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

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 Tile	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 Tile	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

Nothing in particular.

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 1.2	Renesas RZ/T2H-EVKIT-Evaluation Board Kit for RZ/T2H
IDE	EWARM	9.60.3	IAR Systems IAR Embedded Workbench for Arm | IAR
	e² studio	2025-12	RZ FSP About Flexible Software Package (FSP) for Renesas RZ series
Configurator	FSP Smart Configurator	2025-12	
Flexible Software Package	FSP	4.0.0	
GCC Compiler	GNU ARM Embedded Toolchain	13.3.Rel1	
	GNU ARM A-Profile (AArch64 bare-metal)	13. 2.Rel1	
Emulator	J-Link ™	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	Wire shark		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
	e² studio	2025-12	RZ FSP About Flexible Software Package (FSP) for Renesas RZ series
Configurator	FSP Smart Configurator	2025-12	
Flexible Software Package	FSP	4.0.0	
GCC Compiler	GNU ARM Embedded Toolchain	13.3.Rel1	
	GNU ARM A-Profile (AArch64 bare-metal)	13. 2.Rel1	
Emulator	J-Link ™	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	Wire shark		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

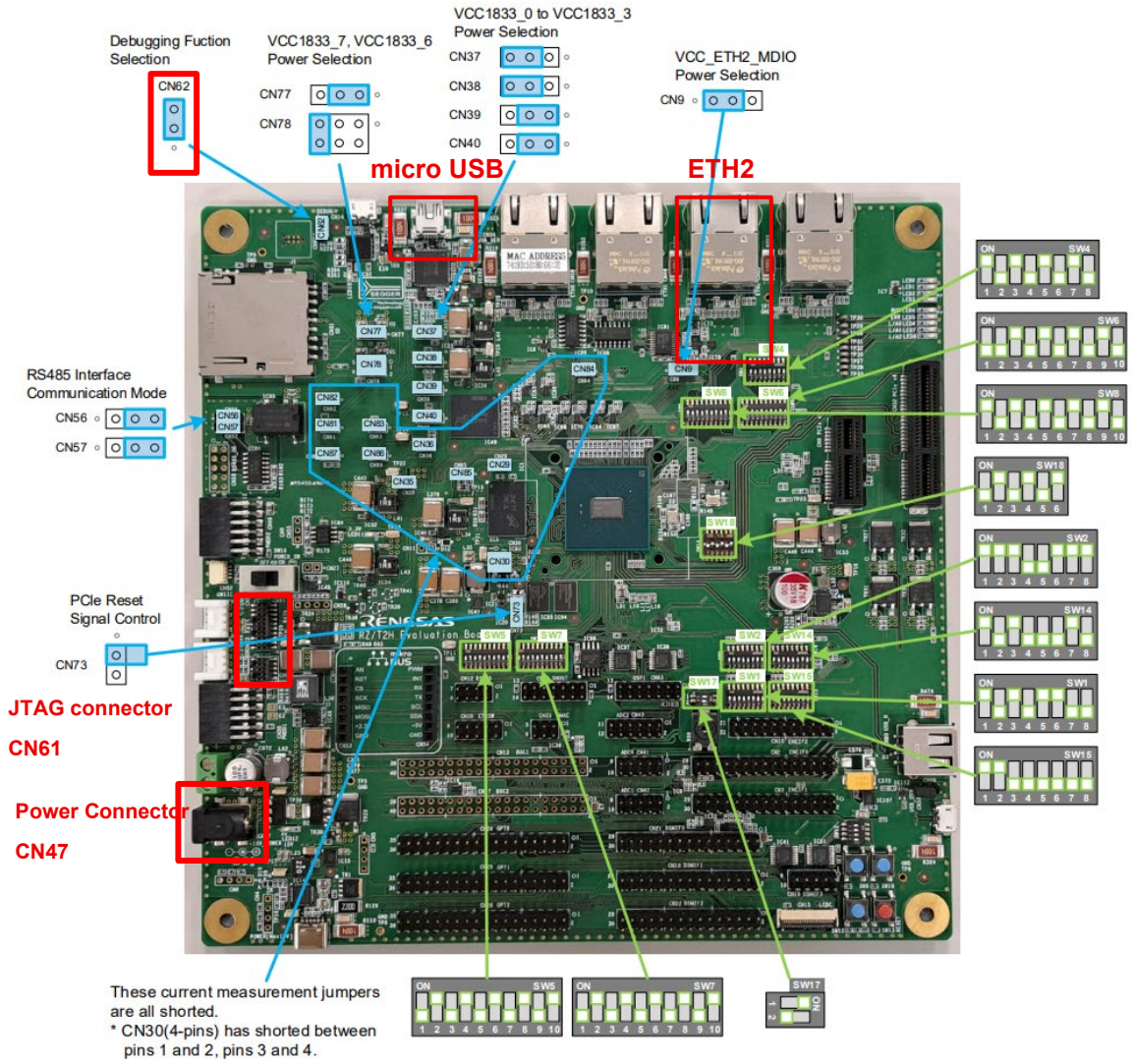


Figure 4.1 RZ/T2H Evaluation Board

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. Connect the Ethernet cable to ETH2.

4.2 RZ/N2H Evaluation Board

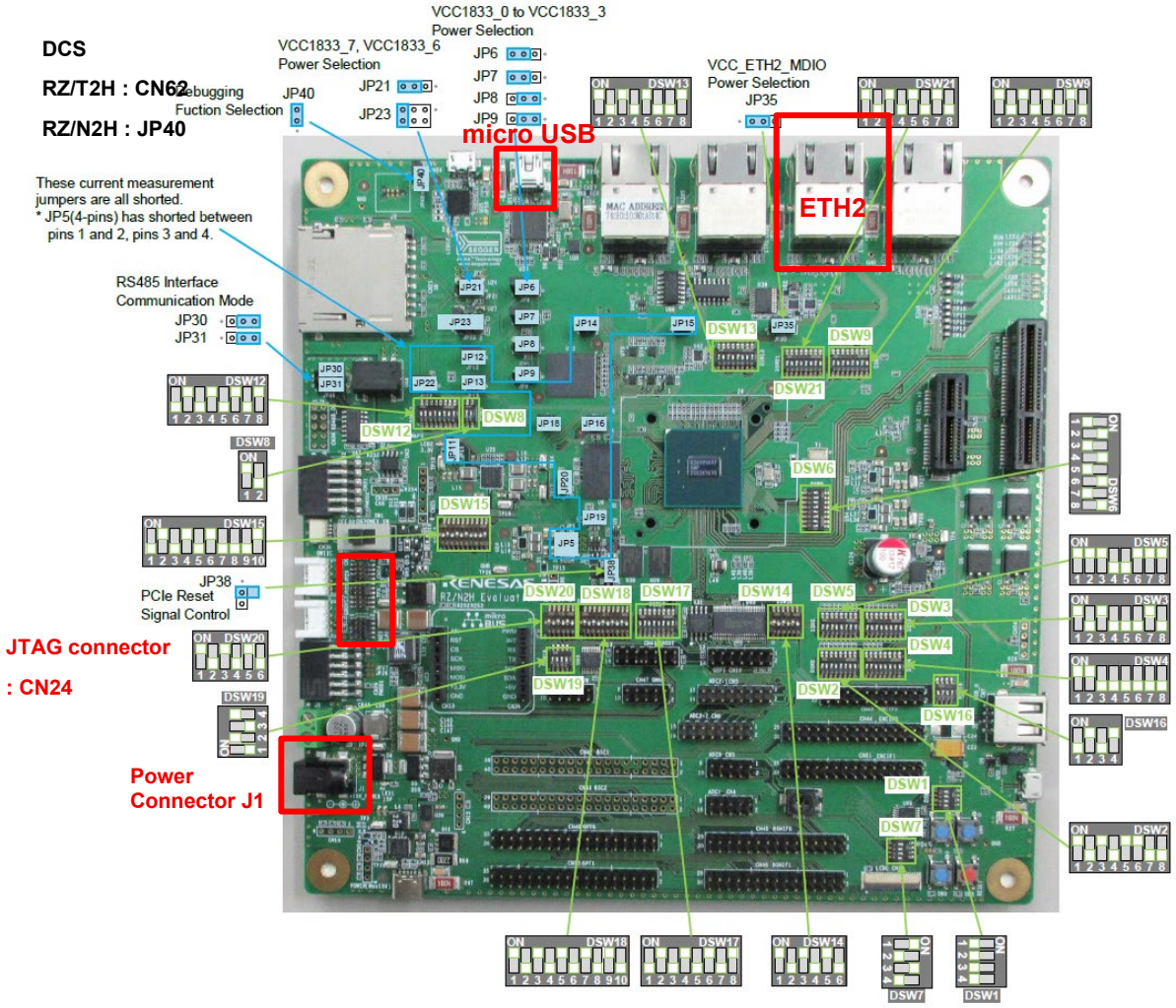


Figure 4.2 Z/N2H Evaluation Board

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. Connect the Ethernet cable to ETH2.

4.3 Connecting the Debugger

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

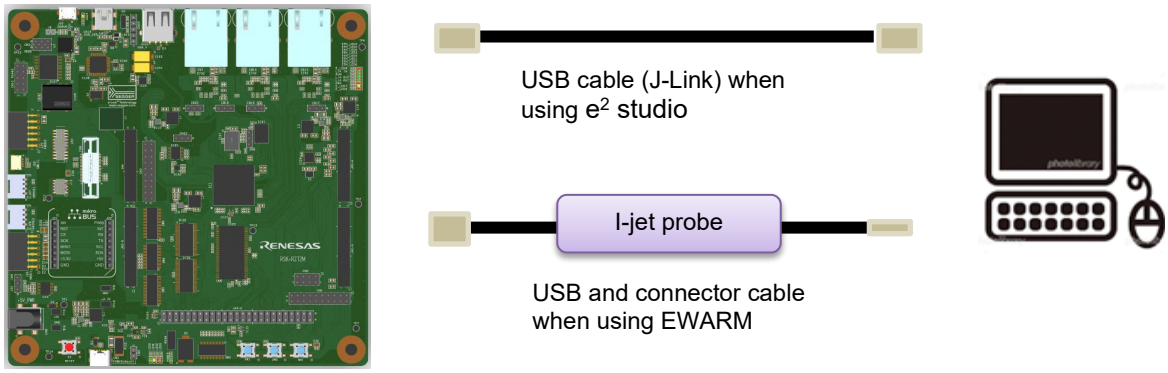


Figure 4.3 Debugger Connection

5. Host PC Settings

See "7. Demonstration of the Sample Application"

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.

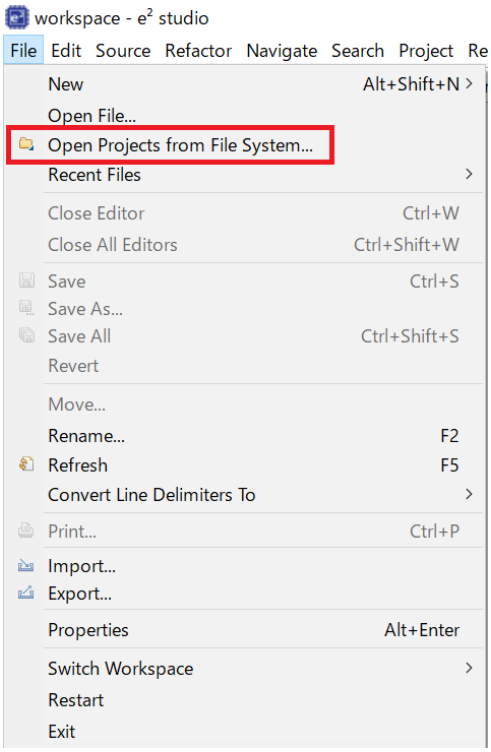


Figure 6.1 e²studio File tab

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\e2studio\RZT2H_EVB_lwIP_cpu0
    - When using RZ/N2H
      project\rzn2h_evb\cr52_cr52\e2studio\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\e2studio\RZT2H_EVB_lwIP_cpu1
    - When using RZ/N2H
      project\rzn2h_evb\cr52_cr52\e2studio\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\e2studio\RZT2H_EVB_lwIP_cpu0
    - When using RZ/N2H
      project\rzn2h_evb\cr52_0-ca55_2\e2studio\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\e2studio\RZT2H_EVB_lwIP_cpu1
    - When using RZ/N2H
      project\rzn2h_evb\ cr52_0-ca55_2\e2studio\RZN2H_EVB_lwIP_cpu1

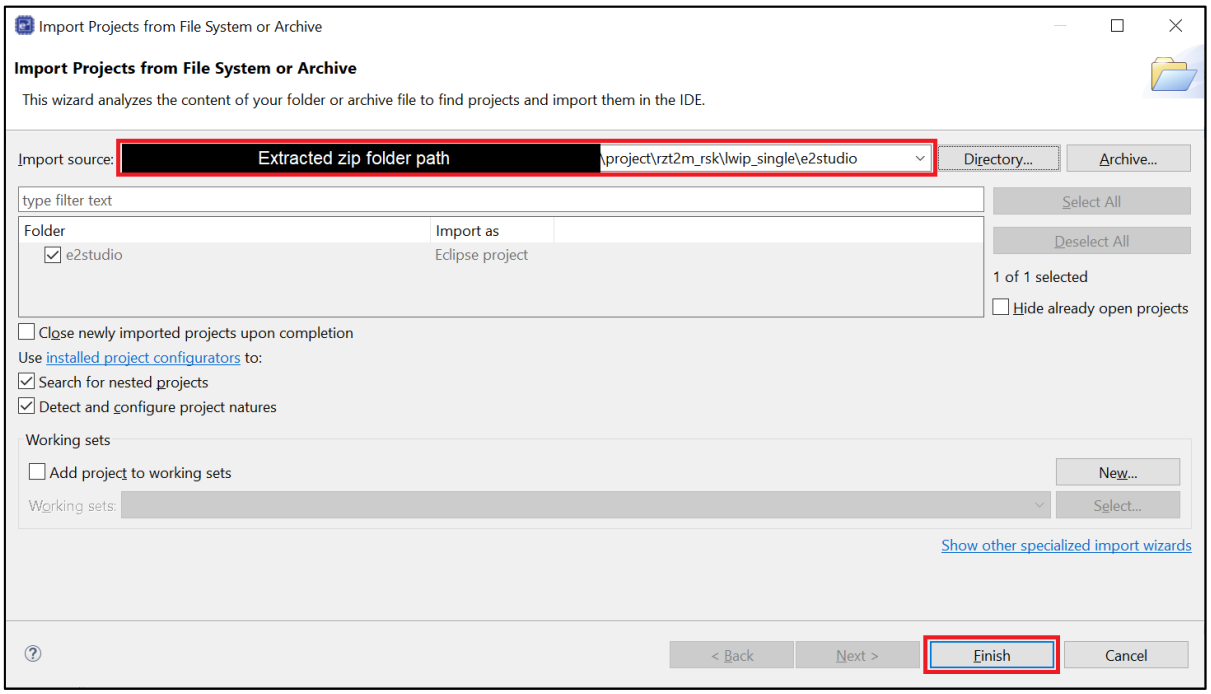


Figure 6.2 Import project on e²studio

6.1.2 Board IP address setting.

Set the IP address in the following procedure.

1. The IP address is set in src/renesas/application/lwip_port_instance.c.

   The port number is set in src/renesas/application/lwip_port_main.c.

   The following addresses are used in the example:

   IP address : 192.168.10.170

   Subnet mask : 255.255.255.0

   Default gateway : 192.168.10.1

   Port number : 65000

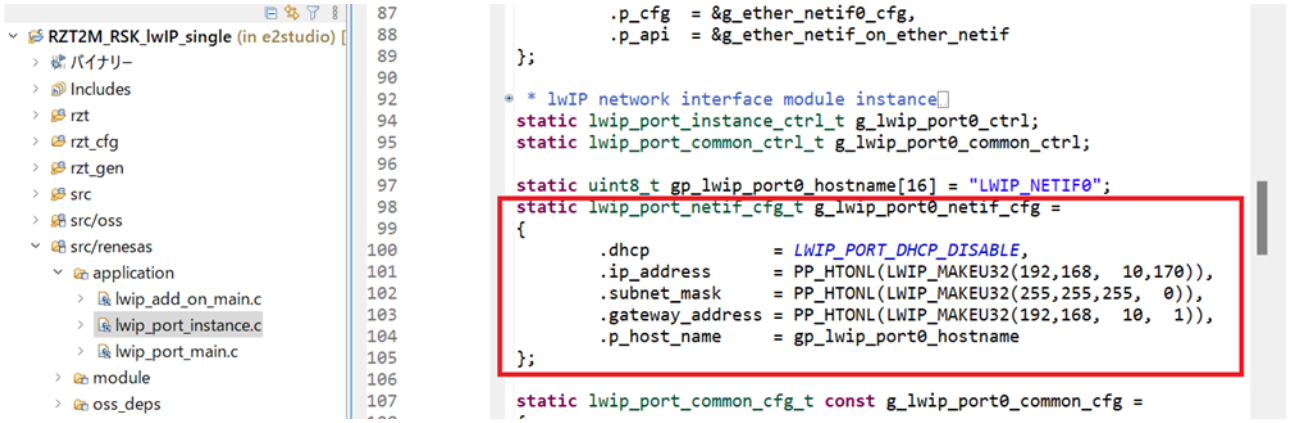


Figure 6.3 Static IP address

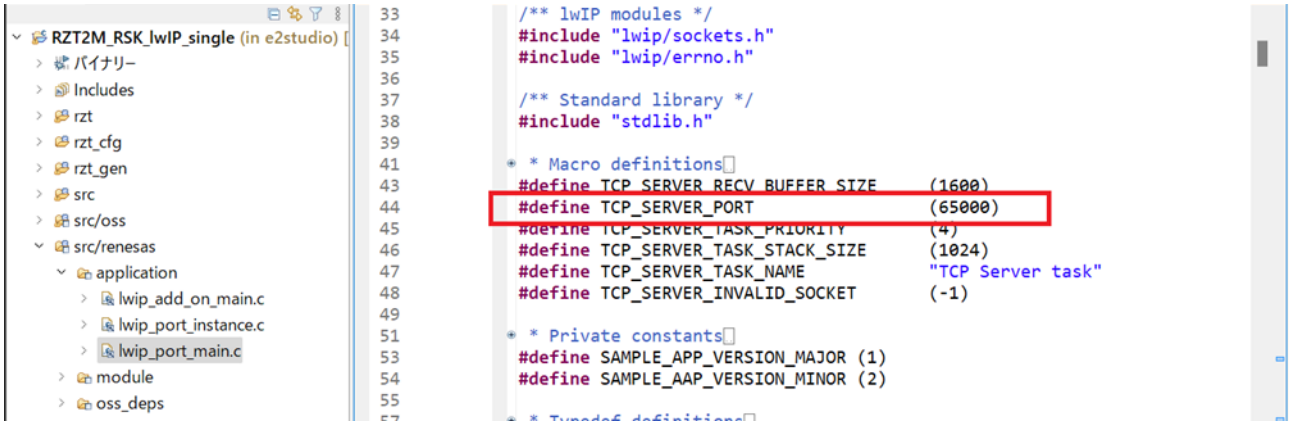


Figure 6.4 Port number

6.1.3 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 you use anything other than the “ld” file in the folder, unexpected results may occur.
- If you change the boot mode or FSP version, the “ld” file will be rewritten at the time of code generation, so please save it in advance and restore it to the script before building.

  If there is no change, there is no rewriting.

