RZ/N2L

Industrial Ethernet Protocol Auto-Detection

Introduction

Today the variety of Industrial Ethernet protocols typically requires production and delivery of the same hardware with different software flavors, since a monolithic approach with one single application is difficult to handle.

This application note presents an easy boot loader solution for RZ/N2L that allows the detection of the used Industrial Ethernet protocol and loads the appropriate application during run-time from flash to RAM. The different requirements of RAM consumption and execution speed are supported by flexible usage of internal SystemRAM, external SDRAM or external HyperRAM. The existing Ethernet protocol applications just require minor changes to startup code and linker scripts.

The major features of the industrial protocol auto-detection sample program are listed below.

- During startup the incoming Ethernet packets to the RZ/N2L-RSK board ports are read to find out the used Ethernet Protocol. EtherType and Source MAC address are used for this.
- Based on the detected protocol, the corresponding application is copied from Flash memory to RAM and executed from there.

Target Device

RZ/N2L

When applying the sample program covered in this application note to another microcomputer, modify the program according to the specifications for the target microcomputer and conduct an extensive evaluation of the modified program.

Referenced Documents

[1] RZ/T2, RZ/N2 Getting Started with Flexible Software Package, Rev 1.06,
r01an6434ej0106-rzt2-rzn2-fsp-getting-started.pdf

[2] Renesas Starter Kit+ for RZ/N2LUser's Manual,
r20ut4984eg0103-rskplus-rzn2l-v1-um.pdf

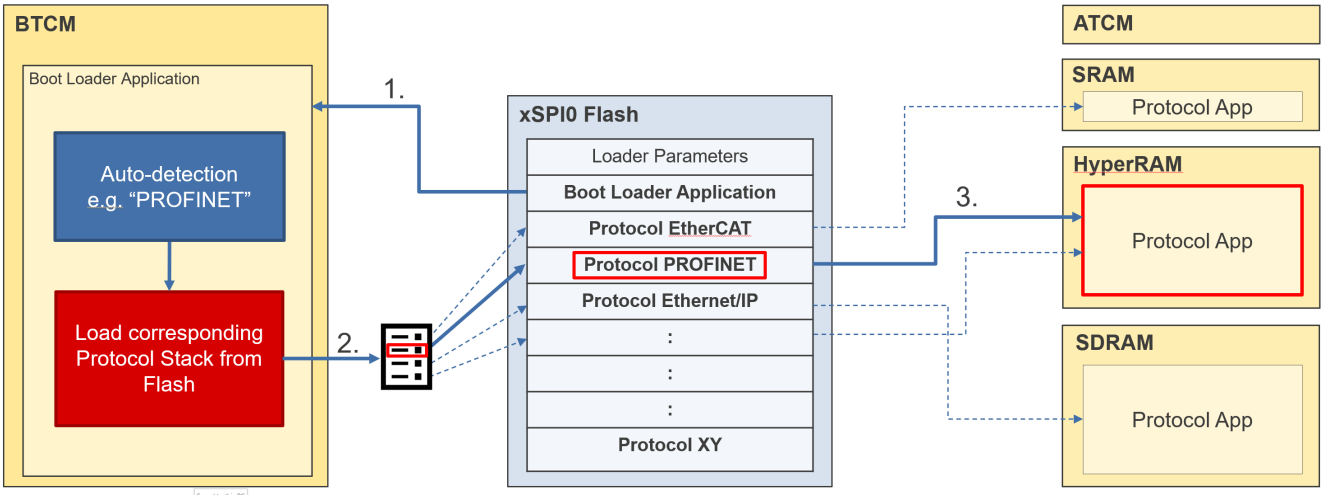
[3] EtherCAT Sample Application, Quick Start Guide: EtherCAT Slave Software,
r01an6178ej0120-rzn2l-ecat.pdf

Contents

1. Overview

1.1 Loading of Protocol Application by Boot Loader

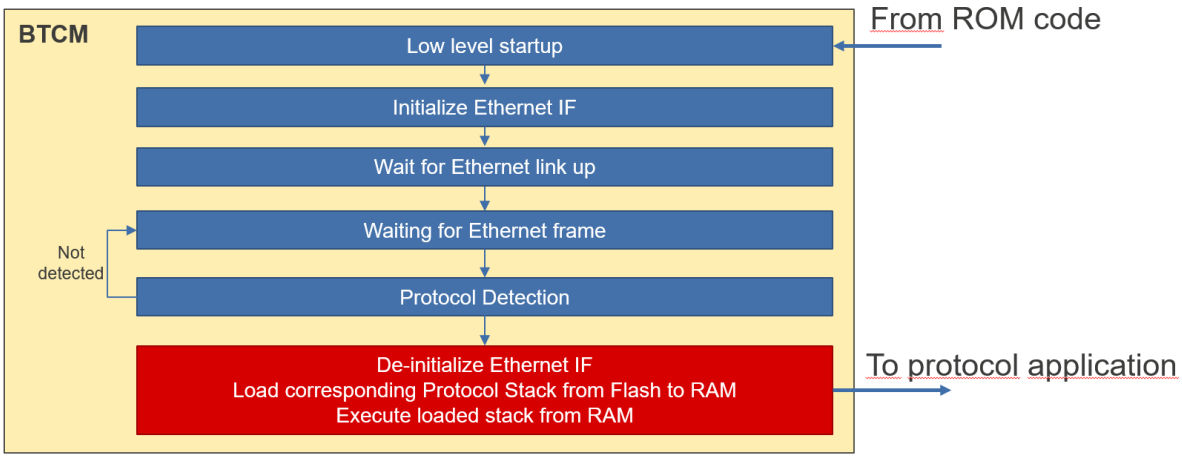
The startup code in ROM copies the boot loader application from the flash to the internal RAM according to the loader parameters also stored in the flash. The boot loader is started. The Industrial Ethernet protocol detection is performed as described below in detail. Depending on the result the boot loader copies the corresponding code and data from the flash to the RAM and finally starts the application from that location.



