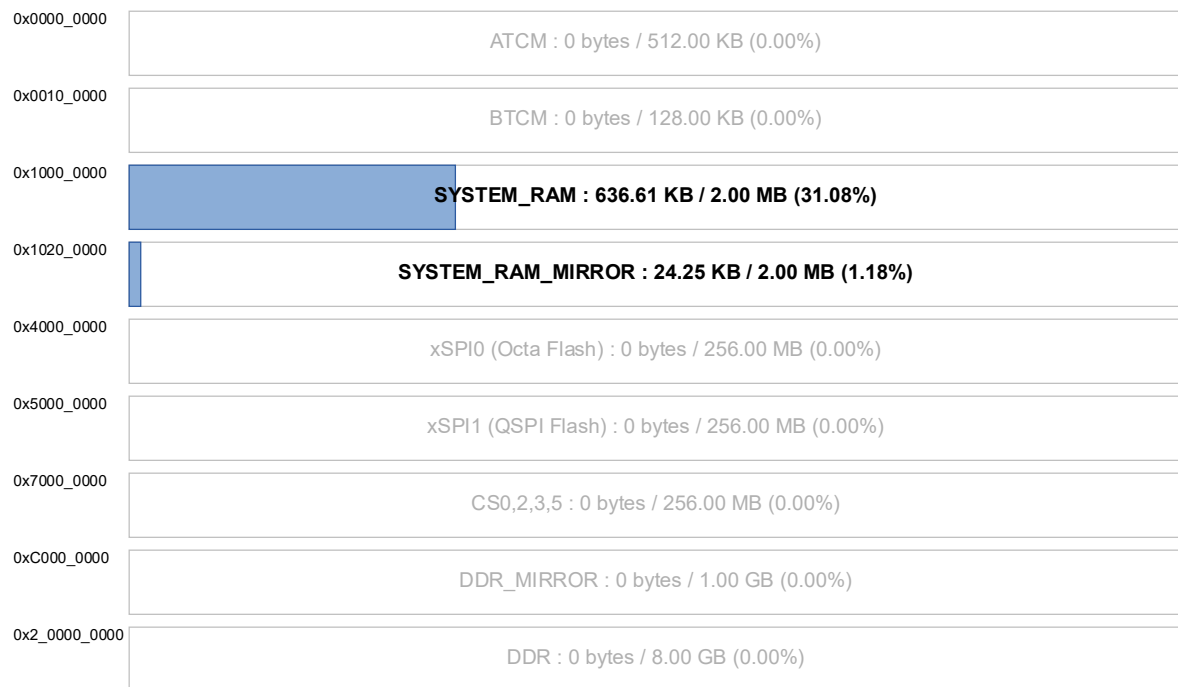


Figure 8.54. Memory footprint of the CPU1 project (EWARM)

SYSTEM_RAM

Total Usage : 636.61 KB

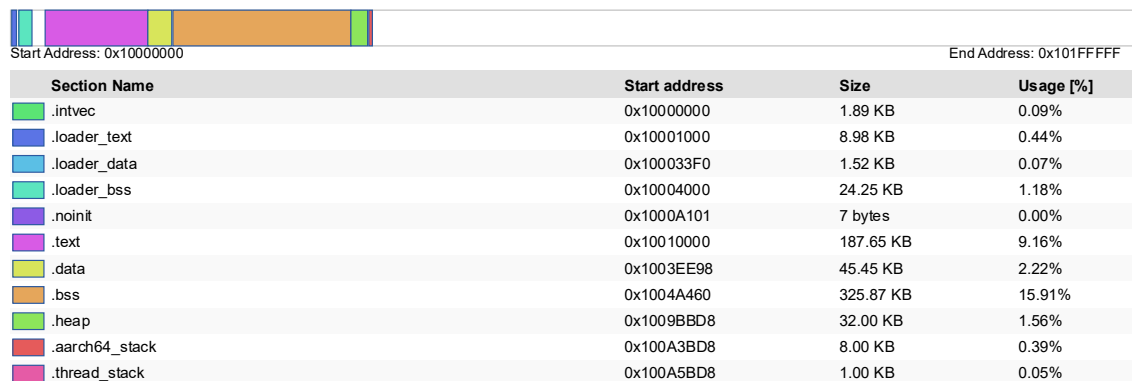


Figure 8.55. SYSTEM_RAM footprint of the CPU1 project (EWARM)

SYSTEM_RAM_MIRROR

Total Usage : 24.25 KB



Figure 8.56. SYSTEM_RAM (MIRROR) footprint of the CPU1 project (EWARM)

8.6 CPU Utilization

This shows the CPU Utilization (task operation rate) of each device.

Please note that the measurements were performed using a simplified calculation method with the FreeRTOS API in e² studio, and therefore should be considered as reference values only.

1. RZ/T2H CR52
 - iperf communication in progress (10 seconds)
CPU usage: 73%
 - Echo server waiting for clients
CPU usage: 23%
2. RZ/T2H CA55
 - iperf communication in progress (10 seconds)
CPU usage: 34%
 - Echo server waiting for clients
CPU usage: <1%
3. RZ/N2H CR52
 - iperf communication in progress (10 seconds)
CPU usage: 73%
 - Echo server waiting for clients
CPU usage: 23%
4. RZ/N2H CA55
 - iperf communication in progress (10 seconds)
CPU usage: 34%
 - Echo server waiting for clients
CPU usage: <1%

Revision History

Rev.	Date	Description	
		Page	Summary
1.60	Jul 31, 2026	-	Standardization of fonts and other improvements
		-	Improvements such as standardizing table formatting
		-	Corrections of typos and other errors
		P5-P10, P12-P15, P32-,P37 P40-P47	Multi-interface support
1.50	Apr, 13 2026	-	Revised New Edition

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.

- 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.
- 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. 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/.