1. Click the “Configuration.xml”.

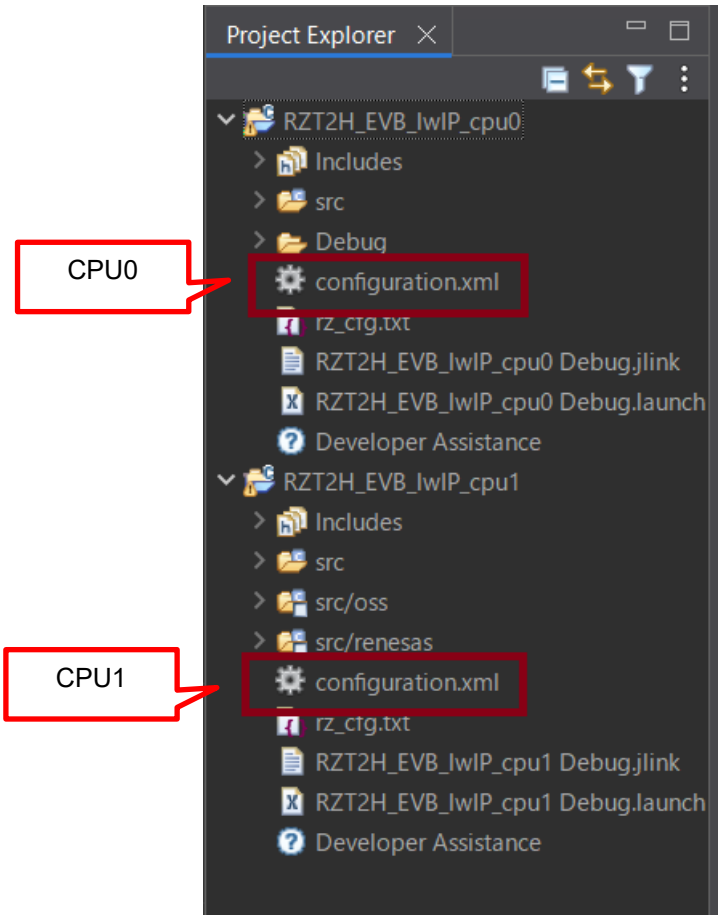


Figure 6.5 Configuration

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)**

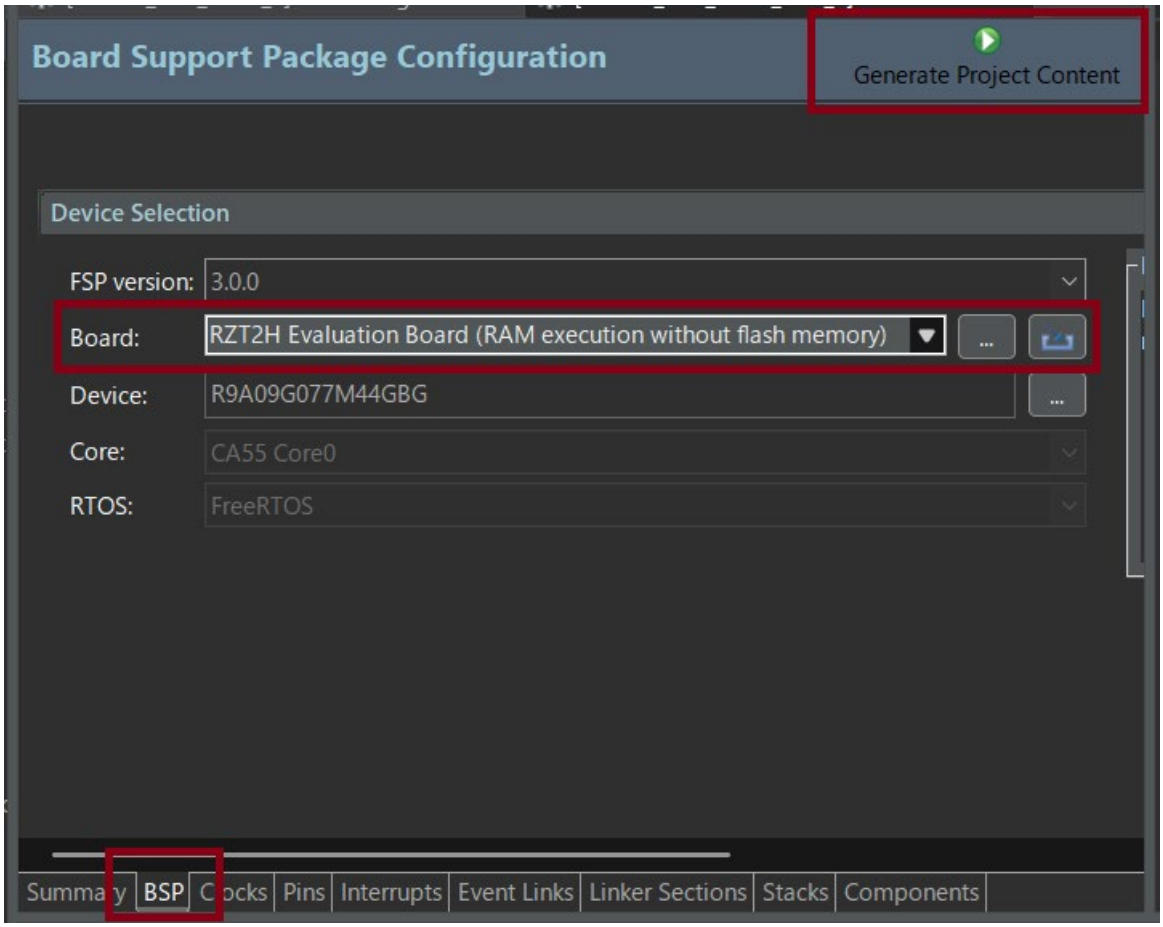


Figure 6.6 Generate Project Content

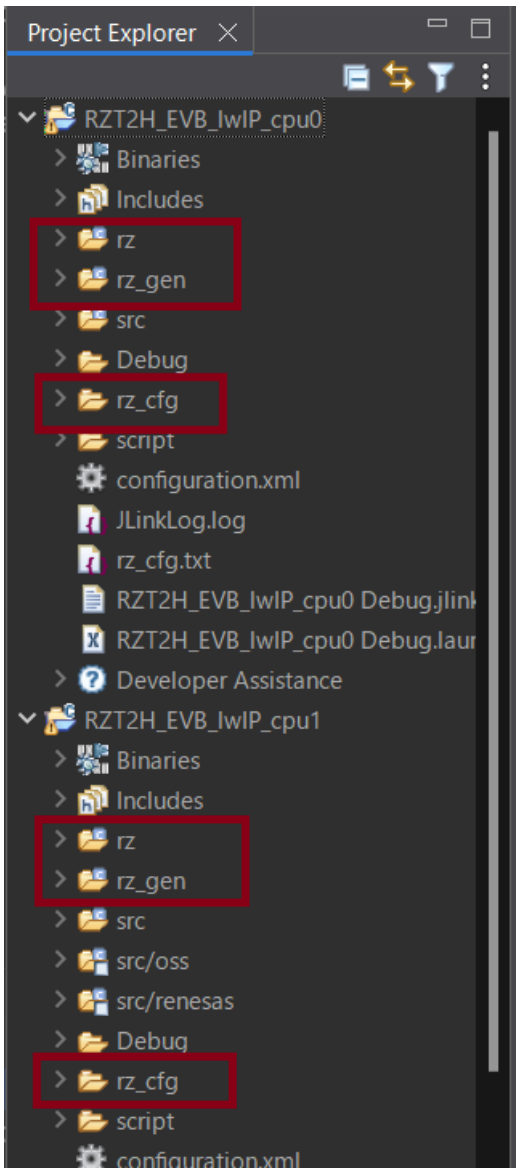


Figure 6.7 Generate project folder

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

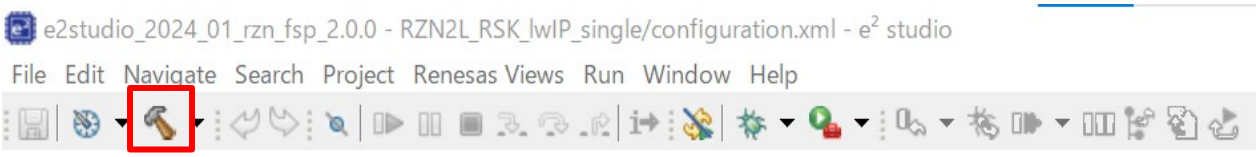


Figure 6.8 Build button

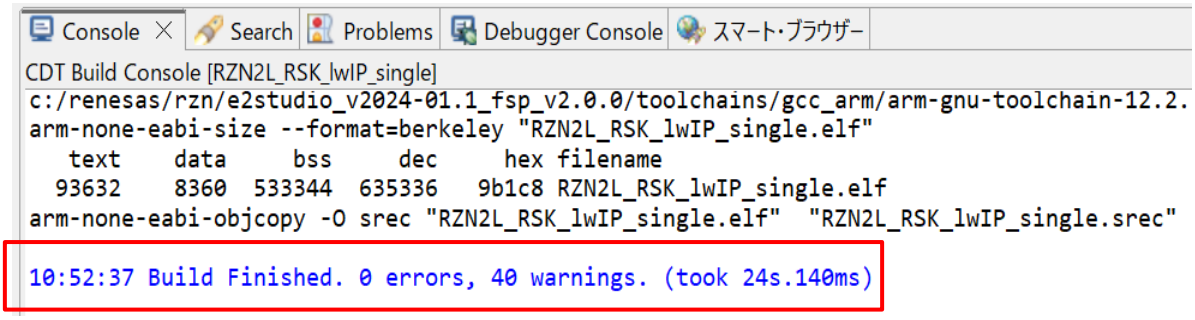


Figure 6.9 Build message

6.1.4 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.

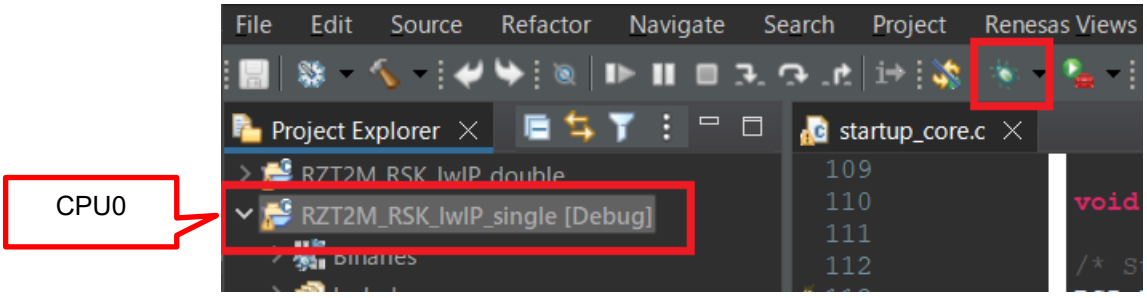


Figure 6.10 CPU0 Debug button

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."

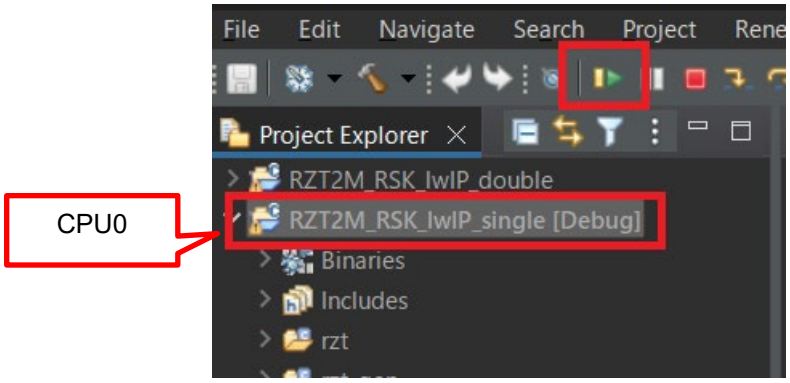


Figure 6.11 CPU0 Run button

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

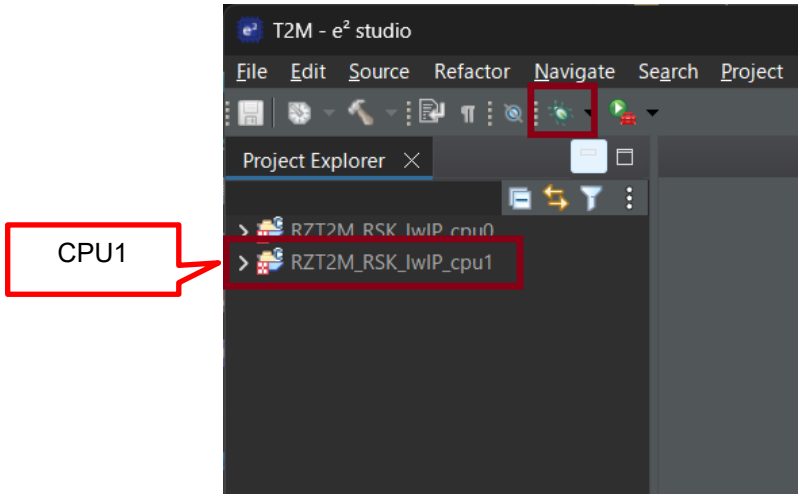


Figure 6.12 CPU1 Debug button

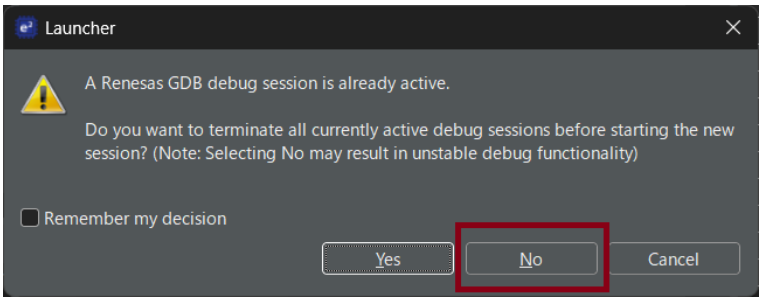


Figure 6.13 CPU1 Duplicate confirmation

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."

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

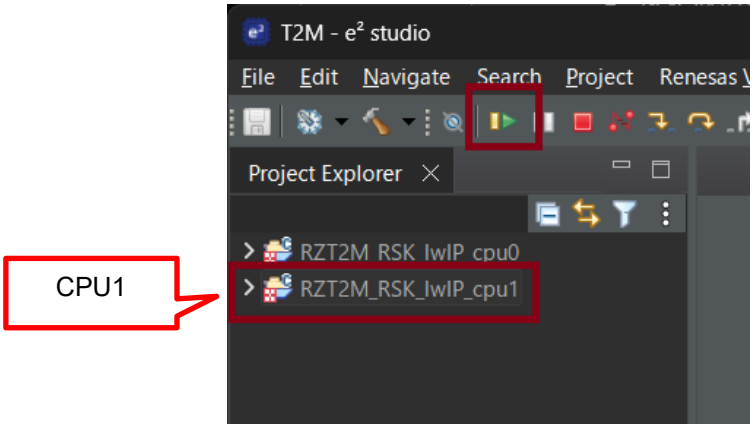


Figure 6.14 CPU1 Run button

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.

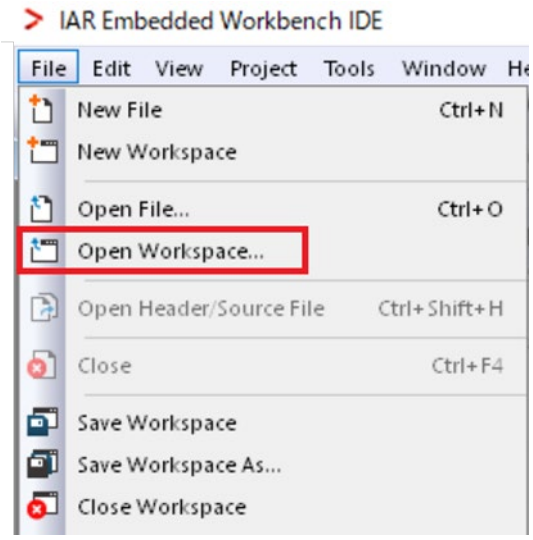


Figure 6.15 EWARM file tab

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 Board IP address setting.

Set the IP address on the following procedure.

1. The IP address is set in \common\renesas\application\lwip_port_instance.c.

   The port number is set in \common\renesas\application\lwip_port_main.c.

   The following addresses are used in the example:

   IP address : 192.168.10.170

   Subnet mask : 255.255.255.0

   Default gateway : 192.168.10.1

   Port number : 65000

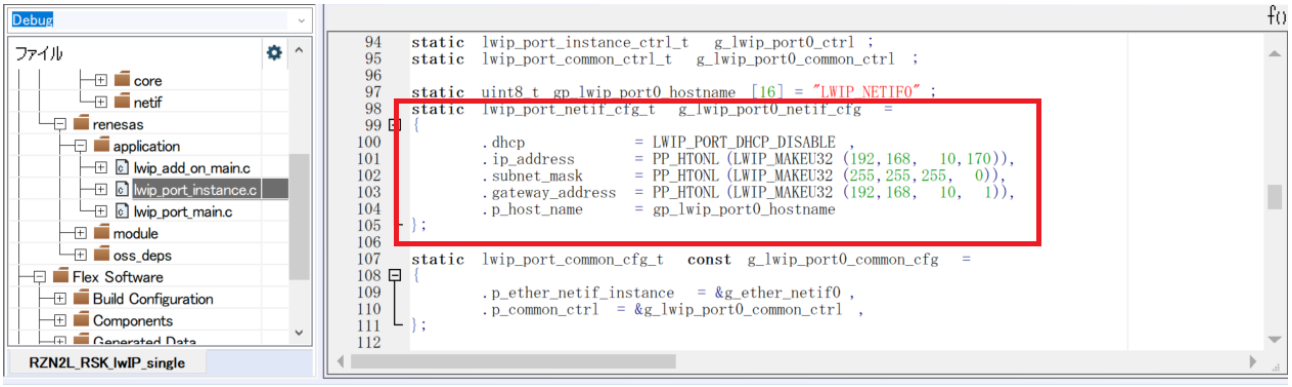


Figure 6.16 Static IP address

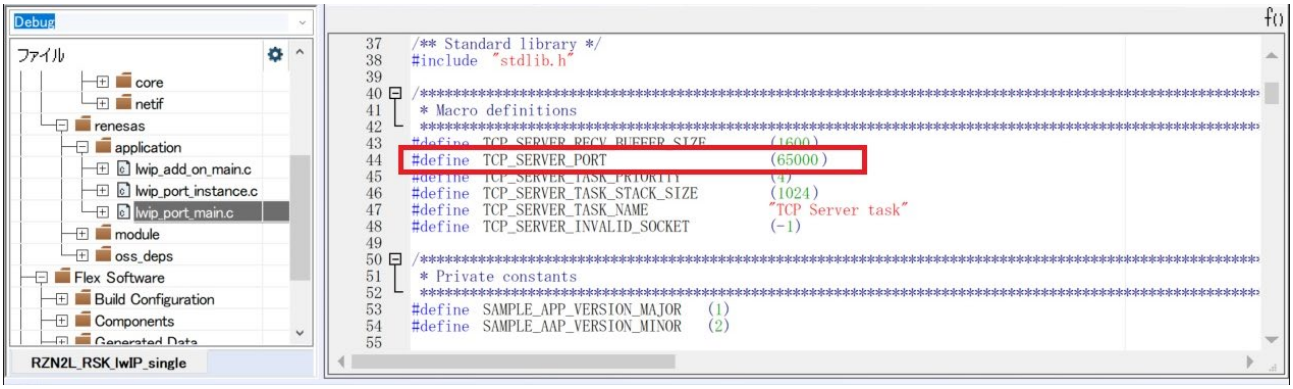


Figure 6.17 Port number

6.2.3 How to generate source code and how to build.

1. Notes on Code Generation.
   - Using an icf file other than the one in the script folder may result in unexpected results.
   - If you change the boot mode or FSP version, the icf file will be rewritten during code generation. Therefore, save it beforehand and restore it to the previously saved icf file before building. If there are no changes, the file will not be rewritten.
   - 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.

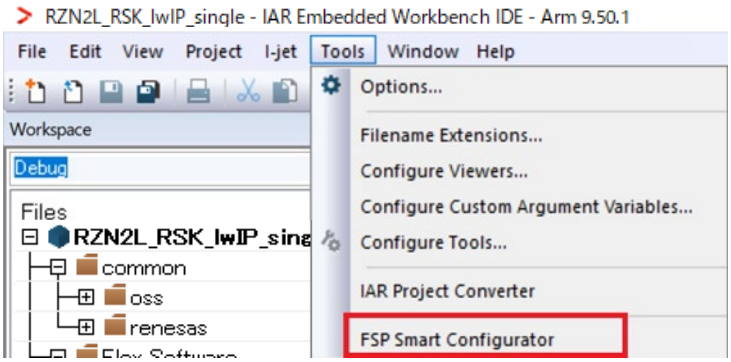


Figure 6.18 Tools tab

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)**

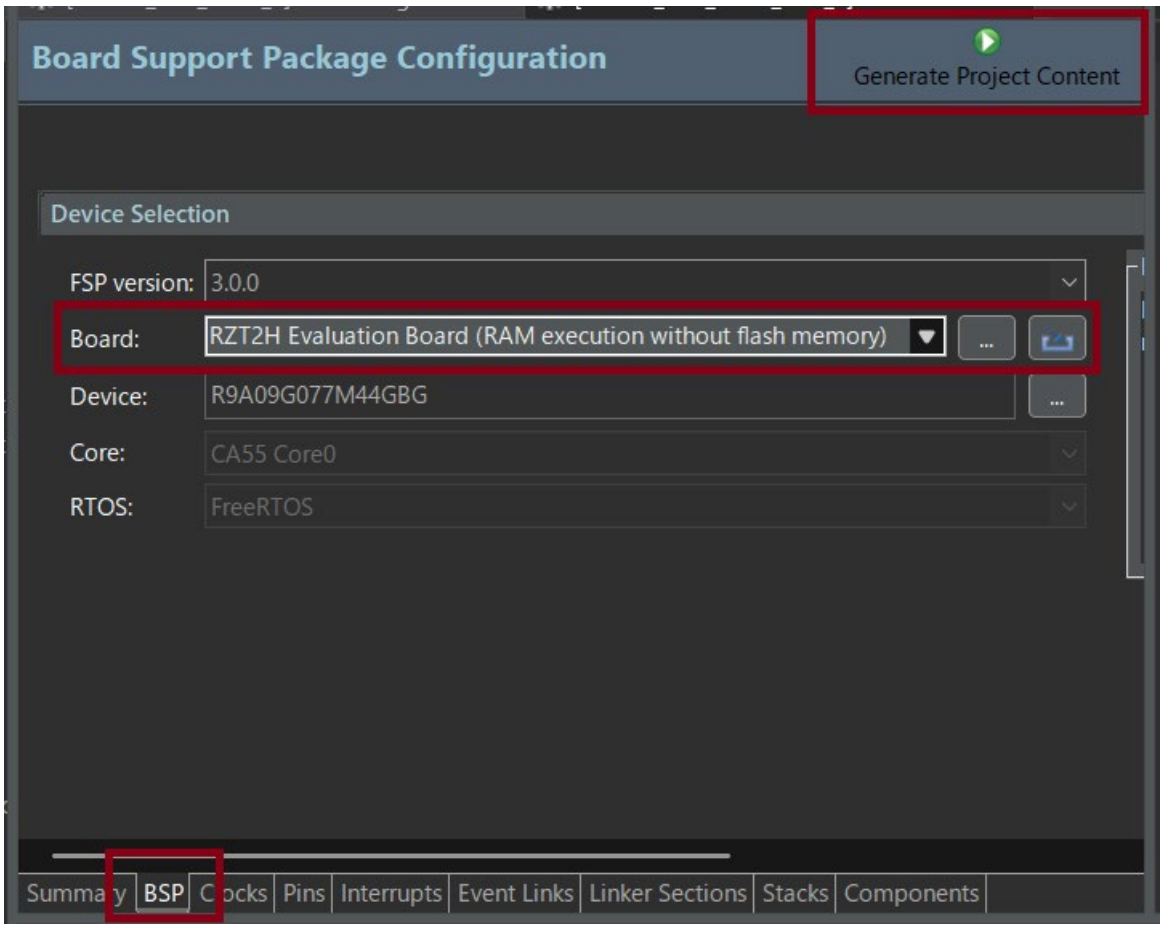


Figure 6.19 FSP SC Smart Configurator

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.