The user must define the type of RAM and specify the locations for each protocol application. A table in the boot loader application holds the flash source address, the RAM destination address, and the size of the applications. The needed values are automatically filled in during the linker process, based on the linker scripts of the applications.

1.2 Ethernet Protocol Auto-Detection

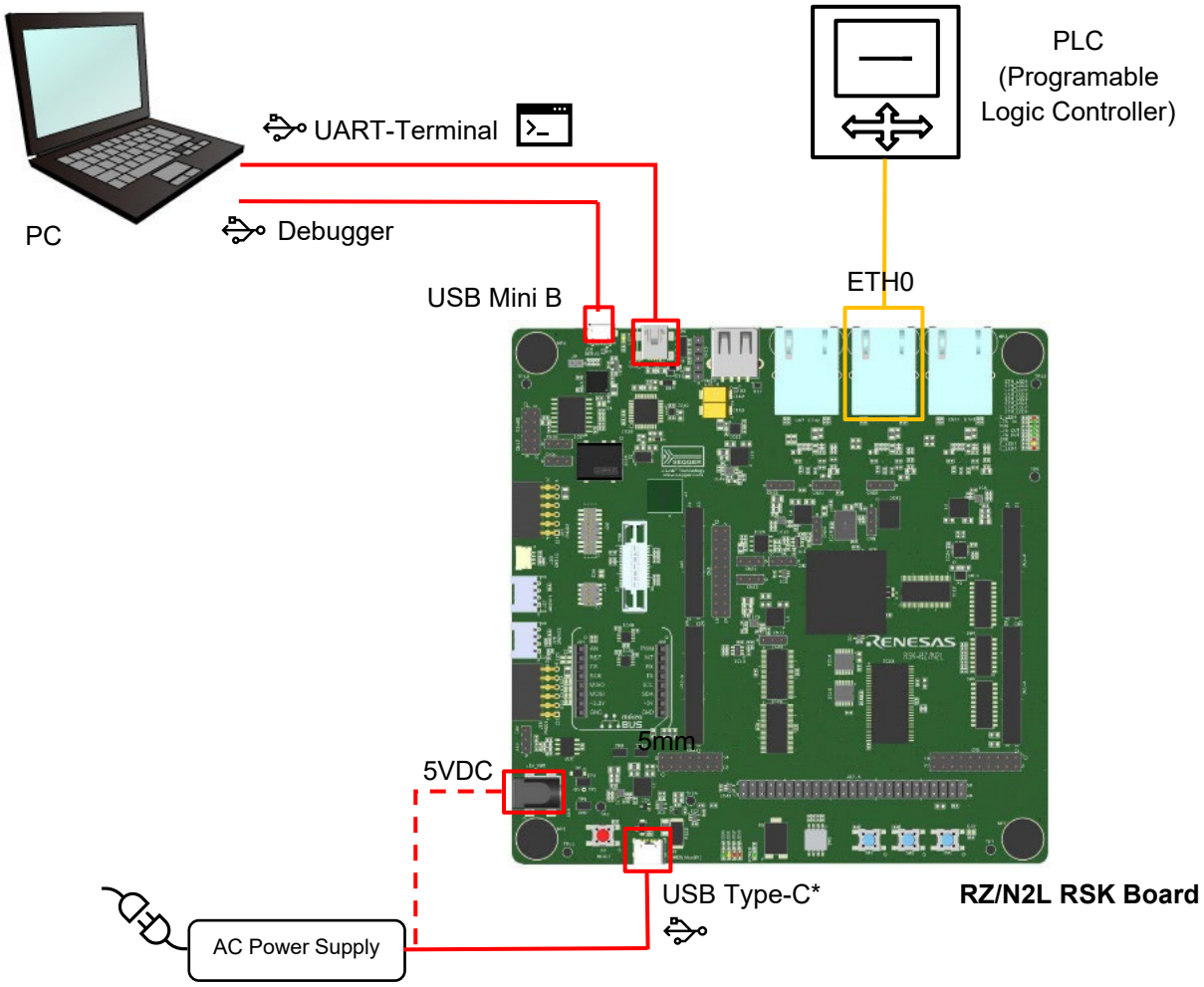
After the low-level startup, the boot loader initializes the hardware to bring up the Ethernet interface. The software evaluates the EtherType field of incoming Ethernet packets to detect the used protocol in the network. If a certain number of frames of a protocol were received, the boot loader copies the corresponding application from the flash to the RAM and starts the execution. Previous hardware initializations are reverted before.



2. Specification and Operating Environment

2.1 System Overview

The connection of the RZ/N2L RSK Board to PC, PCL and Power is shown in the figure below.



2.2 Operating Environment

The sample code covered by this application note is for the hardware and software environment below.

Table 1.1 Requirements

Item	Vendor	Description
Board	Renesas Electronics	Renesas Starter Kit+ for RZ/N2L (RTK9RZN2L0S00000BE)
IDE	Renesas Electronics	• e² studio 2023-04 • RZ/N2L Flexible Software Package (FSP) v1.2.0 • ARM Toolchain 9.3.1.20200408
Emulator	Segger	Hardware: J-Link (onboard) Software: J-Link V7.80b
Software PLC	Beckhoff Automation CODESYS Development GmbH	TwinCAT3 CODESYS V3.5

2.3 Board Setup

This section describes the required DIP-switch and jumper settings for this Industrial Ethernet Protocol Auto-Detection application use case. The boot mode is set to xSPI0 (x1 boot serial flash).
Please refer to the RZ/N2L RSK board user's manual and schematic for the board details.