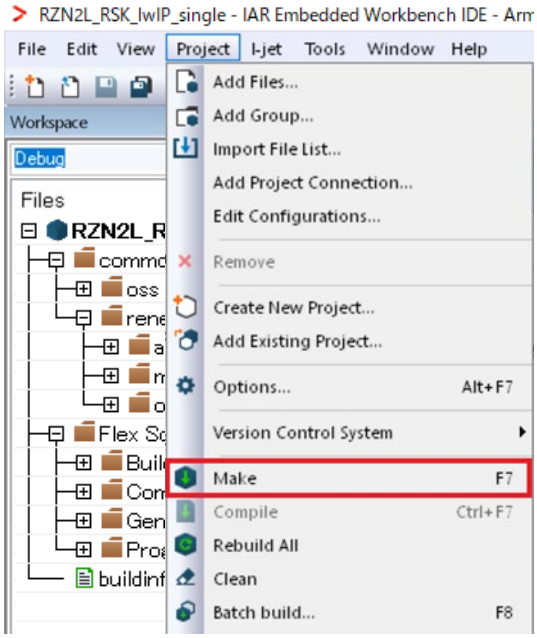


Figure 6.20 Make button 1

Figure 6.21 Make button 2

Figure 6.22 Make Build console

6.2.4 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.

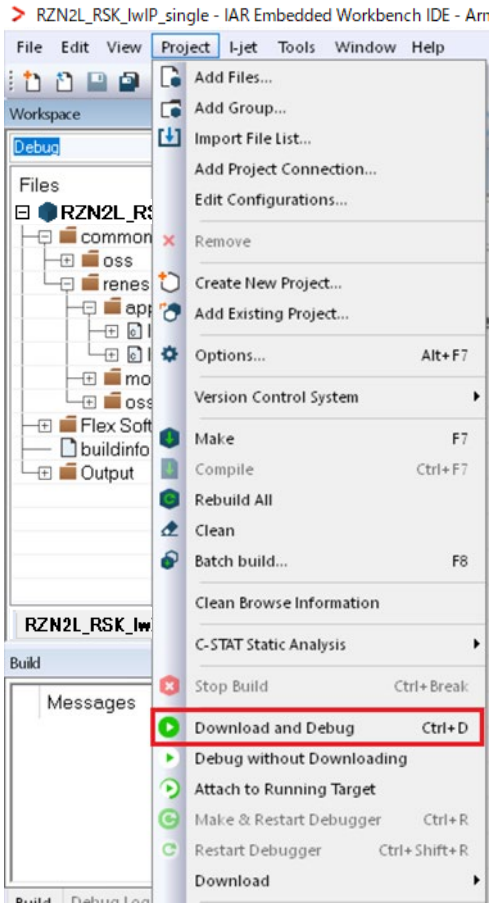


Figure 6.23 Download and Debug

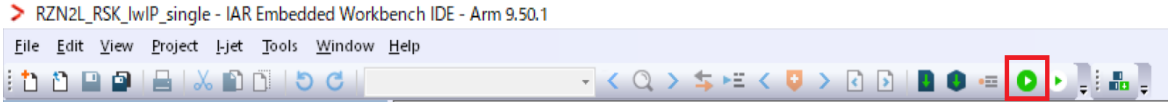


Figure 6.24 Debug button

3. 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.
4. 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.

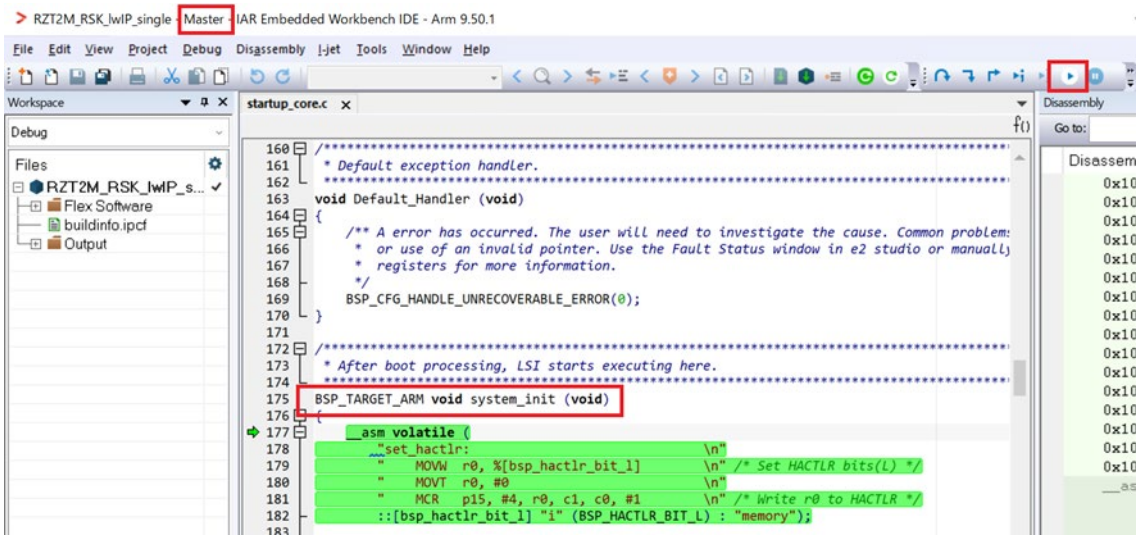


Figure 6.25 CPU0 Go button

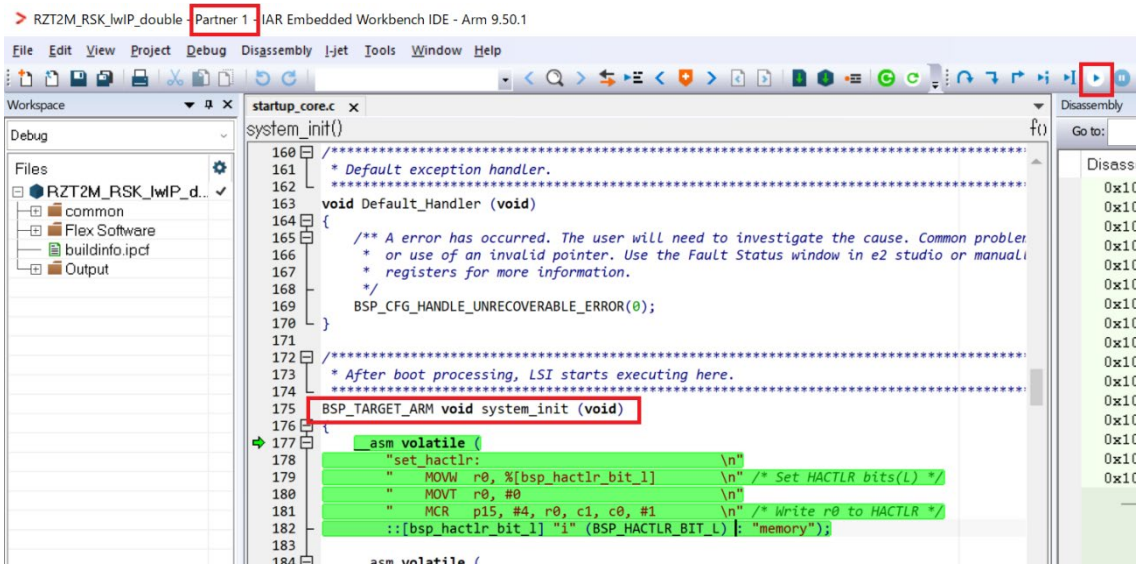


Figure 6.26 CPU1 Go button

7. Demonstration of the Sample Application

The sample program demonstrates the echo server functionality and the LWIP add-ons "iperf" and "SNTP" in the "apps" folder, which can be run with a conditional compiler.

7.1 Connecting to a PC

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

1. Connect the "USB to Serial Port" port shown in "4. Hardware Settings" to your PC 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.

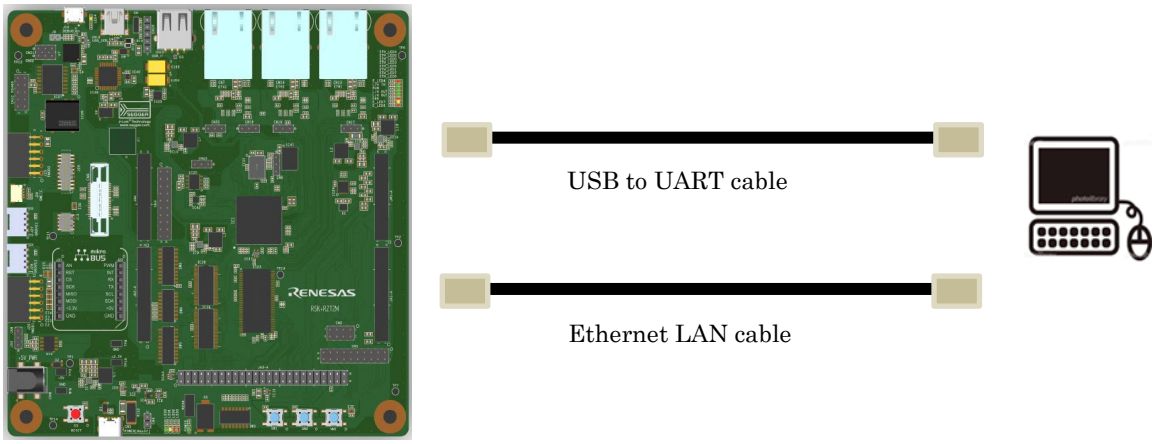


Figure 7.1 Communication terminal connection

7.2 Echo Server

After running the sample application, the following screen will appear and the echo server will start, allowing you to send characters from your PC via telnet.

Please send the following steps.

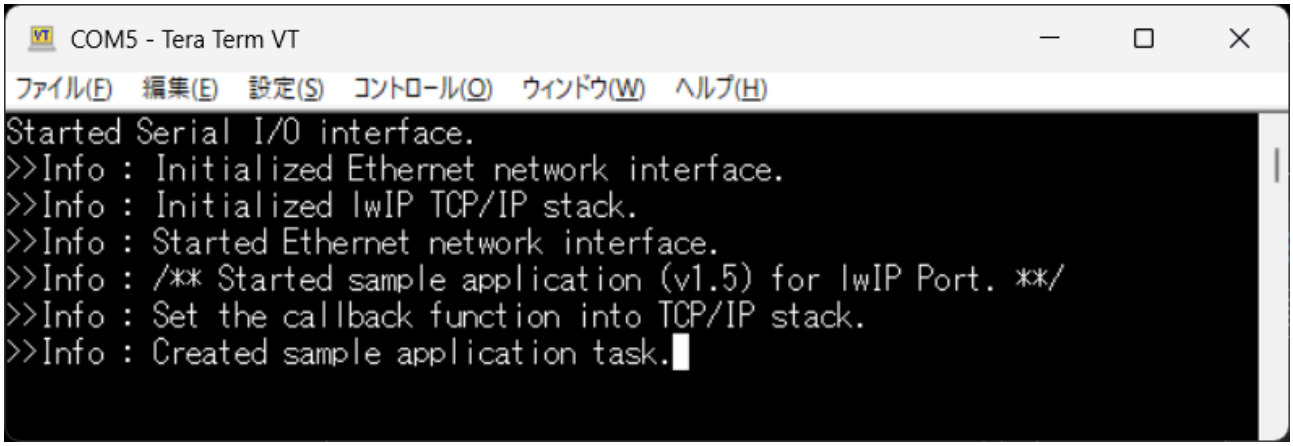


Figure 7.2 Startup screen

1. Open "Tera Term" on your PC. Set it up as shown below.

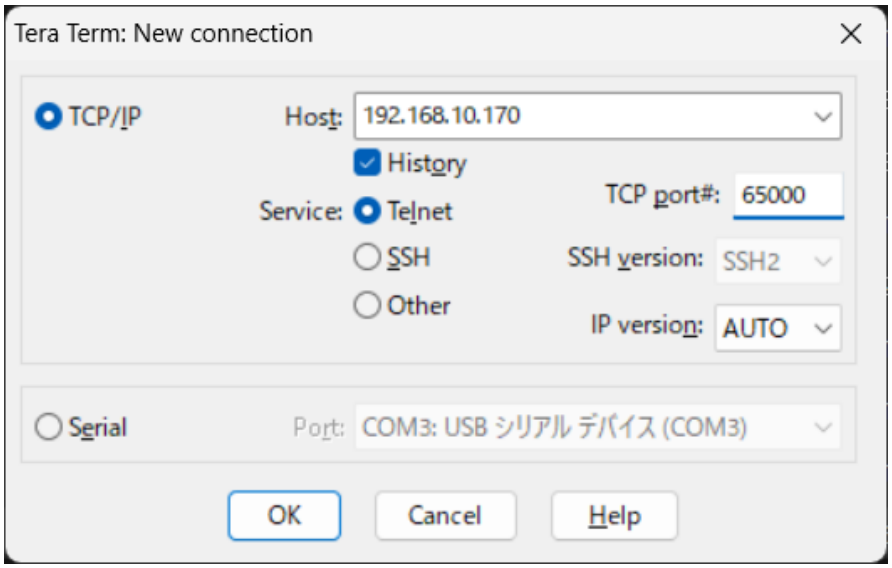


Figure 7.3 Setting connection for Tera Term

2. Enter any characters and press Enter.

   In the following screen capture, the characters you entered are returned by the board.

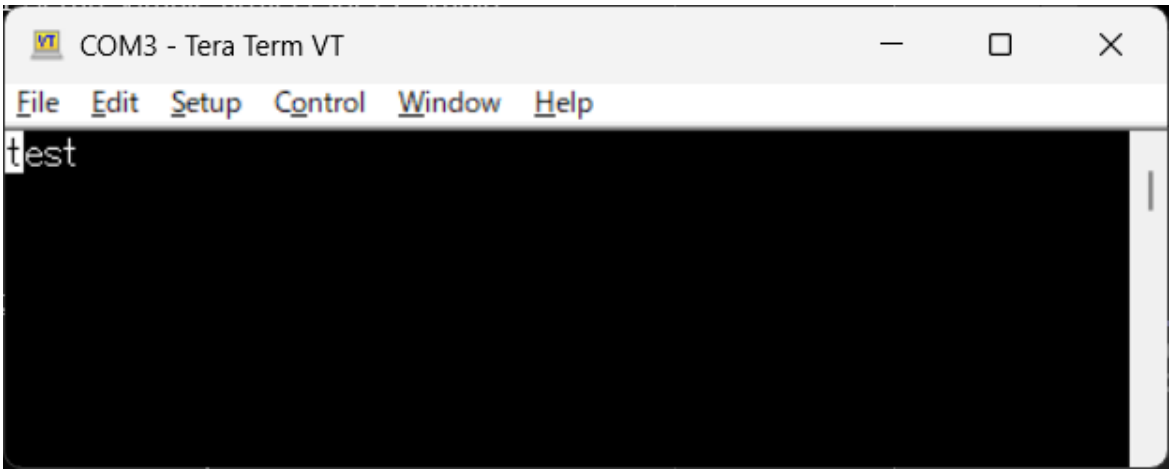


Figure 7.4 Demonstration

7.3 lwIP application.

This sample program can run the following applications as lwIP add-ons.
Please note that this function is available in sample program version 1.2 or later.

Please note that adding this application will not affect other demonstration functions, but please perform any functions other than those described in this section after closing this application.

As a demonstration, a working application.

- SNTP client function
- lwiperf (iperf2) server function
- lwiperf (iperf2) client function

7.3.1 Steps to enable the application.

Each application can be started by enabling conditional compilation, and the settings are shown below.

1. Setting values for conditional compilation

Application	Setting values
SNTP client	Bit0=1: valid integer : 1 0: invalid integer : 0
lwiperf (iperf2) server	Bit1=1: valid integer : 2 0: invalid integer : 0
lwiperf (iperf2) client	Bit2=1: valid integer : 4 0: invalid integer : 0

2. Settings at build time
In each build environment, set the above setting values for the application you want to enable in the builder.

➢ e² studio
Follow the steps below to set it up.
In addition, when enabling multiple functions, please set with OR of each setting value.

- Project menu "Properties" → "C/C++ General" → "Paths and symbols" →"_LWIP_ADD_ON_APP" select it and press the edit button on the right to set it.
- After setting, follow the procedure in Section 3 of "6.1.3" to build.

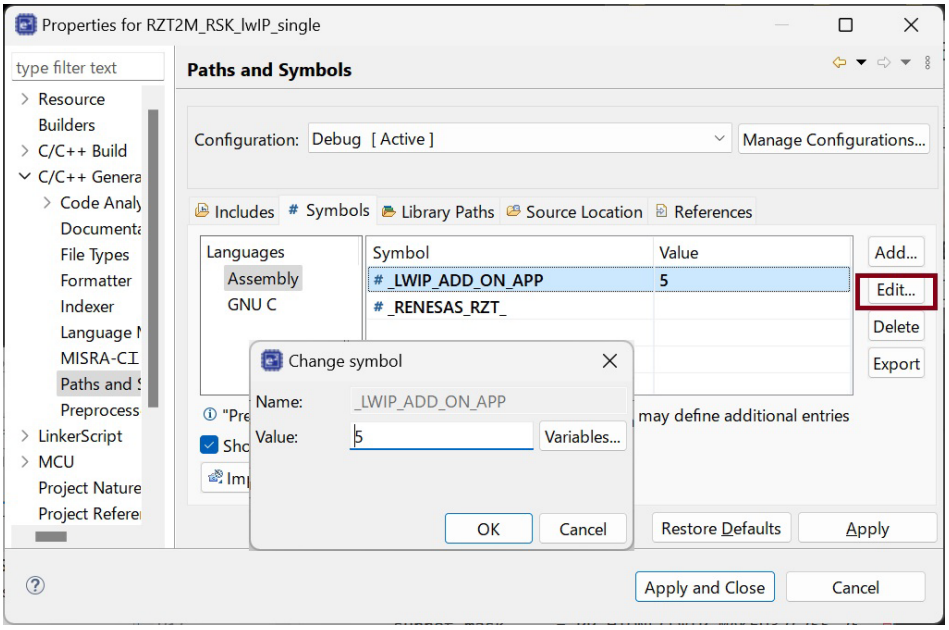


Figure 7.5 e² studio

➢ EWARM

Follow the steps below to set it up.
In addition, when enabling multiple functions, OR each setting value and set it.

- Project menu "Option" → "C/C++ Compiler" → "Preprocessor" →
  Set "_LWIP_ADD_ON_APP=5" directly in the symbol definition field.
- After setting, follow the procedure in Section 3 of "6.2.3" to build.

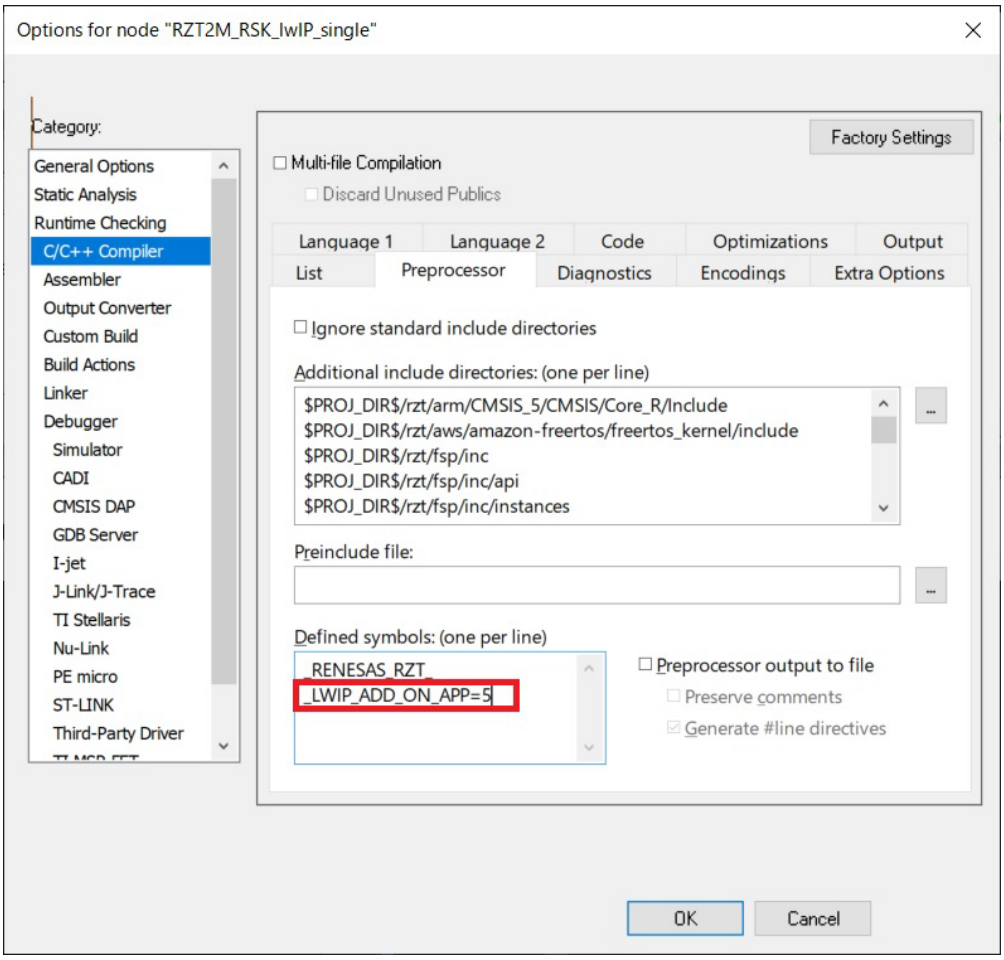


Figure 7.6 EWARM

3. Setting items required by the application
   The required settings for each application are listed below.
   Note that the IP addresses shown in red frames in Figure 7.7 and Figure 7.8 are setting examples, so be sure to set them according to your usage environment.

Application	Defined name	Setting contents
SNTP client	SNTP_SERVER_IP Figure 7.7 Figure 7.8	IP address of the SNTP (NTP) server to connect to
lwiperf (iperf2) server	Use the "IP address" setting in 6.1.2 or 6.2.2	iperf2(lwiperf) server IP address
lwiperf (iperf2) client	IPERF_SERVER_IP Figure 7.7 Figure 7.8	IP address of the iperf2 server to connect to

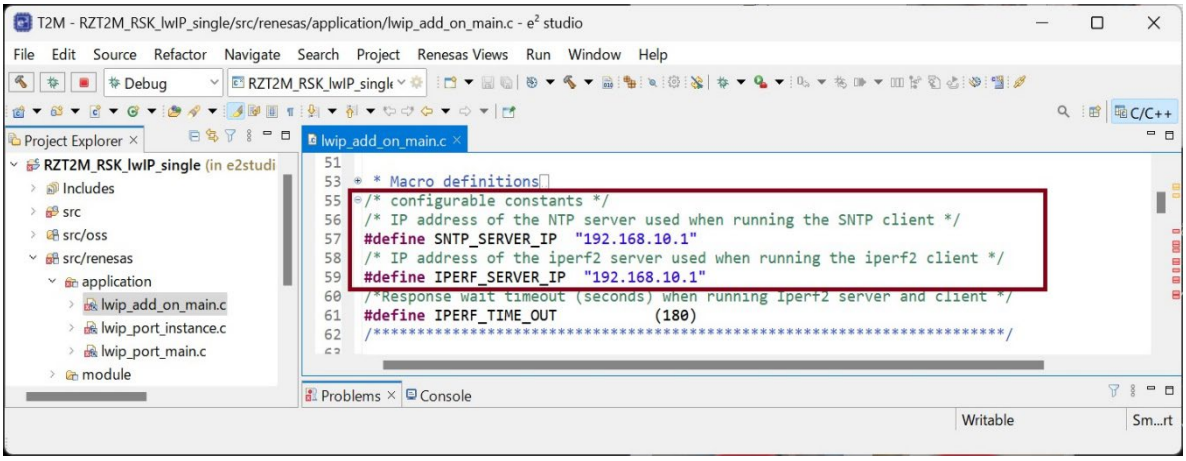


Figure 7.7 e² studio

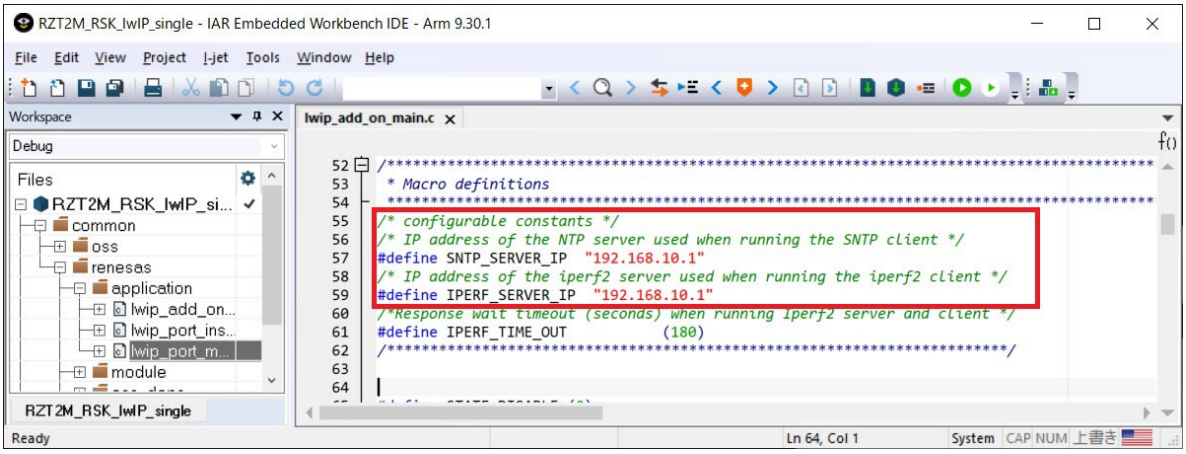


Figure 7.8 EWARM

3. Reference information about connection destinations for each application
   The table below shows the applications to which each application connects.
   Each application does not specify the connection destination application.

Application	Connection destination application	how to get
SNTP client	Use Windows NTP server function	Enable the normally disabled NTP server functionality with a registry change. The setting method is introduced on the website.
lwiperf (iperf2) server lwiperf (iperf2) client	"Iperf 2" is a free software for Windows.	Please do a web search for "Iperf 2". Example of site https://sourceforge.net/projects/iperf2/

7.3.2 Execution steps for each application

The applications set in 2 of “7.3 ” will start in the following order according to the steps in” 6.1.4” or “6.2.4”. Here are the steps:

There is no difference in the procedure between e^2 studio and EWARM.

1. In the application startup order, the startup order decreases in the right direction.
   SNTP client > lwiperf (iperf2) server > lwiperf (iperf2) client
2. The steps to start each application are shown below.

➢ SNTP client
- Before starting the sample program, start the NTP server on the connected Windows PC.
- After displaying the sample program startup LOG, the SNTP client function starts, displays the next LOG, and waits for a response from the NTP server.
  "<< SNTP client application start >>"
- It depends on the communication environment, but normally the notification time will be displayed as shown below after waiting for a response for about 5 seconds, and the process will end.
  "<< SNTP client application completed [09.08.2023 00:00:00] >>"
- If 150 seconds have passed while waiting for a response from the NTP server, the following LOG will be displayed and the process will end.
  "<< SNTP client application end with error >>"

➢ lwiperf (iperf2) server
- After displaying the startup LOG of the sample program, if the lwiperf (iperf2) server function is enabled, the following LOG will be displayed, waiting for a connection from the lwiperf (iperf2) client.
  "<< IPERF2 server application start >>"
- After the above start LOG is displayed, start the lwiperf (iperf2) client from the connected Windows PC.
- Although it depends on the communication environment, it usually takes about 15 seconds after starting communication from the connection destination Windows PC, and the communication report is displayed as shown below and ends.
  "<< IPERF2 server application completed>>"
  "IPERF report: type=0, remote: 192.168.10.1:56861, total bytes: xxxxxxx, duration in ms: xxxxx, kbits/s: xxxxx "
- If there is no connection from the lwiperf (iperf2) client within 180 seconds, the following LOG will be displayed and the process will end.
  " << IPERF2 server application end with error >>"

➢ lwiperf (iperf2) client
- Before starting the sample program, start the iperf2 server on the connected Windows PC.
- After the sample program startup LOG is displayed, the lwiperf (iperf2) client function starts, the next LOG is displayed, and it waits for a response from the iperf2 server.
  "<< IPERF2 client application start >>"
- Depending on the communication environment, it usually takes about 15 seconds to display a communication report as shown below and finish.
  "<< IPERF2 client application completed >>"
  "IPERF report: type=1, remote: 192.168.10.1:5001, total bytes: xxxxxxx, duration in ms: 10000, kbits/s: xxxxx "
- If 180 seconds have passed while waiting for a response from the iperf2 server, it will display the following LOG and exit.
  "<< IPERF2 client application end with error >>"

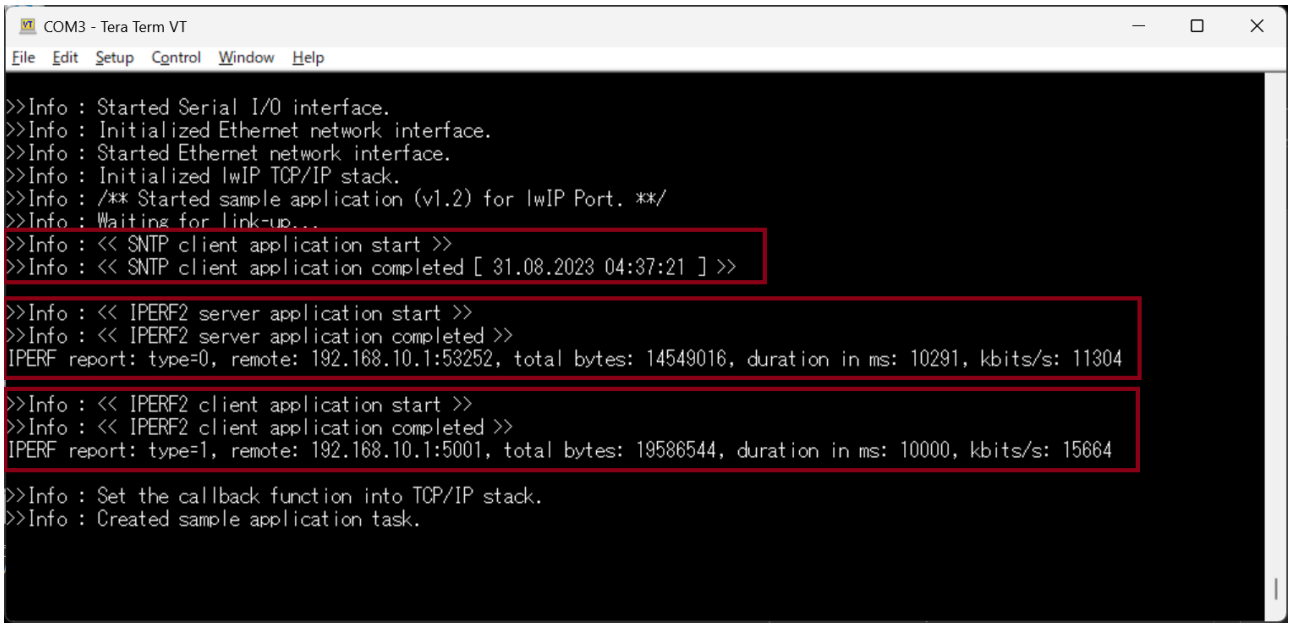


Figure 7.9
Execution example with SNTP client, lwiperf (iperf2) server and lwiperf (iperf2) client enabled

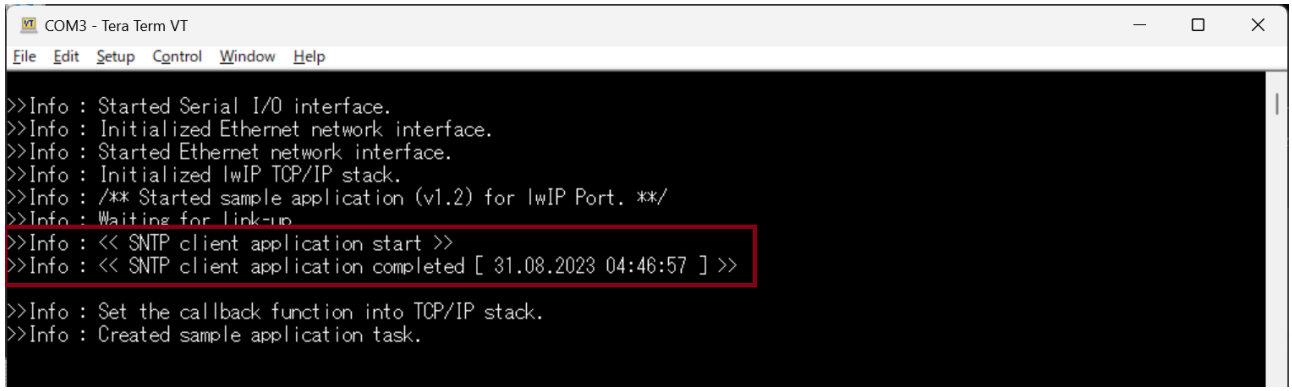


Figure 7.10 Execution example with lwiperf (iperf2) client enabled

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

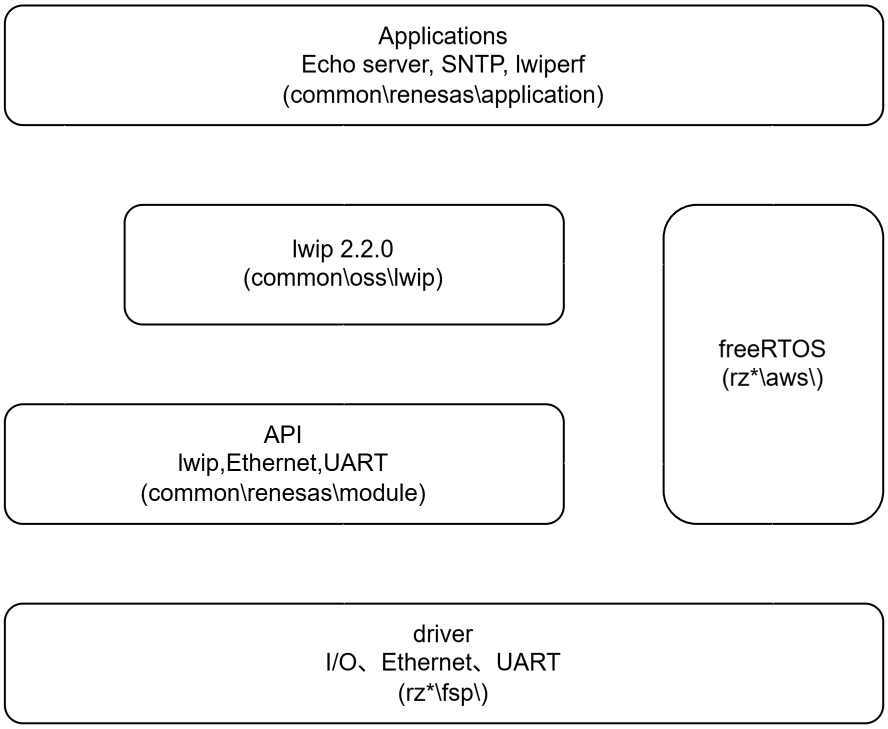


Figure 8.1. Software Architecture Diagram

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	
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	task_code	3	1024	Wake up every 1 second	Ethernet link status monitoring
	task_code	4	2048	Received events	Waiting for received Ethernet data
lwIP Port	task_code	3	1024	Queue notification events	LINK status notification processing
lwip	lwIP	4	1024	Received events	Reception processing within lwip

Note: FreeRTOS schedules tasks based on priority values, with higher values indicating higher priority.

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.

Figure 8.2. sequence

8.2.3 Mirroring function for received packets

Mirroring function for received packets

This shows the function to notify additional modules of data received from the driver, similar to lwIP.

1. Overview

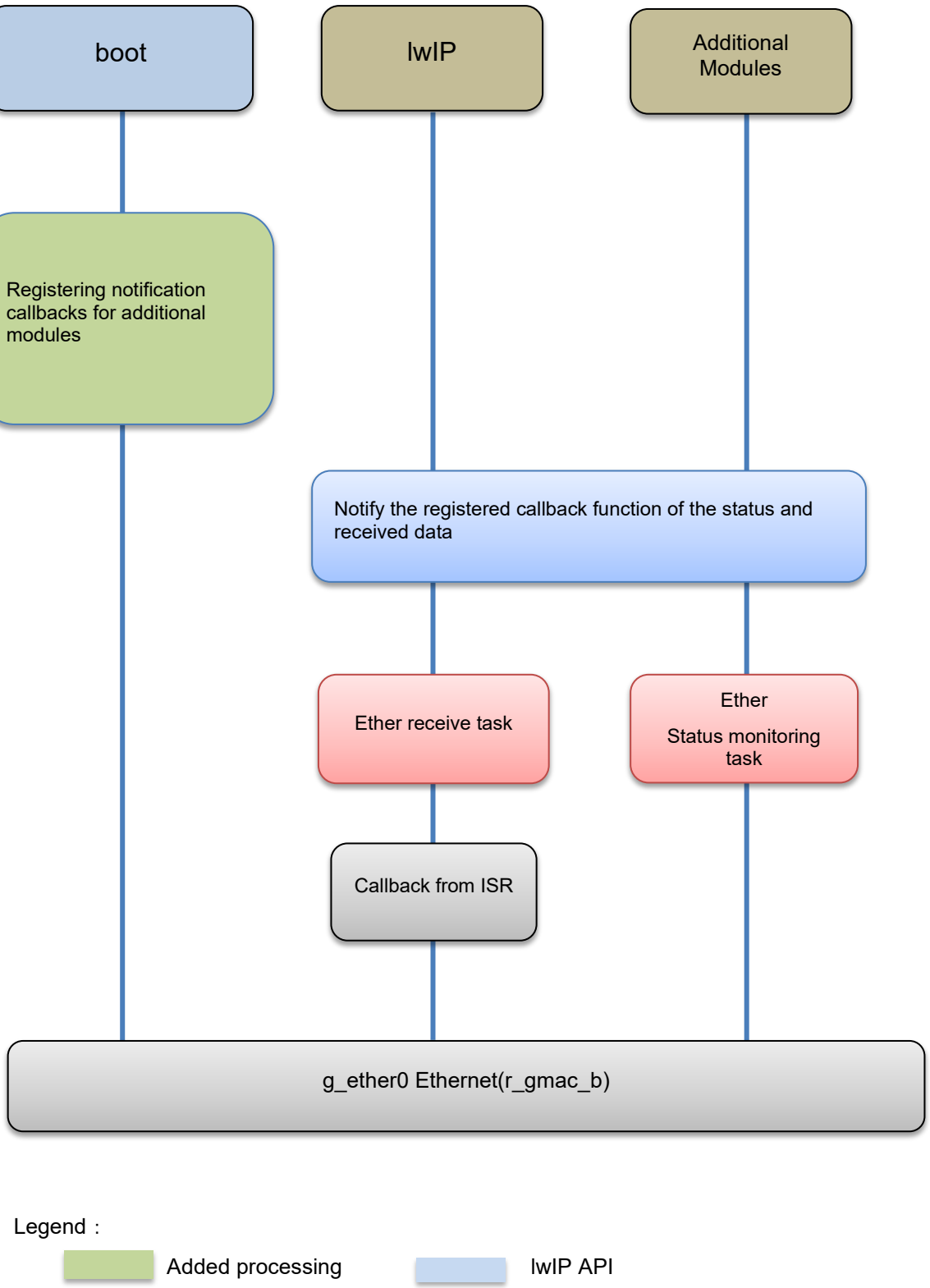


Figure 8.3. sequence

2. Interface

To enable the mirroring feature, register a callback function with the lwIP API to receive received data (MAC frames).

➤ Registering the Callback Function

Sample source code is commented out in the lwIP function "um_lwip_port_ether_netif.c / um_lwip_port_ether_netif_open."

➤ Notification Types

- ETHER_NETIF_CALLBACK_EVENT_LINK_UP
  Ethernet status: Notified when a link is established. No notification data.
- ETHER_NETIF_CALLBACK_EVENT_LINK_DOWN
  Ethernet status: Notified when a link is disconnected. No notification data.
- ETHER_NETIF_CALLBACK_EVENT_RECEIVE_ETHER_FRAME
  An Ethernet receive event notifies the received MAC frame (RAW) as a parameter.

  Note that the parameter buffer "p_frame_packet" must be freed after use.

8.2.4 Ethernet Sending API

If you want to send MAC frames (RAW) directly without going through lwIP, a sample code is commented out in the source code below.

The sample code is commented out in the lwIP function "um_lwip_port_mainf.c/115".

8.3 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.

8.3.1 e² studio 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.
- xxxxxxxxx.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.