DIP switches

SW4

1	2	3	4	5	6	7	8
ON	ON	ON	ON	OFF	OFF	ON	OFF

SW8

1	2	3	4	5	6	7	8	9	10
OFF	ON	OFF	ON	OFF	OFF	OFF	OFF	OFF	OFF

SW11

1	2	3	4	5	6	7	8	9	10
ON	OFF	OFF	OFF	OFF	OFF	OFF	OFF	OFF	OFF

Jumpers

Table 1.2 Renesas Starter Kit+ for RZ/N2L jumper setting

Jumper number	Setting
CN8	Short 2-3
CN17	Short 1-3
CN20	Short 1-2
CN21	Short 1-2
CN22	Short 1-2
CN24	Short 2-3
CN25	Short 1-2
CN27	Short 1-2
CN29	Short 1-2
CN31	Short 1-2
CN32	Short 1-2
J9	Open

2.4 Serial Terminal

The settings for the serial terminal for debug output are as follows:

- Speed 115200 Baud
- Data 8 bit
- No parity
- 1 stop bit
- No flow control

2.5 Software Elements

The software consists of multiple software elements based on the RZ/N2L FSP as follows.
For detailed description of the individual packages please refer to the package documentation.

2.5.1. Boot Loader Software

RZ/N2L Group Example of separating loader program and application program projects V1.1.0

As addition the bootloader initializes the ethernet interface and checks the protocol of the incoming ethernet frames. Depending on the protocol the corresponding application binary is copied from the SPI flash to the specified RAM location and started from there.

2.5.2. Industrial Protocol Auto-Detection Program

The auto-detection program is a simple program based on the RZ/N2L FSP. The program is capable to detect some of the market leading industrial Ethernet protocols (PROFINET, EtherCAT, EtherNet/IP) by reading the EtherType of incoming packets.

2.5.3. EtherCAT

Renesas RZ/N2L Group EtherCAT Sample program Package V1.2.0

Modifications are applied to the low level code in startup.c and the linker script.

2.5.4. PROFINET

GOAL Profinet Demo from port industrial automation GmbH (https://www.port.de).

RZ/N2L: 2.24.0 (Date of update: 29.09.2023)

https://portgmbh.atlassian.net/wiki/spaces/GOALR/pages/639762433/GOAL+-+Renesas+RZ+N2L-RSK

GOAL Profinet Demo

This demo is a full feature protocol stack, but it is 1h time limited.

Selected demo code: “01_simple_io”

Modifications are applied to the low-level code in startup.c and system.c, and the linker script fsp_ram_execution.ld. The original files are available with the extension `.fsp120` or `.goal`.

2.5.5. EtherNet/IP

GOAL Ethernet/IP Demo from port industrial automation GmbH (https://www.port.de).

RZ/N2L: 2.24.0 (Date of update: 29.09.2023)

https://portgmbh.atlassian.net/wiki/spaces/GOALR/pages/639762433/GOAL+-+Renesas+RZ+N2L-RSK

GOAL Ethernet/IP Demo

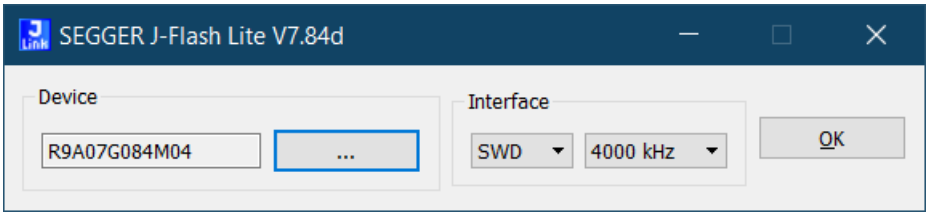
This demo is a full feature protocol stack, but it is 1h time limited.

Selected demo code: “10_led_demo”

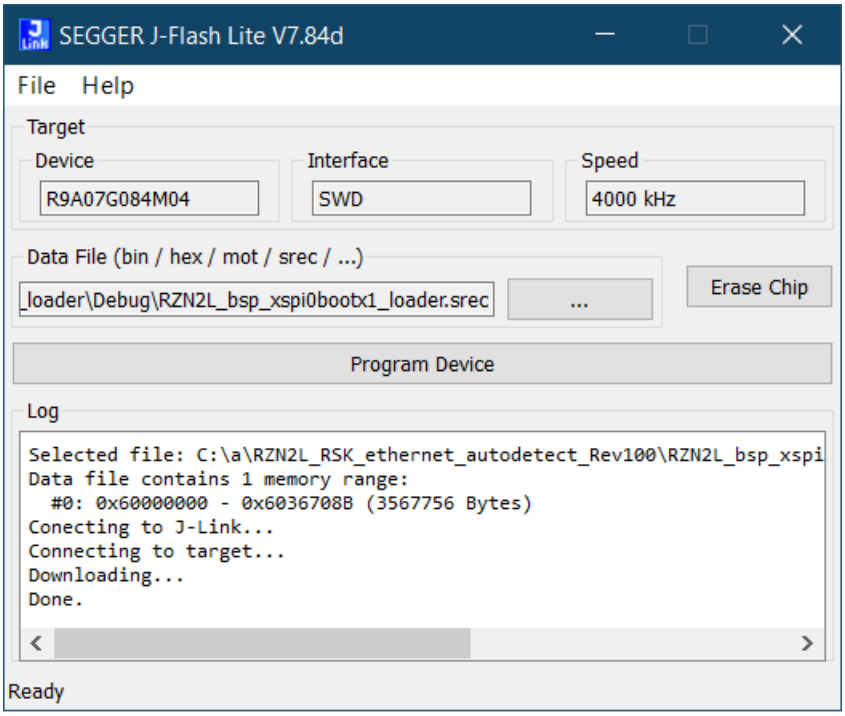
Modifications are applied to the low-level code in startup.c and system.c, and the linker script fsp_ram_execution.ld. The original files are available with the extension `.fsp120` or `.goal`.

3. Running the Sample Application as Binary

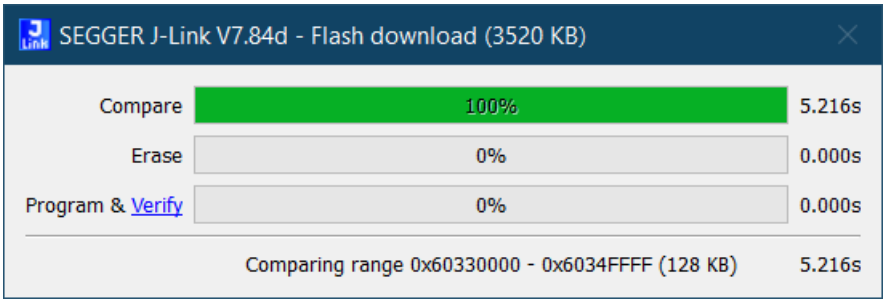
Open the SEGGER J-Flash Lite tool, and select the RZ/N2L device “R9A07G084M04”:



Select the s-record binary of the whole application RZN2L_bsp_xspi0bootx1_loader.srec in folder RZN2L_RSK_ethernet_autodetect_Rev100\RZN2L_bsp_xspi0bootx1_loader\Debug.
Then press “Program Device”:



During comparing, erasing and flashing the following window will appear:

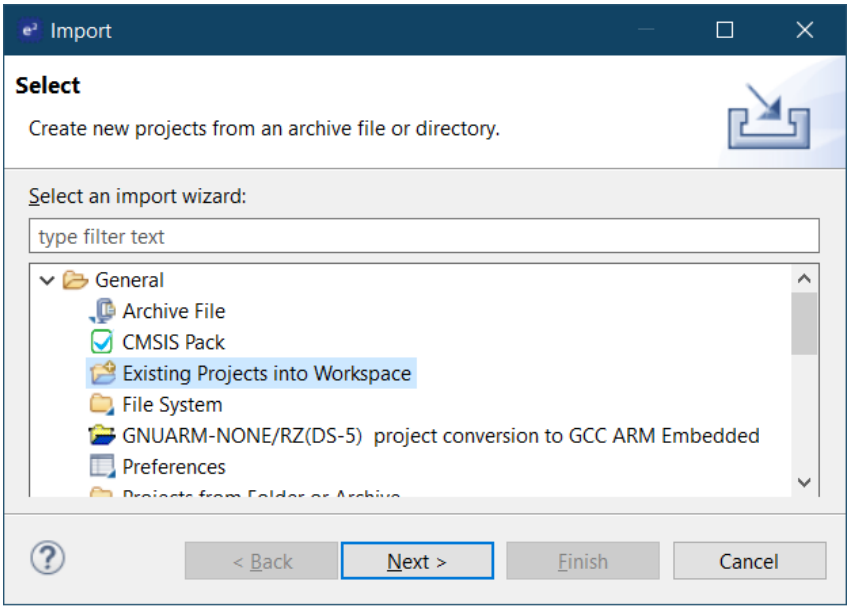


Continue with chapter 5.