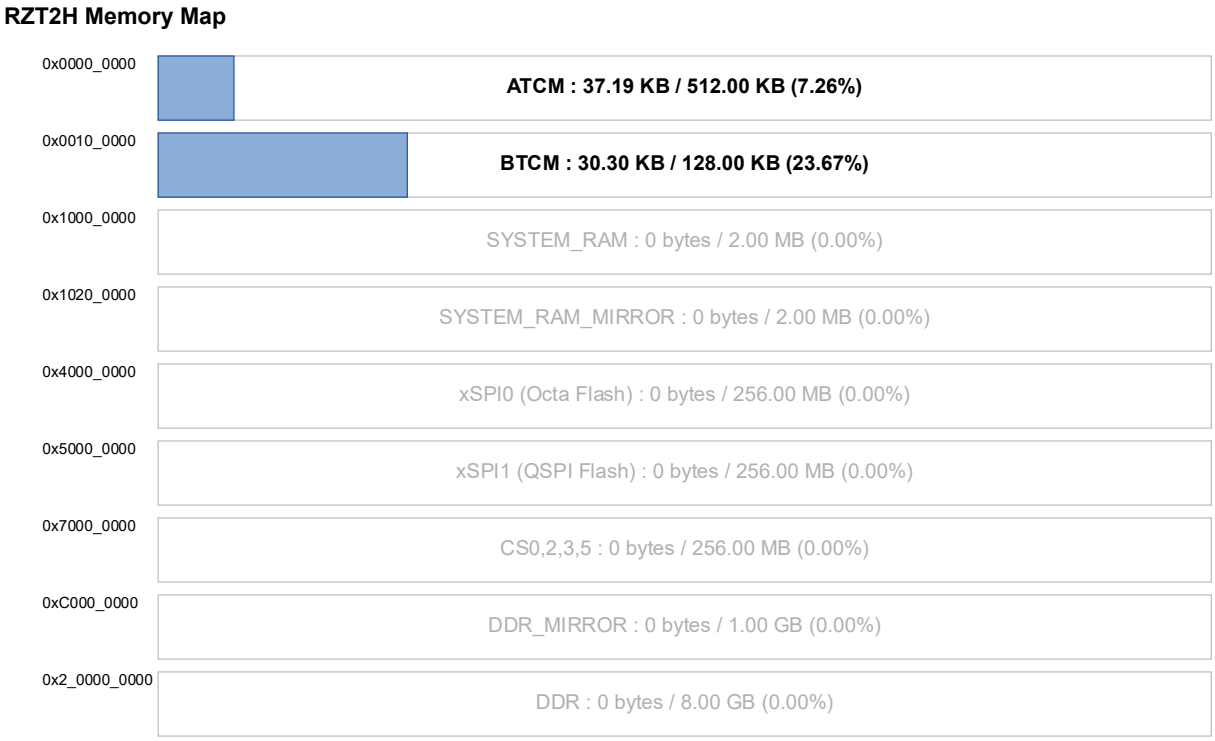
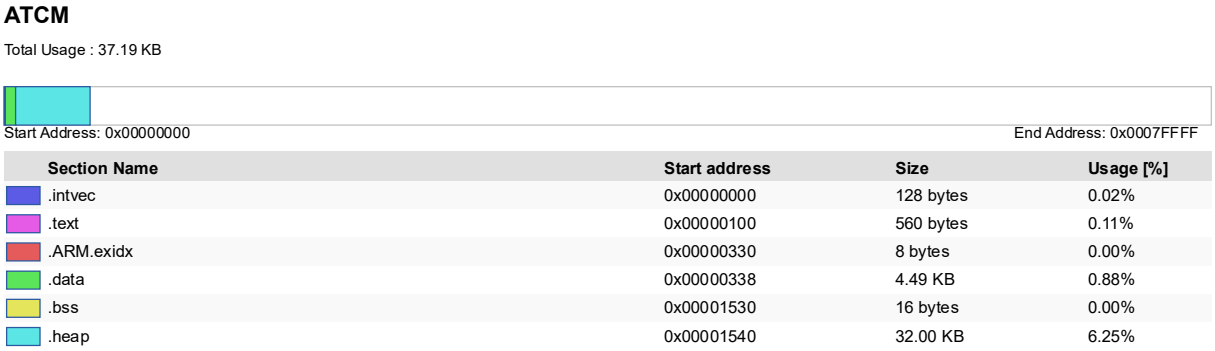


Figure 8.4. Memory footprint of the CPU0 project (e² studio)



Section Name	Start address	Size	Usage [%]
.intvec	0x00000000	128 bytes	0.02%
.text	0x00000100	560 bytes	0.11%
.ARM.exidx	0x00000330	8 bytes	0.00%
.data	0x00000338	4.49 KB	0.88%
.bss	0x00001530	16 bytes	0.00%
.heap	0x00001540	32.00 KB	6.25%

Figure 8.5. ATCM footprint of the CPU0 project (e² studio)

Section Name	Start address	Size	Usage [%]
.loader_text	0x00102000	4.16 KB	3.25%
.loader_data	0x001030A8	208 bytes	0.16%
.loader_bss	0x00103178	209 bytes	0.16%
.loader_stack	0x00103250	24.00 KB	18.75%

Figure 8.6. BTCM footprint of the CPU0 project (e² studio)

➤ CPU1(CR52_1)
The application running on CPU1 will perform functions other than blinking the LED.

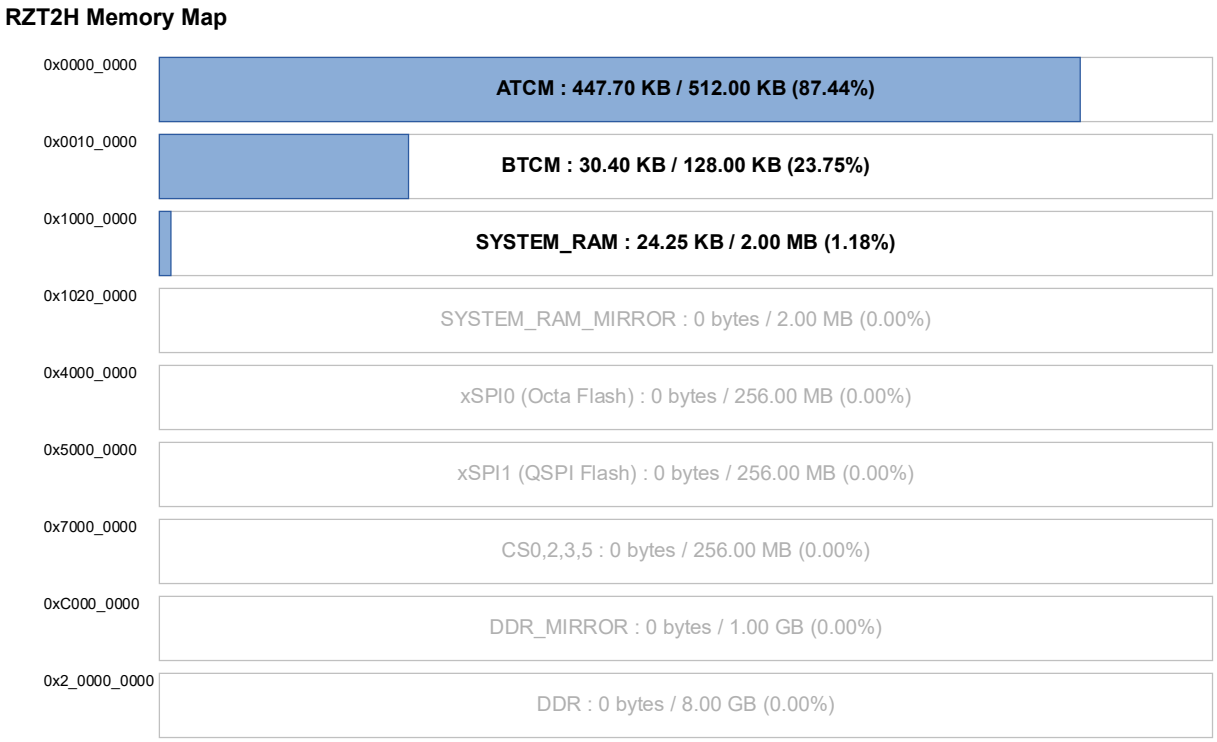
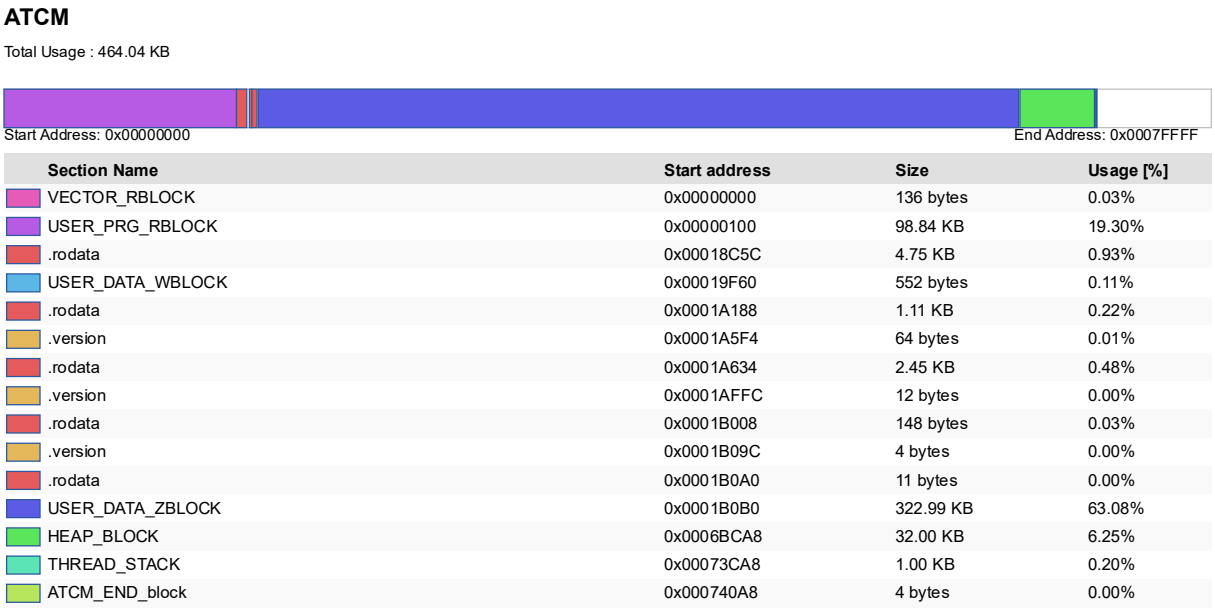


Figure 8.7. Memory footprint of the CPU1 project (e² studio)



Section Name	Start address	Size	Usage [%]
VECTOR_RBLOCK	0x00000000	136 bytes	0.03%
USER_PRG_RBLOCK	0x00000100	98.84 KB	19.30%
.rodata	0x00018C5C	4.75 KB	0.93%
USER_DATA_WBLOCK	0x00019F60	552 bytes	0.11%
.rodata	0x0001A188	1.11 KB	0.22%
.version	0x0001A5F4	64 bytes	0.01%
.rodata	0x0001A634	2.45 KB	0.48%
.version	0x0001AFFC	12 bytes	0.00%
.rodata	0x0001B008	148 bytes	0.03%
.version	0x0001B09C	4 bytes	0.00%
.rodata	0x0001B0A0	11 bytes	0.00%
USER_DATA_ZBLOCK	0x0001B0B0	322.99 KB	63.08%
HEAP_BLOCK	0x0006BCA8	32.00 KB	6.25%
THREAD_STACK	0x00073CA8	1.00 KB	0.20%
ATCM_END_block	0x000740A8	4 bytes	0.00%

Figure 8.8. ATCM footprint of the CPU1 project (e² studio)

BTCM

Total Usage : 30.76 KB

Start Address: 0x00100000 End Address: 0x0011FFFF

Section Name	Start address	Size	Usage [%]
LDR_PRG_RBLOCK	0x00102000	4.11 KB	3.21%
LDR_DATA_WBLOCK	0x00103070	204 bytes	0.16%
LOADER_STACK	0x00103140	24.00 KB	18.75%
LDR_DATA_ZBLOCK	0x00109140	2.45 KB	1.91%
SOFTWARE_RESET_ZBLOCK	0x00109B0D	1 bytes	0.00%
BTCM_END_block	0x00109B10	4 bytes	0.00%

Figure 8.9. BTCM footprint of the CPU1 project (e² studio)

SYSTEM_RAM

Total Usage : 24.25 KB

Start Address: 0x10000000 End Address: 0x101FFFFF

Section Name	Start address	Size	Usage [%]
.noncache	0x10180000	24.25 KB	1.18%

Figure 8.10. 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.

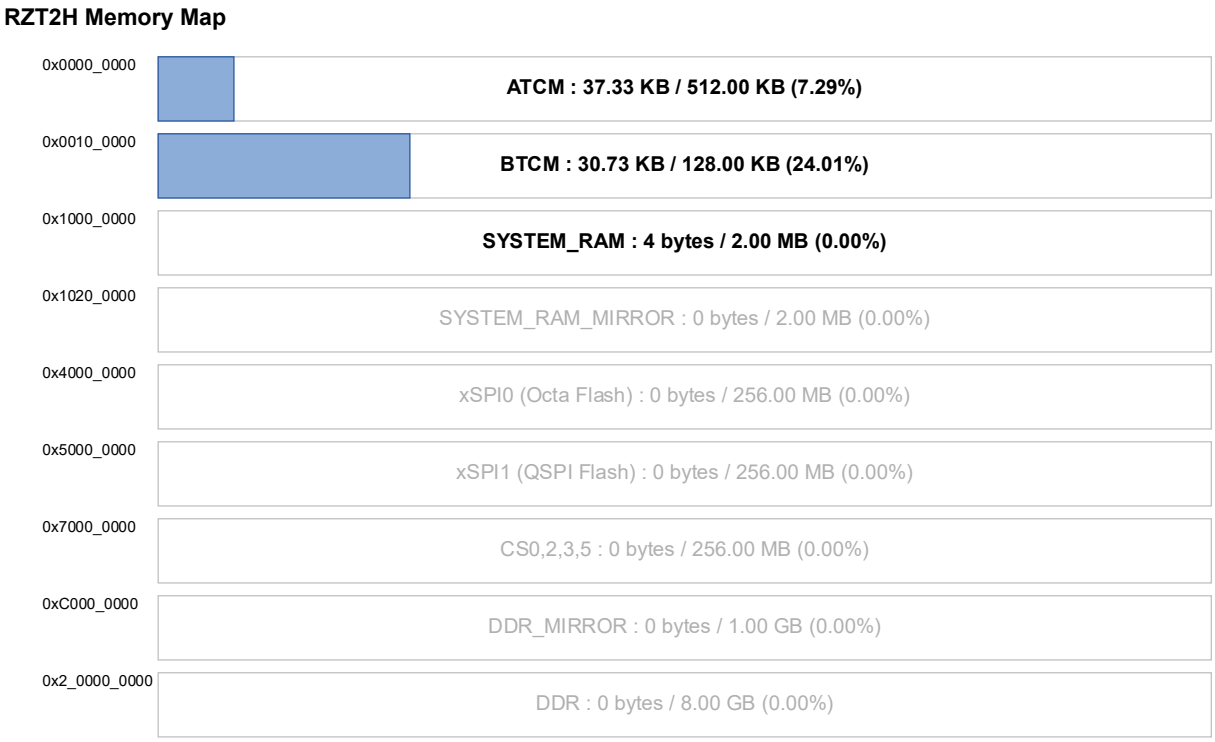
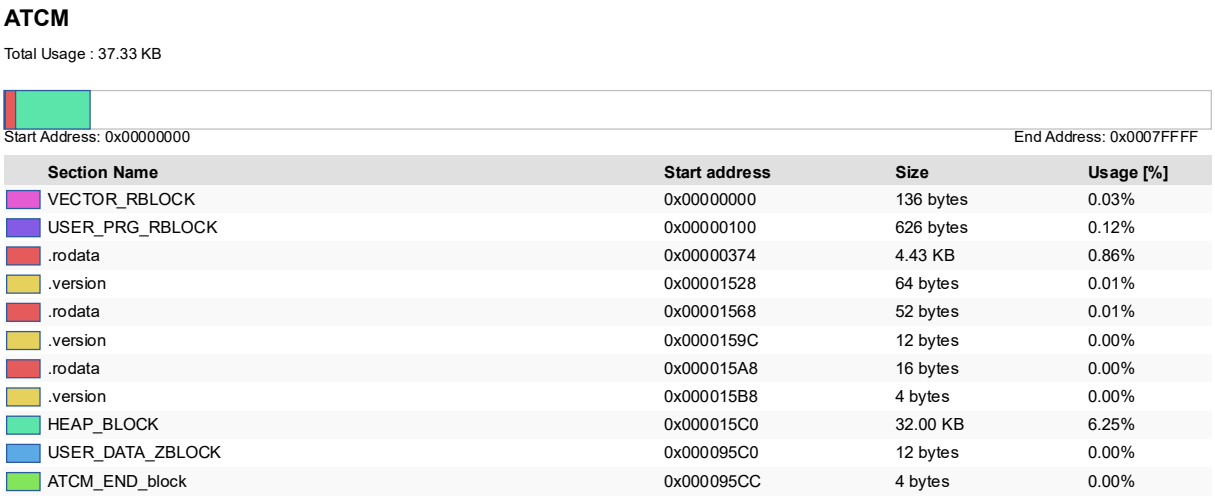
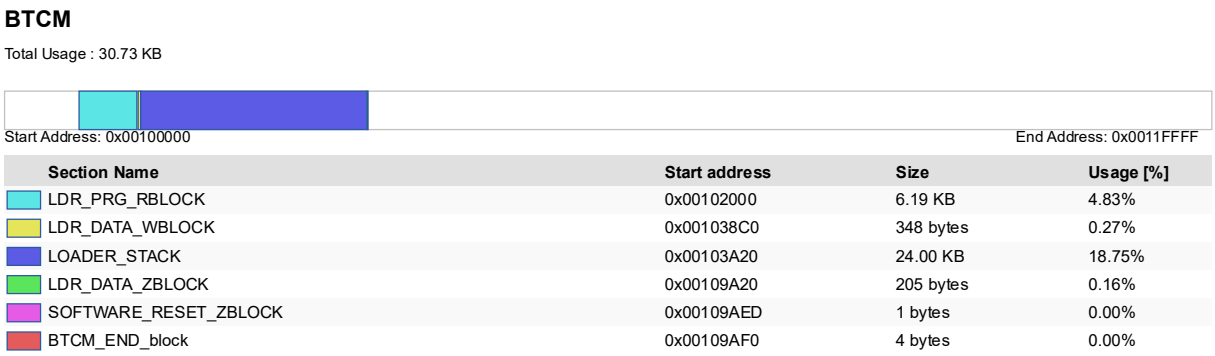


Figure 8.11. Memory footprint of the CPU0 project (e² studio)



Section Name	Start address	Size	Usage [%]
VECTOR_RBLOCK	0x00000000	136 bytes	0.03%
USER_PRG_RBLOCK	0x00000100	626 bytes	0.12%
.rodata	0x00000374	4.43 KB	0.86%
.version	0x00001528	64 bytes	0.01%
.rodata	0x00001568	52 bytes	0.01%
.version	0x0000159C	12 bytes	0.00%
.rodata	0x000015A8	16 bytes	0.00%
.version	0x000015B8	4 bytes	0.00%
HEAP_BLOCK	0x000015C0	32.00 KB	6.25%
USER_DATA_ZBLOCK	0x000095C0	12 bytes	0.00%
ATCM_END_block	0x000095CC	4 bytes	0.00%

Figure 8.12. ATCM footprint of the CPU0 project (e² studio)



BTCM

Total Usage : 30.73 KB

Start Address: 0x00100000 End Address: 0x0011FFFF

Section Name	Start address	Size	Usage [%]
LDR_PRG_RBLOCK	0x00102000	6.19 KB	4.83%
LDR_DATA_WBLOCK	0x001038C0	348 bytes	0.27%
LOADER_STACK	0x00103A20	24.00 KB	18.75%
LDR_DATA_ZBLOCK	0x00109A20	205 bytes	0.16%
SOFTWARE_RESET_ZBLOCK	0x00109AED	1 bytes	0.00%
BTCM_END_block	0x00109AF0	4 bytes	0.00%

Figure 8.13. BTCM footprint of the CPU0 project (e^2 studio)

➤ CPU1(CA55_2)
The application running on CPU1 will perform functions other than blinking the LED.