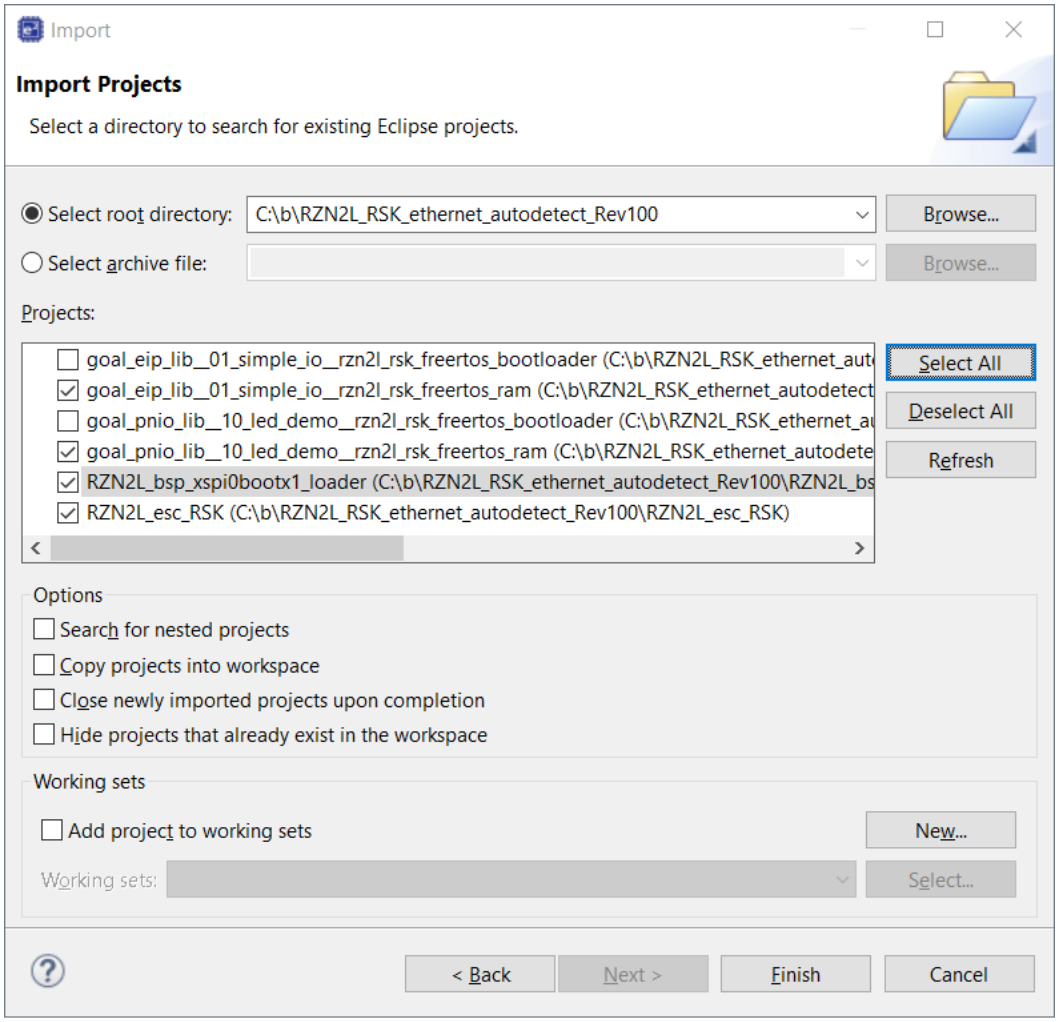
4. Running the Sample Application from Source

4.1 Software Installation

1. Create a new folder. The path to this folder should be short to avoid exceeding the limit of path length of the Windows OS.
2. Unzip the package RZN2L_RSK_ethernet_autodetect_Rev100.zip to the folder.
3. Start e²studio.
4. Select the created folder (1) as new workspace.
5. Select [File] – [Import] – [Existing Projects into Workspace]. Press [Next].



6. Select [Select root directory], click the [Browse] button, and select the folder from (1).
7. Select the following four projects:
   - RZN2L_bsp_xspi0bootx1_loader
   - RZN2L_esc_RSK
   - goal_eip_lib__01_simple_io__rzn2l_freertos_ram
   - goal_pnio_lib__10_led_demo__rzn2l_rsk_freertos_ram



8. Click the "Finish" button.
9. The FSP files are already created and available. Calling the FSP Smart Configurator is not required.

Note: If the FSP files shall be generated with FSP Smart Configurator again, make copies of the linker script and the file `startup.c` before. Modifications were applied to those files to support the loader approach. The generated files might be overwritten during generation.

4.2 EtherCAT Source Files

Due to license conditions, it is not possible to provide the EtherCAT source code from Beckhoff in this package. Please refer to [3] chapter 3.3 “Generating the Slave Stack Code” how to generate the required source code. The generated files must be copied to the folder `.\RZN2L_esc_RSK\src\ethercat\beckhoff\Src`.

4.3 Building the Application

The linker adds the binaries of protocol applications to the boot loader application. Therefore, the protocol applications must be built first.

1. First build the three ethernet protocol applications.
   Select each application, select [Project] – [Build All] from the e²studio menu.

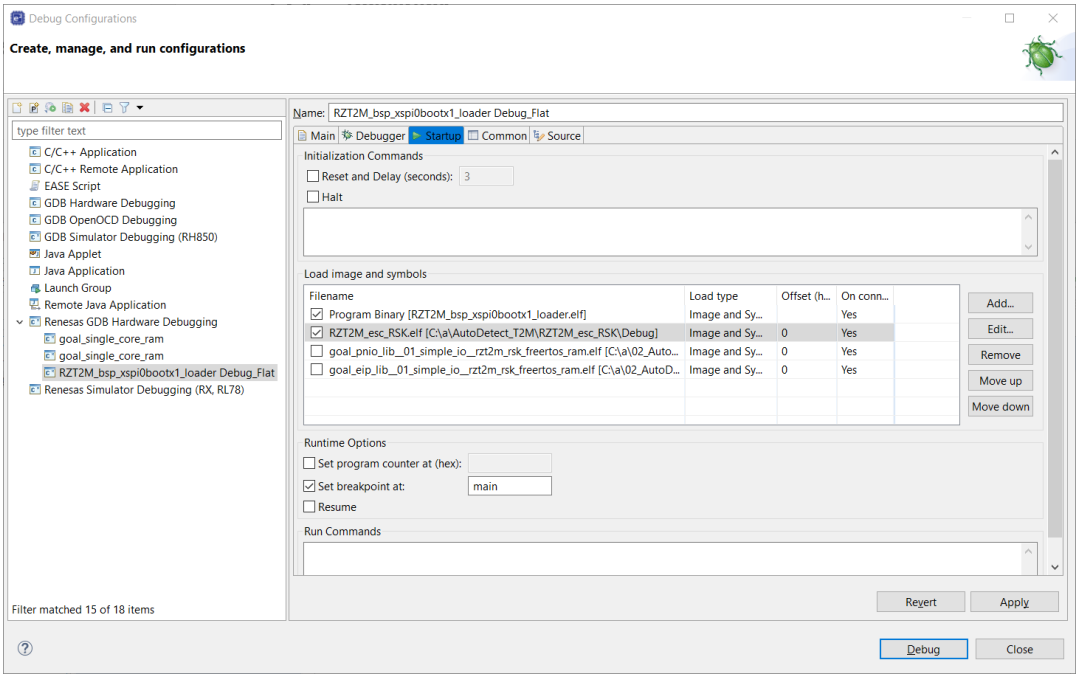
2. After successful building of those applications, build the loader application.

4.4 Debugging the Application

For source code debugging the debugger must map the source code to the binary that is currently executed. Normally, this mapping is fixed during runtime. However, here the loaded ethernet protocol application is selected during runtime. The source code differs depending on the detected ethernet protocol. The debugger must be instructed which application shall be debugged.

This can be selected in the e²studio dialog [Debug-Configurations] – [Renesas GDB Hardware Debugging] - [RZN2L_bsp_xspi0bootx1_loader_Debug_Flat] – [Startup].

Enable only one of the applications for [Load image and symbols] at a time that shall be debugged:



In the Project Explorer select the project [RZN2L_bsp_xspi0bootx1_loader] as active, e.g. by double-click. (Starting one of the applications directly will fail due to missing initializations.)

Start debugging with [Run] – [Debug].

The application binary is downloaded to the RZ/N2L RSK board and started. Breakpoint will be hit at `system_init` and `main` function. Press [Run] – [Resume] in this case to continue.

5. Testing

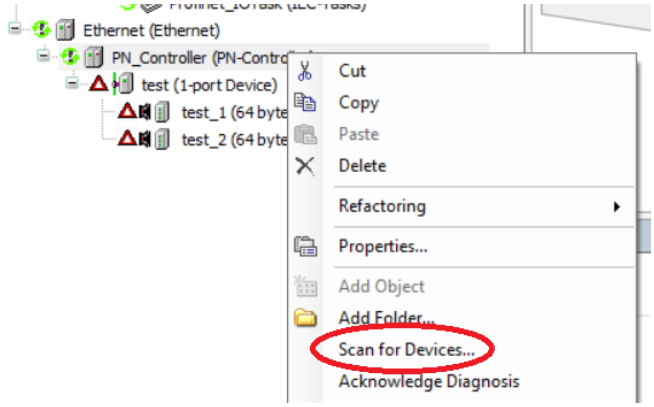
Connect the board to your PC as shown in Chapter 2.1. Start a terminal software like TeraTerm and set the interface parameter as given in chapter 2.4.

Install automation and control software like TwinCAT (supporting EtherCAT) or CodeSys (supporting EtherCAT, PROFINET, and Ethernet/IP) to check the connection and to control the board. Please refer to the corresponding sample applications and related documentation how to set up TwinCAT and Codesys to operate with the original sample applications.

Enable TwinCAT/CodeSys for operational mode as described in the mentioned documentation. In TwinCAT this means doing “Restart TwinCAT System in Config Mode”. In CodeSYS this means starting the Gateway, the PLC, and performing Login and Start.

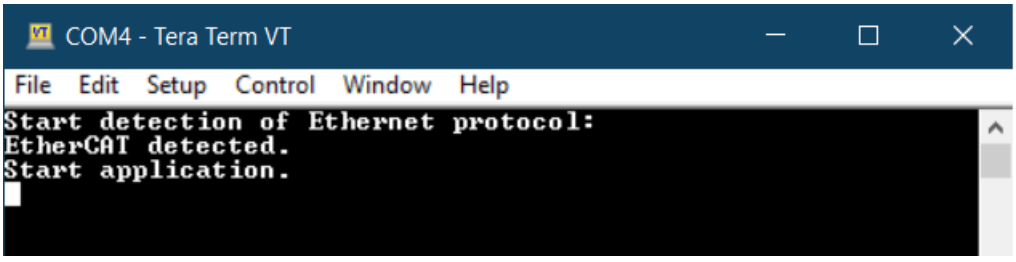
Start the Auto-Detection Software via debugger or from flash. The Serial Terminal should show the output “Start detection of Ethernet protocol”.

To trigger the generation of packets of the intended protocol, it might be necessary to stop and start the tools. In CodeSys calling “Scan for Devices” is helpful:



If enough frames of one protocol have been received by the Auto-Detection software, the found protocol is given, e.g. “EtherCAT detected”. The corresponding application is copied from Flash to RAM and started. Accordingly, “Start application” is printed to the terminal.

For example, a successful detection of EtherCAT should look as follows:



Attention:

Ethernet/IP uses the EtherType value of normal IP packets. This means the software cannot distinguish directly between Ethernet/IP and other traffic. So, as second criterion the MAC Vendor ID (also called “organizationally unique identifier” (OID)) can be used if a Rockwell Device is used.

In a test environment with a PC the MAC has an arbitrary OID, therefore the check of the MAC is disabled in the Auto-Detection software. The check for Rockwell or a dedicated OID can be enabled and configured in `ead_app.c` of the loader application.

A typical PC continuously sends various frames that are interpreted as Ethernet/IP if the MAC address is not considered. There is a risk that Ethernet/IP is detected incorrectly. To overcome this issue the requirement for successful detection of Ethernet/IP are 25 frames without any EtherCAT or Profilnet frames in between, but in rare cases, the Auto-Detection might fail by detecting the wrong protocol.