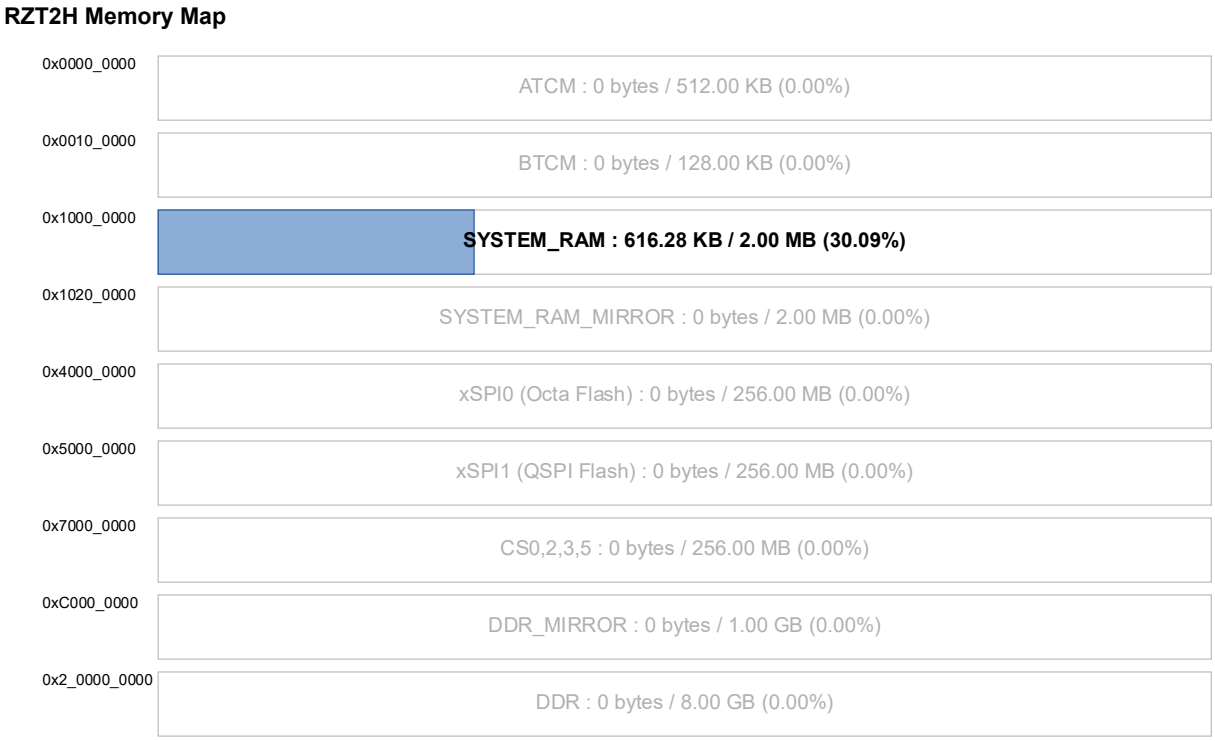
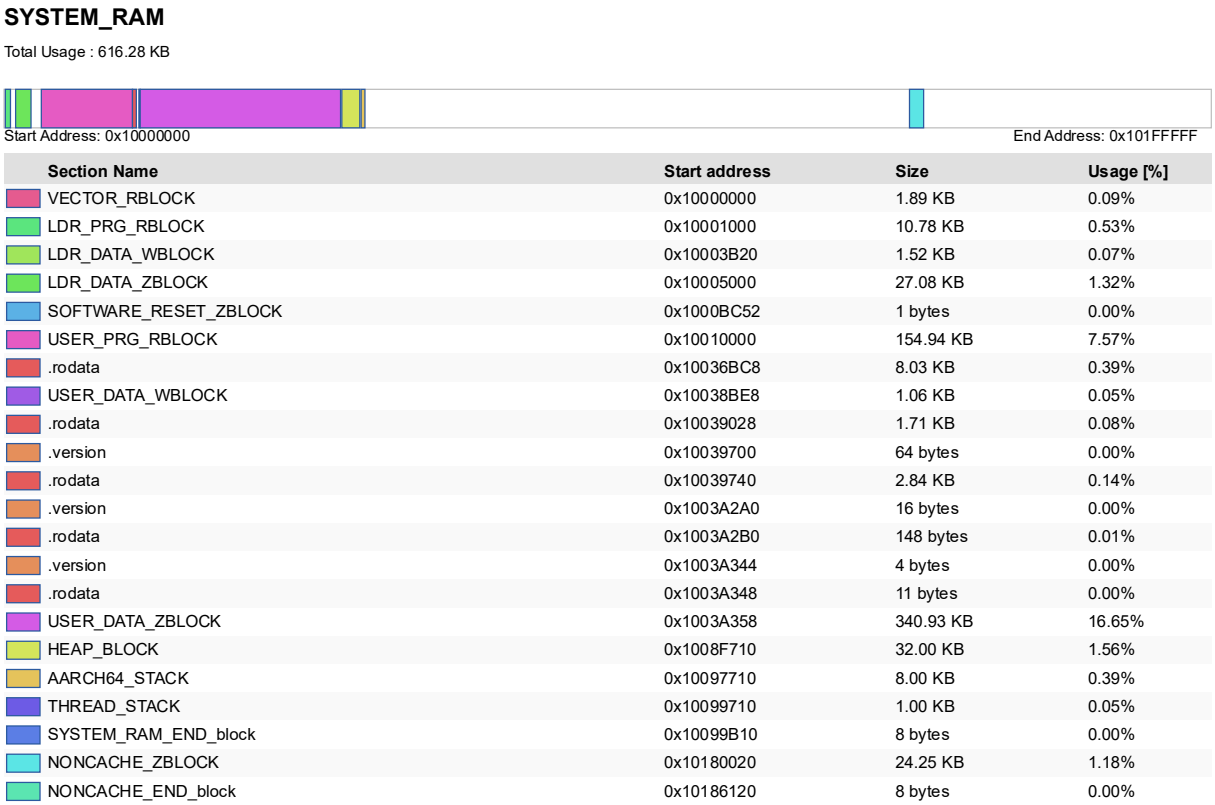


Figure 8.14. Memory footprint of the CPU1 project (e² studio)



Section Name	Start address	Size	Usage [%]
VECTOR_RBLOCK	0x10000000	1.89 KB	0.09%
LDR_PRG_RBLOCK	0x10001000	10.78 KB	0.53%
LDR_DATA_WBLOCK	0x10003B20	1.52 KB	0.07%
LDR_DATA_ZBLOCK	0x10005000	27.08 KB	1.32%
SOFTWARE_RESET_ZBLOCK	0x1000BC52	1 bytes	0.00%
USER_PRG_RBLOCK	0x10010000	154.94 KB	7.57%
.rodata	0x10036BC8	8.03 KB	0.39%
USER_DATA_WBLOCK	0x10038BE8	1.06 KB	0.05%
.rodata	0x10039028	1.71 KB	0.08%
.version	0x10039700	64 bytes	0.00%
.rodata	0x10039740	2.84 KB	0.14%
.version	0x1003A2A0	16 bytes	0.00%
.rodata	0x1003A2B0	148 bytes	0.01%
.version	0x1003A344	4 bytes	0.00%
.rodata	0x1003A348	11 bytes	0.00%
USER_DATA_ZBLOCK	0x1003A358	340.93 KB	16.65%
HEAP_BLOCK	0x1008F710	32.00 KB	1.56%
AARCH64_STACK	0x10097710	8.00 KB	0.39%
THREAD_STACK	0x10099710	1.00 KB	0.05%
SYSTEM_RAM_END_block	0x10099B10	8 bytes	0.00%
NONCACHE_ZBLOCK	0x10180020	24.25 KB	1.18%
NONCACHE_END_block	0x10186120	8 bytes	0.00%

Figure 8.15. SYSTEM_RAM 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.

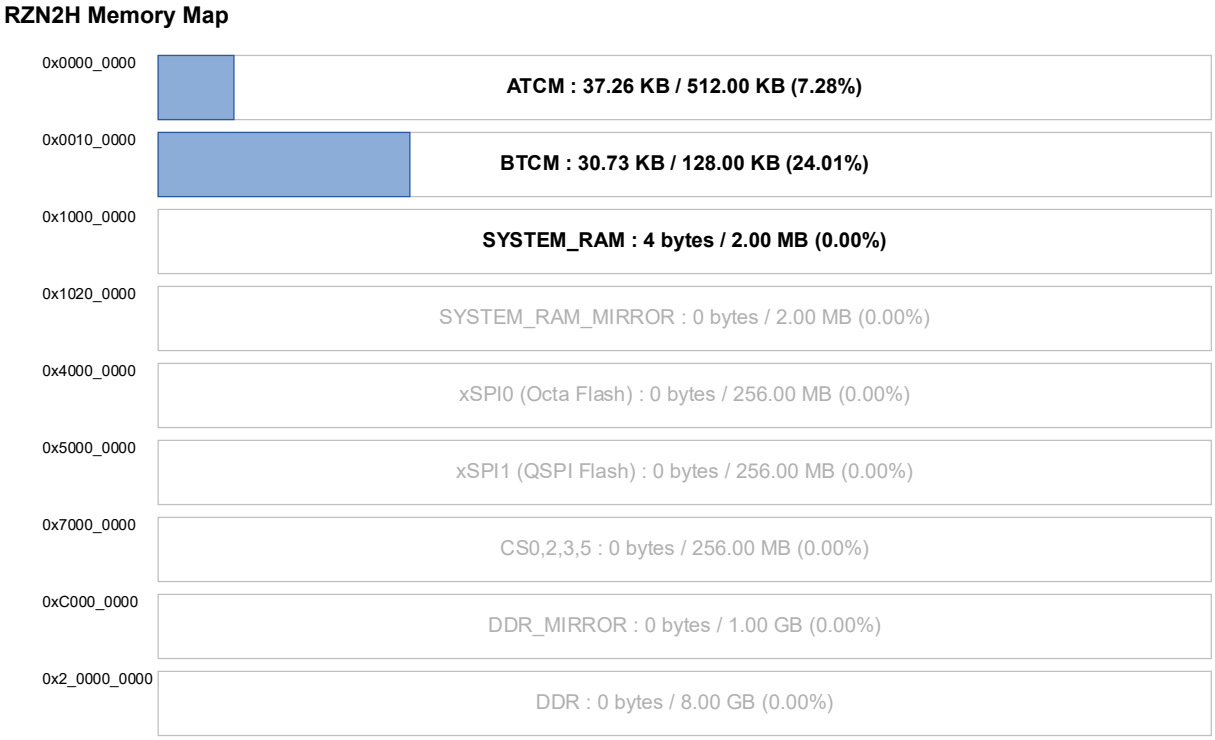
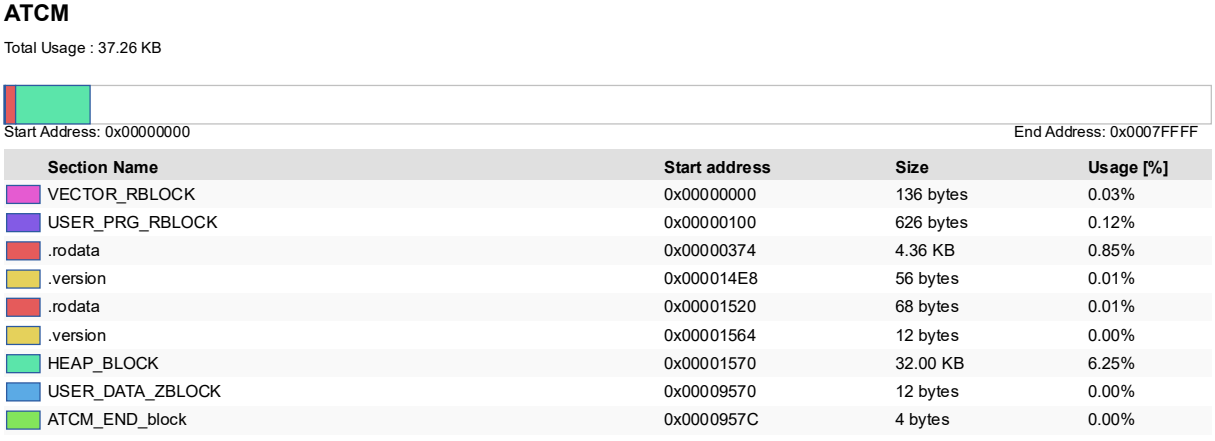
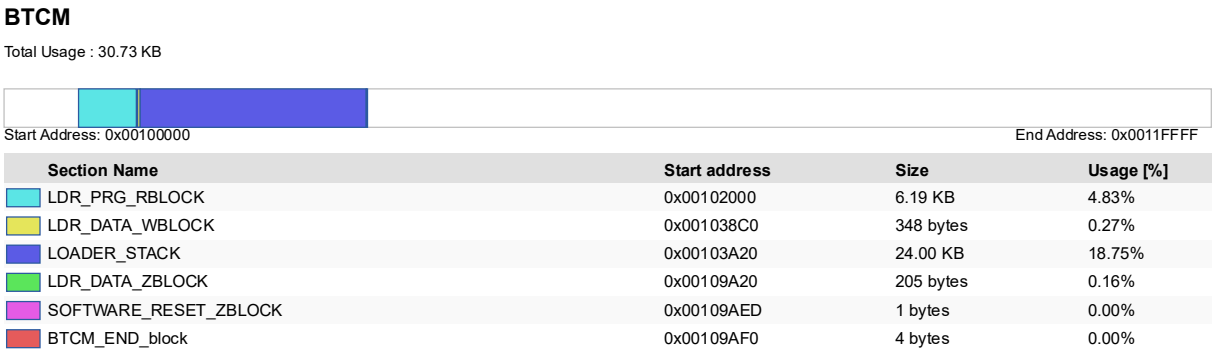


Figure 8.16. Memory footprint of the CPU0 project (e² studio)



Section Name	Start address	Size	Usage [%]
VECTOR_RBLOCK	0x00000000	136 bytes	0.03%
USER_PRG_RBLOCK	0x00000100	626 bytes	0.12%
.rodata	0x00000374	4.36 KB	0.85%
.version	0x000014E8	56 bytes	0.01%
.rodata	0x00001520	68 bytes	0.01%
.version	0x00001564	12 bytes	0.00%
HEAP_BLOCK	0x00001570	32.00 KB	6.25%
USER_DATA_ZBLOCK	0x00009570	12 bytes	0.00%
ATCM_END_block	0x0000957C	4 bytes	0.00%

Figure 8.17. ATCM footprint of the CPU0 project (e² studio)



Section Name	Start address	Size	Usage [%]
LDR_PRG_RBLOCK	0x00102000	6.19 KB	4.83%
LDR_DATA_WBLOCK	0x001038C0	348 bytes	0.27%
LOADER_STACK	0x00103A20	24.00 KB	18.75%
LDR_DATA_ZBLOCK	0x00109A20	205 bytes	0.16%
SOFTWARE_RESET_ZBLOCK	0x00109AED	1 bytes	0.00%
BTCM_END_block	0x00109AF0	4 bytes	0.00%

Figure 8.18. BTCM footprint of the CPU0 project (e^2 studio)

➤ CPU1(CR52_1)
The application running on CPU1 will perform functions other than blinking the LED.

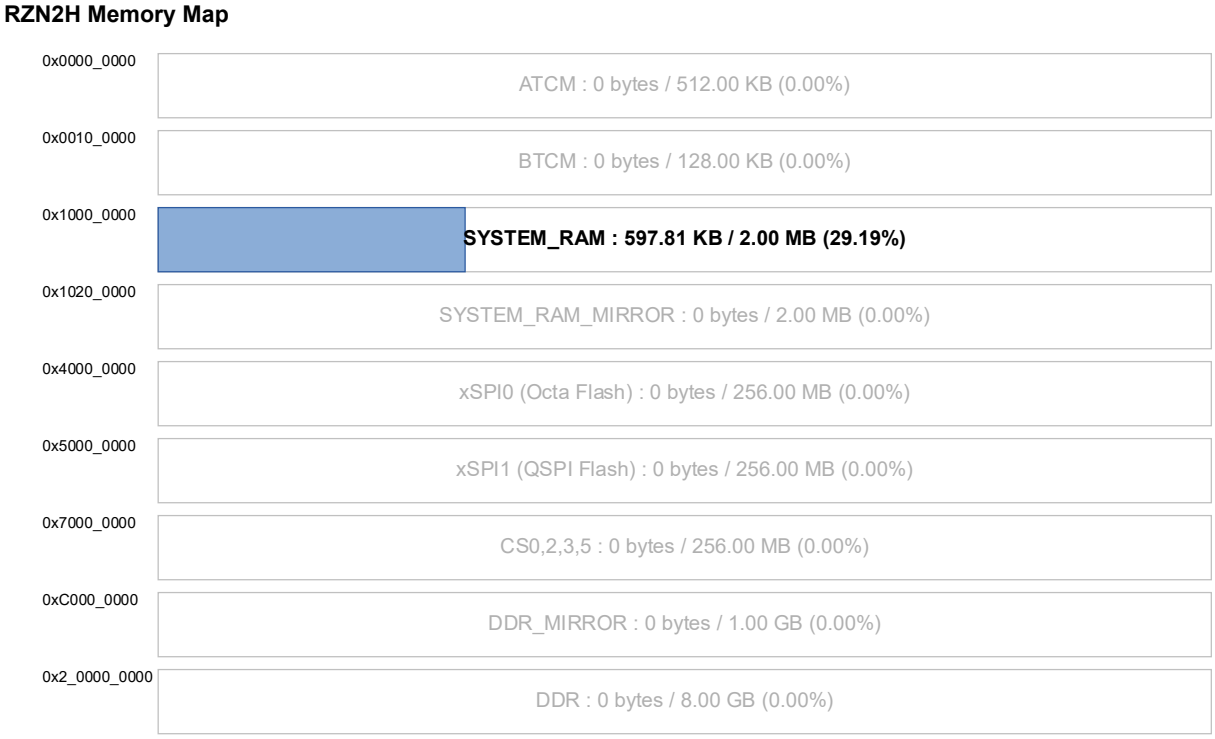


Figure 8.19. SYSTEM_RAM footprint of the CPU1 project (e² studio)

SYSTEM_RAM

Total Usage : 597.81 KB

Section Name	Start address	Size	Usage [%]
VECTOR_RBLOCK	0x10000000	1.89 KB	0.09%
LDR_PRG_RBLOCK	0x10001000	10.71 KB	0.52%
LDR_DATA_WBLOCK	0x10003AE0	1.52 KB	0.07%
LDR_DATA_ZBLOCK	0x10005000	27.08 KB	1.32%
SOFTWARE_RESET_ZBLOCK	0x1000BC52	1 bytes	0.00%
USER_PRG_RBLOCK	0x10010000	140.62 KB	6.87%
.rodata	0x10033278	6.47 KB	0.32%
USER_DATA_WBLOCK	0x10034C58	944 bytes	0.05%
.rodata	0x10035008	1.38 KB	0.07%
.version	0x10035590	56 bytes	0.00%
.rodata	0x100355C8	2.08 KB	0.10%
.version	0x10035E18	8 bytes	0.00%
.rodata	0x10035E20	16 bytes	0.00%
.version	0x10035E30	4 bytes	0.00%
.rodata	0x10035E34	7 bytes	0.00%
USER_DATA_ZBLOCK	0x10035E40	339.79 KB	16.59%
HEAP_BLOCK	0x1008AD70	32.00 KB	1.56%
AARCH64_STACK	0x10092D70	8.00 KB	0.39%
THREAD_STACK	0x10094D70	1.00 KB	0.05%
SYSTEM_RAM_END_block	0x10095170	8 bytes	0.00%
NONCACHE_ZBLOCK	0x10180020	24.25 KB	1.18%
NONCACHE_END_block	0x10186120	8 bytes	0.00%

Figure 8.20. Memory 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

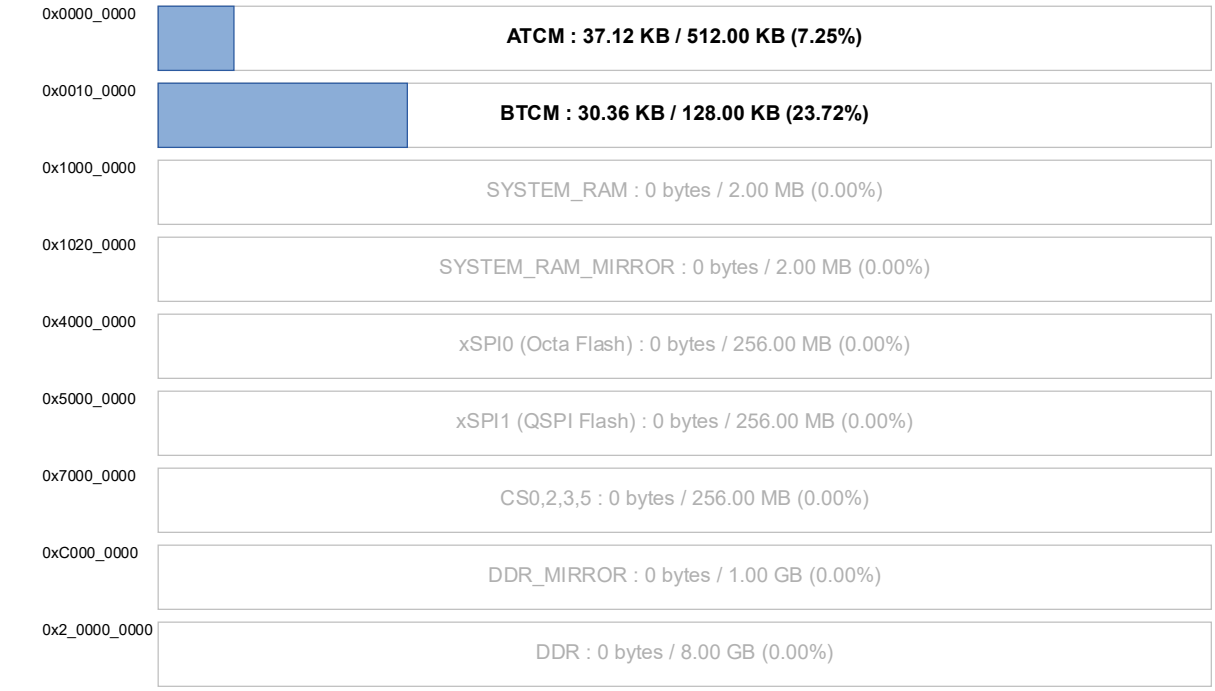


Figure 8.21. Memory footprint of the CPU0 project (e² studio)

ATCM

Total Usage : 37.12 KB

Start Address: 0x00000000 End Address: 0x0007FFFF

Section Name	Start address	Size	Usage [%]
.intvec	0x00000000	128 bytes	0.02%
.text	0x00000100	560 bytes	0.11%
.ARM.exidx	0x00000330	8 bytes	0.00%
.data	0x00000338	4.43 KB	0.87%
.bss	0x000014F0	16 bytes	0.00%
.heap	0x00001500	32.00 KB	6.25%

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

BTCM

Total Usage : 30.36 KB

Start Address: 0x00100000 End Address: 0x0011FFFF

Section Name	Start address	Size	Usage [%]
.loader_text	0x00102000	5.81 KB	4.54%
.loader_data	0x00103740	352 bytes	0.27%
.loader_bss	0x001038A0	209 bytes	0.16%
.loader_stack	0x00103978	24.00 KB	18.75%

Figure 8.23. BTCM footprint of the CPU0 project (e² studio)

➤ CPU1(CA55_2)