5.1 Using Ethernet Tester

To test the auto-detection of ethernet protocols with an ethernet tester, the frames to be sent have to be prepared as follows:

Protocol	EtherType	Source MAC Address
EtherCAT	0x88A4	(don’t care)
PROFINET	0x8892	(don’t care)
Ethernet/IP	0x0800	BC:F6:85:xx:xx:xx (Rockwell OID, if check is enabled)
Powerlink	0x88AB	(don’t care)

5.2 Testing without Ethernet Traffic

To test the loading of the ethernet protocol application without ethernet traffic, the detection can be commented out and the protocol can be set to a fixed value in function `hal_entry()` in `hal_entry.c` in the loader application.

E.g., to load PROFINET without detection the code should look like:

```
/* enable ethernet and detect protocol */
//printf("Start detection of Ethernet protocol:\n");
//detected_protocol = ead_main();

/* select a dedicated application for testing */
//detected_protocol = e_PROTOCOL_ETHERCAT;
detected_protocol = e_PROTOCOL_PROFINET;
//detected_protocol = e_PROTOCOL_ENIP;
printf("Ethernet Protocol %d explicitly set.\n", detected_protocol);
```

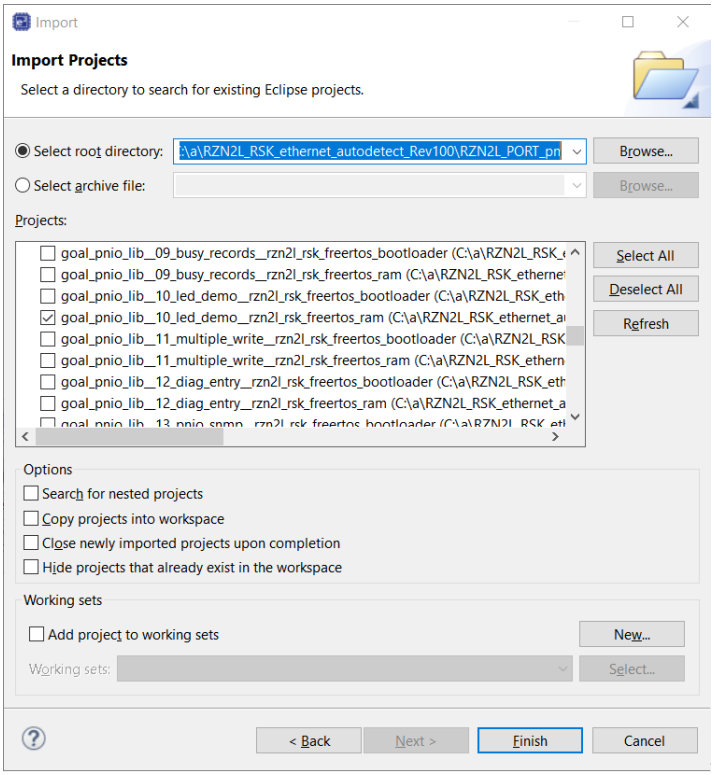
6. Integration

This chapter describes how an application is added to the loader.
As example the PROFINET stack of port GmbH is used.

6.1 Import a Sample Application to the Workspace

Create a new subfolder in the existing workspace of the Auto-Detection application, e.g “RZN2L_PORT_pn” for the PROFINET application. Extract the content of the provided zip-File to this folder.

Click “File” – “Import” and select “Existing Projects into Workspace” in the section “General”. Enter the created folder as root folder and deselect all found projects except the application of one sample, e.g. “10_led_demo”. (The loader application aside is not needed).



6.2 Generate FSP Files

Make a copy of the existing linker script “fsp_ram_execution.ld” in folder
`projects\goal_pnio_lib\10_led_demo\e2studio\renesas\rzn2l_rsk_freertos\ram\script`
This file contains some adaptations and will be overwritten by a default one when generating the files with the Smart Configurator.

Open the Smart Configurator by double clicking the `configuration.xml` in the Project Explorer. Generate the files by clicking the button in right upper corner.

Replace the generated linker script by the copy.

6.3 Setting e²studio Environment

To include the applications to the loader, the type must be pure binary. In e²studio, select [Project] -> [C/C++ Project Settings] -> [Cross ARM GNU Create Flash Image] -> [General]. Select “Output file format” as “Raw Binary”.

6.4 Adapting Low-Level Code

The files `startup.c` and `system.c` in folder <app>\e2studio\renesas\rzn2l_rsk_freertos\ram\rzn\fsp\src\bsp\cmsis\Device\RENESAS\Source must be adapted to fit to the initialization already done by the boot loader. The existing files can be taken as reference.

Build the application project.

6.5 Adding the Application to the Loader Project

Create a `.S` file to include the binary of the application to the loader.

```
.section .APP2_IMAGE_QSPI_FLASH_section, "ax", %progbits
.incbin "../../RZN2L_PORT_pn/projects/goal_pnio_lib/10_led_demo/e2studio/renesas/
rzn2l_rsk_freertos/ram/Single-Core/goal_pnio_lib__10_led_demo__rzn2l_rsk_freertos_ram.bin"
```

6.6 Linker Script of the Loader

Add directives in the linker script `fsp_xspi0_boot_loader.ld` to specify the locations in flash and RAM:

```
.APP2_IMAGE_SYSRAM 0x30000000 : AT (0x30000000)
{
    APP2_IMAGE_SYSRAM_start = .;
    KEEP(*(APP2_IMAGE_SYSRAM))
}


.APP2_IMAGE_QSPI_FLASH_section 0x60200000 : AT (0x60200000)
{
    APP2_IMAGE_QSPI_FLASH_section_start = .;
    KEEP(./src/flash_section_PN.o(.APP2_IMAGE_QSPI_FLASH_section))
    APP2_IMAGE_QSPI_FLASH_section_end = .;
}
APP2_IMAGE_QSPI_FLASH_section_size = SIZEOF(.APP2_IMAGE_QSPI_FLASH_section);
```