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

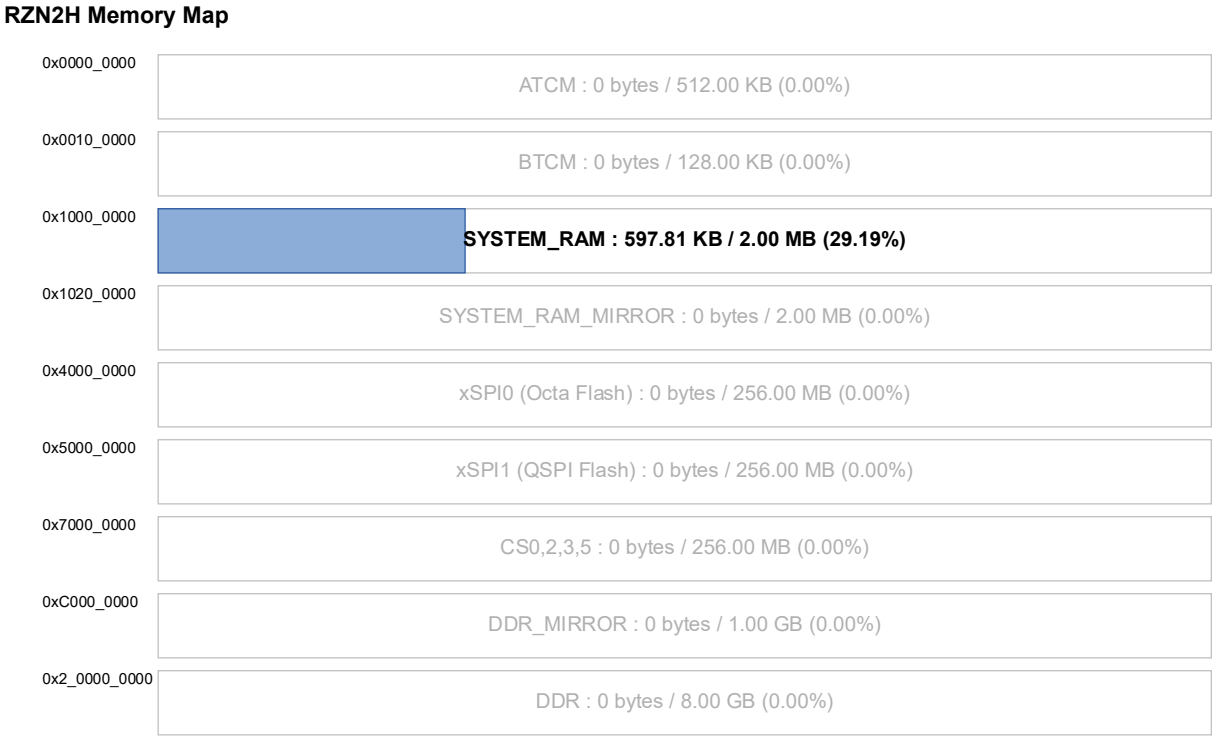


Figure 8.24. Memory footprint of the CPU1 project (e² studio)

Section Name	Start address	Size	Usage [%]
VECTOR_RBLOCK	0x10000000	1.89 KB	0.09%
LDR_PRG_RBLOCK	0x10001000	10.71 KB	0.52%
LDR_DATA_WBLOCK	0x10003AE0	1.52 KB	0.07%
LDR_DATA_ZBLOCK	0x10005000	27.08 KB	1.32%
SOFTWARE_RESET_ZBLOCK	0x1000BC52	1 bytes	0.00%
USER_PRG_RBLOCK	0x10010000	140.62 KB	6.87%
.rodata	0x10033278	6.47 KB	0.32%
USER_DATA_WBLOCK	0x10034C58	944 bytes	0.05%
.rodata	0x10035008	1.38 KB	0.07%
.version	0x10035590	56 bytes	0.00%
.rodata	0x100355C8	2.08 KB	0.10%
.version	0x10035E18	8 bytes	0.00%
.rodata	0x10035E20	16 bytes	0.00%
.version	0x10035E30	4 bytes	0.00%
.rodata	0x10035E34	7 bytes	0.00%
USER_DATA_ZBLOCK	0x10035E40	339.79 KB	16.59%
HEAP_BLOCK	0x1008AD70	32.00 KB	1.56%
AARCH64_STACK	0x10092D70	8.00 KB	0.39%
THREAD_STACK	0x10094D70	1.00 KB	0.05%
SYSTEM_RAM_END_block	0x10095170	8 bytes	0.00%
NONCACHE_ZBLOCK	0x10180020	24.25 KB	1.18%
NONCACHE_END_block	0x10186120	8 bytes	0.00%

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

8.3.2 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)

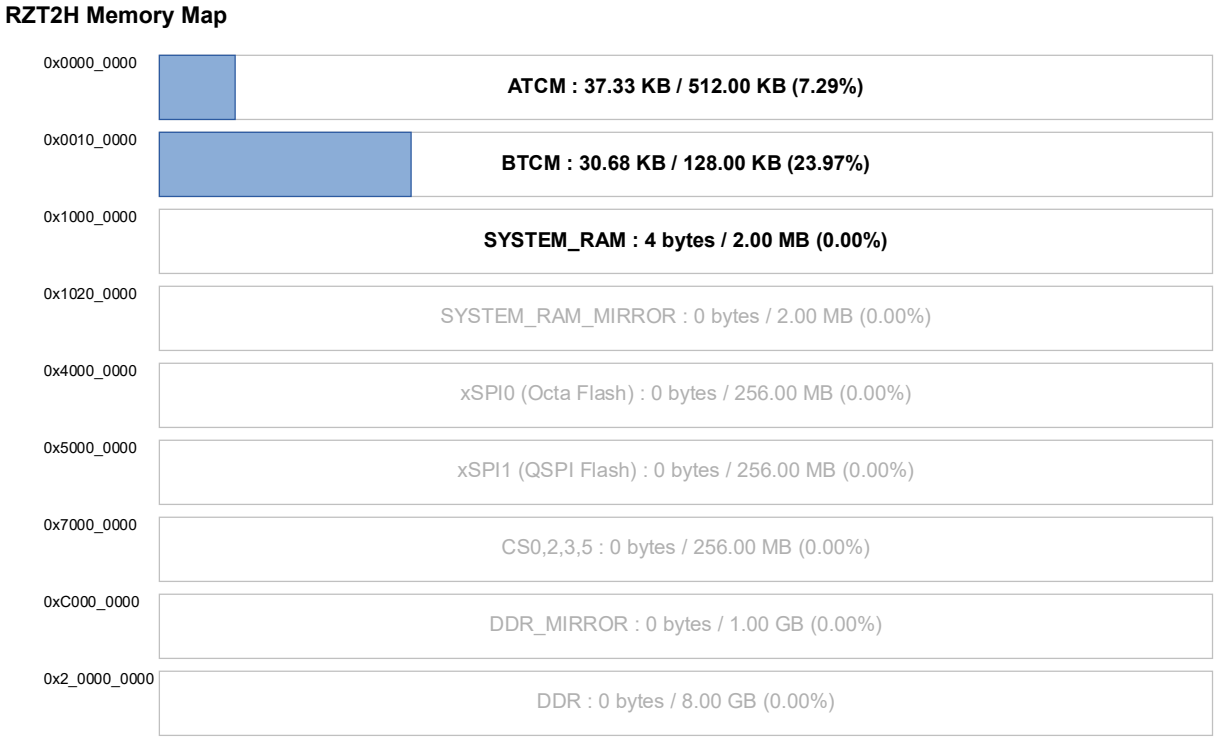


Figure 8.26. Memory footprint of the CPU0 project (EWARM)

ATCM

Total Usage : 37.33 KB

Start Address: 0x00000000 End Address: 0x0007FFFF

Section Name	Start address	Size	Usage [%]
VECTOR_RBLOCK	0x00000000	136 bytes	0.03%
USER_PRG_RBLOCK	0x00000100	626 bytes	0.12%
.rodata	0x00000374	4.43 KB	0.86%
.version	0x00001528	64 bytes	0.01%
.rodata	0x00001568	52 bytes	0.01%
.version	0x0000159C	12 bytes	0.00%
.rodata	0x000015A8	16 bytes	0.00%
.version	0x000015B8	4 bytes	0.00%
HEAP_BLOCK	0x000015C0	32.00 KB	6.25%
USER_DATA_ZBLOCK	0x000095C0	12 bytes	0.00%
ATCM_END_block	0x000095CC	4 bytes	0.00%

Figure 8.27. ATCM footprint of the CPU0 project (EWARM)

BTCM

Total Usage : 30.68 KB

Start Address: 0x00100000 End Address: 0x0011FFFF

Section Name	Start address	Size	Usage [%]
LDR_PRG_RBLOCK	0x00102000	6.13 KB	4.79%
LDR_DATA_WBLOCK	0x00103888	348 bytes	0.27%
LOADER_STACK	0x001039E8	24.00 KB	18.75%
LDR_DATA_ZBLOCK	0x001099E8	205 bytes	0.16%
SOFTWARE_RESET_ZBLOCK	0x00109AB5	1 bytes	0.00%
BTCM_END_block	0x00109AB8	4 bytes	0.00%

Figure 8.28. BTCM footprint of the CPU0 project (EWARM)

➤ CPU1(CR52_1)
The application running on CPU1 will perform functions other than blinking the LED.

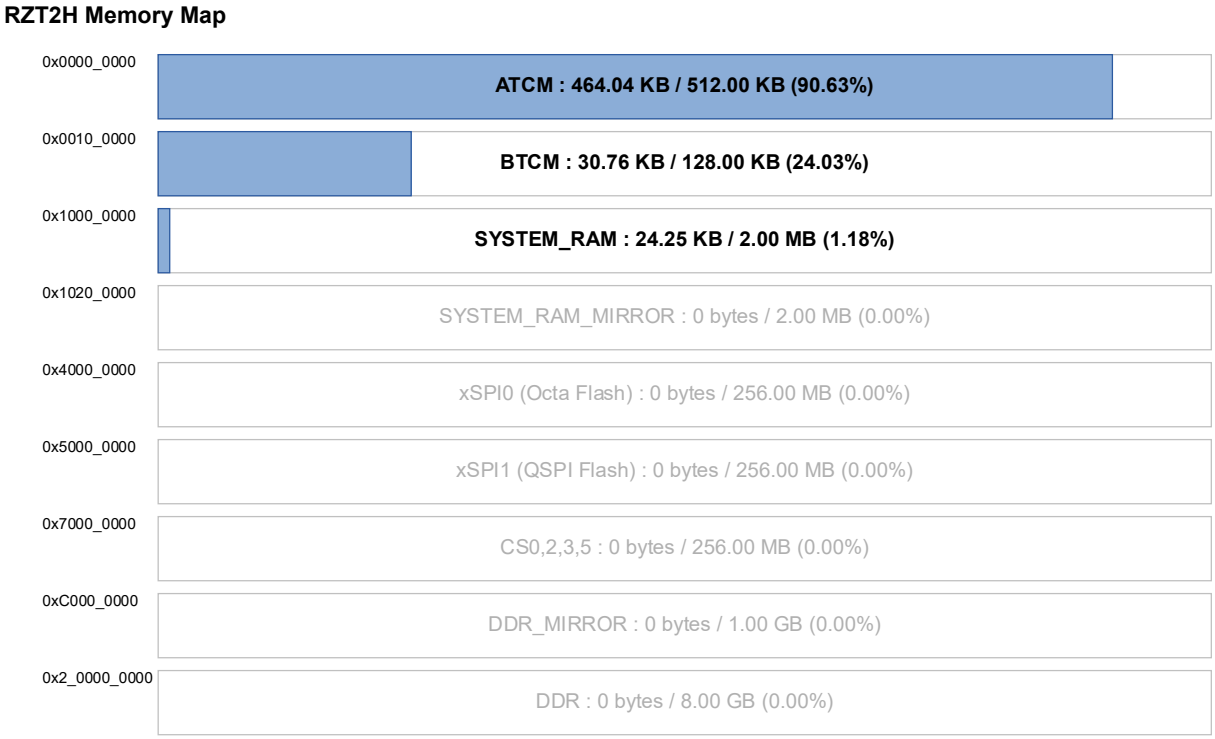


Figure 8.29. Memory footprint of the CPU1 project (EWARM)

ATCM

Total Usage : 464.04 KB

Start Address: 0x00000000 End Address: 0x0007FFFF

Section Name	Start address	Size	Usage [%]
VECTOR_RBLOCK	0x00000000	136 bytes	0.03%
USER_PRG_RBLOCK	0x00000100	98.84 KB	19.30%
.rodata	0x00018C5C	4.75 KB	0.93%
USER_DATA_WBLOCK	0x00019F60	552 bytes	0.11%
.rodata	0x0001A188	1.11 KB	0.22%
.version	0x0001A5F4	64 bytes	0.01%
.rodata	0x0001A634	2.45 KB	0.48%
.version	0x0001AFFC	12 bytes	0.00%
.rodata	0x0001B008	148 bytes	0.03%
.version	0x0001B09C	4 bytes	0.00%
.rodata	0x0001B0A0	11 bytes	0.00%
USER_DATA_ZBLOCK	0x0001B0B0	322.99 KB	63.08%
HEAP_BLOCK	0x0006BCA8	32.00 KB	6.25%
THREAD_STACK	0x00073CA8	1.00 KB	0.20%
ATCM_END_block	0x000740A8	4 bytes	0.00%

Figure 8.30. ATCM footprint of the CPU1 project (EWARM)

BTCM

Total Usage : 30.40 KB

Start Address: 0x00100000 End Address: 0x0011FFFF

Section Name	Start address	Size	Usage [%]
.loader_text	0x00102000	3.73 KB	2.92%
.loader_data	0x00102EF0	208 bytes	0.16%
.loader_bss	0x00102FC0	2.46 KB	1.92%
.loader_stack	0x001039A0	24.00 KB	18.75%

Figure 8.31. BTCM footprint of the CPU1 project (EWARM)

SYSTEM_RAM

Total Usage : 24.25 KB

Start Address: 0x10000000 End Address: 0x101FFFFF

Section Name	Start address	Size	Usage [%]
.noncache	0x10180000	24.25 KB	1.18%

Figure 8.32. 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.

RZT2H Memory Map

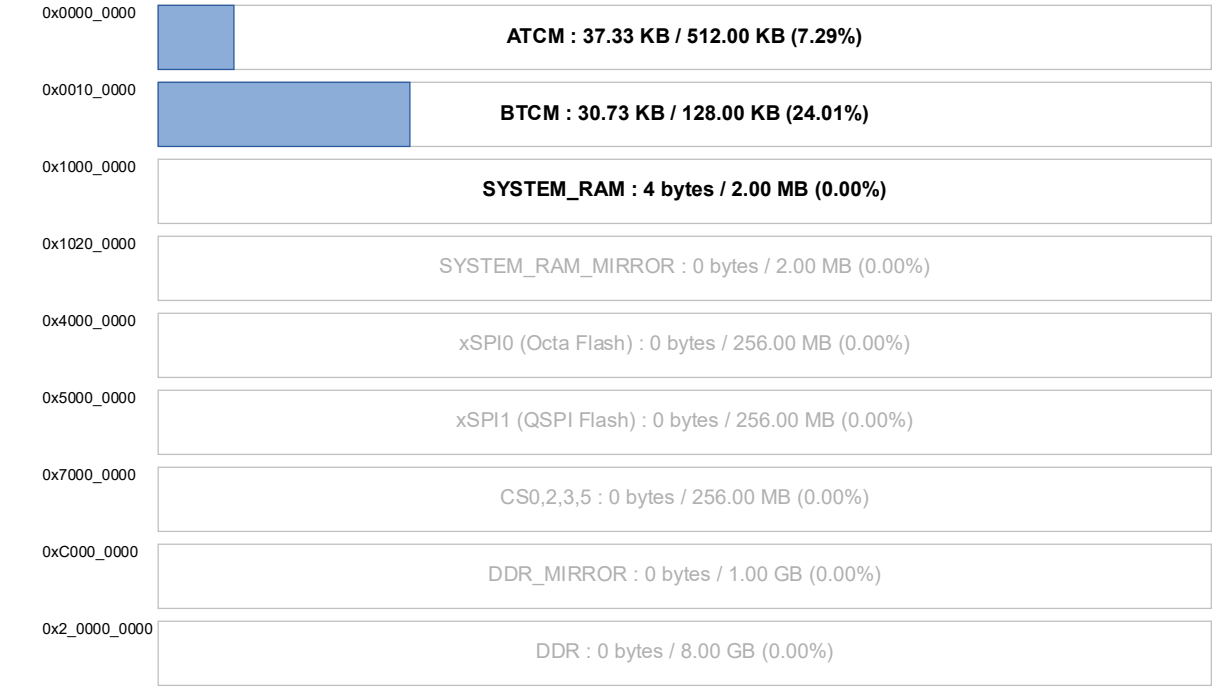


Figure 8.33. Memory footprint of the CPU0 project (EWARM)

ATCM

Total Usage : 37.33 KB

Start Address: 0x00000000 End Address: 0x0007FFFF

Section Name	Start address	Size	Usage [%]
VECTOR_RBLOCK	0x00000000	136 bytes	0.03%
USER_PRG_RBLOCK	0x00000100	626 bytes	0.12%
.rodata	0x00000374	4.43 KB	0.86%
.version	0x00001528	64 bytes	0.01%
.rodata	0x00001568	52 bytes	0.01%
.version	0x0000159C	12 bytes	0.00%
.rodata	0x000015A8	16 bytes	0.00%
.version	0x000015B8	4 bytes	0.00%
HEAP_BLOCK	0x000015C0	32.00 KB	6.25%
USER_DATA_ZBLOCK	0x000095C0	12 bytes	0.00%
ATCM_END_block	0x000095CC	4 bytes	0.00%

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

BTCM

Total Usage : 30.73 KB

Start Address: 0x00100000 End Address: 0x0011FFFF

Section Name	Start address	Size	Usage [%]
LDR_PRG_RBLOCK	0x00102000	6.19 KB	4.83%
LDR_DATA_WBLOCK	0x001038C0	348 bytes	0.27%
LOADER_STACK	0x00103A20	24.00 KB	18.75%
LDR_DATA_ZBLOCK	0x00109A20	205 bytes	0.16%
SOFTWARE_RESET_ZBLOCK	0x00109AED	1 bytes	0.00%
BTCM_END_block	0x00109AF0	4 bytes	0.00%

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

➤ CPU1(CA55_2)
The application running on CPU1 will perform functions other than blinking the LED.

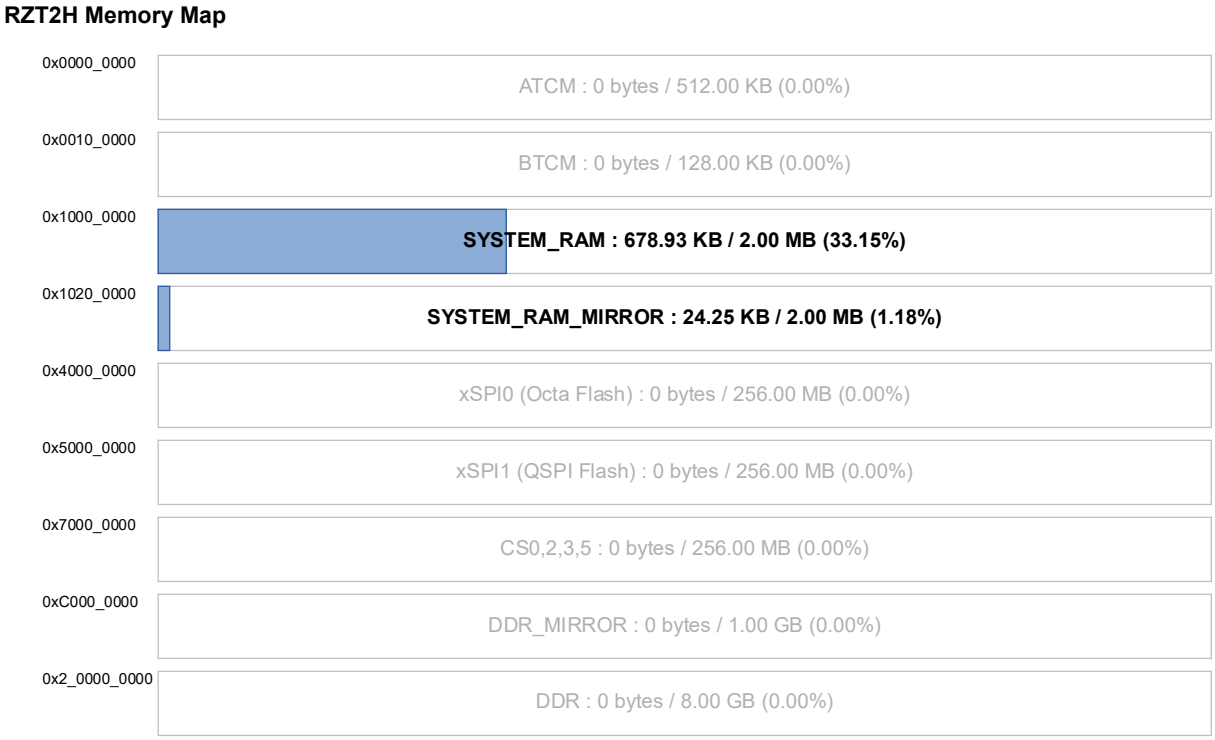
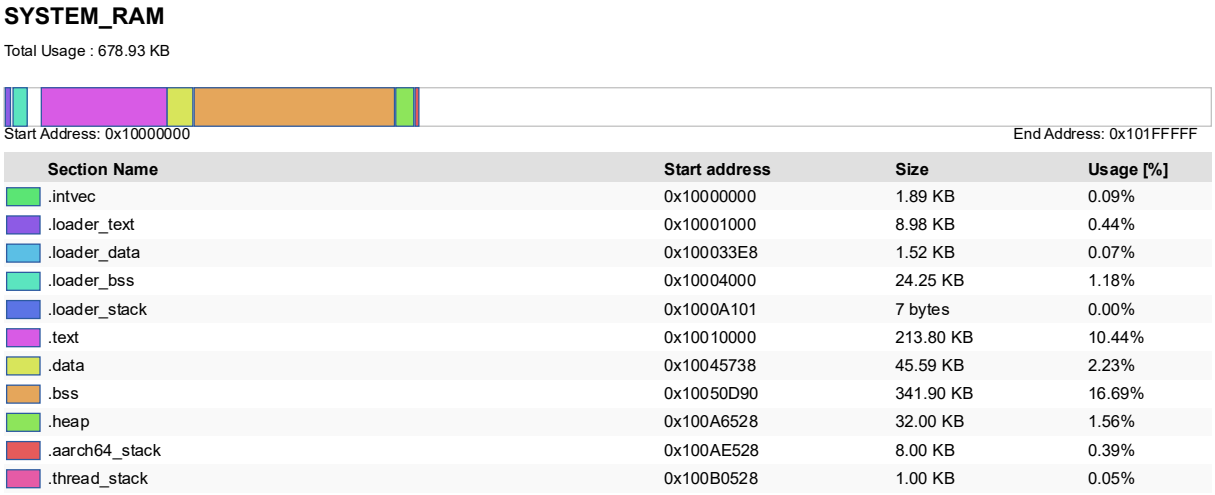


Figure 8.36. Memory footprint of the CPU1 project (EWARM)



Section Name	Start address	Size	Usage [%]
.intvec	0x10000000	1.89 KB	0.09%
.loader_text	0x10001000	8.98 KB	0.44%
.loader_data	0x100033E8	1.52 KB	0.07%
.loader_bss	0x10004000	24.25 KB	1.18%
.loader_stack	0x1000A101	7 bytes	0.00%
.text	0x10010000	213.80 KB	10.44%
.data	0x10045738	45.59 KB	2.23%
.bss	0x10050D90	341.90 KB	16.69%
.heap	0x100A6528	32.00 KB	1.56%
.aarch64_stack	0x100AE528	8.00 KB	0.39%
.thread_stack	0x100B0528	1.00 KB	0.05%

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

Figure 8.38. 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.

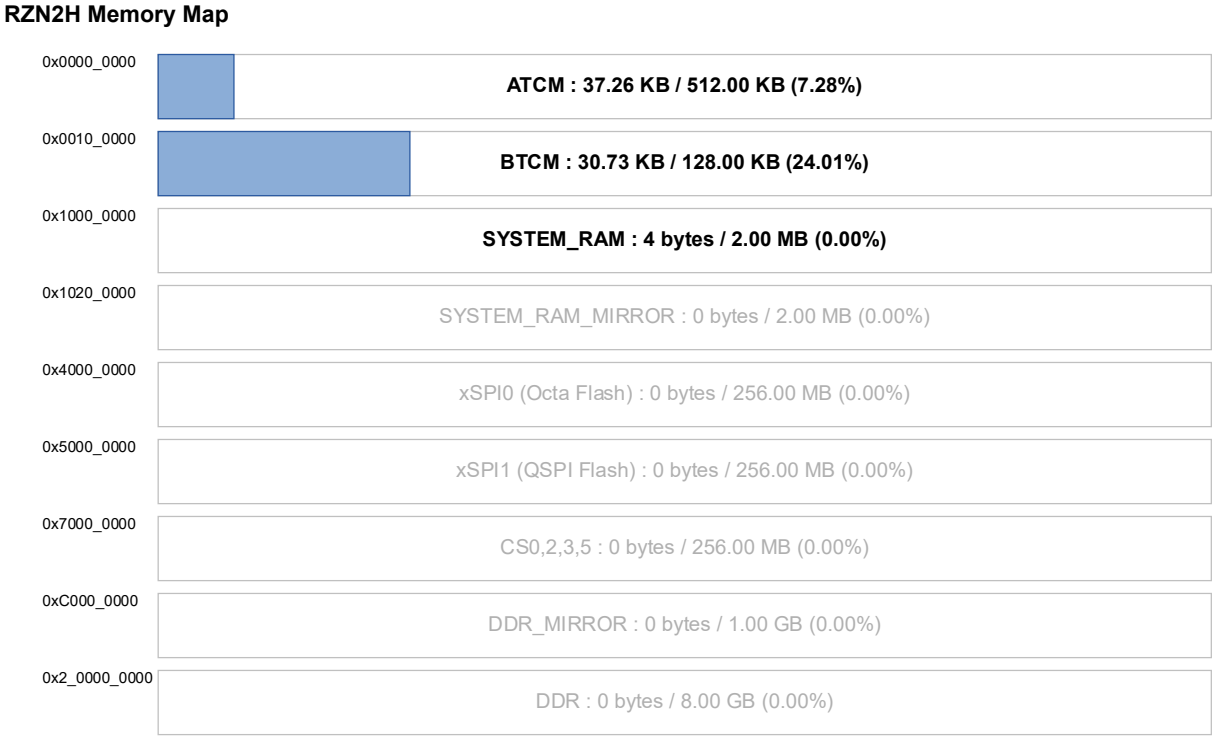


Figure 8.39. Memory footprint of the CPU0 project (EWARM)

ATCM

Total Usage : 37.26 KB

Start Address: 0x00000000 End Address: 0x0007FFFF

Section Name	Start address	Size	Usage [%]
VECTOR_RBLOCK	0x00000000	136 bytes	0.03%
USER_PRG_RBLOCK	0x00000100	626 bytes	0.12%
.rodata	0x00000374	4.36 KB	0.85%
.version	0x000014E8	56 bytes	0.01%
.rodata	0x00001520	68 bytes	0.01%
.version	0x00001564	12 bytes	0.00%
HEAP_BLOCK	0x00001570	32.00 KB	6.25%
USER_DATA_ZBLOCK	0x00009570	12 bytes	0.00%
ATCM_END_block	0x0000957C	4 bytes	0.00%

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

BTCM

Total Usage : 30.73 KB

Start Address: 0x00100000 End Address: 0x0011FFFF

Section Name	Start address	Size	Usage [%]
LDR_PRG_RBLOCK	0x00102000	6.19 KB	4.83%
LDR_DATA_WBLOCK	0x001038C0	348 bytes	0.27%
LOADER_STACK	0x00103A20	24.00 KB	18.75%
LDR_DATA_ZBLOCK	0x00109A20	205 bytes	0.16%
SOFTWARE_RESET_ZBLOCK	0x00109AED	1 bytes	0.00%
BTCM_END_block	0x00109AF0	4 bytes	0.00%

Figure 8.41. 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

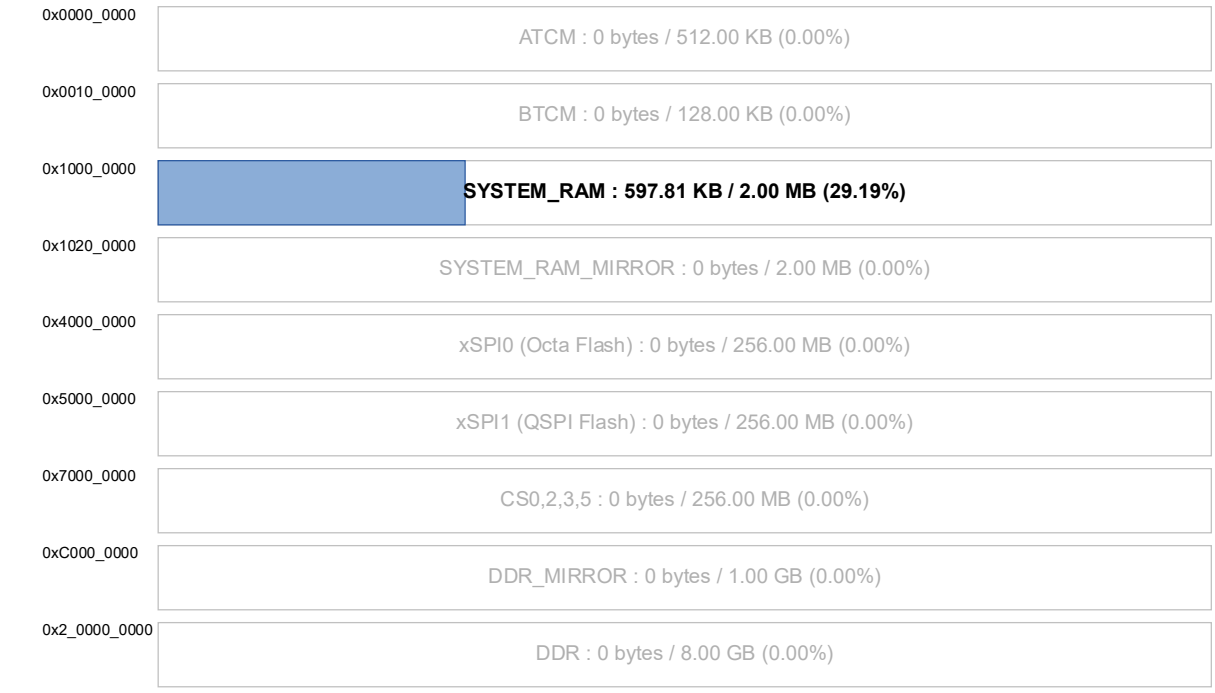


Figure 8.42. Memory footprint of the CPU1 project (EWARM)

SYSTEM_RAM

Total Usage : 597.81 KB

Start Address: 0x10000000 End Address: 0x101FFFFF

Section Name	Start address	Size	Usage [%]
VECTOR_RBLOCK	0x10000000	1.89 KB	0.09%
LDR_PRG_RBLOCK	0x10001000	10.71 KB	0.52%
LDR_DATA_WBLOCK	0x10003AE0	1.52 KB	0.07%
LDR_DATA_ZBLOCK	0x10005000	27.08 KB	1.32%
SOFTWARE_RESET_ZBLOCK	0x1000BC52	1 bytes	0.00%
USER_PRG_RBLOCK	0x10010000	140.62 KB	6.87%
.rodata	0x10033278	6.47 KB	0.32%
USER_DATA_WBLOCK	0x10034C58	944 bytes	0.05%
.rodata	0x10035008	1.38 KB	0.07%
.version	0x10035590	56 bytes	0.00%
.rodata	0x100355C8	2.08 KB	0.10%
.version	0x10035E18	8 bytes	0.00%
.rodata	0x10035E20	16 bytes	0.00%
.version	0x10035E30	4 bytes	0.00%
.rodata	0x10035E34	7 bytes	0.00%
USER_DATA_ZBLOCK	0x10035E40	339.79 KB	16.59%
HEAP_BLOCK	0x1008AD70	32.00 KB	1.56%
AARCH64_STACK	0x10092D70	8.00 KB	0.39%
THREAD_STACK	0x10094D70	1.00 KB	0.05%
SYSTEM_RAM_END_block	0x10095170	8 bytes	0.00%
NONCACHE_ZBLOCK	0x10180020	24.25 KB	1.18%
NONCACHE_END_block	0x10186120	8 bytes	0.00%

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

■ For CR52 and CA55 projects

➤ CPU0(CR52_0)

RZN2H Memory Map

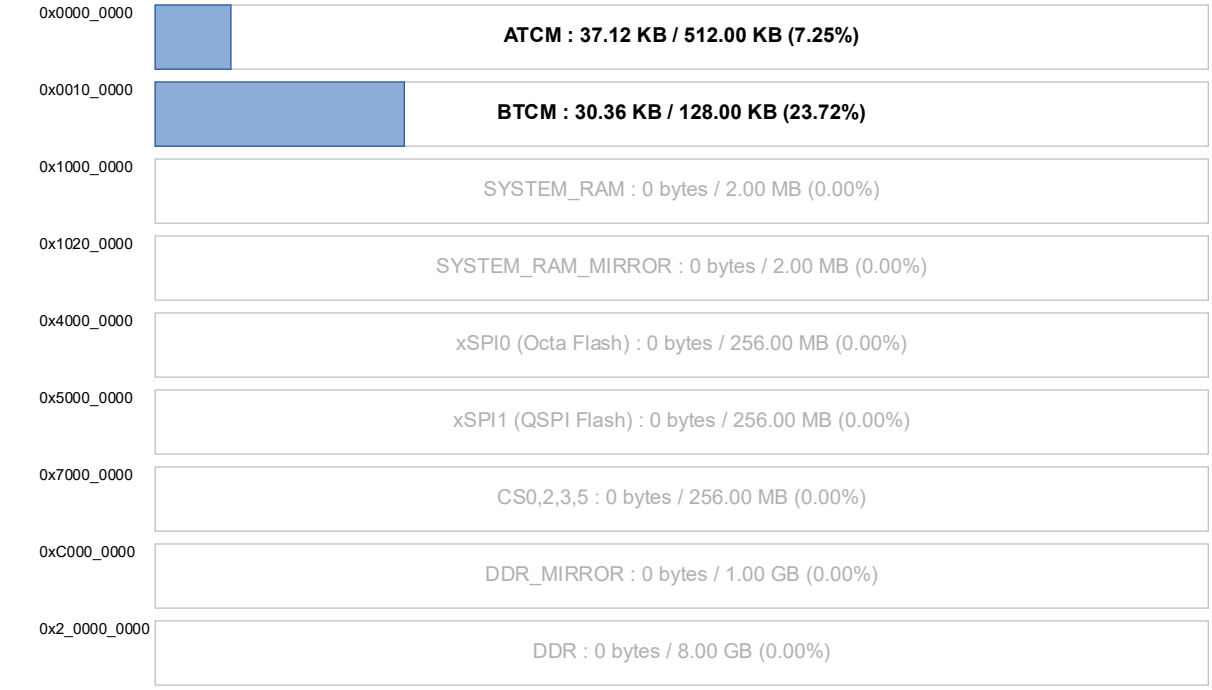


Figure 8.44. Memory footprint of the CPU0 project (EWARM)

ATCM

Total Usage : 37.12 KB

Start Address: 0x00000000 End Address: 0x0007FFFF

Section Name	Start address	Size	Usage [%]
.intvec	0x00000000	128 bytes	0.02%
.text	0x00000100	560 bytes	0.11%
.ARM.exidx	0x00000330	8 bytes	0.00%
.data	0x00000338	4.43 KB	0.87%
.bss	0x000014F0	16 bytes	0.00%
.heap	0x00001500	32.00 KB	6.25%

Figure 8.45. ATCM footprint of the CPU0 project (EWARM)

BTCM

Total Usage : 30.36 KB

Start Address: 0x00100000 End Address: 0x0011FFFF

Section Name	Start address	Size	Usage [%]
.loader_text	0x00102000	5.81 KB	4.54%
.loader_data	0x00103740	352 bytes	0.27%
.loader_bss	0x001038A0	209 bytes	0.16%
.loader_stack	0x00103978	24.00 KB	18.75%

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

➤ CPU1(CA55_2)

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

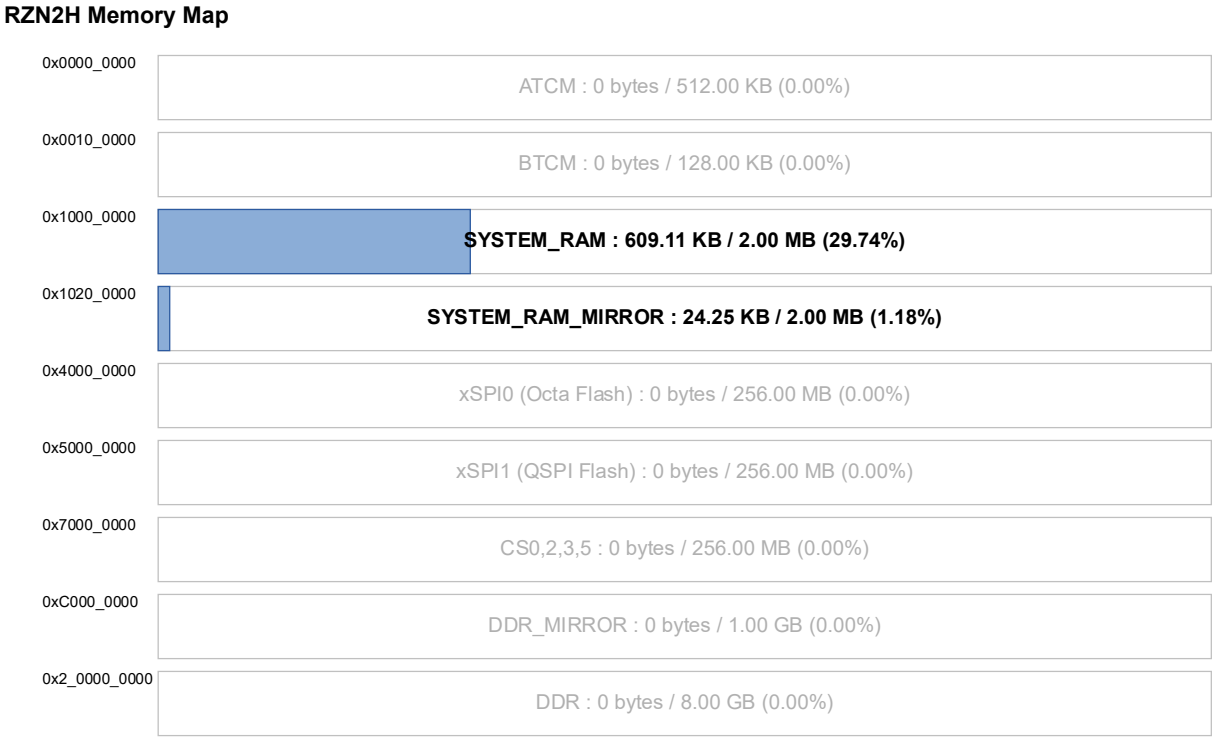
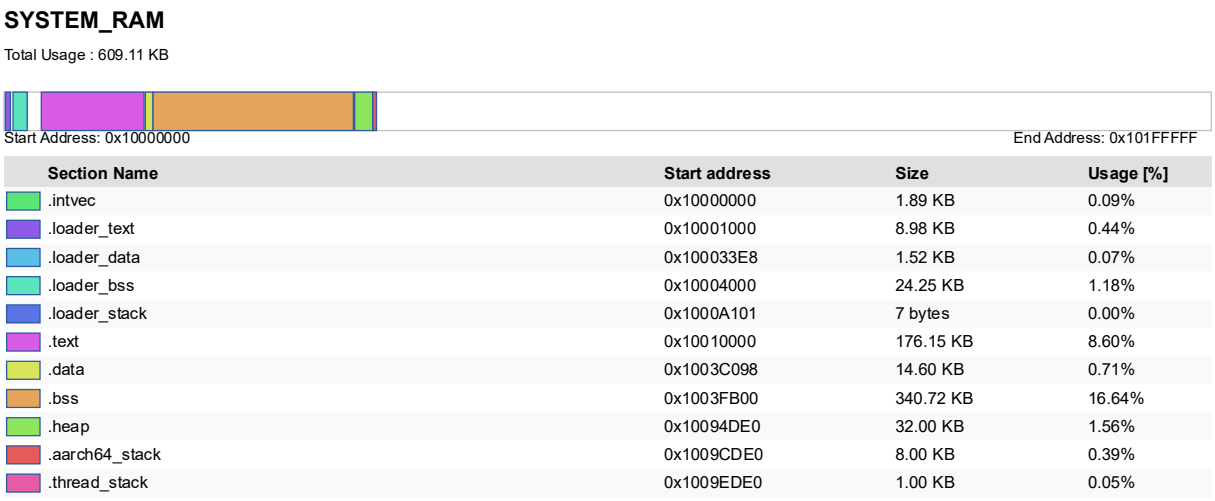


Figure 8.47. Memory footprint of the CPU1 project (EWARM)



Section Name	Start address	Size	Usage [%]
.intvec	0x10000000	1.89 KB	0.09%
.loader_text	0x10001000	8.98 KB	0.44%
.loader_data	0x100033E8	1.52 KB	0.07%
.loader_bss	0x10004000	24.25 KB	1.18%
.loader_stack	0x1000A101	7 bytes	0.00%
.text	0x10010000	176.15 KB	8.60%
.data	0x1003C098	14.60 KB	0.71%
.bss	0x1003FB00	340.72 KB	16.64%
.heap	0x10094DE0	32.00 KB	1.56%
.aarch64_stack	0x1009CDE0	8.00 KB	0.39%
.thread_stack	0x1009EDE0	1.00 KB	0.05%

Figure 8.48. SYSTEM_RAM footprint of the CPU1 project (EWARM)

Section Name	Start address	Size	Usage [%]
.noncache	0x10380000	24.25 KB	1.18%

Figure 8.49. SYSTEM_RAM (MIRROR) footprint of the CPU1 project (EWARM)

8.4 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: 75%
   - Echo server waiting for clients
     CPU usage: 23%
2. RZ/T2H CA55
   - iperf communication in progress (10 seconds)
     CPU usage: 35%
   - Echo server waiting for clients
     CPU 使用率 <1%
3. RZ/N2H CR52
   - iperf communication in progress (10 seconds)
     CPU usage: 74%
   - Echo server waiting for clients
     CPU usage: 23%
4. RZ/N2H CA55
   - iiperf communication in progress (10 seconds)
     CPU usage: 35%
   - Echo server waiting for clients
     CPU usage: <1%

Revision History

Rev.	Date	Description	
		Page	Summary
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.

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/.