6.7 Add the new Application to the Loader Table

Symbols of the linker script are used in the source code to define the source address, the destination address and the size of the application binary. Add in `loader_table.c`:

```
extern uint32_t APP2_IMAGE_QSPI_FLASH_section_start;
extern uint32_t APP2_IMAGE_QSPI_FLASH_section_size;

extern uint32_t APP2_IMAGE_SYSRAM_start;

#define application2_prg_flash_addr (&APP2_IMAGE_QSPI_FLASH_section_start)
#define application2_prg_start_addr (&APP2_IMAGE_SYSRAM_start)
#define application2_prg_size       (&APP2_IMAGE_QSPI_FLASH_section_size)

const loader_table table[TABLE_ENTRY_NUM] BSP_PLACE_IN_SECTION("CPU0_LOADER_TABLE") =
{
  { (uint32_t *)application1_prg_flash_addr, (uint32_t *)application1_prg_start_addr,
          (uint32_t)application1_prg_size, (uint32_t)TABLE_ENABLE },
  { (uint32_t *)application2_prg_flash_addr, (uint32_t *)application2_prg_start_addr,
          (uint32_t)application2_prg_size, (uint32_t)TABLE_ENABLE },
  { (uint32_t *)TABLE_INVALID_VALUE, (uint32_t *)TABLE_INVALID_VALUE,
          (uint32_t)TABLE_INVALID_VALUE, (uint32_t)TABLE_DISABLE },
```

```
  { (uint32_t *)TABLE_INVALID_VALUE, (uint32_t *)TABLE_INVALID_VALUE,
          (uint32_t)TABLE_INVALID_VALUE, (uint32_t)TABLE_DISABLE }
};
```

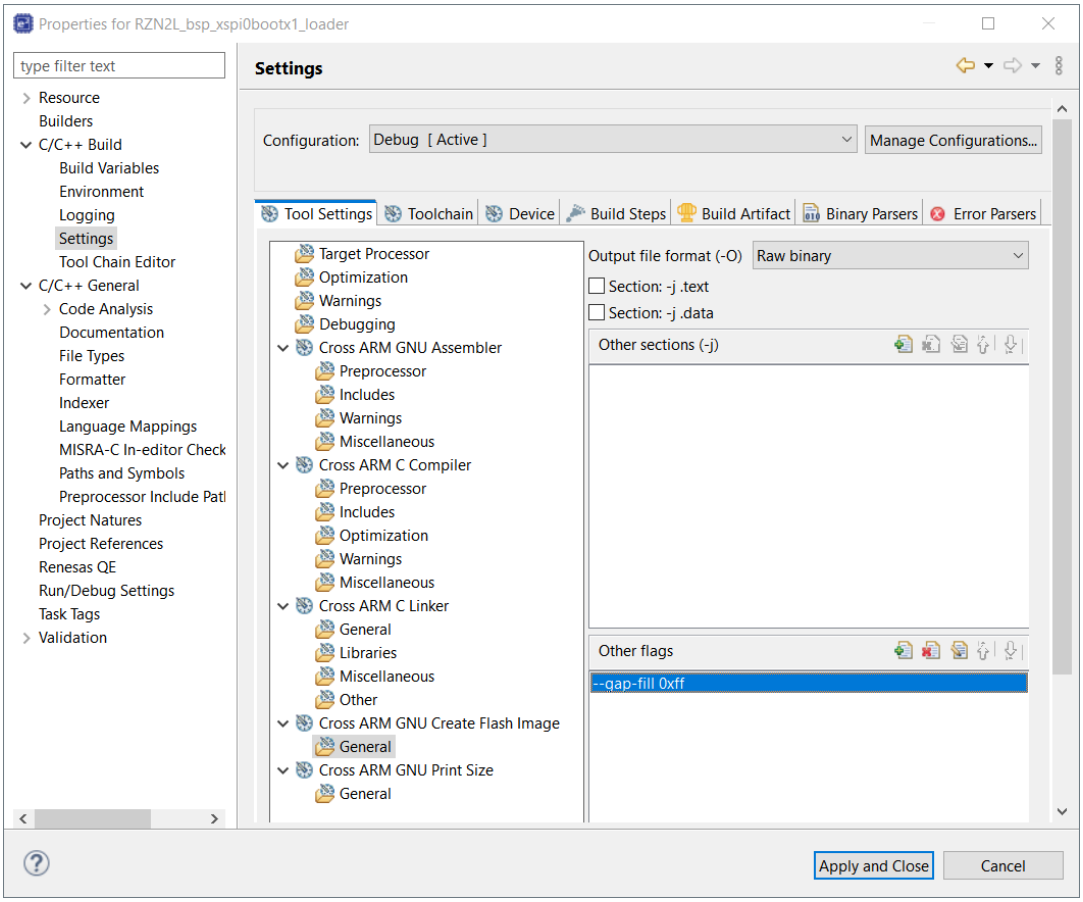
6.8 e²studio

For building and debugging two changes are needed in e²studio.

6.8.1. Output Raw Binary

Including the compiled application files to the loader application by the linker requires the binary format.

In the project of the application change the output format to raw binary. This binary is included by the .S file of the loader.



6.8.2. Loading Symbols for the Application

In the project of the loader go to menu [Run] - [Debug Configurations]. Add an entry for the new application and select the ELF-file from the workspace.

7. Revision History

Revision	Date	Description
1.00	Dec 18, 2023	First edition issued.